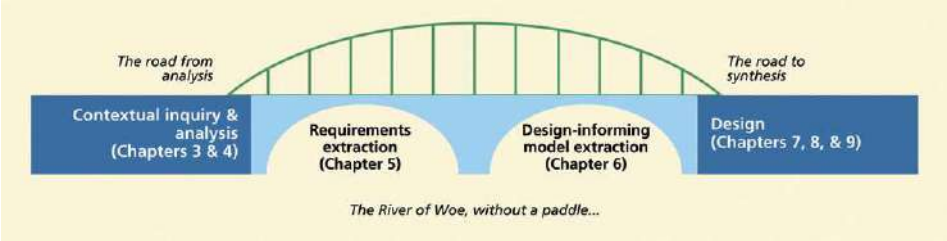

Figure 5-2
Overview of the bridge to design.

- The gap is the demarcation between the old and the new—between studying existing work practice and existing systems and envisioning a new work space and new system design space.

This chapter is about how we begin to bridge this gap with requirements as shown in Figure 5-2.

5.2 NEEDS AND REQUIREMENTS: FIRST SPAN OF THE BRIDGE

5.2.1 What Are "Requirements"?

Almost everyone understands the basic meaning. The term refers to a statement of what is needed to design a system that will fulfill user and customer goals. But when you start getting specific, it is a term that can mean something different to just about everyone associated with developing interactive software systems. To one, it is about ascertaining all the functionality needed to do the job. To another it is a compilation of all the user tasks needed to do the job.

In the UX domain, interaction design requirements describe what is required to support user or customer work activity needs. To that end we are also concerned with functional requirements to ensure the usefulness component of the user experience. Finally, we will have requirements to fulfill the need for emotional impact and long-term phenomenological aspects of the user experience.

5.2.2 Requirements "Specifications"

Before we get into extracting requirements from contextual data, let us look briefly at the forms interaction design requirements can take. One term we often think of when coupled with "requirements" is "specifications."

In past software engineering traditions, a formal written requirements document was de rigueur and could even designate details about how the corresponding software is to be implemented, including such software stuff as

Usability

Usability is the pragmatic component of user experience, including effectiveness, efficiency, productivity, ease-of-use, learnability, retainability, and the pragmatic aspects of user satisfaction.

Phenomenological Aspects of Interaction

Phenomenological aspects (deriving from phenomenology, the philosophical examination of the foundations of experience and action) of interaction are the cumulative effects of emotional impact considered over the long term, where usage of technology takes on a presence in our lifestyles and is used to make meaning in our lives.

object models, pseudo-code, use cases, and software structure. However, currently in software engineering and software requirements engineering there is an increasing recognition that:

- Detailed formal requirements cannot ever be complete.
- Detailed formal requirements cannot ever be 100% correct.
- Detailed formal requirements cannot be prevented from changing throughout the lifecycle.

Domain-Complex Systems

Domain-complex systems are systems with high degree of intricacy and technical content in the corresponding field of work. Often, characterized by convoluted and elaborate mechanisms for how parts of the system work and communicate, they usually have complicated workflow containing multiple dependencies and communication channels. Examples include an air traffic control system and a system for analyzing seismic data for oil exploration.

As a result, there appears to be a trend toward abandoning the detailed requirements specifications in favor of ascertaining the important features and capabilities.

Often people from a software engineering background expect a similar kind of requirements specification for the user interface. However, on the UX side we are talking about only *interaction design requirements*, nothing about software or implementation. Also, as we will see, it is not easy to lay down that same kind of requirements specification for the interaction design, nor is it particularly useful to try.

However we specify our requirements, there is a broad range of acceptability for completeness and detail. For domain-complex systems, with many requirements for compliance and risk avoidance, you may need a rather complete specification of requirements.

Our approach to interaction design requirements follows directly from the contextual data that we have gathered and analyzed. The result is not just a monolithic specification, but a variety of descriptions that, while not necessarily like software specifications, are each part of the whole that constitutes the interaction design requirements specification.

Therefore, at the end of the day, or more likely the end of the week, requirements extraction produces an assortment of deliverables, each of which can be thought of as a kind of "specification"—for needs and requirements and for design-informing models such as personas, tasks, user experience goals, or usage scenarios. That is why all those activities and deliverables are brought together in this chapter and the next.

5.2.3 Software and Functional Implications of Interaction Design Requirements

User needs are really not just interaction needs. Usability and UX include usefulness that we get from functionality. Often an initial requirement extracted from contextual data first appears as a requirement for a broad

overall system capability—that is, it expresses a need for both functionality and user interface support.

As an example, a Ticket Kiosk System requirement might state that a user should be able to buy tickets for up to 10 different events in one session or transaction. We recommend that you devise a way to record the functional needs that correspond to user needs and requirements revealed in this process and pass them on to your software engineering counterparts. It will help them be aware of needed functionality and will help you both stay on the same page during the project.

5.3 FORMAL REQUIREMENTS EXTRACTION

This process of extracting needs and requirements is similar to data interpretation and consolidation sessions of contextual analysis in that it involves a group sitting down together and going over a large amount of data, including the WAAD and evolving design-informing models. But here it is actually easier because much of the hard work is already done.

5.3.1 Walking the WAAD for Needs and Requirements

At the end of Chapter 4 we recommended doing a "wall walk," a walkthrough of contextual data in the WAAD. It is now time for your team to get re-immersed in work activity data; this time with the focus of the walkthrough on extracting needs and requirements rather than iteratively improving the data. The general idea is to traverse the hierarchical WAAD structure and focus on extracting requirement statements from work activity notes.

5.3.2 Switching from Inductive to Deductive Reasoning

Extracting requirements from the WAAD calls for a deductive thinking process. It is deductive because each work activity note in the WAAD is treated as the major premise in a logical syllogism. The second "premise" is everything you know about UX and interaction design. The conclusion of this syllogism is a statement of user needs and requirements you deduce from the work activity note, something we capture in a "requirement statement."

To clarify with a small example from MUTTS and the Ticket Kiosk System, a WAAD note, say in node C19, that says "I am concerned about security and privacy of my transactions" can imply a design requirement (at a high level): "Shall protect security and privacy of ticket-buyer transactions." In the design,

this requirement might be at least partially met by a timeout feature to clear the screen between customers. Note that at this level, requirements can be a mix of interaction and functional requirements.

5.3.3 Preparation

Select a requirements team, including people you think will be best at deductive reasoning and creativity. You will need both UX and software people represented, plus possibly system architects and maybe managers.

This team approach enhances SE-UX communication because the SE and UX roles are working together at a crucial point in their mutual lifecycles, describing and funneling the different kinds of requirements to the places they will be used. Choose a requirements team leader and a recorder, a person experienced in writing requirements.

You may need a requirements "record" template in a word processing document, a spreadsheet, or a database schema to capture the requirement statements in a consistent and structured format in an interaction design requirements document (or requirements document, for short in this context). The requirements team will work in the room where the WAAD is posted on the wall.

If there is a need for all to see each requirement statement, you can connect the recorder's computer to a screen projector and show the requirements document on an open part of the wall. The leader is responsible for walking the team through the WAAD, traversing its hierarchical structure systematically and keeping the team on track.

5.3.4 Systematic Deduction of Needs as "Hinges" to Get at Requirements

Start by letting everyone walk through the WAAD, individually and silently, to accommodate those who need to think quietly and to allow everyone to write notes about ideas for requirements. Then begin the main part of the process. As the leader walks the team through the WAAD, one node and one note at a time, the team works together to ask what user needs, if any, are reflected in this work activity note and the hierarchical labels above it.

Such user needs are still expressed in the perspective of the user and in the work domain. Although the user need is not documented in the requirements document, it is an important "hinge" in the mental process of getting from work activity notes to requirements. This interim step will become almost automatic with only a little practice.

5.3.5 Terminology Consistency

This pass through the contextual data is a chance to standardize terminology and build consistency. Your contextual data will be full of user comments about an infinitude of usage and design concepts and issues. It is natural that they will not all use exactly the same terms for the same concepts.

For example, users of a calendar system might use the terms "alarm," "reminder," "alert," and "notification" for essentially the same idea. Sometimes differences in terminology may reflect subtle differences in usage, too. So it is your responsibility to sort out these differences and act to help standardize the terminology for consistency issues in the requirements document.

5.3.6 Requirement Statements

Next, the team translates each user need into one or more interaction design requirement statements. Each requirement statement describes a way that you decide to support the user need by providing for it in the interaction design. Ask what new or more specific user interface feature you should see in the design to support the user needs implied by this WAAD note. There is not necessarily a one-to-one correspondence between work activity notes in the WAAD and needs or requirements.

A given work activity note might not generate a need or requirement. The ideas in some notes may no longer be relevant in the envisioned system design. Sometimes one work activity note can produce more than one need. A single need can also lead to more than one requirement. Examples of work activity notes, user needs, and corresponding requirements are coming soon.

Now the recorder writes the requirement statement in the requirements document by first finding the appropriate headings and subheadings. If the necessary headings are not already in the requirements document, now is the time to add them and grow the document structure as the process continues.

Interaction requirements often imply functional requirements for the system, which you may also capture here for communicating to your software people. For example:

> Interaction requirement: "Ticket buyers shall be able to see a real-time preview of available seating for a venue."
>
> Corresponding system requirement: "System shall have networked infrastructure to poll all kiosk transactions as they are happening and coordinate with the venue seating data to 'lock and release' selected seats."

This is a good time for the software team members to work in parallel and capture those inputs to software requirements here so that they are not lost.

These inputs will be transformed into software requirements specifications in the software requirements process, a separate process done only by the software team and not part of our scope here. Although software requirements gathering is not officially part of the interaction requirements extraction process, it is a shame to not take advantage of this opportunity to provide valuable software requirements inputs, based on real contextual data. This is also a good opportunity for you, the interaction designer, to coordinate with your software engineering teammates about your mutual requirements.

In a requirement statement it is typical to use the phrase "Users shall be able to . . ." and can be followed by a rationale statement explaining the relationship of the requirement to the user need and how the requirement was determined from that need. A "notes" statement can also be part of a requirement statement. Such notes are not always necessary but they document discussion points that may have come up within the extraction process and need to be preserved for designers to consider in their process.

5.3.7 Requirement Statement Structure

A generic structure of a requirement statement that has worked for us is shown in Figure 5-3. A requirements document is essentially a set of requirement statements organized on headings at two or more levels.

For systems where risk is high and traceability is important, each requirement is tagged with the WAAD source node ID, which serves as a link back to the source of this requirement statement within the WAAD. The WAAD in turn has a link back to its source in raw work activity data.

Later, if a question arises about a particular need or requirement, the connection to original work activity data and the person who was its source can be traced to find the answers (sort of the UX lifecycle analog of a software requirements traceability matrix).

Because we use the WAAD node ID as a link this way, someone should ensure that all WAAD nodes are labeled with some identification number before the extraction process begins. We use A, B, C, . . . for the highest-level nodes under the root node. Under node A, we use AA, AB, AC, . . . , and for the work activity notes

Figure 5-3
Generic structure of a requirement statement.

Name of major feature or category
Name of second-level feature or category
Requirement statement [WAAD source node ID]
Rationale (if useful): Rationale statement
Note (optional): Commentary about this requirement

themselves we use the group ID plus a number, such as AB1 and AB2; this is the identifier that goes in the "WAAD source node ID" part of a requirement.

Security
Privacy of ticket–buyer transactions
Shall protect security and privacy of ticket-buyer transactions [C19]
Note: In design, consider timeout feature to clear screen between customers.

Figure 5-4
Example requirement statement.

As an example, consider the work activity note that said "I am concerned about privacy and security of my transactions." In Figure 5-4 we show how the resulting requirement statement fits into the requirement statement structure of Figure 5-3.

5.3.8 Requirements Document Structure

We show two levels of headings, but you should use as many levels as necessary for your requirements.

As an example of an extracted requirement for the Ticket Kiosk System, suppose in our contextual inquiry a user mentioned the occasional convenience of shopping recommendations from Amazon.com. The resulting requirement might look like what is shown in Figure 5-5.

Example: Extracting a Requirement Statement for the Ticket Kiosk System

Note CA9 within the WAAD for MUTTS says "I sometimes want to find events that have to do with my own personal interests. For example, I really like ice skating and want to see what kinds of entertainment events in the nearby areas feature skating of any kind." This user work activity statement implies the user need, "Ticket buyers need to find various kinds of events."

Labels on a group at a higher level imply a feature or topic of "Finding events" so we use that as the heading for this requirement in the requirements document. Lower-level labels in the WAAD narrow it down to "Direct keyword search by event description"; we will use that for our subheading.

We can then write the requirement in Figure 5-6.

Figure 5-5
Sample requirement statement for the Ticket Kiosk System.

Note that this comment, also in the WAAD, "I sometimes want to find events that have to do with my own personal interests," could lead to consideration of a requirement to maintain personal profiles of users.

Transaction flow
Recommendations for buying
Ticket-buyer purchases shall be supported by recommendations for the purchase of related items. [DE2].
Implied system requirement: During a transaction session the Ticket Kiosk System shall keep track of the kinds of choices made by the ticket buyer along with the choices of other ticket buyers who bought this item. [DE2].
Note: Amazon.com is a model for this feature.

Finding events
Direct keyword search by event description
Ticket buyers shall be able to find (e.g., search) by content to identify relevant current and future events [CA9].
Browse events by parameters
Ticket buyers shall be able to browse by category, description, location, time, rating, and price.

Figure 5-6
Example requirement statement for the Ticket Kiosk System.

5.3.9 Continue the Process for the Whole WAAD

In the Ticket Kiosk System example, you will also extract requirements for all the different ways you search and browse event information, such as requirements to search by event category, venue, date range, and so on. Take the time here to pick *all* the fruits; it is too easy to neglect the connections of rationale to user work activities and lose much of the advantage gained from the contextual analysis work.

After each requirement statement is written, it is very important for the whole team to see—for example, by projection display—or hear—for example, by the recorder reading it—the statement to ensure that the written statement represents their concept of what it was supposed to be.

Later when reviewing and finalizing the requirements document, we may find that not every "requirement" extracted from the WAAD will eventually be met because of cost, other constraints, and how our own knowledge and experience temper the process, but that kind of judgment comes later. For now, just keep cranking out the requirements.

5.3.10 Keep an Eye out for Emotional Impact Requirements and Other Ways to Enhance the Overall User Experience

When extracting requirements, most people will think of the functional requirements first, feeding usefulness. Most people might think of usability goals next, feeding UX targets. But do not forget that we are on a quest to design for a fabulous user experience and this is where you will find opportunities for that, too.

In addition to getting at routine requirements for tasks, functions, and features, seek out those indefinable evolving characteristics essential to a quality usage experience. Because factors related to emotional impact or phenomenological aspects may not be as clear-cut or explicit as functional or other interaction requirements, you have to be alert for the indicators.

Work activity notes with user concerns, frustration, excitement, and likings offer opportunities to design a system to address emotional issues. Especially look out for work activity notes that make even an oblique reference to "fun" or "enjoyment" or to things like data entry being too boring or the use of colors being unattractive. Any of these could be a clue to ways to provide a more rewarding user experience. Also, be open minded and creative in this

phase; even if a note implies a need that is technologically difficult to address, record it. You can revisit these later to assess feasibility and constraints.

5.3.11 Extrapolation Requirements: Generalization of Contextual Data

User statements in a WAAD can be quite narrow and specific. You may need to generate extrapolation requirements to broaden existing contextual data to cover more general cases.

For example, ticket buyers using MUTTS, in anticipation of a kiosk, might have expressed the need to search for events based on a predetermined criterion but said nothing about browsing events to see what is available. So you might write an extrapolation requirement about the obvious need also to browse events (as we did in Figure 5-6).

As another example, in our WAAD for MUTTS, a ticket buyer speaks about the desirability of being able to post an MU football ticket for exchange with a ticket in another location in the stadium to be able to sit with their friends. In our extrapolation requirement statement we broadened this to "Ticket buyer shall be able to post, check status of, and exchange student tickets." And we added a relationship note: "Will require ticket-buyer user 'accounts' of some kind where they can login using their MU Passport IDs."

In another work activity note a user mentioned it would be nice to be able to select seats from all available seats in a given price category. This translates to a requirement to display seating availability and to be able to filter that list of available seats (such as by price categories). Seat selection *assumes* the existence of a lock and release mechanism of some sort, something we perhaps did not yet have in the requirements document. This is a technical requirement to give the buyer a temporary option on the selected seats until the transaction is completed or abandoned. So we added an extrapolation requirement to cover it:

> Shall have a default time interval for locking available seating while the ticket buyer is making a choice.
>
> Rationale: If a ticker buyer has not performed any actions with the interface in a certain amount of time, we assume the ticket buyer has left the kiosk or at least abandoned the current transaction.

The timeout will release those seats back to an available pool for others to access from other kiosks.

Another work activity note said, "I often plan to attend entertainment events with friends." At first, we thought this comment was just a passing remark about how he would use it. It did not seem to imply a requirement because it did not say anything directly about a feature.

On reflection, however, we could easily broaden it slightly to imply a possible need to communicate with those friends and, with a bit more extrapolation, maybe facilitate sending tickets or event information to them via email. This extrapolation could well be beyond the scope of the user's intent and it could be beyond the scope of the current project, but it should be saved as an input about a potential future feature and, more importantly, as a chance to provide a great user experience.

This example is a good one because it starts with a statement about usage. And that is what contextual data are about, so we should not have missed seeing an implied requirement because "it did not say anything about a feature." It is our job to come up with requirements implied by usage statements.

In balance, while extrapolation requirement statements may be necessary and valuable, we should be careful with them. To be sure, we distinguish them by calling them (and tagging them as) extrapolation requirements, which must be taken back to users for confirmation as real needs or requirements. This validation can result in a thumbs up and you can include it in your requirements document or it can result in a thumbs down and you can eliminate that requirement.

5.3.12 Other Possible Outputs from the Requirements Extraction Process

In addition to requirement statements, a work activity note in a WAAD can lead to certain other outputs, discussed in the following subsections.

Questions about missing data

Sometimes, as you go deeper into the implications of contextual data, you realize there are still some open questions. For example, in our contextual inquiry for MUTTS, while we were putting together requirements for the accounting system to aggregate sales at the end of the day, we had to face the fact that the existing business manages tickets from two independent systems. One is the local ticket office sales and the other is from the national affiliate, Tickets4ever .com. During our contextual inquiry and analysis we neglected to probe the interaction between those two and how they reconciled sales across those two systems.

System support needs

You may also occasionally encounter system requirements for issues outside the user experience or software domains, such as expandability, reliability, security, and communications bandwidth. These are dealt with in a manner similar to that used for the software requirements inputs. A few examples from the MUTTS WAAD illustrate:

> Work activity note: "Identity theft and credit card fraud are huge concerns for me."
>
> System requirement: "System shall have specific features to address protecting ticket buyers from identity theft and credit card fraud." (This "requirement" is vague but it is really only a note for us to contact the systems people to figure out potential solutions to this problem.)
>
> Work activity note: "When I am getting tickets for, say, a controversial political speaker, I do not want people in line behind me to know what I am doing."
>
> System requirement: "Physical design of kiosk shall address protecting privacy of a ticket buyer from others nearby."

Marketing inputs

Sometimes a comment made by a user during contextual inquiry might make a good input to the marketing department as a candidate sound bite that can be adapted into advertising copy. This is a good opportunity to communicate with the marketing people and help cement your working relationship with them.

Example: Requirements Extraction for the Ticket Kiosk System

Here are a few selected requirements extracted from the MUTTS WAAD that we are using to inform requirements for the Ticket Kiosk System.

Shopping cart

Existence of feature

> Ticket buyer shall have a shopping cart concept with which they can buy multiple items and pay only once [BBA1-4]

Accessibility of shopping cart

> Ticket buyer shall be able to view and modify shopping cart at all times [BBA3]

Shopping cart versatility

Ticket buyer shall be able to add different kinds/types of items (example, different events, sets of tickets for the same event) [BBA4]

Note: This requirement is important because it has implications on how to display shopping cart contents with dissimilar types of objects in it.

Transaction flow

Timeouts

Extrapolation: Ticket buyer shall be supported by a timeout feature [BCA]
Rationale: To protect ticket buyer privacy

Extrapolation: Ticket buyer shall be made aware of the existence and status of timeout, including progress indicator showing remaining time and audible beep near the end of the timeout period [BCA] [BCA1]

Extrapolation: Ticket buyer shall have control to reset the timeout and keep the transaction alive

Extrapolation: Ticket buyer's need to keep transaction alive shall be supported by automation, timer reset triggered by ticket buyer activity

Immediate exit

Ticket buyer shall be able to make a quick exit and reset to the home screen [BCB1]

Rationale: Important for kiosks in bus stations where user may have to quit in the middle of a transaction and to protect their privacy

Ticket buyer shall have a way to quickly return to a specific item they were viewing just prior to an immediate exit [BCB1]

Note: Ticket buyer shall be able to use an event ID number for direct access next time or the system can potentially do it using an "account" and restore state.

Recommendations for buying

Extrapolation: Ticket buyer purchases shall be supported by recommendations for related items [BCB2]

Extrapolation: Ticket buyer shall be able to say no to recommendations easily [BCB2]

Transaction progress awareness

Ticket buyer shall be able to track the progress of the entire transaction (what is done and what is left to do) using, for example, a "bread crumb" trail [BCB3-4]

Ticket buyer reminders

Ticket buyer shall receive reminders to take the ticket and MU Passport/credit card at the end of each transaction [BCC1-2]

Checkout

Ticket buyer shall have, before making a payment, a confirmation page showing exactly what is being purchased [BCD1]

Ticket buyer shall receive actual ticket and not just confirmation [BCD2]
Rationale: For maintaining ticket buyer trust
Note: This is a huge issue involving marketing, high-level business decisions, and hardware (printer) reliability and kiosk maintenance

Ticket buyer shall be able to use cash, credit cart, debit card, or MU Passport for payment [BCD3]

Note: For cash transaction it is difficult to recognize and dispense change [BCD4], and attracts vandals and thieves [BCD5]

System requirements

Performance

The system shall have a good response time to make transactions fast (so ticket buyers do not miss the bus) [BCB5]

Exercise

See Exercise 5-1, Extracting Requirement Statements for Your System

5.3.13 Constraints as Requirements

Constraints, such as from legacy systems, implementation platforms, and system architecture, are a kind of requirements in real-world development projects. Although, as we have said, much of the interaction design can and should be done independently from concerns about software design and implementation, your interaction design must eventually be considered as an input to software requirements and design.

Legacy System

A legacy system is a system with maintenance problems that date back possibly many years.

Therefore, eventually, you and your interaction design must be reconciled with constraints coming from systems engineering, hardware engineering, software engineering, management, and marketing. Not the least of which includes development cost and schedule, and profitability in selling the product.

What restrictions will these constraints impose on product scope? Are product, for example, a kiosk, size and/or weight to be taken into account if, for example, the product will be on portable or mobile equipment? Does your system have to be integrated with existing or other developing systems? Are there compliance issues that mandate certain

features? Constraints arise from the problems of legacy systems, limitations of implementation platforms, demands of hardware and software, budgets, and schedules.

Example: Constraints for MUTTS

A hardware constraint for the existing working environment of MUTTS is the necessity of keeping the secure credit card server continuously operational. An inability of the ticket office to process credit card transactions would essentially bring their business to a halt. They have only one "general purpose" technician on staff to care for this server plus all the other computers, network connections, printers, scanners, and so on.

In addition, the physical space of the MUTTS office is constrained, a constraint that should also show up in the physical model (Chapter 6), and work areas can become cramped on busy days. Their office space is leased, a fact that is not likely to change in the near future, so a more efficient work flow is desirable. Sometimes the air conditioning is inadequate.

The constraints will show significant differences in going from MUTTS to the Ticket Kiosk System. Here are some example constraints that might be anticipated in the Ticket Kiosk System, mostly about hardware (systems engineering people would probably add quantitative standards to be met in some cases):

- Special-purpose hardware for the kiosk
- Rugged, "hardened" vandal-proof outer shell
- All hardware to be durable, reliable
- Touchscreen interaction, no keyboard
- Network communications possibly specialized for efficiency and reliability
- If have a printer for tickets (likely), maintenance must be an extremely high priority; cannot have any customers pay and not get tickets (e.g., from paper or ink running out)
- Need a "hotline" communication feature as backup, a way for customers to contact company representatives in case this does happen

Exercise

See Exercise 5-2, Constraints for Your System

5.3.14 Prioritizing Requirements

A drawback of affinity diagrams is that they do not contain priority information, so every note has the same weight as any other note. A note about a major task has the same significance as a passing comment. As a result, the extracted requirements are also unprioritized. To remedy this, as part of

the validation process, ask your customer and users to prioritize the requirements.

At a minimum they can point out the key requirements and the requirements that are "also-rans." These can be separated into different sections of a requirements document or distinguished by a color-coding scheme.

With a bit more effort you can tag each requirement with an importance rating. Later, you will use these priority ratings to decide which design-informing models to focus on. For example, important tasks will be the ones chosen as the basis for representative scenarios.

Often, as the result of prioritizing, you and your customer achieve a realization of, and mutual understanding about, the fact that some requirements cannot be met realistically in the current product version and must be set aside for consideration in the future.

5.3.15 Taking Requirements Back to Customers and Users for Validation

After your own review, it is time to take the requirements document or requirements WAAD back to the customer and users for validation. This is a critical step for them because it gives them a chance to offer inputs and correct misconceptions before you get into design. It also helps solidify your relationship as partners in the process.

For each work role, schedule a meeting with the representative users, preferably some from the ones you have interviewed or otherwise interacted with before, and some new users. Walk them through the requirements to make sure your interpretation of requirements from the work activity notes is accurate.

Work Role

A work role is defined and distinguished by a corresponding job title or work assignment representing a set of work responsibilities. A work role usually involves system usage, but some work roles can be external to the organization being studied.

Pay close attention to feedback from new users who are looking at the requirements for the first time. They may provide valuable feedback on anything you missed or new insights into the needs. Remember that these users are experts in the work domain, but probably not in the domains of interaction design or software development, so protect them from technical jargon.

5.3.16 Resolve Organizational, Sociological, and Personal Issues with the Customer

When you take your requirements to the customer for validation, it is also a good opportunity to resolve organizational, social, and personal issues. Because your requirements reflect what you intend to put into the design, if heeded, they can flash early warning signs to customers and users about issues of which your team

may be unaware, even after thorough contextual inquiry. Especially if your requirements are pointing toward a design that changes the work environment, the way work is done, or the job descriptions of workers, your requirements may give rise to issues of territoriality, fear, and control.

Changes in the workflow may challenge established responsibilities and authorities. There may also be legal requirements or platform constraints for doing things in a certain way, a way you cannot change, regardless of your arguments for efficiency or better user experience. Organizational, social, and personal issues can catch your team by surprise because they may well be thinking mostly about technical aspects and design at this point.

Work Activity Affinity Diagram

A work activity affinity diagram (WAAD) is an affinity diagram used to sort and organize work activity notes in contextual analysis, pulling together work activity notes with similarities and common themes to highlight common work patterns and shared strategies across all users.

5.4 ABRIDGED METHODS FOR REQUIREMENTS EXTRACTION

5.4.1 Use the WAAD Directly as a Requirements Representation

To save time and cost, the WAAD itself can be taken as a set of implicit requirements, without formally extracting them. On the WAAD you created in contextual analysis, highlight (e.g., using a marker pen) all groups or individual work activity notes that imply requirements and design ideas directly or indirectly. The way a WAAD note can represent a requirement is: you must cover, include, or accommodate (in the interaction design) the issue, idea, or concept expressed in the note.

To use the Ticket Kiosk System example of customer security and privacy again, the work activity note says, "I am concerned about the security and privacy of my transactions." Instead of rewriting this as a formal requirement statement in a requirements document as we did previously, you just interpret it directly as you read it to "shall protect security and privacy of ticket-buyer transactions."

This requirement may immediately generate ideas about how to solve the problem in the design, such as by automatic timeout and and/or a limited viewing angle on the physical kiosk. You should also document these design ideas immediately, while you can, as notes directly on the WAAD.

You will acquire the ability to look at the WAAD with an interpretative eye and see the work activity notes as more explicit requirements. Clear and crisply written work activity notes will help make this mental step of interpretation easier.

5.4.2 Anticipating Needs and Requirements in Contextual Analysis

In anticipation of the need to extract requirements here, we can introduce a shortcut in contextual analysis, adjusting the process for work activity note synthesis and saving some cost. The shortcut involves doing some interpretation of the raw data, on the fly, to move it more rapidly to reflect requirements.

For example, consider a work activity note from the MUTTS interviews that says: "After the lottery results for an MU football game are out, students who won try to exchange tickets with others so they and their friends can sit together." From this, you can move more rapidly toward needs and requirements by restating it as: "Some MU football ticket lottery winners need an ability to go to a kiosk and trade tickets with other winners so they can sit with their friends."

5.4.3 Use Work Activity Notes as Requirements (Eliminate the WAAD Completely)

Another efficient abridgement technique, for experienced practitioners, is eliminating the WAAD altogether and using the bins of sorted work activity notes as requirements. Building a WAAD is about organizing large amounts of data to identify underlying themes and relationships.

If your contextual inquiry did not result in a huge number of work activity notes (a likely case in an abridged approach), you can identify relationships by just manipulating the work activity notes themselves. But you still have to make the mental step of interpretation to deduce requirements on the fly.

Intentionally left as blank