Working Through Task-Centered System Design

Saul Greenberg
Department of Computer Science, University of Calgary, Canada
saul@cpsc.ucalgary.ca

This chapter presents a 'how-to' tutorial to a version of Lewis and Rieman's Task Centered System Design methodology. Using an example of an interface being developed for a catalogue store, we show in detail how a practitioner can identify key tasks, use those tasks to do a rudimentary requirements analysis, and how one can evaluate prototype designs through a task-centered walkthrough.

Keywords. Task centered system design, walkthroughs, scenarios, requirements analysis.

Introduction

In 1993, Clayton Lewis and John Rieman introduced Task Centered System Design (TCSD), a highly practical discount-usability engineering methodology (Lewis and Reiman, 1993). At its essence, TCSD is a process where designers:

- articulate concrete descriptions of real-world people doing their real-world tasks;
- use these descriptions to determine which users and what tasks the system should support;
- prototype an interface that satisfy these requirements; and
- evaluate the interface by performing a task-centered walkthrough.

Because TCSD is simple to learn and apply, I have been teaching it for almost a decade within an introductory Human Computer Interaction (HCI) for computer scientists. I could only devote about four classes to this method, so I reworked the Lewis and Rieman material into a short form that provides students with a terse explanation of the process, as well as a worked example that illustrates how to apply it to a problem. Students then use TCSD in their first assignment to analyze a real world problem of their choosing, and to develop and evaluate an interface that solves this problem. Most students find TCSD to be an eye-opener in terms of its simplicity and effectiveness—they are surprised at how well it informs their interface design and how well it lets them evaluate the nuances of their designs.

This chapter summarizes my reworked approach to Task-Centered System Design. It both paraphrases and adds to the Lewis and Rieman material. The first part of this chapter details the main steps of the TCSD process. The second part applies this process to an actual example.

Greenberg, S. (2004) Working through Task-Centered System Design. in Diaper, D. and Stanton, N. (Eds) *The Handbook of Task Analysis for Human-Computer Interaction.* Lawrence Erlbaum Associates. p49-66.
This version differs in minor ways from the published version.

Phase 1. Identification

In the first phase of task-centered system design, you identify specific users of the system and articulate example realistic tasks that they would do. Your goal is to produce a manageable list of representative users and tasks that give realistic coverage of who would use the system to do what kinds of tasks. To achieve this goal, you need to first discover what tasks users do, then write these up as task descriptions, and finally validate the descriptions to make sure they represent reality. These steps are detailed below.

A. Discovering the Tasks that Users Do

TCSD strives for realism. This means you should discover how real people do their real tasks. Yet depending upon your situation, you may or may not be able to access these real people. Consequently, you should select one of the approaches below that best fits your situation.

The ideal: Observing and/or interviewing the real end user. Get in touch with current or potential users. These users may now be using paper methods, competing systems, or antiquated systems for doing their tasks. Observe them as they do their task activities, and interview them about what they are doing (Beyer and Holtzblatt, 1998). For example, if you are interested in customers who purchase items in a store, you should observe and talk to store customers as they move about the store. These interviews and observations are critical. They transform 'the user' from an abstract notion into real people with real needs and concerns. It will help you put a face on the faceless, and will help you understand where they are coming from.

Second best: Interviewing the end-user representative. If you absolutely cannot get in direct contact with end users, you can carefully select and interview end-user representatives as stand-ins. These *must* be people who have direct contact with end users, and have intimate knowledge and experience of their needs and what they do. It is crucial that this client representative has a deep and real (rather than idealized) understanding of what end-users actually do. People who work "in the trenches" with the end-user are the best bet. For example, you can talk to a store's front-line sales staff about their customers if you cannot observe or talk to the customers directly. A better option is to interview these front-line staff as they deal with the customers; this way you can observe first-hand what customers do.

When all else fails: Make your beliefs of the end-users and task space explicit. If you cannot get in touch with real end-users or their representatives, use your team members to articulate expected end-users and tasks. Because this runs the serious risk of producing end-user and task descriptions that bear no relation to reality, you should do this only as a last resort. Still, you will at least produce a diverse list of expected end-users and their tasks (because you have a diverse team), and it will put your beliefs and assumptions on

the table. You can always show these to your clients later, and hopefully you will be able to compare them to real-world situations to see if these tasks indeed reflect what real end-users do.

No matter which approach you choose, you have to determine your stopping conditions i.e., when you should stop gathering and generating user and task descriptons. In practice, you will notice ever-increasing repetition as you do your observations and interviews. You simply stop when new types of people and tasks are rare, and when it is no longer cost-effective to continue.

B. Developing good task descriptions

You must write up the results of your observations and interviews as good task descriptions. These task descriptions adhere to five very important criteria.

It describes what the user wants to do but does not say how the user would do it. The description should not include any interface mechanics about how the task is actually carried out. That is, you do not want to detail task steps that are peculiar to the system being used. What you are really doing here is identifying the person's goal, as well as the concrete steps they would take to achieve this goal no matter what system was being used (Cooper, 1999). This is important, because you will want to use these tasks to generate several alternative designs that let a user accomplish the task in quite different ways. Similarly, you will use these tasks to compare different interface design alternatives in a fair way.

It is very specific. A description is concrete. It says exactly what the user wants to do, including the actual items the user would eventually want to input (somehow) into the system and what information or results the user will want out of it. This is important because it provides concrete rather than imaginary data for the types of information the system must handle.

It describes a complete job. The description should flow through all aspects of the task, starting at the very beginning and concluding at the very end. This is important because the complete description forces you to consider how interface features will work together. You can also use the complete description to contrast how information input and output is carried through a particular interface design. That is, you can ask: "Where does information come from? Where does it go? What has to happen next?"

It says who the users are and reflects their real interests. The description should name real people, and should include what they know or don't know about performing the task and using computers. This is important because the success of a design is strongly influenced by what people know. Because you need to observe real people to do this, you will tend to find tasks that illustrate required functionality in a person's real world context. If the task names real people, you can go back to them and ask them about any information you are missing. You will eventually use this information to see if people

realistically have the desire, knowledge and/or capabilities to accomplish their task on your system design.

As a set, the task descriptions identify a broad coverage of users and task types. Collectively, the descriptions should identify the typical 'expected' users, the occasional but still important users, and the unusual users. Similarly, they should identify typical routine tasks, infrequent but important tasks, and unexpected or odd tasks. This is important because you will need a way to decide the coverage of your system design i.e., which tasks and user groups must be included in the design, and which could be left out. Because your initial set of descriptions may have similar entries, you should reduce it by choosing particular user and task stories that best represent your expected classes of users and tasks. In practice, you should end up with a manageable number of descriptions that still gives good coverage.

C. Validate the tasks.

Your final step in the identification phase is to get a reality check of your task descriptions. You can do this by circulating a description back to the person it is describing, or to the end-user representatives who regularly interact with them in their task setting. These people should see if it fairly summarizes the activities. Specifically, they should check to see if the set of descriptions adequately covers the potential end-users of your product, if the different tasks really represent what people do, and if their details are realistic. You should ask details omitted from the original task description, get corrections, clarifications, and suggestions. Re-write these as corrected task descriptions.

This step is critical if you used a client representative, or if your team just 'made up' what they thought were good descriptions. While it may not be possible for you to interview and observe many real clients to get the descriptions, you can probably get one to comment on a compiled list of prototypical tasks.

Phase II. User-Centered Requirements Analysis

You will rarely design a system that handles all possible users and tasks equally well. This could be because you do not have the budget to develop an all-encompassing system, or because the diversity of possible users and tasks is too high to be handled by a common system, or because you cannot cost-justify certain features. As a rule of thumb, most systems are considered successful if they have about 90% coverage, that is, if 90% of the people can do 90% of the tasks reasonably well. This also means that these systems exclude 10% of the people and tasks. The next phase in TCSD is for you to decide what people and tasks will be included or excluded from your design. This list will become your basic user-centered requirements analysis of your system design.

A. Deciding which User Types to Include

You need to identify which user types will be supported by your design. Because each description identifies a representative user, you can separate these into user types. While you want to be careful not to over-stereotype people, you will likely find that some of the groups are clearly separate, with quite distinct needs and goals. In a store setting, for example, this could be store customers *vs* sales clerks. As well, you may find that some users differ considerably from each other, even though the tasks they wish to accomplish are similar. You may find people with different levels of computer experience, or with different levels of knowledge and experience of the actual task. You should go through the list and make some hard decisions about who to include in your system design, as follows.

1. *Absolutely must include.* The system design must support these user types. They are the basic audience, and leaving them out would seriously undermine the purpose of entire system.
2. *Should include if possible.* These user types are of lesser importance or are perhaps somewhat atypical. The system design should strive to accommodate them if possible. However, it is acceptable if they must do somewhat more work to use the system, or if they are excluded altogether (perhaps because other workarounds exist), or if their inclusion is deferred to the next system release.
3. *Exclude.* These user types are rare, or unimportant, or quite different from the core users, or cannot be justified from a cost perspective, or have workarounds where they do not need the system. While these users may be able to use the existing system, the system design should not go out of its way to accommodate them.

B. Deciding which Tasks to Include

Similarly, you need to identify what tasks will be handled efficiently and effectively by your design. Because each description is task-centered, you can order the task descriptions by the following criteria.

1. *Absolutely must include.* These are key tasks, and identify the essential things that people would do with the system. They are typically frequent and important tasks.
2. *Should include if possible.* These tasks should be included if budget and time permits. While still important, they are perhaps somewhat rarer. If they are not included in version 1 of the system, they should be included in version 2.
3. *Could include.* These are lesser tasks that could be supported by the system, but only if it can be included in the design almost ‘for free’. That is, if the necessary features can be easily added without impacting the rest of the interface, or if the task can be accommodated simply by the way the system supports the other tasks, then it can be included.
4. *Exclude.* These tasks are unimportant and/or so rare that no effort should be made to include them into the system.

Phase III. Design through Scenarios

With the descriptions and requirements in hand, you can now start thinking about the interface design. Each description creates the character and plot of a story. You generate design possibilities by exploring how specific designs support the telling of this story. Each design should consider how its features work together to help a person accomplish their real work. Each design should account for the expected user knowledge and motivation, where it takes into consideration the real world contexts of real users (Carroll, 2000).

As design ideas unfold, you can judge and quickly modify your interface by seeing how well it supports the story told by your core set of user/task descriptions. That is, you can perform an 'in the small' task-centered walkthrough (discussed in Phase 4) to see how well your interface and its features support particular user types and tasks.

Phase IV. Evaluate via Task-Centered Walkthroughs

A *usage scenario* combines an interface design with one of your user/task descriptions. In this phase, you choose a scenario and perform a *task centered walkthrough* of it (Nielson and Mack 1994). With a walkthrough, you tell a concrete story about what a particular user would do and see step-by-step when performing his or her particular task on the interface (Carroll, 2000).

Walkthroughs are an excellent and low-cost way to evaluate your interface, for you will quickly discover trouble spots as you move through the task. At its cheapest, you can do it by yourself, with no need for end-user involvement. However, walkthroughs tend to produce richer results when they are performed with others on your team, particularly if other members have perspectives that differ from yours e.g., designers, implementers, and end-users (Bias, 1994).

Lewis and Rheiman's algorithm for performing a task-centered walkthrough is surprisingly simple and easy to do.

```
Select one of the task scenarios
For each of the user's step/action in the task:
        Can you build a believable story that motivates the user's actions?
        Can you rely on user's expected knowledge and training about system?
        If you cannot:
                You have located a problem in the interface
                Note the problem and any comments or solutions that come to mind
        Once a problem is identified, assume it has been repaired
Go to the next step in the task
```

To make this walkthrough algorithm work effectively, you must put yourself in the mind and context of the end user. You are essentially role-playing. You must stay true to the spirit of what the person is trying to accomplish, what they know, and what is reasonable for them to do. The story you tell must be complete and must ring true. The

story should start at the very beginning of the task, perhaps even before the person touches the computer. For each expected step in the task as prescribed by the interface design, you have to ask if the person will know what they have to do next, whether they will know how to use the interface controls to do it, and whether they can comprehend the feedback provided by the system. You should continue through the task to the bitter end, even when you discover that your design is so terrible that it should be abandoned. This is because the other problems you see may help you avoid them in successive designs, and may even give you insights into new designs.

A Working Example: Cheap Shop

This sections steps through a working example to illustrate how this process can be applied.

The Situation

Cheap Shop is a catalog-based department store known for its low cost merchandise. A customer shops by browsing one of the paper catalogs scattered around the store. As the customer finds each desired item, he or she enters its item code from the catalog onto an order form. The customer then gives this form to a sales clerk at the front counter. After a modest time (about 3-8 minutes), the warehouse clerk delivers the items from the back room to the sale clerk at the front counter. The sales clerk passes it to the customer. The customer checks the items, and pays the sales clerk for the items they want. An example item for the catalog as well as a filled in form is illustrated in Figure 1.

Cheap Shop has contracted you to evaluate an in-store computer system that they have prototyped, where customers would use this prototype system to indicate and buy the items they want. The system would then send this request to the warehouse, after which the items will appear at the front counter for further processing by a clerk.

If the prototype has major problems, you can either suggest how it can be repaired or propose a completely new design.

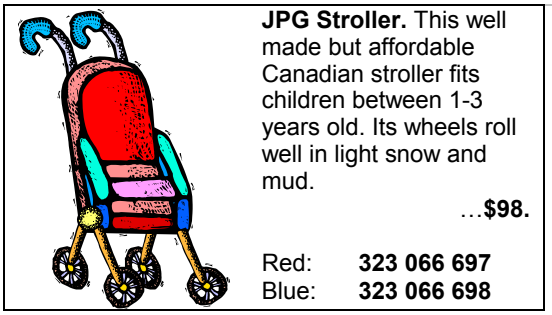


Figure 1a: The catalog entry for the stroller

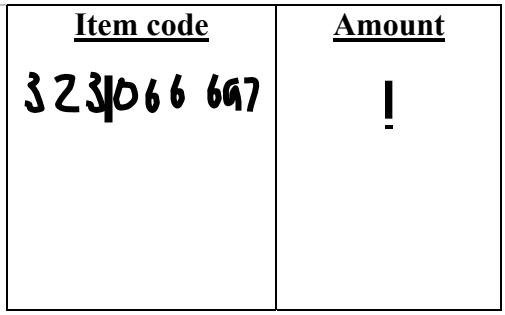
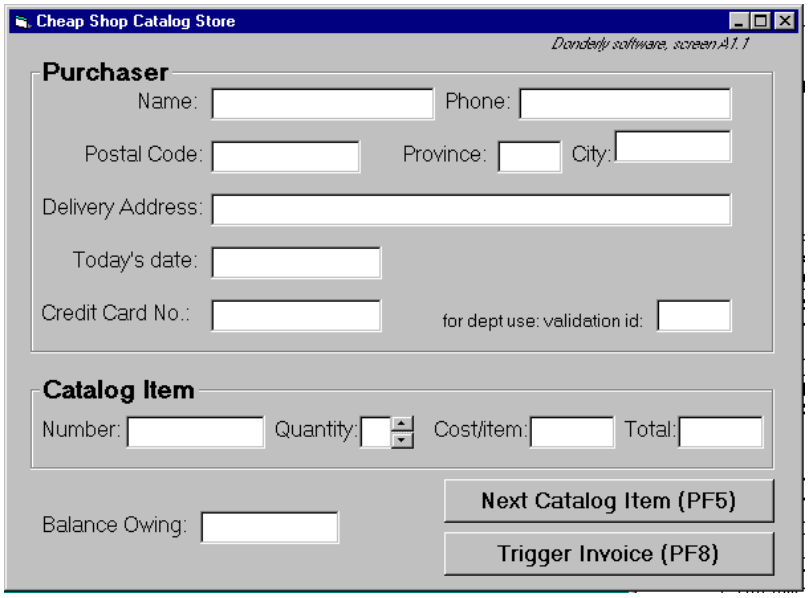


Figure 1b: The filled-in order form for the stroller

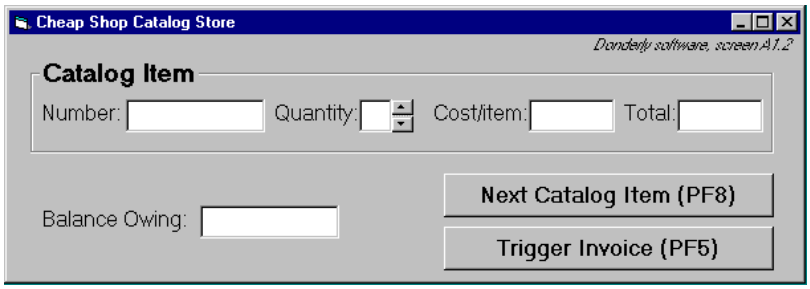
The Cheap Shop Prototype

The prototype illustrated in Figure 2 is intended to be available on all Cheap Shop Department Store computers. Shoppers in the store decide on the item they want by browsing the catalog, and can then purchase items by entering the relevant information into these screens.

Screen 1



Screen 2



Prototype specifications.
To order the first item:
- shoppers follow the sequence on screen 1 to enter their personal information and their first order;
- text is entered via keyboard, and the tab or mouse is used to go between fields.

To order additional items:
- shoppers fill in screen 2 after clicking *Next Catalog Item* (can be repeated).

To complete an order:
- shoppers click 'Trigger Invoice';
- the system automatically tells shipping and billing about the order;
- the system returns to a blank screen #1.

To cancel the order:
- shoppers do not enter input for 30 seconds (as if they walk away);
- the system will then clear all screens and return to the main screen.

Input checking:
- all input fields checked when either button is pressed;
- erroneous fields will blink for 3 seconds, and will then be cleared;
- the shopper can then re-enter the correct values in those fields.

Figure 2: The Cheap Shop Prototype

Producing User and Task Descriptions for Cheap Shop

We collected descriptions by monitoring customer activity at the Cheap Shop store. We validated each description by interviewing the customer and by asking them if it reflected what they did. We also sat behind the counter with the store clerks, where we observed what they did and how customers and clerks talked to each other. Later, we gave the complete set of descriptions to the store clerks (the client representative) and asked them if the descriptions typified what they saw in terms of customer requests. Three of our descriptions are included below. Notice that they follow the criteria for good task descriptions identified earlier.

Task 1. Fred Johnson, who is caring for his demanding toddler son, wants a good quality umbrella stroller (red is preferred, but blue is acceptable). He browsers the catalog and chooses the JPG stroller (cost $98., item code 323 066 697). He pays for it in cash, and uses it immediately. Fred is a first-time customer to this store, has little computer experience, and says he types very slowly with one finger.

Discussion. Fred has many properties of our typical expected user: many customers are first time shoppers, and a good number have no computer experience and are poor typists. Similarly, the task type is routine and important. Many people often purchase only one item, and a good number of those pay by cash. As with Fred, people often have a general sense of what they want to buy, but decide on the actual product only after seeing what is available.

Task 2. Mary Vornushia, an elderly arthritic woman, is price-comparing the costs of a child's bedroom set, consisting of a wooden desk, a chair, a single bed, a mattress, a bedspread, and a pillow all made by Furnons Company. She takes the description and total cost away with her to check against other stores. Three hours later, she returns and decides to buy everything but the chair. She pays by credit card and asks for the items to be delivered to her daughter's home at 31247 Lucinda Drive, in the basement suite at the back of the house.

Discussion. Like Mary, a reasonable number of store customers are elderly, with infirmities that inhibit their physical abilities. A modest number of them also enjoy comparison shopping, perhaps because they have more time on their hands or because they are on low income. Although this would be considered a 'major' purchase in terms of the total cost, the number of items purchased is not unusual. Delivery of large items is the norm, and many customers pay by credit card for larger orders.

Task 3. John Forham, the sole sales clerk in the store, is given a list of 10 items by a customer who does not want to use the computer. The items are: 4 pine chairs, 1 pine table, 6 blue place mats, 6 "lor" forks, 6 "lor" table spoons, 6 "lor" teaspoons, 6 "lor" knives, 1 "tot" tricycle, 1 red ball, 1 "silva" croquet set. After seeing the total, the customer tells John he will take all but the silverware, and decides to add 1 blue ball to the list. The customer starts paying John by credit card, but then changes his mind and decides to pay cash. The customer tells John he wants the items delivered to his home the

day after tomorrow. While this is occurring, 6 other customers are waiting for John. John has been on staff for 1 week, and is only partway through his training program.

Discussion. This task introduces the clerk as a system user. While every store will have a few clerks, they are vastly outnumbered by the number of customers using the system. Because the store has a high turnover in its staff, new employees such as John are also common. Thus John reflects a 'rare' but important group of users. The task that John is asked to do by the customer, while complex, is fairly typical i.e., people making large numbers of purchases often ask the clerk to help them. Similarly, clerks mention that customers often change their mind partway through a transaction i.e., by changing what they want to buy and/or by changing how they want to pay for it. Customers, however, rarely give specific delivery dates, with most wanting delivery as soon as possible. Lineups for clerks do happen during busy times.

The Task Walkthrough.

Because we already have an interface in hand, we begin by performing a task centered walkthrough of it. The example below shows our walkthrough analysis performed with a scenario that combines our first task description with this interface.

For reporting purposes, each walkthrough report is preceded by a description of the scenario i.e., which task description and which interface is being analyzed. The table itself tells the story, where it records the step-by-step results of the task walkthrough algorithm. The 1st column describes each task step in sequence. The 2nd column asks if that person has the knowledge or training to do this step, and if it is believable that the person would be motivated to do what is asked of them. The final column records problem details, comments, and (optional) solutions to any problems.

Interface: Cheap Shop prototype #1.
Description #1. Fred Johnson, who is caring for his demanding toddler son, wants a good quality umbrella stroller (red is preferred, but blue is acceptable). He browsers the catalog and chooses the JPG stroller (cost $98., item code 323 066 697). He pays for it in cash, and uses it immediately. Fred is a first-time customer to this store, has little computer experience, and says he types very slowly with one finger.

The sequence in Table 1 illustrates the value of starting the walkthrough at the very beginning of the task. Notably, we see that we are missing information about whether paper or electronic catalogs will be used, how the computerized system is situated in the store environment, and whether signage or other instructional material will tell customers what to do. While doing task centered design, you should examine what information is missing, and list what assumptions you are making. You should validate these assumptions, as incorrect ones can profoundly affect how the interface will perform in the actual setting. Even when assumptions reveal issues 'outside' the actual interface being designed, they can be critical to its success.

Task step	Knowledge? Believable? Motivated?	Comments/solutions.
a Enters store	ok	
b Looks for catalog	Ok if paper catalog is used, but what if the catalog is on-line?	Finding paper catalogs is not a problem in the current store. However, we were not told if the paper catalog would still be used, or if the catalog would be made available on line. ***Note:*** ask Cheap Shop about this. If they are developing an electronic catalog, we will have to consider how our interface will work with it. For now, we assume only a paper catalog is used.
c Finds red JPG stroller in catalog	Ok	The current paper catalog has proven itself repeatedly as an effective way for customers to browse Cheap Shop merchandise and to locate products.
d Looks for computer	Modest problem	As a first time customer, Fred does not know that he needs to order through the computer. Unfortunately, we do not know how the store plans to tell customers that they should use the computer. Is there a computer next to every catalog (so its association can be inferred), or are there limited number of computers on separate counters? Are there signs telling Fred what to do? ***Note:*** ask Cheap Shop about the store layout and possible signage. ***Possible solution:*** Instead of screen 1, a startup screen can clearly indicate what the computer is for e.g., "Order your items here" in large letters.

Table 1: From entering the store to finding the computer

The next sequence in Table 2 illustrates fundamental problems that arise as one walks through task step details, such as how Fred selects and moves between fields, and how Fred enters input. It also illustrates how the inspector can do a reality check on walkthrough steps in the large i.e., whether the expected sequence of activities matches Fred's goals, and whether Fred is willing to enter the expected information. In this case, we see several serious problems that must be repaired.

e Enters name	No motivation to do this!	Fred's task is to buy the stroller, but the scenario shows that the system is asking him for his name. Fred may be reluctant to do so if (say) he believes that he will be added to a mailing list without his permission. ***Note.*** Ask Cheap Shop why they are asking for the customer's name and other contact information.
f Selects the name field	Knowledge lacking. Fred does not know how to select a field.	To enter his name, Fred is expected to click and type into the first text field on this form. Yet Fred has little computer experience, and thus he may not know what to do. He may also be reluctant to experiment with the system. ***Possible solutions:*** a) have the first field pre-selected, with the cursor in it. b) have a poster next to the computer describing these basic acts.

g Types his name	Knowledge lacking: Fred types poorly, does not know name format.	Because Fred types poorly, text entry will be slow and tedious. This further dampens Fred's motivation, as he is entering information that is unimportant to the task. Fred is uncertain about formats: does he type his name as 'Fred Johnson' or 'Johnson, Fred'?
h Moves to phone field	Knowledge lacking	Fred may not know how to tab or mouse over to the next field because of his unfamiliarity with computers.
i Fill in phone, postal code, province, and city.	Poor motivation. Poor format knowledge	If Fred can complete steps e-h, he will be able to continue with the following fields. However, motivation will decrease even further as Fred painfully types unnecessary information into the system. Fred continues to have formatting concerns about how he should enter information. Should the phone number include the area code, spaces and/or dashes? Should he spell out the province or use the abbreviation? Should he leave a space in the postal code?
j Enter delivery address	Violates the task	Fred will use the stroller immediately, but the system asks for his delivery address. Fred may incorrectly assume that he is filling in the wrong form, and may give up. We also noticed that the order of the contact information does not follow the typical flow i.e., one would expect 'Name, Address, City, Province, Postal Code', 'Phone' rather than the odd order shown in the form.
k Enters today's date	No motivation	This is an odd field… why should Fred enter the date when the system already knows it? Can he skip it? If he does fill it in, he would be quite lucky to enter a recognizable date format.
l Enters credit card information	Violates the task	Fred is paying by cash, and thus he is unwilling enter his credit card number. He is also concerned that others may see his credit card information as he types it onto the screen. Finally, this seems an odd place to ask for payment information. Most stores ask for it at the end of transaction, not at the beginning.
m Ignore validation id	Ok	While Fred will likely do the right thing, this field should not be here. It has nothing to do with Fred's task. ***Possible solution.*** Remove it.
Steps e-m	Not needed for task.	***Possible solution.*** This entire part of the interface is not needed or is, at best, optional (e.g., if it is for getting onto the mailing list). Delete it entirely or move it into a very secondary area that can be filled in after the transaction is completed.

Table 2: Entering personal information

The task steps in Table 3 illustrate how poorly the interface handles the most critical part of the task, where Fred specifies the item he wants to buy, and tries to complete the transaction. We also see that the interface is full of jargon and unneeded or poorly designed interface components.

n Enter item number for the JPG red stroller	Is motivated, but has problems. Error-prone.	The item number for Fred's stroller is written in the paper catalog as 323 066 397. Because catalogs are common, he may be able to figure out what he has to do. However, the format is a bit mysterious – should he include spaces or not? If the paper catalog is in an awkward place, Fred will have to rely on his memory to enter the number or he will constantly be running back and forth between the catalog and the computer. Because Fred is a poor typist, he may have difficulty typing the number correctly.
o Enter quantity	Knowledge low, motivation high	Fred wants one stroller only. However, this 'spinner widget' is somewhat mysterious to Fred. Because he does not know computers, he will likely not know that he can type the amount directly or just click the arrows to select the quantity. ***Partial solution.*** Have the spinner show a 1 by default.
p Enter cost/item	Motivation low	Why should Fred enter the cost? Surely the system knows this. If this field is actually used to display the cost, then it has the wrong visual affordance as it looks like a text box. Perhaps Fred would be willing to enter a deeply discounted cost, but this will probably be treated as a system error.
q Enter total	Motivation low	See above point
r Enter balance owing	Motivation low, knowledge low	See above point. Fred will also be uncertain about how this field differs from the 'Total' field.
s Click Trigger invoice or press PF5	Knowledge low	Being inexperienced with computers, Fred may not recognize or know how to use a clickable button. The 'PF5' label is also mysterious, as Fred does not recognize it as a keyboard shortcut. Fred will find the meaning of 'Trigger Invoice' cryptic, as it is in the language of the database system rather than in his language. This may leave him at a loss of what to do. *Possible solutions.* Remove the PF5 label. Change 'Trigger Invoice' to something more meaningful.

Table 3: Buying the stroller

As with the opening task sequence in Table 1, the sequence in Table 4 illustrates how the closing of the task must recognize factors that go beyond the interface. In this case, we see a serious problem of how the electronic part of the task flows through to the physical completion of the task.

t Wait for item at sales counter	Knowledge low, motivation high	Fred has to go to the sales clerk and wait for the item to appear. Yet he may not know this, especially because the computer returns to the initial empty screen. Has the transaction completed successfully? Is there signage that says what has to happen? ***Possible solution.*** Provide a final screen that tells Fred what he has bought and what he has to do next.
u Get item from Sales Clerk.	Knowledge low, motivation high	If other items are appearing aside from Fred's, he may not know which items are his unless the boxes are clearly labeled or if the box size and shape give it away. Similarly, the sales clerk has no easy way to identify whose items have appeared, unless the name given in the Name field is somehow attached to the items. ***Possible solution.*** After completing Trigger Invoice, the system could print out a sheet listing the chosen items which Fred can then give to the sales clerk.
v Pay for item in cash	Ok	While this is straight-forward, there is a question about how the clerk will tally up this bill. This is the clerk's problem, but we don't want Fred to wait excessively.
w Use it immediately	Ok	

Table 4: Picking up the stroller and paying for it

The above sequences in Tables 1-4 walk through the correct task sequence. What happens when errors or other events occur that disrupt this sequence? Table 5 illustrates a set a sequences dealing with problems. It shows that walkthroughs are also effective for discovering how the system design deals with problems, and with events arising from the end-user's real-world context.

1 Event: interruptions and timeout.		
a Deal with toddler	Knowledge high, motivation high	Fred's toddler starts demanding his attention part way through this task (say after he has entered the item number). Fred comforts his child.
b Deal with timeout	Knowledge low, motivation low	Unfortunately, this took more than 30 seconds, which means that the system has cleared the screen. Fred has to re-enter all this information, which he will likely not do. Note that a similar problem will happen if Fred lingers too long on any step in this task.
2 Error: incorrect item number		
a Recognize error message	Knowledge low	If Fred enters an incorrect item number, the system will blink that field for 3 seconds and then clear it. It is extremely unlikely that Fred will know what this means.
b Enter corrected item number	Knowledge low, motivation medium	Even if Fred realizes that he has made a mistake entering the item number, he will be uncertain about what he did wrong (since the number is no longer there), or how to correct it.
3 Event: red stroller unavailable		
a No red stroller is in stock	Knowledge low	If there is no red stroller in stock, how does Fred find this out? Will the sales clerk tell him (in which case the clerk needs to identify Fred)?

b Reenter all information for blue stroller	Motivation low	We cannot imagine that Fred would be willing to go through this whole process again, especially because his demanding toddler is likely loosing patience. ***Possible solution:*** As the customer selects and item, the interface should clearly indicate if it is in stock.

Table 5: Interruptions, errors, and exceptions

While Tables 1-5 appear fairly detailed, they do not cover all task steps. Some steps are intentionally left out because they are identical to previously seen task steps. For example, after identifying problems with selecting and moving between fields as well as typing (steps f-h in Table 2), these details are skipped in other task steps. Other steps are left out because of oversights i.e., they simply were not thought about. For example, no mention is made in Table 5 about how the customer or clerk unpacked the box the stroller came in, or what was done with it afterwards. While oversights will happen, they can be reduced by adding more people to the team and by grounding the task by real observations. For example, videotaping a person doing a real task forces one to account for all visible steps.

Walkthroughs of the interface using the other two descriptions yield other problems. For the second description, we identify a user (Mary) who cannot use the mouse or type because she is arthritic. What should be done with her? We also see serious problems when multiple items are ordered; the system gives no feedback of what was entered, and there is no way to correct errors without re-entering everything from scratch. There is also no easy way for Mary to price-compare (since no printout is provided), nor is there any way for her to recall information that she previously entered. Details are also a problem: there is no easy way for her to tell the system about the unusual delivery address (i.e., that it is the basement in the back). For the third description, we see that the system is completely unsuitable for use by the store clerk. Item entry is too slow, and the clerk will not be able to keep up with the changes requested by the customer. This slowness also affects other customers waiting in line. We also see that the system cannot accommodate delayed delivery.

The bottom line is that this design is a disaster, and it should be completely revamped.

Determining User-Centered Requirements

Before redesigning the interface, we revisited the descriptions and made decisions on which users and tasks should be supported by the design. We split the main user types and tasks into the following categories and dimensions.

Customers: • first time *vs* repeat customers • computer knowledgeable *vs* computer naïve • typists *vs* non-typists • willing *vs* unwilling to use the computer • people with disabilities who may have trouble with fine motor control ***Sales clerks:*** • experienced and trained • new staff member; has passed introductory training session …	***Choosing merchandise*** • one item • multiple items • modifying the selected list of items ***Pay by*** • cash • credit or debit card • invoice ***Reviewing cost*** • individual item cost • total costs • comparison shopping ***Merchandise pick-up*** • immediate • delivery …

Table 6 Categories of user types and tasks

In terms of user types, we decided that the system design must include first time and repeat customers who are computer-knowledgeable and who are willing to use the computer. These make up a large customer base, and they have skills that should allow them to use the system with minimal effort. The requirements should also include customers who are computer naïve and who are non-typists, for these too are also a large part of the customer base. However, the system can exclude those who may have disabilities that interfere with their ability to enter their data in the system; those people are fairly rare, and they can simply ask the sales clerk to help them. We also recommend that a different system be designed for sales clerks, because their needs are unlike those of typical customers.

In terms of tasks, the system design requirements must let people choose and easily modify a list of merchandise comprising at least one to seven items, where they can quickly determine the individual and total cost. They should be able to pay by cash, credit or debit card, and take the merchandise away with them. These are all important and very frequent tasks that customers do. The system should also let people buy more than seven items, although this can be deferred as the large majority of people buy seven items or less at Cheap Shop. The system could include the ability to let people comparison shop (e.g., by printing out the screen) or retrieve previous orders. The system can exclude delivery or payments by invoice as these are fairly rare and are best handled by the sales clerk.

Interface Design.

From these requirements and from our knowledge of the problems detected in our walkthrough, we prototyped the interface illustrated below (this non-functional prototype was created in about 45 minutes).

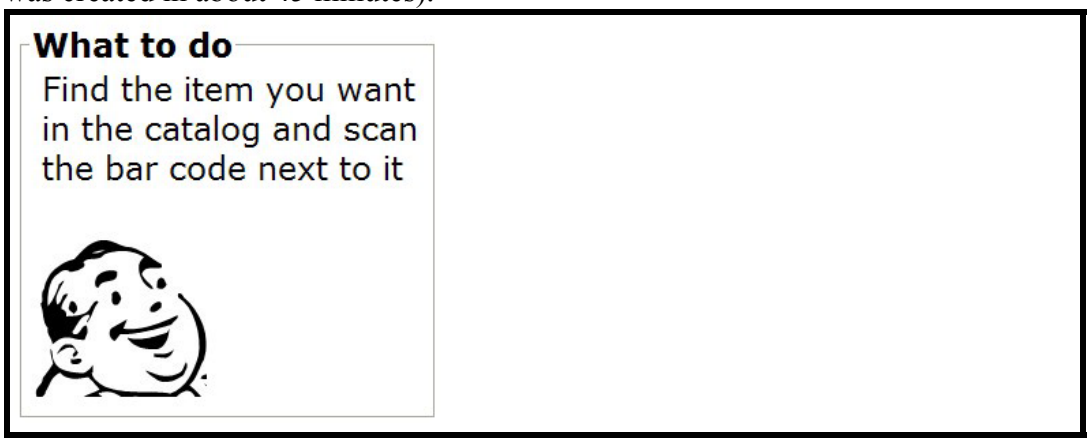


Figure 3a: The initial screen of the new prototype

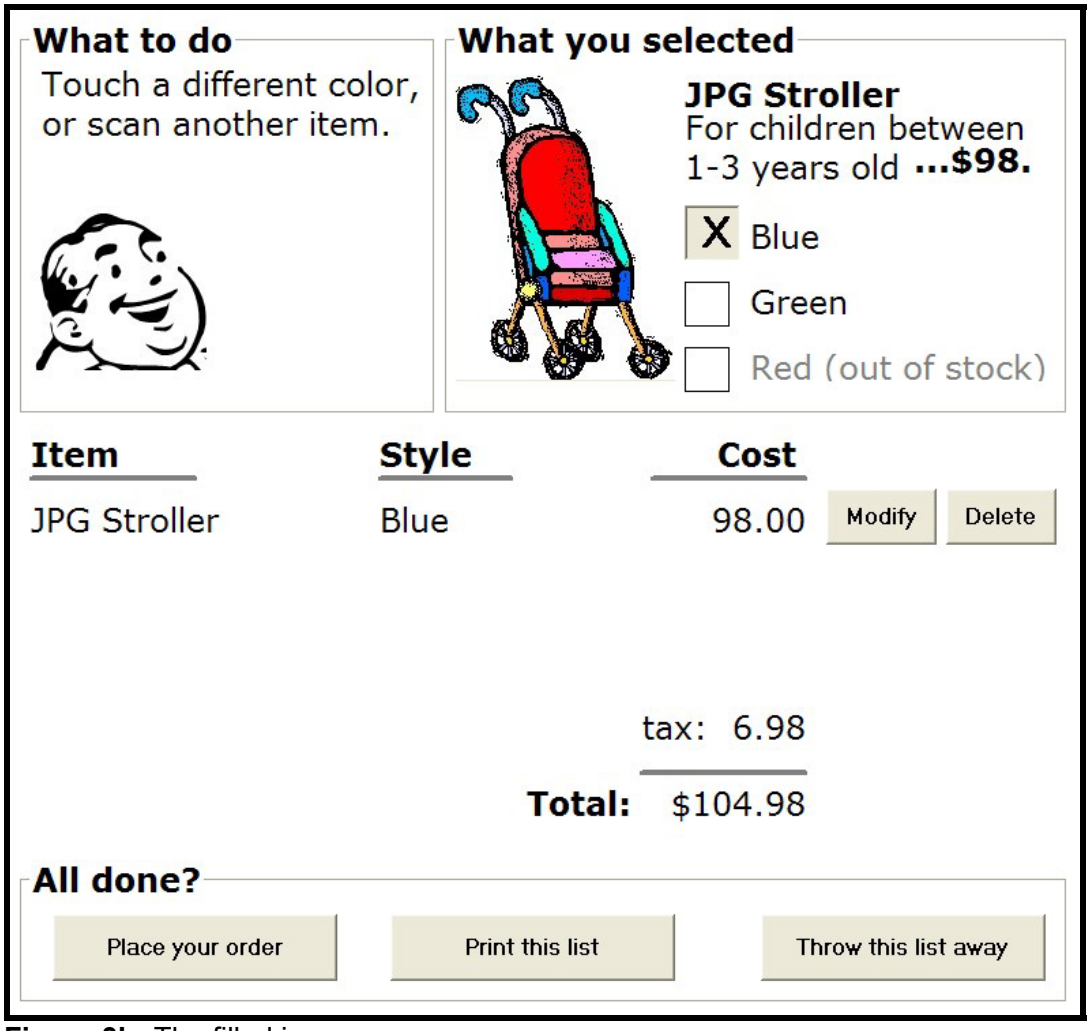


Figure 3b: The filled in screen

It differs considerably from the previous version. Because people may not know how to type or use a mouse, we use only a bar code reader and a touch screen for input. Paper catalog descriptions of items will be modified to include bar codes. A 'getting started' instructional poster is located next to the computer (not shown) that pictorially tells people how to use the bar code reader, and that tells them the sales clerk would gladly help them if they have any problems. Because we want this system to let people easily select and purchase just a few items (perhaps changing their mind on the way), the interface portrays itself as a dynamic shopping list. Because people may be unfamiliar with the system, context-sensitive on-screen instructions prompts the user every step of the way. To illustrate how this works, we briefly revisit the first task in the two screen snapshots illustrated in Figure 3. In the initial screen, the 'wizard" tells Fred how to add an item to the shopping list. Fred complies and finds his stroller in the catalog and scans the bar code next to it. The 2nd screen shows the result: a picture of the stroller, its description and its cost appears, and it is automatically added to the shopping list. Because the stroller comes in several colors, the wizard tells Fred he can change the current color (chosen by default) by touching the color he wants (this would automatically modify the shopping list). Fred wanted a red stroller, but sees it is out of stock so he leaves it as blue. At this point, the system lets the customer scan in other items, or delete or modify items already in the list by touching the appropriate button located at an item's right. If the modify button is touched, the item reappears in the "What you selected" box. Fred just wants the stroller, so he touches the 'Place your order' button at the bottom. A copy of the shopping list prints out (including its own unique bar code) and the wizard tells Fred to give it to the sales clerk at the front counter (not shown). The clerk takes the order from Fred, and uses it to collect the items as they appear from the warehouse. Fred is ready to pay by cash, so the clerk displays the order on his screen by scanning the barcode on the printout and processes it through his own system.

This prototype should handle other tasks as well. For the comparison shopper in Description 2, she can choose to print the list: in this case the printout appears but the order is not placed. If the shopper comes back later to order the items, she simply scans the barcode on the printout, which puts her shopping list back onto the screen (instructions on the form tell them that they can do this). We are still uncertain if she can do this because of her arthritis, but it could be possible with mild cases.

While we created this solution to address some of the problems identified in the task analysis, we don't really know how good these fixes are. We do know that major redesigns such as this one will likely introduce new problems, both large and small!

Thus we go back to the beginning of our task walkthrough process. However, it is now your turn. Does this interface work? Do a detailed task-centered walkthrough to discover where it succeeds and where it fails. There are certainly flaws in it!

Conclusions

Task-centered system design is a very effective discount usability engineering method suitable for many interface development problems. It offers a big 'bang for the buck', where you can probably achieve a reasonable design with only modest extra effort. The caveat is that it is not perfect. Compared to other more precise techniques, you will likely miss a task or user group, or you may overlook a task nuance during the walkthrough. I would not use TCSD to design an air traffic control system, but I would use it if I had to produce a non-critical system under a limited budget. I would also supplement it with other evaluation techniques, such as heuristic evaluation (Nielsen and Mack, 1994) and user testing (Dumais and Redish, 1993).

This chapter is sufficient to get you started on your own task-centered system design project. If this method excites you, you should also read Lewis and Reiman's book on the topic (1993). They go into further detail, and describe how the TCSD method relates to other aspects of interface design and evaluation.

There are other good sources on related information. Beyer and Holtzlatt (1998) describe the contextual interview, which is an excellent way to discover detailed user and task descriptions. Carroll (2000) goes into great detail on scenario-based design. Cooper (1999) presents a variant on TCSD called goal-centered system design. Nielsen and Mack (1994) collects readings about other discount interface inspection methods. Finally, my own teaching materials on this topic are available at www.cpsc.ucalgary.ca/~saul/hci_topics/topics/tasks.html. Materials include PowerPoint presentations, student assignments, and teaching tips.

Acknowledgments. The National Sciences and Engineering Research Council of Canada (NSERC) provided funding. This material has been refined over years of teaching University of Calgary undergraduate students; I owe them my gratitude.

References

Beyer, H. & Holtzblatt, K. (1998) Contextual Design: Defining Customer-Centered Design. Morgan-Kaufmann.

Bias, R. (1994) The Pluralistic Usability Walkthrough: Coordinated Empathies. in J. Nielsen & R. Mack Usability Inspection Methods, pp.63-76, John Wiley, 1994.

Carroll, J. (2000) Making Use: Scenario-Based Design of Human-Computer Interactions. MIT Press.

Cooper, A. (1999) The Inmates are running the Asylum: Why High-tech Products Drive Us Crazy and How to Restore the Sanity. SAMS.

Dumas, J. & Redish, J. (1993) A Practical Guide to Usability Testing. Ablex.

Lewis & Reiman (1993) Task Centered User Interface Design: A Practical Introduction. University of Colorado, Boulder. Shareware book available from ftp.cs.colorado.edu/pub/cs/distribs/clewis/HCI-Design-Book/.

Nielsen, J. & Mack, R. (1994) Usability Inspection Methods, John Wiley & Sons.