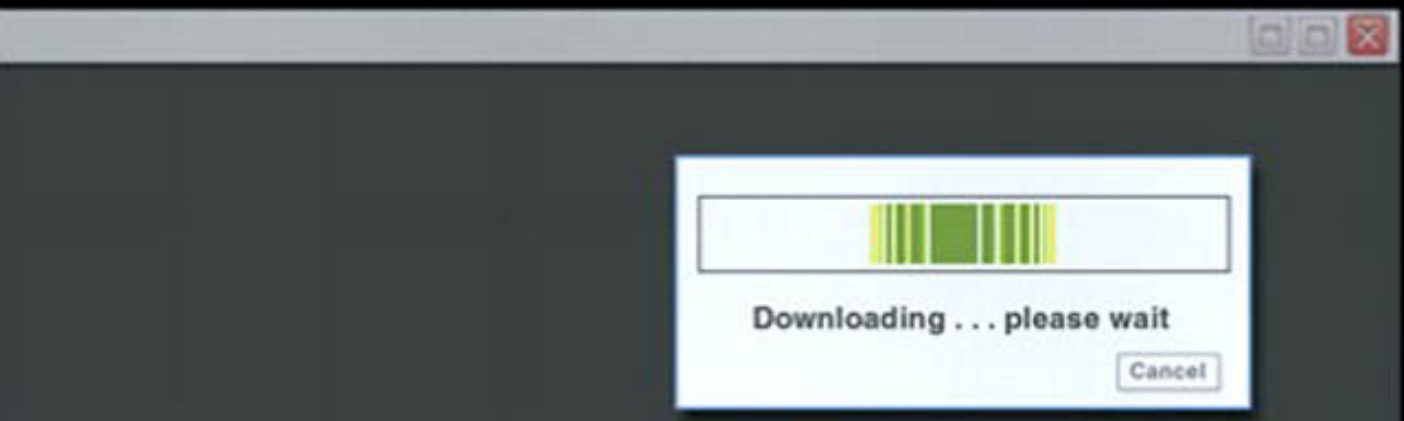
Designing and Engineering Time

The Psychology of Time Perception in Software



Steven C. Seow, Ph.D.

This Book Is Safari Enabled

Library of Congress Cataloging-in-Publication Data:
Seow, Steve.
Designing and engineering time : the psychology of time perception in software / Steve Seow. — 1st ed.
p. cm.
ISBN 0-321-50918-8 (pbk. : alk. paper) 1. Software engineering. 2. Human-computer interaction—Psychological aspects. 3. User-centered system design 4. Time perception. I. Title.
QA76.758.S456 2008
005.1—dc22

2008003570

ISBN-13: 978-0-321-50918-5
ISBN-10: 0-321-50918-8
Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing May 2008

Editor-in-Chief
Karen Gettman

Acquisitions Editor
Chris Guzikowski

Development Editor
Chris Zahn

Managing Editor
Gina Kanouse

Project Editor
Meg Shaw

Copy Editor
Keith Cline

Indexer
Cheryl Lenser

Proofreader
Jovana San Nicolas-Shirley

Publishing Coordinator
Raina Chrobak

Book Designer
Anne Jones

Compositor
Nonie Ratcliff

Preface

This book was written with several practitioners and professionals in mind. The first group is the software engineers. Not everyone has had the opportunity (or wants) to take classes in human-computer interaction, usability, or user interface design, and not every company can afford to hire usability professionals to assist with optimizing the user friendliness of a website or the discoverability of the new features of an application. If you are a developer or an architect, you are one for a good reason: Your magic and craft is turning ones and zeros into a solution that makes life and work better. However, just as it is good to know what to do if you get a flat tire and not rely on a mechanic, it is good for you to be able to apply simple concepts to improve the usability and appeal of your solution.

The second group of readers I had in mind consists of usability professionals. This group includes a subset of my colleagues at Microsoft whose job responsibilities and talent make them the "voice of the user" to the rest of the company to ensure that our products and services are easy to use, easy to learn, and so on. If you are like my colleagues, you care not only about how a product or service is used or consumed, but also how it is perceived, what emotions it may conjure up, and what behaviors it may cause. This book provides the research, the logic, and other useful knowledge to help you understand how to deal with and correct usability issues arising from time and timing.

A third group of readers is everyone else who has some responsibility to ensure that a solution is delivered with quality and value. This includes program and product managers, testers, marketing professionals, and all other decision makers involved in putting a solution in front of the user. Reality is such that we have to make tradeoffs, compromises, and workarounds. Reality also dictates that we work with the resources that we have. This book takes that into consideration and provides as much practical guidance as possible on how to set reasonable and informed tradeoffs.

Regardless of your background, one of the characteristics of this book that will catch your attention is the mishmash of information, anecdotes, examples, theories, and practices from a variety of disciplines and industries well beyond the purview of software engineering and human-computer interaction: service and retail, food and beverage, culinary, psychology, sociology, animal research, business management, entertainment,

banking, and communication, just to name a few. This eclectic presentation reflects the essence of the topic of the book—the multiple facets and universal experience of time.

This human experience of time is difficult to describe, let alone improve for the human-computer interaction. This is where leveraging knowledge from other disciplines and industries becomes vital. For example, one of my earliest epiphanies about managing the experience of time came when I was standing in the checkout line at a local Costco. While standing in line, I noticed a store employee brandishing a scanner gun and furiously zapping the merchandise in the shopping cart of a customer in front of me. When she was done, she accosted me and demanded to see my Costo card. I surrendered my card, which she zapped with the gun in a split second, before she started rummaging through my merchandise. She hunted and zapped the UPC codes within a few seconds, and when she scanned the last item, proceeded to ditch me like she did with the guy in front of me. Not known for being shy, I asked her point-blank, "What did you just do?" She nonchalantly explained that she had just rung my items up and that all I have to do when I get to the cashier is pay. Skeptical of how that works and worried that I might end up paying for someone else's merchandise, I quizzed her about how the cashier would know which group of merchandise is mine. She looked at me over her drooping reading glasses, waved the gun, and replied that she had scanned my card.

A few seconds after she walked away without giving any time for a follow-up question, it dawned on me how brilliant an idea that was. Sure enough, when I got to the register, the cashier simply scanned my card and announced how much I needed to pay. My merchandise went from the shelf, suffered a little manhandling, and straight into the trunk of my car. Costco has effectively made very intelligent use of my waiting time and has expedited my check-out process!

It wasn't long after my Costco experience that I realized that these proven techniques are applicable to the design of computer software. Why can't we ask the users all the questions up front and then proceed to start the lengthy installation? Why can't we load what users want to use immediately and then continue to load other features that are less important in the background? Why can't we provide instructions on how to use a product while the product is installing? We can, and we sometimes do, but not nearly enough because we keep seeing users throw their hands up in the air in frustration and hear them describe an application, website, or some computer as "stupid." How can this be?

At the time of this writing, the microprocessor found in the average home personal computer can perform billions of instructions per second. That is a lot of *brain* power. Surely, the computer cannot be stupid. What users are telling us is not that the computer doesn't have enough *brains,* but that it doesn't have enough *mind.* We are talking about

having enough mind to be cognizant of what is important and valuable to the user, enough mind to be clear and proper about communicating to the user, and enough mind to be considerate and intelligent about how a user's time is expensed efficiently.

This book, like a spin-off or sequel to the *Wizard of Oz,* is attempting to give the computer a mind (or more accurately, more of it). The objective of this book is to ensure that the user experience is *expeditious.* This is done by not blindly applying what has worked for other people, but to go one step further to understand why it works. I have divided the chapters into two essential parts. The first is organized by topic and provides specific knowledge, guidance, and other recommendations on issues pertaining to perception, user tolerance, responsiveness, detecting differences, progress indication, expressing time, and so forth. The second part comprises two compilations. The first is a compilation of proven techniques culled from psychological principles, business practices, industry research, and other sources. (Yes, you will find a technique here that speaks to my Costco experience.) The second is a compilation of violations that comprises the user experience in terms of how time and perception was managed. Here you will find some painfully familiar practices, such as the Time-Fluctuation Phenomenon mentioned by *PC Magazine*'s John C. Dvorak.

I hope that this book will elicit from you two responses that I get when I give talks related to the subject matter. The first is *schadenfreude*—a form of pleasure that you get from the misfortune of other people. This is commonly expressed by my audience as laughter when I relate the pain that people have to experience as a result of bad design in products and services. The second is the *bobble-head effect*—the approving nods from the audience, typically seen well before I deliver the punch line or finish describing an anecdote. Both are positive signs for me as a speaker because they indicate that the audience identifies with what they are hearing. So, if you find yourself somewhat giggly or catch yourself nodding your head as you read this book, then, to me, this book has earned its keep.

7

Expressing Time

In many instances, especially in progress indications, you need to communicate to your user some aspect of time, such as remaining time or estimated completion time. Although it may seem trivial and superficial, how you express time in the user interface (UI) can determine how your user experiences and perceives your solution. Simple words—for good or bad—can alter perception and affect tolerance. This chapter discusses when and how you should express time in the UI of your solution and introduces the use of time anchors.

The Timing of Time: Past, Present, Future

Besides *what* you express (phrases, time units, etc.) and *how* you express it (text, graphical, etc.), you also need to consider *when* you provide timing information to your user. For example, telling a person who is about to stand in line how long it will take to get to the front of the line has a different effect and serves a different purpose than informing the person who is already standing halfway in the line. Simply put, when you release information can make or break an experience. We all have our share of horror stories of how the mistiming of some information, sometimes by a matter of mere seconds, changed the course of events.

Users will use any information revealed by the UI—by design or otherwise—to form a perception of an interaction or process. This shouldn't be a surprise because we all do something similar every day. We make predictions based on patterns we see (long lines equal long wait), evaluate the quality of things based on signs and symptoms (blemishes equal carelessness), and form theories about why things happen the way they did (broken because of poor-quality parts). Likewise, users will use *any* timing information to help them understand and decide how they feel about and respond to the duration of an interaction.

Consider the following questions:

- How long will this take?
- How much longer will this take?
- How much time did that take?

Each of these three questions relates to a different temporal perspective; that is, where you are in temporal relation to some event. We can perceive durations from three basic temporal perspectives. First, we can anticipate, set expectations, or predict the duration of an event before it begins. Because the event has not happened, we'll describe this as *prospective.* In the UI, informing users how long a download will take, for example, is giving users a prospective estimate of the duration of the download. Second, we assess the duration of an event in *real time* while it is transpiring. A great example is reporting remaining time as the download is progressing. Finally, we can also evaluate the duration after event has transpired. We describe this last one as *retrospective.* Telling users how much time the download took after it has completed is an example. As mentioned, *when* you provide information about the duration of some event can influence user perception and shape user experience. Let's take a closer look at each of the perspectives.

Prospective: Tickle-Me-Elmo

It is not immediately intuitive that perception can be established, let alone be affected, before the actual experience. The key factor in prospective assessments is that a certain degree of judgment may have already been passed before the actual experience. If the judgment is negative, people can be hesitant in proceeding because they are, in essence, predicting that the experience will not match up or meet up to their expectation. An inordinate number of factors can go into forming or influencing the perception of a solution. In the retail industry, it is commonly known that people choose not to buy, use, or even try a specific solution based on its association, reputation, brand, or even price (not necessarily too expensive, but also too cheap!). On the flip-side, for the same reasons, people will go through hell and high water to get their hands on a product. In 1996, for instance, marketing hype and demand for the then-new Tickle-Me-Elmo dolls fueled fights among Christmas shoppers, some of whom reportedly paid upward of $1,500 for the $30 doll.

REPORT TIME PROSPECTIVELY WHEN

- Users need to decide if they can "afford" to start the process.
- Users would likely want to attend to other tasks.
- The process is very long or captive.

Keep in mind that most users do not devote their entire time and undivided attention to using your solution. More likely, users will have a few applications running, not to mention other active noncomputing tasks happening (talking to a customer on the phone, watching TV, etc.). Users, like computers, are efficient multitaskers. To a great extent, the mental gymnastics they have to perform to attend to several tasks simultaneously is desirable because it translates to productivity and proficiency: "While the file is downloading, let me find that e-mail Marketing sent me so that we can look at it together while we are on the phone."

Any process that hijacks or compromises the user's ability to multitask will break the user's flow and experience. Imagine if the file download requires a dial-up connection that runs on the same phone line or if the e-mail program is hogging so much network bandwidth that the file download is halted or significantly slowed down. Therefore, give users an estimate of how much time so that they can decide whether they can afford to start or engage the process (see Figure 7.1). Examples of these include instances when the process is very time-consuming or captive—that is, when the process will commandeer some level of the application, operating system, or the entire computer system such that users have to wait out the process before proceeding.

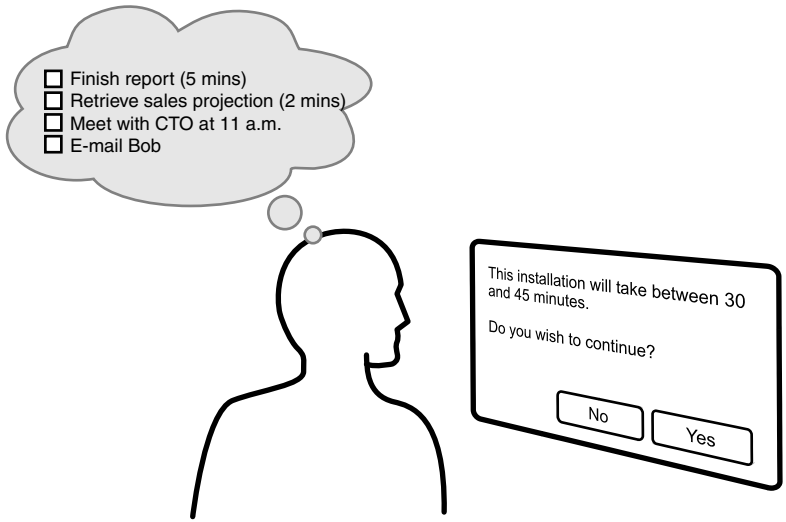


FIGURE 7.1

When time estimates are reported to the users prospectively, it allows users to decide whether they can afford to proceed with the process without breaking user flow or compromising other priorities.

Real Time: Scratch-and-Sniff

Perception obviously changes moment to moment as more information about something becomes available as it is being experienced. The retail world learned a long time ago that getting people to experience some product or service (by sight, touch, taste, smell, or sound) is one of the most powerful marketing techniques because perception built through direct experience is lasting compared to perception built vicariously through commercials and ads. This gave birth to the world of free samples, movie trailers, trial memberships, 30-day trial software, test drives, model homes, and Scratch-and-Sniff stickers. Although these work before the actual experience, they give consumers a taste, or better, a promise of the full experience. There is a flip-side to having real-time information, too. When the actual experience is not what was promised or expected, informed consumers will want to bail out. That gave birth to the world of refunds, money-back guarantees, and store credits!

REPORT TIME IN REAL TIME WHEN

- Ongoing processes and operations are too technical or meaningless to the user.
- There is a need for users to know and act on elapsed time.

Giving users real-time information about an ongoing process was the distinctive feature behind the Class A and C progress indications mentioned in Chapter 6, "Progress Indication." Providing remaining time, for example, typically provides assurance that progress is being made and progressively informs users that the wait is getting shorter. When detail about what is being done (work units) is too technical or meaningless to the user, show remaining time rather than remaining work. For example, detail about a few files being copied from one folder to another may be meaningful to mainstream users, but detail about a few hundred files being copied from the installation DVD to a temporary folder is not. Therefore, showing files copied in the former case may be acceptable, but showing time remaining might be better in the latter case. Figure 7.2 shows another example of hiding what is not meaningful and only showing what is meaningful in the UI.

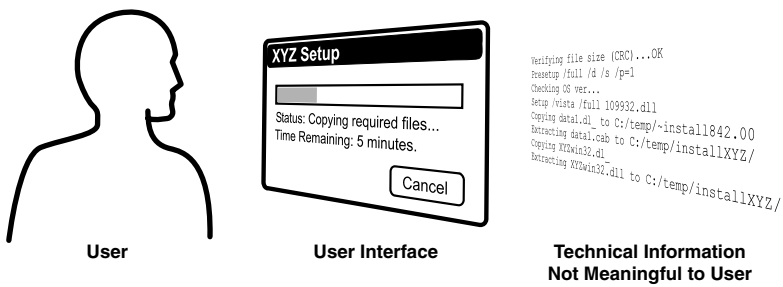


FIGURE 7.2

When detail about the process is too technical or meaningless to the user, report estimated remaining time to the user in real time.

Providing elapsed time is another way to provide real-time information, but to reiterate an important point, report elapsed time with care. As a rule of thumb, elapsed time should be used only when it is meaningful for the user to see it, such as for diagnostic purposes and performance evaluation. As a footnote, not displaying elapsed time doesn't mean not tracking elapsed time. Prompting users for actions when waits become usually long is a good idea when tracking elapsed time is useful.

Retrospective: Worst Episode Ever!

It is natural for people to evaluate a product or service after experiencing it (like the comic book guy in *The Simpsons* whose catch phrase is "Worst episode ever!") Thankfully, research has shown that it is possible to negate some ill effects of long waits with compelling value or highly desirable outcomes, particularly with great service.
Understanding how people look back at an experience is critical because there is a high likelihood that they will repeat or reengage the same experience if they found it enjoyable. Repeating or returning to what was deemed rewarding is a basic principle called Law of Effect. Beyond returning or repeating the experience, people also become pro bono marketing mouthpieces if the experience was positive (so called word-of-mouth marketing or WOMM) or a vitriolic critic if the experience was negative. In this modern age of blogs, ensuring that people look back at an experience and evaluate it positively is even more important.

Report Time Retrospectively When

- It is meaningful or valuable for users to know how long a process took.
- Diagnostic measurement or performance assessments is necessary.

Reporting elapsed time retrospectively (after a process has been completed) should be done only when it is meaningful or valuable for the user to know how long a process has taken, such as during diagnostic measurements or performance assessments (see Figure 7.3). For most mainstream applications, there is little value in telling users how much time was taken. If there is any doubt, fill in the blank: "My users will use the reported elapsed time to _____." Other than "tell how time has elapsed," whatever you can fill in the blank should reflect your users, their needs, and how they use your solution. If you cannot fill in the blank, reporting elapsed time might very well be a bad idea in your solution.

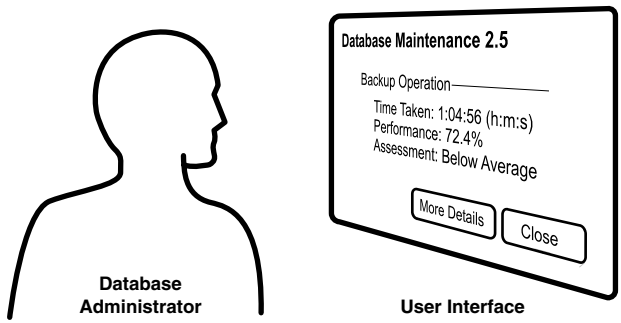


FIGURE 7.3

Report time elapsed retrospectively only when it is valuable for users to see the amount of time taken to complete the process. This diagram illustrates an example of a database administrator viewing the output of a maintenance program that informs the administrator whether the performance of the maintenance was acceptable.

Talking Time

To drink eight glasses of water every day is a recommendation some people heed religiously. The idea here is that there are health benefits, such as preventing dehydration, by simply drinking around eight glasses of water every day. This recommendation is simple until one starts to get technical about the volumetric quantity of glasses. Of course, the idea in using "glasses" is that it is easy and more practical to express than ounces or milliliters. Other than contexts in which precision is critical, such as brain surgery or aerospace engineering, for most parts of our lives we do not use or need precise measurements because estimations suffice. Perception is often inaccurate, and for most activities in life and work, it doesn't need not be. In day-to-day activities, we often rely on common objects when estimating and expressing quantity: The meatball was about the size of a golf ball. The new credit card-size digital camera is for sale next month. This way of speaking is common for estimating volume, mass, distance, and so forth.

For estimation of time, we are more likely to use proper time units (seconds, minutes, hours) instead of using references to objects or events. For example, not many people (at least in the Western culture) will state that they waited for table at a restaurant for about four to five times the amount of time it takes to boil an egg, or that the food took

about 50 Hail Marys to be served. One possible instance of when we do not use proper time units is when we compare one duration with another. For example, "By the time we got our table, we could have finished dinner at the other restaurant." Nevertheless, we're prone to simplify the estimation of time as we do for the estimation of other measures.

Time Anchors

Think about your last meaningful conversation with someone. How long did the conversation last? If you have to use time units such as seconds and minutes, there is a high likelihood that you will use whole numbers, like 1, 2, 5, or 10, to describe the duration. When we are asked to characterize durations of trivial events, we seldom give precise estimates such as 10.7 seconds or 5.17 minutes unless we are deliberately clocking the duration with a stopwatch or a wristwatch. Instead, we gravitate toward particular numbers to estimate durations. I call these *time anchors* because people tend to anchor their estimation to one or more of these numbers (see Figure 7.4). The term *anchor* is used to highlight the fact that although we know that an event lasted less than or more than five minutes, we still gravitate toward five when we have to verbalize an estimation.

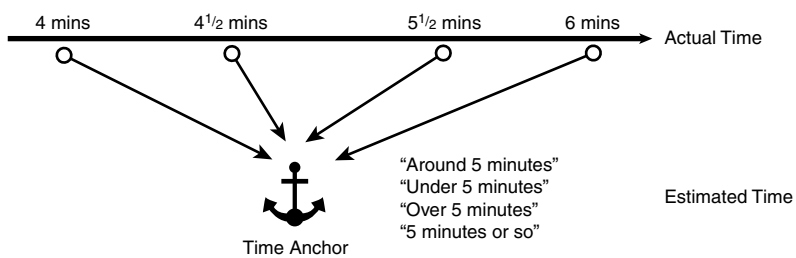


FIGURE 7.4

Although the average person can detect differences between two durations, the tendency is to use time anchors to give time estimates.

To illustrate the effects of time anchors, consider that following statements:

- The conversation with my manager lasted no more than five minutes.
- He got up to the stage and froze for about 30 seconds.
- He was more than ten minutes late for the meeting.

Now consider how odd it is if these statements did not use time anchors:

- The conversation with my manger lasted no more than 4.3 minutes.
- He got up to the stage and froze for about 34 seconds.
- He was more than 9.44 minutes late for the meeting.

Using precise units (in contexts that are trivial, casual, noncritical, or colloquial) gives the impression that the given time estimates are exact, which either leads people to trustingly assume that you timed the event or invites them to verify the accuracy of the estimates given. If you know that your conversation with your manager lasted for 4.3 minutes, chances are you weren't paying attention, were you?

Time Anchor Matrix

For time estimations under a few hours, people tend to gravitate toward the numbers 1, 2, 3, 5, 10, 15, 20, and 30. That is, when asked to estimate a short duration, people are prone to using one or more numbers, such as "about ten seconds" or "two to three minutes," in their estimates. This is observable for durations in the magnitude of hours, but to a lesser extent. A quick way to remember these numbers is to express them in what I have called a *Time Anchor Matrix* (see Figure 7.5).

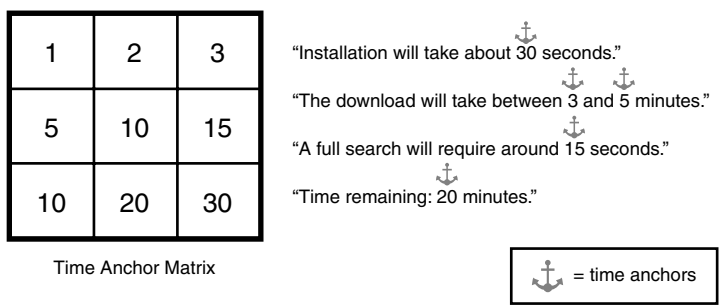


FIGURE 7.5

The numbers toward which people gravitate when expressing time can be easily remembered in the Time Anchor Matrix.

The average person can tell the difference between four minutes and eight minutes, so the matrix doesn't imply that we are only capable of estimating time using these whole numbers, or that we perceive everything in the world in chunks of time dictated by these numbers. Rather, it suggests that these are easy and practical numbers we are

comfortable and confident using when we have to describe the length of time, particularly in impromptu and trivial contexts.

Why people gravitate toward these numbers is not clear, but it is highly likely that our sexagesimal (base-60) clock system has a strong influence. Another obvious influence is language and culture. In certain parts of the world, quantifying time in seconds and minutes is not typical. In some Muslim countries, time is commonly expressed relative to the five daily Muslim prayers. In Israel, time is commonly expressed relative to an hourly news broadcast.

Talking Time

Just as humans discovered that they can communicate with aliens in musical notes in Steven Spielberg's *Close Encounters with the Third Kind* or in mathematical languages in the late Carl Sagan's novel *Contact,* we can use the Time Anchor Matrix to communicate time estimates to the user. The three main flavors of time estimation are ranges, limits, and countdowns.

Ranges: Between X and Y

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 four minutes, we can state (display in the UI) a range that spans over four minutes. Referring to the Time Anchor Matrix, we see that 3 and 5 are the integers that are the two surrounding candidates, and therefore we state that the process will take between three and five minutes (see Figure 7.6).

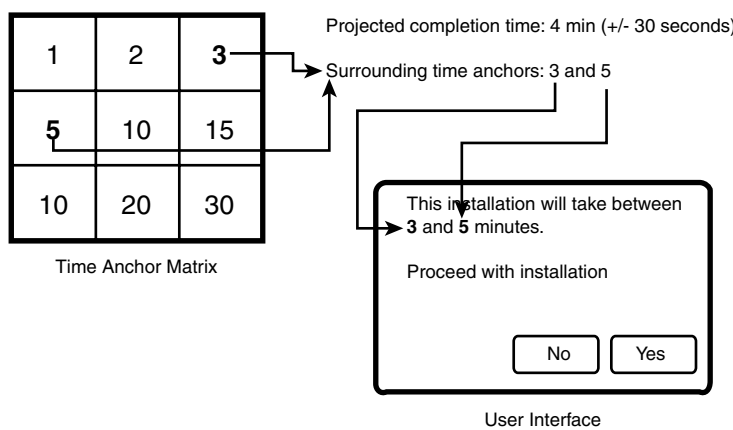


FIGURE 7.6

An example of how to use the Time Anchor Matrix to express a range of durations in the UI.

A rule of thumb for specifying ranges is not to skip over any successive time anchor. For example, we don't want to state 5 to 15 minutes because that skips over 10. The reason for not using wide ranges can be illustrated with a little imagination: Compare your response to the promise of two hypothetical cable companies, one promising that their service technician will be at your residence between 10 a.m. and 12 p.m., and the other stating that their technician will be there between 10 a.m. and 3 p.m. When the difference between two time anchors is too large, we begin to feel that the range could reasonably be tighter. Why a wide range would cause annoyance is possibly tied to a classic psychophysical principle called the Weber-Fechner Law that was mentioned in Chapter 5, "Detecting Timing Differences." Without going into detail, it is likely that using successive numbers theoretically makes it more difficult for people to perceive and "insert" one or more time anchors in between the high and low ends of the range.

Limits: Less Than or More Than X

There are two ways to express time limits, and each serves very different purposes. Lower limits tell users that a particular duration will take at least X amount of time. Upper limits tell users that a particular duration will not take more than X amount of time. You should use lower limits with care because it is essentially a warning of inevitable wait, and such statements are typically made to brace an individual for the wait or delay: *The road trip to Vancouver will take at least three hours. The package will take at least one week to get to Singapore. The house inspection will take more than two hours.* In contrast, an upper limit is a guarantee of completion: *You will reach Vancouver by midnight. You will receive the package within the month. The house inspection will be done in one hour.* Lower limits warn, and upper limits promise.

When it is not possible or wise to state a range, use the latter to state the longest possible duration in a statement such as "less than five minutes." When stating upper limits, round up to the next element in the Time Anchor Matrix. For example, if we are confident that a particular process will be completed in 7 minutes and 50 seconds, we'll state that the process will take under 10 minutes. Use the former when there is a need to warn users prospectively that a process will take a long time, especially if it is captive process when users cannot interact with parts of the application, operating system, or the entire machine until the process is completed. These are instances when you have some confidence that it will definitely take at least a certain amount of time. In such cases, state lower limits and do likewise in rounding up to the next highest time anchor. For example, if a process will take at least 3 minutes and 45 seconds, inform users that the process will take at least 5 minutes (see Figure 7.7). Underpromise and overdeliver.

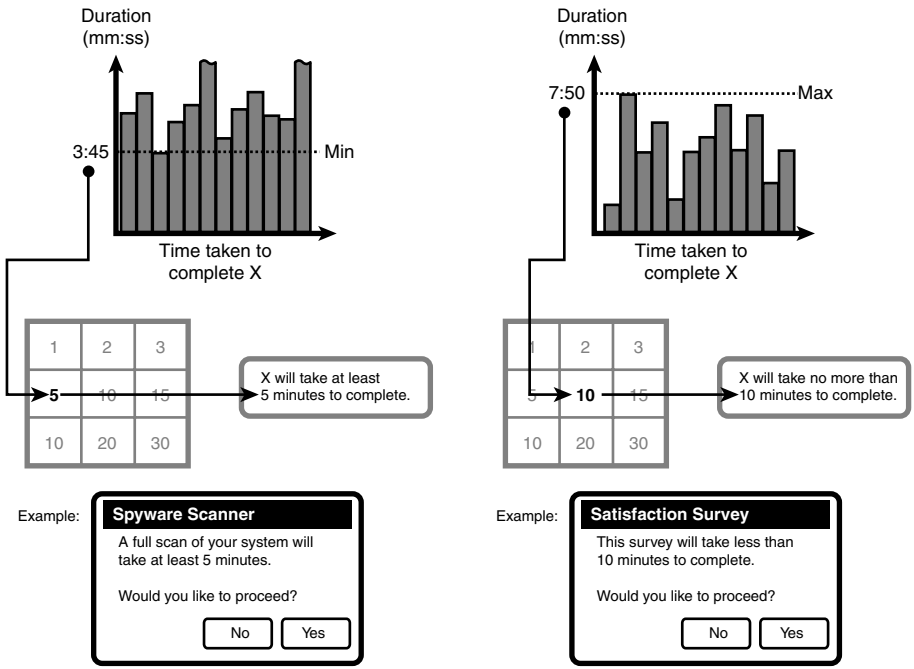


FIGURE 7.7

Two illustrations of how lower limits (left) and upper limits (right) are used. Lower limits warn user of unavoidable wait, which functions to help users decide whether they can proceed with the process (spyware scanning, for example). Upper limits guarantee that a process will complete by a certain time, and thus functions well as a persuasive mechanism to encourage users to proceed.

The argument against using specific and precise numbers, such as 7 minutes and 50 seconds or even 8 minutes, is that 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 apparently promised. This expectation is likely due to the fact that your statement is construed as one that has been made after rigorous timing and performance testing and that you have somehow locked down eight minutes with precision. In contrast, when you use the next higher integer in the matrix, ten minutes, it will more likely be perceived as a rough estimation because the colloquial prevalence of "ten minutes."

Remaining Time: Z, Y, X...

Unless there is a compelling need to report elapsed time (0:01…0:02…0:03…), it is always better to use a time-remaining or count-down timer (0:54…0:53…0:52…) if there is any need to show a timer at all. Reserve such use of timers to relatively shorter durations that are around ten minutes and shorter. Imagine watching a count-down timer from one hour: 1:00:00…0:59:59…0:59:58…0:59:57…. Obviously this would be too frustrating to watch!

Because a time-elapsed timer decrements like a clock or a stopwatch, providing it alongside a lengthy process is equivalent to inviting users to time the process, which will increase the chances of shooting yourself in the foot. Although the use of time-remaining timers is relatively and generally better, if the countdown is set to "tick" by the second, it can have the same adverse effect as a time-elapsed timer. This is where the time anchors come in handy. Instead of specifying every single second between 10 minutes and 0 seconds (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…).

As a footnote, when reporting remaining time, never allow remaining time to increase. At most, remaining time can remain unchanged momentarily, but it should never get longer, or worse, fluctuate. If your remaining time is prone to fluctuations, making remaining time difficult to predict, remaining time might not be the best progress unit to use.

Couple of Whiles

Sometimes stating ranges or upper limits is not possible because the estimated time depends on factors that are too variable or too difficult to predict. One notorious example is accessing a resource such as a database over the network. Even if the network connection speed can be determined, factors beyond the connection speed can adversely affect the access to the database. The dilemma of not being able to predict how long something will last and the need to provide some time estimate commonly leads to the use of phrases such as "a few moments," "please wait," or "a while." This practice might not be the best approach because such phrases are highly ambiguous and do not provide users with any certainty or comfort about the process.

Time and timing-related terms, such as "a few moments" or "a while," can be problematic because they are subjectively perceived and heavily dependent on context, not to mention culture, language, age, and so forth. That is, it is unlikely that such terms are quantified and perceived similarly from one person to the next, between the service provider

and the consumer, and between us and the recording on the phone that keeps promising that that someone will be with us "momentarily." The pain of being put on hold on the phone for a seemingly indefinite time really stems from *uncertainty,* and the cure then is to provide some certainty.

As an example, the customer service departments in some companies make it a practice to have staff answer a phone call as it comes but quickly inform the caller that she will be put on hold: "ABC company, can you hold please?" This practice works to a certain degree because it provides some *certainty* to the user that he or she has probably dialed the right number and has at least reached a live person on the line. Likewise, the user benefits from some certainty about *any* work being done even if a time estimate is not possible. The following are some remedies and implementations to consider in lieu of using ambiguous terms:

1. Give Non-temporal Information

The first remedy is to treat this issue as a progress indication. When the completion of the process cannot be projected and no status is available to the user, we essentially have a Class B scenario (see Figure 6.2) or a Busy/Working indication. Therefore, we want to do the exercise described in Figure 6.5 to find a way to move this to the lower-bottom quadrant to make it a Class C progress indication. In other words, we want to find all possible information and display the meaningful ones to the user (such as numbers of lines read, number of projects searched, updated queue information, compiling phase, etc.) as the process is ongoing.

2. Timers and Timeout

Although the elapsed time typically should not be used in the UI, it can be used under the covers to implement a mechanism that will respond to abnormal delays. This is found in many telecommunication devices, such as cell phones when the device will stop ringing after a fixed number of rings or a fixed amount of time. In the software world, we typically find such timeout mechanism in Internet browsers or some network-related solutions. The key in implementing this is to determine what actions to take at what time. The first step is to use data or models to map out the distribution of latencies and associate them with success and failure rates. For example, you might find that beyond 30 seconds of inactivity, failure rates are at 90%. In this case, you might want to provide a means for the user to continue, abandon, or restart the process when a delay goes beyond 30 seconds. As a relevant footnote, beyond ten seconds (captive class, see Chapter 4, "Responsiveness"), an "escape hatch," such as a Cancel or Retry button, is highly recommended.

Time Grammar and Etiquette

You should observe a few simple rules when expressing time in the UI. You will find more techniques and violations in later chapters, but here are five immediate ones to pay attention to.

1. Singularize Singular Units

It is always good practice to write the extra few lines of code to ensure that the time units are appropriately singular when necessary. For example, "1 minutes" should be "1 minute," and "1 seconds" should be "1 second." If you need to display time units that are smaller than one, use plurals, as in "0 seconds" or "0.5 minutes." (A better expression for "0 seconds" would be with a term such as *complete* or *done*.) A way to remember when to use singular units is to remember that singulars are used for exactly one, no more and no less.

2. Zero Means Finished!

In reporting remaining time, "0 seconds" implies that the process is complete and therefore should not reflect any more ongoing process. Sometimes, another process kicks in after the first process has completed, such as the unpacking of a downloaded file. What the users see, however, is a process that appears to be perpetually almost done but never does finish. The remedy is to either inform users what new process has begun or include the time needed by the extra processes into the remaining time estimates.

3. Express Time Units Consistently

The common practice is to express time units numerically, such as "This installation will take 1 to 2 minutes." It is possible to express in words, such as "one to two minutes" but never mix the two ("one to 2 minutes"). Double-digit time units are better expressed numerically, such as "15 minutes" as opposed to verbally "fifteen minutes."

4. Between X and Y

When the preposition *between* is used, make sure the conjunction *and* is used, too, as in "This installation will taken between three *and* five minutes" not "This installation will taken between three *to* five minutes." Using *to* as in "This installation will take one *to* two minutes" is fine.

5. Avoid Ambiguous Phrases

Use of the phrases *momentarily* or *a while* might be lead to more user annoyance than not. Do not use these ambiguous terms just because a process has unpredictable completion time. Refer to Chapter 6 and consider the right class of progress indication to use. The term *second* in "We'll be with you in a second," is extremely overused and is not taken as literally as it reads. An informal survey showed that the median expectation of "a second" was around six seconds, whereas "a minute" and "an hour" are likely to be taken more literally. Adjectives and adverbs, such as *immediately* and *instantly,* are also ambiguous.

Summary

When timing information is expressed in the UI is as important as how to express it. Whether information is shown prospectively, in real time or retrospectively can significantly influence user's perception, behavior, and experience. This chapter provided some guidance for expressing time. In expressing time units, use time anchors to express ranges, limits, and remaining time to prevent users from thinking that estimations are exact. Some time expression grammatical rules and etiquette are also spelled out, such as singularizing singular units.

Rabbit Hole

Prospective Versus Retrospective Time

Teigen, K. H. and K. I. Karevold (2005). Looking back versus looking ahead: Framing of time and work at different stages of a project. *Journal of Behavioral Decision Making,* 18, 229–246.

Underestimation and Overestimation of Prospective Time

Roy, M. M., N. J. S. Christenfeld, and C. R. M. McKenzie (2005). Underestimating the duration of future events memory: Incorrectly used or memory bias. *Psychological Bulletin,* 131, 738–756.

Zauberman, G. and J. G. Lynch (2005). Resource slack and propensity to discount delayed investments of time versus money. *Journal of Experimental Psychology: General,* 134, 23–37.

Writing Styles

The Chicago Manual of Style Online. (Author's note: Available online at www.chicagomanualofstyle.org/indexT.html. Various guidance under "Time.")

Anchors and Estimation of Time

Konig, C. J. (2005). Anchors distort estimates of expected duration. *Psychological Reports,* 96, 253–256.

Index

Symbols

A

B

C

D

E

F

G–H

I

J–K–L

M

N

O

P

Q–R

S

T

U

V

W–Z