CHAPTER

UX Evaluation: Data Analysis

26

If it ain't broke, it probably doesn't have enough features.

– Anonymous

Highlights

- Analyzing quantitative data:
  - Comparing results with usability targets.
  - Deciding whether you can stop iterating.
- Analyzing qualitative UX problem data.
- Cost-importance analysis to prioritize UX problems to fix.
- Lessons from the field.

Informal Summative Evaluation

A quantitative summative UX evaluation method that is not statistically rigorous and does not produce statistically significant results (Section 21.1.5.2).

Summative Evaluation

Quantitative evaluation with a goal to assess, determine, or compare a level achieved of a parameter (such as usability), especially for assessing improvement in the user experience due to formative evaluation (Section 21.1.5).

26.1 INTRODUCTION

26.1.1 You Are Here

We begin each process chapter with a "you are here" picture of the chapter topic in the context of The Wheel, overall UX design lifecycle template (Fig. 26-1). In this chapter, we show how to analyze the UX evaluation data you collected. The techniques described here generally apply regardless of the evaluation method used to obtain the data.

26.2 ANALYZE QUANTITATIVE DATA

If you collected quantitative UX data from informal summative evaluation, now is the time to analyze it.

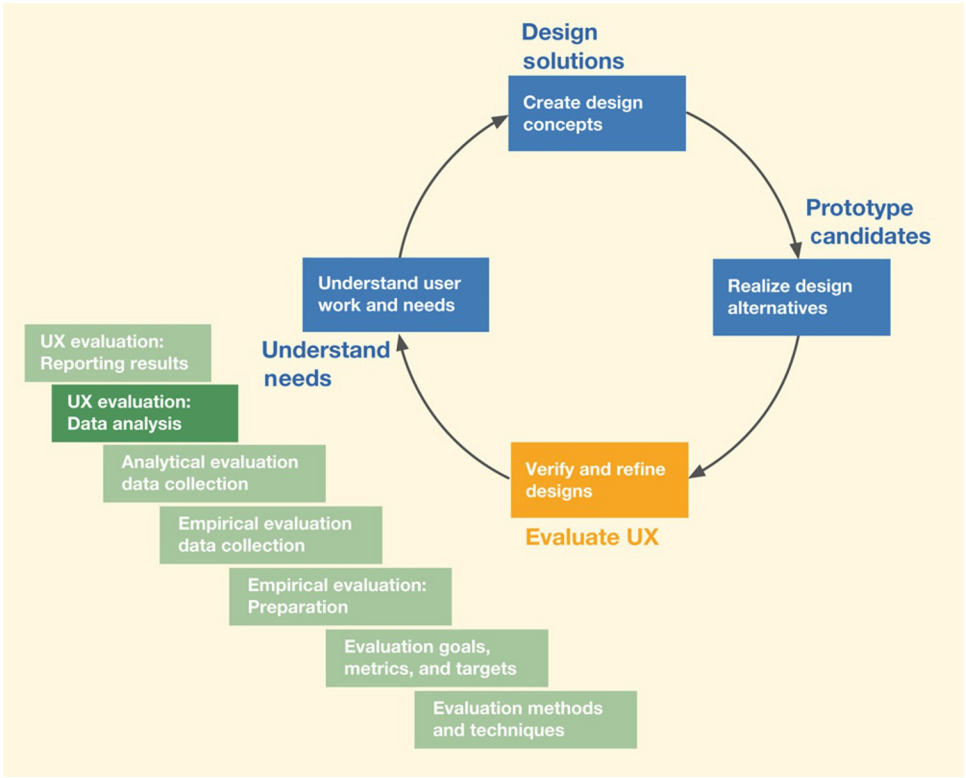

Fig. 26-1

You are here, in the chapter on data analysis within the Evaluate UX lifecycle activity, in the context of the overall Wheel lifecycle process.

Quantitative UX Evaluation Data

Numeric data taken from a measurement, such as user performance metrics or opinion ratings, used in informal summative evaluation to assess a level of achievement (Section 21.1.4.1).

Target Level

A quantitative statement of an aimed-at or hoped-for value for a UX metric, such as time on task, number of errors, questionnaire rating; the operationally defined criterion for success per that metric (Section 22.10).

26.2.1 Use Simple Descriptive Statistics

As we have said, our informal quantitative data analysis does not include inferential statistical analyses. Rather, it uses simple "descriptive" statistics (such as averages) to make an engineering determination as to whether the UX design has met the UX target levels. If the design has not yet met those targets, qualitative analysis will indicate how to modify the design to improve the UX ratings and help converge toward those goals in subsequent cycles of formative evaluation.

The first step in analyzing quantitative data is to compute averages, or whatever metrics you are using, for timing, error counts, questionnaire ratings, and so on, as stated in the UX targets (Section 22.10).

You don't usually need to worry about standard deviation values. For example, if three participants are all very close in performance times for a particular task or three questionnaires have nearly the same values for a question, the numbers should give you pretty good confidence; those averages are meaningful. If there is

a big spread, however, you should find out why there is such a variance (e.g., one user spent a huge amount of time in an error situation). Sometimes, it can mean that you should try to run a few more participants.

26.2.2 Treat Subjective Quantitative Questionnaire Data as Simply as Possible

There are many different questionnaires and some (the System Usability Scale, for example [Bangor, Kortum, & Miller, 2008]) have specialized ways of assessing and interpreting the results. For our purposes we like to use just averages. As an example, to obtain a single numerical result for a questionnaire-based metric, we might use the average of scores on questions 1, 2, 5, and 8 as averaged over all the parts. Or, more often, we'll average the scores over all the questions and over all the participants.

26.2.3 Lining Up Your Quantitative Ducks

26.2.3.1 Filling in the "observed results"

After you compute summary statistics of quantitative data across all your users, put the results in the "Observed Results" column at the end of the UX target table. As an example, partial results from a hypothetical evaluation of the Ticket Kiosk System are shown in Table 26-1 using some of the UX targets established in Chapter 22. The first two targets are based on user performance metrics and the values entered are "3.5 minutes," a count of "2" errors, and an average questionnaire score for Questions 1 through 10 of "7.5," respectively.

Participant

A participant, or user participant, is a user, potential, or user surrogate who helps evaluate UX designs for usability and user experience. These are the people who perform tasks and give feedback while we observe and measure. Because we wish to invite these volunteers to join our team and help us evaluate designs (i.e., we want them to participate), we use the term "participant" instead of "subject" (Section 21.1.3).

Subjective UX Evaluation Data

Data based on opinion or judgment, of evaluator or user (Section 21.1.4.2).

Table 26-1

Example of partial informal quantitative testing results for the Ticket Kiosk System

Work Role: User Class	UX Goal	UX Measure	Measuring Instrument	UX Metric	Baseline Level	Target Level	Observed Results	Meet Target?
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use	Initial user performance	BT1: Buy special event ticket	Average time on task	3 min as measured at the kiosk	2.5 min	3.5 min	No
Ticket buyer: Casual new user, for occasional personal use	Walk-up ease of use for new user	Initial user performance	BT2: Buy movie ticket	Average number of errors	<1	<1	2	No
Ticket buyer: Casual new user, for occasional personal use	Initial customer satisfaction	First impression	Questions Q1–Q10 in questionnaire XYZ	Average rating across users and across questions	7.5/10	8/10	7.5	No

26.2.3.2 Filling in the "meet target?" column

Next, by directly comparing the observed results with the specified target levels, you can tell immediately which UX targets have been met, and which have not, during this cycle of formative evaluation. It is useful to add, as we have done in Table 26-1, yet one more column to the right-hand end of the UX target table, for "Did you meet UX target?" Entries can be Yes, No, or Almost.

In looking at the example results in Table 26-1, in this round of evaluation, you can see that we didn't meet any of these three UX evaluation targets. This is not unusual for an early evaluation of a still-evolving design.

26.2.4 The Big Decision: Can We Stop Iterating?

Now it is time for a major project management decision: Should you continue to iterate? This decision should be a team affair and made at a global level, not just considering quantitative data. Here are some questions to consider:

- Did you simultaneously meet all your target-level goals?
- What is your general team feeling about the conceptual design, the overall UX design, the metaphor, and the user experiences they have observed?
- What role does pressure from management and marketing play in this decision?

If your answers to these questions tell you that you can accept the design as is, you can stop iterating. Or resource limitations also can force you to stop iterating and get on with pushing this version out in the hope of fixing known flaws in the next version. If and when you do decide to stop iterating, don't throw your *qualitative* data away, though; you paid to get it, so keep this round of problem data for next time.

If your UX targets were not met—the most likely situation after the first cycle(s) of testing—and resources permit (e.g., you are not out of time or money), you need to iterate. This means analyzing the UX problems and going back through the lifecycle again to find a way to solve them in order of their cost and effect on the user experience.

26.2.4.1 Convergence toward a quality user experience

Following our recurring theme of using your own thinking and experience in addition to following a process, we point out that this is a good place to use your intuition. As you iterate, you should keep an eye on the quantitative results over multiple iterations: Is your design at least moving in the right direction?

It is always possible for UX levels to get worse with any round of design changes. If you are not converging toward improvement, why not? Are UX

problem fixes uncovering problems that existed but could not be seen before, or are UX problem fixes causing new problems?

26.3 ANALYZE QUALITATIVE UX DATA

26.3.1 Overview

Our friend Whitney Quesenbery gave us this nutshell digest of her approach to usability problem analysis, which she, in turn, adapted from someone else:

> *The team usually includes all the stakeholders, not just UX folks, and we rarely have much time. First, we agree on what we saw. No interpretation, just observation. This gets us all on the same page. Then we brainstorm until we agree on "what it means." Then we brainstorm design solutions.*

Formative analysis of qualitative data is the bread and butter of UX evaluation. The goal of formative data analysis is to identify UX problems and causes (design flaws) so that they can be fixed, thereby improving product user experience. The process of determining how to convert collected data into scheduled design and implementation solutions is essentially one of negotiation in which, at various times, all members of the project team are involved.

Qualitative UX Evaluation Data

Nonnumeric descriptive data taken, for example, while observing user task performance, used to find and fix UX problems (Section 21.1.4.1).

26.3.2 Analysis Preparation Steps

26.3.2.1 Keep a participant around to help with early analysis

In a typical way of doing things, it isn't until the final participant for data collection data analysis is dismissed before the team turns its attention to data analysis. But this approach can put the problem analyst at a disadvantage when the need inevitably arises to ask the participant questions. So, we suggest that you start analyzing qualitative data for each participant while that participant is still present to fill in missing data and clarify ambiguous issues.

26.3.2.2 Multiple sources of raw UX data

Regardless of the source of the raw evaluation data, most of the data analysis we do in this chapter is essentially the same.

Qualitative UX data notes can include:

- Critical incident comment.
- User think-aloud comment.
- UX inspection note.

Inspection (UX)

An analytical evaluation method in which a UX expert evaluates an interaction design by looking at it or trying it out, sometimes in the context of a set of abstracted design guidelines. Expert evaluators are both participant surrogates and observers, asking themselves questions about what would cause users problems and giving an expert opinion predicting UX problems (Section 25.4).

26.3.2.3 Clean up the raw data before your memory fades

However you get data, you probably still have a large number of raw data notes at this point. Many of your notes are likely to be terse observational comments that will be difficult to expand and interpret if:

- There is a delay between data collection and analysis.
- The person performing UX problem analysis is not the same person who observed the incidents and recorded the comments.

Therefore, it is essential to go through your UX data notes to clean up and flesh out the raw data, especially emotional impact data, as soon after data collection as possible, while perishable detailed data are still fresh.

26.3.3 Qualitative UX Data Analysis Steps

Fig. 26-2 illustrates the steps of qualitative data analysis, each of which is discussed in the following sections:

1. Gather up your raw qualitative UX data notes.
2. Extract elemental data notes (just as we did in analysis for usage research).
   a. Each elemental data note should be about exactly one UX problem.
3. Edit elemental data notes into UX problem.
4. Consolidate multiple notes about the same UX problem description.
5. Group related UX problem descriptions so they can be fixed together.

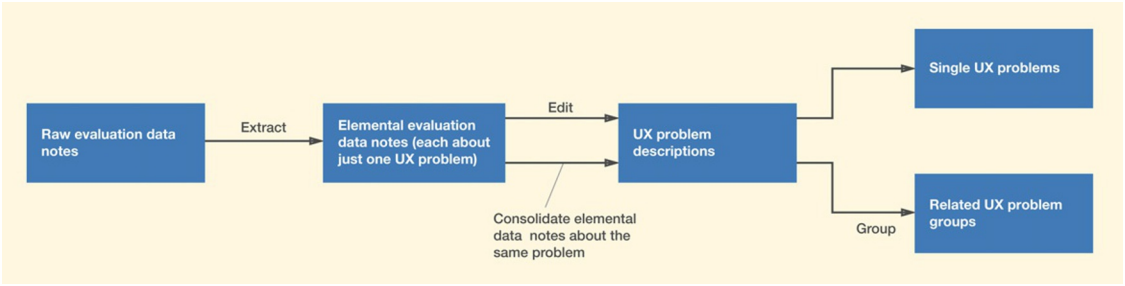

Fig. 26-2
Extracting, editing, consolidating, and grouping elemental UX evaluation data notes into UX problem descriptions.

26.3.3.1 Gather up your raw qualitative UX data notes

Didn't find many UX problems? Better look again at your data collection process. We seldom, if ever, see a UX design for which UX testing does not reveal lots of UX problems. Absence of evidence is not evidence of absence.

26.3.3.2 Extract elemental data notes: Each refers to just one problem

On occasion, participants can experience more than one distinct UX problem at the same time and a resulting UX data note can refer to all of those problems. Look through your data notes for any such notes about more than one UX problem, and separate them into multiple elemental data notes, each about a single UX problem. This is the same way we distilled the essence of raw usage research data notes in Section 8.2.

Elemental Data Note

A data note from either usage research or UX evaluation that is brief, clear, concise, and refers to or relates to exactly one concept, idea, fact, or topic (Section 8.2.2).

Here is an example from one of our UX evaluation sessions for a companion website for the Ticket Kiosk. The participant was in the middle of a benchmark task that required her to order three tickets to a Three Tenors concert. As she proceeded through the task, at one point she could not locate the button (which was below the "fold") to complete the transaction.

When she finally scrolled down and saw the button, the button label said, "Submit." At this point, she remarked, "I am not sure if clicking on this button will let me review my order or just send it in immediately." This is an example of a critical incident note that needs to be divided into separate elemental notes:

1. The button is located where it is not immediately visible.
2. The label is not clear enough to help the user make a confident decision.

26.3.3.3 Edit raw UX data notes into UX problem descriptions

In UX data analysis, we have to sort through the raw UX data notes and extract the essential associated UX problem information. In editing from UX data notes to UX problem descriptions, you are cleaning up the wordings, getting rid of fluff and noise, filling in missing words, making complete sentences, and generally making the descriptions readable to all team members. This is also a good time for a reality check on the value of this problem description, keeping only ones that represent "real" UX problems that are worth fixing.

Depending on the size and complexity of your project, you can keep management of your UX problem descriptions simple by typing them into a word processor or spreadsheet. The earlier you can get your raw critical incident notes packaged as data records, the more expedient the transition to subsequent data analysis.

As you create each UX problem description, think not only of the user but also of the UX team. This means including enough information to make the UX problem description as useful as possible for data analysis so team members can:

- Understand the problem in its usage context.
- Glean insight into its causes and possible solutions.
- Be conscious of relationships among similar problems.
- Come up with suitable redesign solutions.

To that end, we suggest that you consider including these kinds of information:

Problem name: So people can refer to it in other contexts and discussions

Problem statement: Terse one-sentence summary of the problem as an effect or outcome experienced by the user, but not as a suggested solution. You want to keep your options flexible when you do get to considering solutions.

User goals and task information: This information provides problem context to know what the user was trying to do when the problem was encountered.

What the user tried to do, what happened instead, and why: It's important to explain what actually happened and what incorrect assumptions about the design or misunderstandings of how the design works led to it. Explain what the user should have done.

Causes and potential solutions: Although you may not know the problem causes or potential solutions at first, you should explain it if you do know or have any ideas.

26.3.3.4 Consolidate congruent data notes

You are likely to encounter some different data notes about the same UX problem, especially if you have used multiple participants to perform the same tasks. We use the term *congruent* to refer to multiple UX data notes that are about the same underlying UX problem (not just similar problems or problems in the same category).

As you are editing these notes into UX problem descriptions, if a UX problem description already exists for this data note, consolidate and merge it into the existing problem description.

If you have trouble knowing if two problem descriptions are about the same underlying problem, Capra (2006, p. 41) suggests a practical solution-based criterion: "Two problems, A and B, were considered the same [congruent] if fixing problem A also fixes problem B, and fixing problem B also fixes problem A." Capra's approach is based on criteria used in the analysis of UX reports collected in CUE-4 (Molich & Dumas, 2008). The symmetric aspect of this criterion rules out a case where one problem is a subset of the other.

As an example, from our Ticket Kiosk System evaluation, one UX problem description states that the participant was confused about the button labeled "Submit" and didn't know that this button should be clicked to move on in the transaction to pay for the tickets. Another (congruent) UX problem description (as encountered by a different participant) said that the participant complained about the wording of the button label "Submit," saying it didn't help understand where one would go if one clicked on that button.

26.3.3.5 Group related UX problem descriptions to be fixed together

UX problems can be related in many different ways that call for you to consider fixing them at the same time:

- Problems may be in physical or logical proximity (e.g., may involve objects or actions within the same dialogue box).
- Problems may involve objects or actions used in the same task.
- Problems may be in the same category of issues or design features but scattered throughout the UX design.
- Problems may have consistency (or other) issues that require similar treatments.
- Observed problem descriptions are indirect symptoms of common, more deeply rooted, UX problems.
  - A telling indicator of such a deeply rooted problem is complexity and difficulty in its analysis.

The idea is to create a common solution that might be more general than required for a single problem, but which will be the most efficient and consistent for the whole group.

Example: Grouping Related Problems for the Ticket Kiosk System

Consider Table 26-2, adapted with permission, from a student team in one of our classes. In the second column, we list five UX problems, all related to distinguishability of seat categories (e.g., availability). The right-hand column contains corresponding suggested solutions.

These problems may be indicative of a much broader design problem: a lack of effective visual design elements in seat selection part of the workflow. We can group and label all these problems into a problem group, along with a group solution (Table 26-2):

Table 26-2

A group of related UX problems and solutions in a spreadsheet (thanks to Sirong Lin and her student project team)

Seat category distinguishability usability problem	9. User expected graphic of seat layout but missed seeing the button for that at first; kept missing "View Seats"	Group "View Seats" button in layout with other seat purchase task controls
	13. For "Selected seats," no way to distinguish balcony from floor seats because they use the same numbering scheme	Distinguish balcony seats and floor seats with different numbering schemes
	20. In "View Seats" view, participant couldn't determine which seats were already sold, because color choices were confusing	Use different icons or colors to distinguish which seats are already sold
	25. Missed fact that blue seat sections are clickable to zoom in on detailed view of available seats. User left thinking that blue seat section wasn't enough information about *which* seats are available	Show clearly that blue seat sections are clickable. When one is clicked, display the detailed seat information, like location, price, etc. Add legend to explain color usage
	26. Seat availability color-coding scheme problematic. Colors not distinguishable (color blindness?). In detailed seat view, purple wasn't distinguishable as separate between red and blue. Labels in this view not legible enough. Probably should have thicker font (maybe bold would do it)	Change the colors to a better combination with which the user can distinguish the different categories of seats clearly. Use a thicker font

Style Guide

A document fashioned and maintained by designers to capture and describe details of visual and other general design decisions, especially about screen designs, font choices, iconography, and color usage, which can be applied in multiple places. A style guide helps with consistency and reuse of design decisions (Section 17.8.1).

Group 1: Visual designs for seat selection workflow.
Group 1 Solution: Comprehensively revise all visual design elements for seat selection workflow. Update style guide accordingly.

26.3.3.6 Usage research analysis tools work here, too

For projects with large numbers of UX data notes, you might need a way to organize the resulting UX problem descriptions to make logical sense of them. This usually means organize them by category—by feature, area of functionality, task, user activity, etc. Putting your problem descriptions into an affinity diagram can help you:

- Find congruencies.
- Find problem descriptions to be grouped for fixing together.

26.3.3.7 Higher level common issues within groups

When UX problem data include a number of critical incidents or problems that are quite similar, you will group these descriptions together because they are closely related. Then, you usually look for common issues among the problems in the group.

But sometimes, the real problem is not explicit in the commonality within the group, but the problems only represent symptoms of a higher-level problem. You might have to deduce that this higher-level problem is the real underlying cause of these common critical incidents.

For example, in one application we evaluated, users were having trouble understanding several different quirky and application-specific labels. We first tried changing the label wordings, but eventually we realized that the reason they didn't "get" these labels was that they didn't understand an important aspect of the conceptual design. Changing the labels without improving their understanding of the model didn't solve the problem.

26.3.4 UX Problem Data Management

In large projects, management of the sheer numbers of UX problem descriptions can be a challenge, and you might have to adopt a database management approach. We leave it up to you to work this out in a way that works for your project. For more about UX problem data management, see Section 28.10.

26.3.5 Rapid Qualitative Data Analysis

As a rapid approach to qualitative data analysis:

- Just take notes about UX problems in real time during the data collection session.
- Immediately after session, make UX problem records from the notes.

As an alternative, if you have the necessary simple tools for creating UX problem records:

- Create UX problem records as you encounter each UX problem during the session.
- Immediately after the session, expand and fill in missing information in the records.
- Analyze each problem, focusing on the real essence of the problem and noting causes (design flaws) and possible solutions.

26.4 COST-IMPORTANCE ANALYSIS: PRIORITIZING PROBLEMS TO FIX

Cost-importance analysis is an approach to prioritizing the time and effort of fixing UX problems found in UX evaluation based on priority ratios, which are calculated by dividing the importance of making a change by the cost.

Of course, we want to fix all known UX problems after each iteration of evaluation. However, because resources are limited, and time is short, we have to take the engineering approach of prioritizing problems to fix.

We call this cost-importance analysis because it is based on calculating tradeoffs between the cost to fix a problem and the importance of getting it fixed. Cost-importance analysis applies to any UX problem list regardless of what evaluation method or data collection technique was used.

Although these simple calculations can be done manually, this analysis lends itself nicely to the use of a simple spreadsheet. The basic form we will use is the cost-importance table shown in Table 26-3.

26.4.1 Problem

Starting with the left-most column in Table 26-3, we enter a concise description of the problem. Analysts needing to review further details can consult the original problem data notes. We will use some sample UX problems from the Ticket Kiosk System to illustrate how we fill out the entries in the cost-importance table.

Table 26-3

Basic form of the cost-importance table

Problem	Imp.	Solution	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution

In our first example problem, the user had decided on an event to buy tickets for and had established the parameters (date, venue, seats, price, etc.) but didn't realize that it was then necessary to click on the "Submit" button to finish up the event-related choices and move to the screen for making payment. So, we enter a brief description of this problem in the first column of Table 26-4.

Table 26-4

Problem description entered into cost-importance table

Problem	Imp.	Solution	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution
User unaware of the need to click on the "Submit" button to proceed to payment							

26.4.2 Importance to Fix

The next column is for an estimate of the importance to fix the problem, independent of cost. While importance includes severity or criticality of the problem, most commonly used by other authors, this parameter can also include other considerations. The idea is to capture the effect of a problem on user performance, user experience, and overall system integrity and consistency. Importance can also include intangibles such as management and marketing "feelings" and consideration of the cost of not fixing the problem (e.g., in terms of lower user satisfaction).

Because an importance rating is just an estimate, we use a simple scale for the values:

- Importance = *M*: Must fix, regardless
- Importance = 5: The most important problems to fix after the "Must fix" category
  - The UX feature involved is mission critical.
  - The UX problem has a major impact on task performance or user satisfaction (e.g., user cannot complete key task or can do so only with great difficulty).
  - The UX problem is expected to occur frequently and/or could cause costly errors.
- Importance = 3: Moderate impact problems
  - The user can complete the task, but with some difficulty (e.g., it caused confusion and required some extra effort).
  - The problem was a source of moderate dissatisfaction.
- Importance = 1: Low impact problems
  - The problem didn't impact task performance or dissatisfaction much (e.g., mild user confusion or irritation or a cosmetic problem), but is still worth listing.

This fairly coarse gradation of values has proven to work for us. You can customize it to suit your project needs.

26.4.2.1 Importance rating adjustments

We also need some flexibility to assign intermediate values, so we allow for importance rating adjustment factors, the primary one of which is estimated frequency of occurrence. If this problem is expected to occur very often, you might adjust your importance rating upward by one value.

Conversely, if it is not expected to occur very often, you could downgrade your rating by one or more values. As Karat, Campbell, and Fiegel (1992) relate frequency of occurrence to problem severity classification, they ask: Over all the affected user classes, how often will the user encounter this problem?

For example, consider the Ticket Kiosk System problem about users being confused by the button label "Submit" to proceed to payment in the ticket-purchasing transaction. Because this was not shown to be a show-stopper, we initially assigned it an importance of 3. But because it will be encountered by almost every user in almost every transaction, we "promoted" it to a 4, as shown in Table 26-5.

Table 26-5

Estimate of importance to fix entered into cost-importance table

Problem	Imp.	Solution	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution
User unaware of the need to click on the "Submit" button to proceed to payment	4						

Learnability can also be an importance adjustment factor. Some problems have most of their impact on the first encounter. After that, users learn quickly to overcome (work around) the problem, so it does not have much effect in subsequent usage. That could call for an importance rating reduction.

26.4.3 Solutions

The next column in the cost-importance table is for one or more candidate solutions to the problems. Solving a UX problem is redesign, a kind of design, so you should use the same approach and resources as we did for the original design, including consulting your usage research data. Other resources and activities that might help include design principles and guidelines, ideation and sketching, study of other similar designs, and solutions suggested by users and experts. It is almost never a good idea to think of more training or better documentation as a UX problem solution–that is fixing the user, not the UX design.

Example: Solution for Ticket Kiosk System Problem

Coming back to the confusing button label in the Ticket Kiosk System, one obvious and inexpensive solution is to change the label wording to better represent where the interaction will go if the user clicks on that button. Maybe "Proceed to payment" would make more sense to most users.

We wrote a concise description of our proposed fix in the Solution column in Table 26-6.

Table 26-6

Potential problem solution entered into cost-importance table

Problem	Imp.	Solution	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution
User unaware of the need to click on the "Submit" button to proceed to payment	4	Change label wording to "Proceed to Payment"					

26.4.4 Cost to Fix

The Cost column is where you enter your estimate of the cost to fix this problem. You should use the cost to fix the design at the current stage of development in the project. For example, the cost to fix almost any problem in a paper prototype is low, even including the cost to brainstorm a solution. But, in later stages, with medium- to high-fidelity prototypes and programmed prototypes, costs can be substantially higher.

Making accurate estimates of the cost to fix a given UX problem takes practice; it is an acquired engineering skill. But it is nothing new; it is part of our job to make cost estimates in all kinds of engineering and budget situations. For our analysis, costs are stated in terms of resources needed, which almost always translates to person-hours required.

Because this is an inexact process, we usually round up fractional values just to keep it simple. When you make your cost estimates, don't make the mistake of including only the cost to implement the change; you must include the cost of redesign and discussion and, sometimes, even some prototyping and experimentation. If you have a programmed prototype, you might need help from your software developers to estimate implementation costs.

Because it is very easy to change label wordings in our Ticket Kiosk System, we have entered a value of just one person-hour into the Cost column in Table 26-7.

Table 26-7

Estimate of cost to fix entered into cost-importance table

Problem	Imp.	Solution	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution
User unaware of the need to click on the "Submit" button to proceed to payment	4	Change the label wording to "Proceed to Payment"	1				

26.4.4.1 Cost values for problem groups

Table 26-8 shows an example of including a problem group in the cost-importance table. Note that the cost for the group is higher than that of either individual problem, but lower than their sum.

26.4.4.2 Calibration feedback from down the road: Comparing actual with predicted costs

To learn more about making cost estimates and to calibrate your engineering ability to estimate costs to fix problems, we recommend that you add a column to your cost-importance table for actual cost. After you have done the redesign and implementation for your solutions, you should record the actual cost of each and compare with your predicted estimates. It can tell you how you are doing and help you improve your estimates.

26.4.5 Priority Ratio

The next column in the cost-importance table, the priority ratio, is a metric we use to establish priorities for fixing problems. We want a metric that will reward high importance but penalize high costs. A simple ratio of importance to cost fits this bill. Intuitively, a high importance will boost up the priority but a high cost will bring it down. Because the units of cost and importance will usually yield a fractional value for the priority ratio, we scale it up to the integer range by multiplying it by an arbitrary factor, say, 1000.

If the importance rating is "*M*" (for "must fix regardless"), the priority ratio is also "*M*." For all numerical values of importance, the priority ratio becomes:

$$\text{Priority ratio} = (\text{importance}/\text{cost}) \times 1000$$

Table 26-8

Cost entries for problem groups entered into cost-importance table

Problem Group	Problem	Imp.	Solution	Group Solution	Single Costs	Group Cost
Transaction flow for purchasing tickets	7. The user wanted to enter or choose date and venue first and then click "Purchase Tickets," but the UX design required them to click on "Purchase Tickets" before entering specific ticket information	3	Change flow to allow actions in either order and label it so	Establish a comprehensive and more flexible model of transaction flow and add labeling to explain it	3	5
	17. The "Purchase Tickets" button took user to screen to select tickets and commit to them, but then users didn't realize they had to continue on to another screen to pay for them		Provide better labeling for this flow		3	

Example: Priority Ratios for Ticket Kiosk System Problems

For our first Ticket Kiosk System problem, the priority ratio is $(4/1) \times 1000 = 4000$, which we have entered into the cost-importance table in Table 26-9.

Table 26-9

Priority ratio calculation entered into cost-importance table

Problem	Imp.	Solution	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution
User unaware of the need to click on the "Submit" button to proceed to payment	4	Change the label wording to "Proceed to Payment"	1	4000			

In the next part of this example, shown in Table 26-10, we have added several more Ticket Kiosk System UX problems to fill out the table a bit more realistically.

Table 26-10

Priority ratios for more Ticket Kiosk System problems

Problem	Imp.	Solution	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution
User unaware of the need to click on the "Submit" button to proceed to payment	4	Change the label wording to "Proceed to Payment"	1	4000			
Didn't recognize the "counter" as being for the number of tickets. As a result, user failed to even think about how many tickets he needed	*M*	Move quantity information and label it	2	*M*			
Unsure of current date and what date he was purchasing tickets for	5	Add current date field and label all dates precisely	2	2500			
Users were concerned about their work being left for others to see	5	Add a timeout feature that clears the screens	3	1667			
User confused about "Theatre" on the "Choose a domain" screen. Thought it meant choosing a physical theater (as a venue) rather than the category of theatre arts	3	Improve the wording to "Theatre Arts"	1	3000			
Ability to find events hampered by lack of a search capability	4	Design and implement a search function	40	100			
Didn't recognize what geographical area theater information was being displayed for	4	Redesign graphical representation to show search radius	12	333			

Continued

Table 26-10 Priority ratios for more Ticket Kiosk System problems —cont'd

Problem	Imp.	Solution	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution
Didn't like having a "Back" button on second screen because first screen was only a "Welcome"	2	Remove it	1	2000			
Transaction flow for purchasing tickets (group problem; see Table 26-8)	3	Establish a comprehensive and more flexible model of transaction flow and add labeling to explain it	5	600			

Note that although fixing the lack of a search function (the sixth row in Table 26-10) has a high importance, its high cost is keeping the priority ratio low. This is one problem to consider for an Importance = *M* rating in the future. At the other end of things, the next-to-last problem (about the Back button to the Welcome screen) is only Importance = 2, but the low cost boosts the priority ratio quite high. Fixing this will not cost much and will get it out of the way.

26.4.6 Priority Rankings

The next step is to sort the cost-importance table by priority ratios to get the final priority rankings, the order in which to fix the problems.

First, move all problems with a priority ratio value of "*M*" to the top of the table. These are the problems you must fix, regardless of cost. Then sort the rest of the table in descending order by priority ratio. This puts high-importance, low-cost problems (shown at **A** in the upper left-hand quadrant of Fig. 26-3) at the top of the priority list. These are the problems to fix first, the fixes that will give the biggest bang for the buck.

You might think that, in the real world, you won't see many problems having high importance in combination with low cost. You're thinking you have to pay for what you get. But, in fact, we often find many problems of this kind in early iterations. A good example is a badly worded button label. It can completely confuse users, but usually costs almost nothing to fix.

In contrast, the UX problems that sort to the bottom of the priority list are costly to fix with relatively little gain in doing so. You will probably not bother to fix these problems, as shown at **B** in the lower right-hand quadrant of Fig. 26-3.

Quadrants **A** and **B** sort out nicely in the priority rankings. Quadrants **C** and **D**, however, may require more thought. Quadrant **C** represents problems for which fixes are low in cost and low in importance. You will usually just go ahead and fix them to get them off your plate. The most difficult choices appear in quadrant **D** because, although they are of high importance to fix, they are also the most expensive to fix.

No formula will help; you need good engineering judgment. Maybe it is time to request more resources so these important problems can be fixed. That is usually worth it in the long run.

A cost-importance table for sample UX problems in the Ticket Kiosk System, sorted by priority ratio, is shown in Table 26-11.

26.4.7 Cumulative Cost

The next step is simple. In the "Cuml. Cost" column of the table, sorted by priority ratio, is the cost of fixing each problem plus the cost of fixing all the problems above it in the table. See how we have done this for our example Ticket Kiosk System cost-importance table in Table 26-11.

26.4.8 The Line of Affordability

Your team leader or project manager should determine your "resource limit," in person-hours, that you can afford to allocate to making design changes for the current cycle of iteration. For example, suppose that for the Ticket Kiosk System, we have only a fairly small amount of time available in the schedule, about 16 person hours.

Draw the "line of affordability," a horizontal line in the cost-importance table just above the line in the table where the cumulative cost value first exceeds your resource limit. For the Ticket Kiosk System, the line of affordability appears just above the row in Table 26-11 where the cumulative cost hits 27.

If you have time for more learning about your process, it might be interesting to graph the problems in a cost-importance space like that of Fig. 26-3. Sometimes, this kind of graphical representation can give insight into your process, especially if your problems tend to appear in clusters. Your line of affordability will be a vertical line that cuts the cost axis at the amount you can afford to spend on fixing all problems in this iteration.

26.4.9 Drawing Conclusions: A Resolution for Each Problem

It's time for the payoff of your cost-importance analysis. It's time to decide how each problem will be addressed.

Table 26-11

The Ticket Kiosk System cost-importance table, sorted by priority ratio, with cumulative cost values entered, and the "line of affordability" showing the cutoff for this round of problem fixing

Problem	Imp.	Solution	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution
Didn't recognize the "counter" as being for the number of tickets. As a result, user failed to even think about how many tickets he needed	*M*	Move quantity information and label it	2	*M*	1	2	
User unaware of the need to click on the "Submit" button to proceed to payment	4	Change the label wording to "Proceed to Payment"	1	4000	2	3	
User confused about "Theatre" on the "Choose a domain" screen. Thought it meant choosing a physical theater (as a venue) rather than the category of theatre arts	3	Improve the wording to "Theatre Arts"	1	3000	3	4	
Unsure of current date and what date he was purchasing tickets for	5	Add current date field and label all dates precisely	2	2500	4	6	
Didn't like having a "Back" button on second screen because first screen was only a "Welcome"	2	Remove it	1	2000	5	7	
Users were concerned about their work being left for others to see	5	Add a timeout feature that clears the screens	3	1667	6	10	
Transaction flow for purchasing tickets (group problem; see Table 26-8)	3	Establish a comprehensive and more flexible model of transaction flow and add labeling to explain it	5	600	7	15	
Line of affordability (16 person-hours—2 work days)							
Didn't recognize what geographical area theater information was being displayed for	4	Redesign graphical representation to show search radius.	12	333	8	27	
Ability to find events hampered by lack of a search capability	4	Design and implement a search function.	40	100	9	67	

First, you have to deal with your "Must fix" problems, the show-stoppers. If you have enough resources, that is, if all the "Must fix" problems are above the line of affordability, fix them all. If not, you already have a headache. Someone, such as the project manager, has to earn his or her pay today by making a difficult decision.

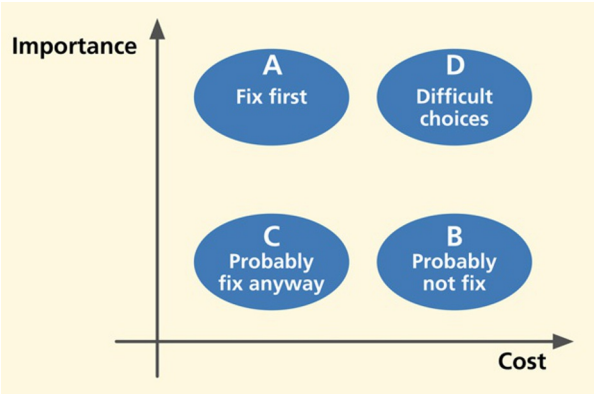

Fig. 26-3
The relationship of importance and cost in prioritizing which problems to fix first.

The extreme cost of a "Must fix" problem could make it infeasible to fix in the current version. Exceptions will surely result in cost overruns, but might have to be dictated by corporate policy, management, marketing, etc. It is an important time to be true to your principles and to everything you have done in the process so far. Don't throw it away now because of some perceived limit on how much you are willing to put into fixing problems that you have just spent good money to find. Quality is remembered long after budget and schedules are forgotten.

Sometimes you have resources to fix the "Must fix" problems, but no resources left for dealing with the other problems. Fortunately, in our example, we have enough resources to fix a few more problems. Depending on their relative proximity to the line of affordability, you have to decide among these choices as a resolution for all the other problems:

- Fix now.
- Fix, time permitting.
- Remand to the "wait-and-see list."
- Table until next version.
- Postpone indefinitely; probably never get to fix.

In the final column of the cost-importance table, write in your resolution for each problem, as we have done for the Ticket Kiosk System in Table 26-12.

Table 26-12

Problem resolutions for Ticket Kiosk System

Problem	Imp.	Solutions	Cost	Prio. Ratio	Prio. Rank	Cuml. Cost	Resolution
Didn't recognize the "counter" as being for the number of tickets. As a result, user failed to even think about how many tickets he needed	*M*	Move quantity information and label it	2	*M*	1	2	Fix in this version
User unaware of the need to click on the "Submit" button to proceed to payment	4	Change the label wording to "Proceed to Payment"	1	4000	2	3	Fix in this version
User confused about "Theatre" on the "Choose a domain" screen. Thought it meant choosing a physical theater (as a venue) rather than the category of theatre arts	3	Improve the wording to "Theatre Arts"	1	3000	3	4	Fix in this version
Unsure of current date and what date he was purchasing tickets for	5	Add current date field and label all dates precisely	2	2500	4	6	Fix in this version
Didn't like having a "Back" button on second screen because first screen was only a "Welcome"	2	Remove it	1	2000	5	7	Fix in this version
Users were concerned about their work being left for others to see	5	Add a timeout feature that clears the screens.	3	1667	6	10	Fix in this version
Transaction flow for purchasing tickets (group problem; see Table 26-8)	3	Establish a comprehensive and more flexible model of transaction flow and add labeling to explain it	5	600	7	15	Fix in this version
Line of affordability (16 person-hours—2 work days)							
Ability to find events hampered by lack of a search capability	4	Design and implement a search function	40	100	9	67	Defer to a future version

Finally, look at your table; see what is left below the line of affordability. Is it what you would expect? Can you live with not making fixes below that line? That is okay, as our engineering approach is aiming for cost-effectiveness, not perfection. You might even have to face the fact that some important problems cannot be fixed simply because they are too costly.

26.4.10 Special Cases

26.4.10.1 Tie-breakers

Sometimes you will get ties for priority rankings. If they don't occur near the line of affordability, it isn't necessary to do anything about them. In the rare case that they straddle the line of affordability, you can break the tie by almost any practical means, for example, your team members may have a personal preference.

In cases of more demanding target systems (e.g., an air traffic control system), where the importance of avoiding problems, especially dangerous user errors, is a bigger concern than cost, you might break priority ties by adjusting the priorities by weighting importance higher than cost in the priority ratio formula.

26.4.10.2 Cost-importance analysis involving multiple problem solutions

Sometimes, you can think of more than one solution for a problem. It is possible that, after a bit more thought, one solution will emerge as best. If, however, after careful consideration, you still have multiple possibilities for a problem solution, you can keep all solutions in the running and in the analysis until you see something that helps you decide.

If all solutions have the same cost to fix, then you and your team will just have to make an engineering decision. This might be the time to implement all of them and retest, using further prototyping to evaluate alternative design solutions for just this one feature.

Usually, though, solutions are distinguished by cost and/or effectiveness. Maybe one is less expensive but some other one is more desirable or more effective; in other words, you have a cost-benefit tradeoff to resolve before entering the chosen solution and its cost into the cost-importance table.

26.4.10.3 Problem groups straddling the line of affordability

If you have a group of related problems right at the line of affordability, the engineering answer is to do the best you can before you run out of resources. If necessary, break the group back apart and do as many pieces as possible. Give the rest of the group a higher importance in the next iteration.

26.4.10.4 Priorities for emotional impact problems

Priorities for fixing emotional impact problems can be difficult to assess. They are often very important because they can represent problems with product or system image and reputation in the market. They can also represent high costs to fix because they often require a broader view of redesign, not just focusing on one detail of the design as you might for a usability problem.

Also, emotional impact problems are often not just redesign problems but might require more understanding of the users and work or play context, which means going all the way back in the process to usage research and a new approach to the conceptual design. Because of business and marketing imperatives, you may have to move some emotional impact problems into the "Must fix" category and do what is necessary to produce an awesome user experience.

26.4.11 Rapid Cost-Importance Analysis

As a rapid version of the cost-importance analysis process:

- Put the problem list in a spreadsheet or similar document.
- Project it onto a screen in a room with pertinent team members to decide priorities for fixing the problems.
- Have a discussion about which problems to fix first based on a group feeling about the relative importance and cost to fix each problem, without assigning numeric values.
- Do a kind of group-driven "bubble sort" of problems in which problems to fix first will float toward the top of the list, and problems you probably cannot fix, at least in this iteration, will sink toward the bottom of the list.
- When you are satisfied with the relative ordering of problem priorities, start fixing problems from the top of the list downward, and stop when you run out of time or money.

26.5 FEEDBACK TO THE PROCESS

Now that you have been through an iteration of the UX process lifecycle, it is time to reflect not just on the design itself, but also on how well your process worked. If you have any suspicions after doing the testing that the quantitative criteria were not quite right, you might ask if your UX targets worked well.

For example, if all target levels were met or exceeded on the very first round of evaluation, it will almost certainly be the case that your UX targets were too lenient. Even in later iterations, if all UX targets are met, but observations during evaluation sessions indicate that participants were frustrated and performed tasks poorly, your intuition will probably tell you that the design is nevertheless not acceptable in terms of its quality of user experience. Then, obviously, the UX team should revisit and adjust the UX targets or add more considerations to your criteria for evaluation success.

Next, ask yourself whether the benchmark tasks supported the evaluation process in the most effective way. Should they have been simpler or more complex, narrower or broader? Should any benchmark task description be reworded for clarification or to give less information about how to do a task?

Finally, assess how well the overall process worked for the team. You will never be in a better position to sit down, discuss it, and document possible improvements for the next time.

26.6 LESSONS FROM THE FIELD

26.6.1 Onion-Layers Effect

There are many reasons to make more than one iteration of the design-evaluate-redesign part of the UX lifecycle. The main reason, of course, is to continue to uncover and fix UX problems until you meet your UX target values. Another reason is to be sure that your "fixes" have not caused new problems. The fixes are, after all, new and untested designs.

Also, in fixing a problem, you can uncover new UX problems obscured by the original problem, preventing participants and evaluators from seeing the new problem, until the top layer of the onion[1] is peeled off by solving that "outer" problem.

26.6.2 UX Problem Data as Feedback to Process Improvement

In our analysis, we are also always on the lookout for *process causes of the problem causes*. It sometimes pays off to look at your UX process to find causes of the design flaws that cause UX problems, places in your process where, if you could have done something differently, you might have avoided a particular kind of design flaw. If you suffer from an overabundance of a particular kind of UX problem and can determine how your process is letting them into the designs, maybe you can head off that kind of problem in future designs by fixing that part of the process.

For example, if you are finding a large number of UX problems involving confusing button or icon labels or menu choices, maybe you can address these in advance by providing a place in your design process where you look extra carefully at the precise use of words, semantics, and meanings of words. You might even consider hiring a professional writer to join the UX team. We ran into a case like this once. For expediency, one project team had been letting their

[1]Thanks to Wolmet Barendregt for the onion-layer analogy.

software programmers write error messages as they encountered the need for them in the code. This situation was a legacy from the days when programmers routinely did most of the user interface. As you can imagine, these error messages were not the most effective. We helped them incorporate a more structured approach to error message composition, involving UX practitioners, without unduly disrupting the rest of their process.

Similarly, large numbers of problems involving physical user actions are indicators of design problems that could be addressed by hiring an expert in ergonomics, human factors engineering, and physical device design. Finally, large numbers of problems involving visual aspects of design, such as color, shape, positioning, or gray shading, might indicate the need for hiring a graphic designer or layout artist.

Exercise 26-1: UX Data Analysis for Your System

Goal: To get some practice with the analysis part of a very simple formative UX evaluation.

Activities: If you are working with a team, get together with your team, including any new participants you picked up along the way.

Fill in the UX target table "Observed results" column.

Together, your team compiles and compares the quantitative results to determine whether UX targets were met.

Review your raw critical incident notes and write a UX problem list.

Organize the UX problem list and perform cost-importance analysis. Using a paper cost-importance table or laptop spreadsheet, list a dozen or more UX problems from critical incidents.

Assign an importance (to fix) rating to each observed problem.

Propose solutions (without doing all the work of redesign).

Group together any related problems and list as single problem.

Assign cost values (in person-hours) to each solution.

Compute priority ratios.

Compile your results:

Move your "Must fix" problems to the top of your cost-importance table.

Sort the remaining problems by decreasing priority ratios to determine the priority rank of UX problems.

Fill in the cumulative cost column.

Assume a hypothetical value for available time resources (something to make this exercise work).

Draw the cutoff, line of affordability.

Finalize your "management" decisions (resolution) about which changes to make now and in the next version.

Deliverables: Summary of quantitative results, written in "Observed results" column in your UX target table form (for comparison with UX targets).

List of raw critical incidents.

Cost-importance table form containing three UX problems selected as interesting to present to class or your work group (complete across all three rows).

Choose someone to give brief a report on your evaluation results.

Schedule: Given the simplicity of the domain, we expect this exercise to take about 30 to 60 minutes.