A common reason users find an interface unfriendly or not intuitive is because it does not support how they really do their work and think about their work. Too often, the software architecture design dictates the interface design, for example, logical system function partitioning and data flow dictate screen or window partitioning and dialog flow.

Analysis identifies current user tasks and the information, concepts, and terminology users use when performing their work. It then describes future tasks from which the interface design work can begin. Analysis is a key phase in designing an interface that is an effective tool for users.

In this chapter we will explore the steps in analysis using a sample application, Check-Ease, software for keeping an individual's checkbook on computer.

**Purpose**
The purpose of the Analysis phase is to document and verify information about the users, their current work, and the vision of their work when the new software is in place. This information feeds directly into the next phase, Design.

**Deliverables**
Deliverables in this phase include:
- **User profiles.** A complete description of your primary and secondary users.
- **Current task analyses.** Thorough descriptions of how users are currently doing their work.
- **Future task descriptions.** High-level descriptions of how users will be doing their work using the new software.
- **Usability specifications.** Identification of which usability issues and areas are of most concern for this project and specific measurements to determine whether the resulting software is usable.
- **Use case scenarios.** A set of specific scenarios for how users will interact with the new software, for use in the next phase, Design.

**Preparation**
Before you begin analysis, you should put together an analysis consisting of:
- **One interface designer.** This person is the one who most needs to use the analysis results. Because of this, he or she will make sure the work stays focused and in the right amount of detail.
- **Two content experts/users.** Since you are describing the way users will work in detail, it is critical that you have people present who can really describe the work on a detailed level. You want more than one content expert/user because often the best information comes out of listening to two users talk to each other. During the part of analysis where you describe the future, it is important that the users be knowledgeable about future plans and have the authority to say how the tasks will or should be done.
- **One technical person.** You may not need to have a technical person involved in all the working sessions for analysis, but you should have one identified that is available to answer questions as they arise about planned system opportunities and constraints.
- **A managerial level person.** If a manager would be helpful in answering questions for a particular step you should have one there. There may be times, however, when including the manager may prevent your users from talking freely. You may need to have some sessions without a manager present.

**Process**
The Analysis phase includes both learning about and documenting users’ current tasks, as well as documenting and verifying the future tasks from the user’s point of view. The steps in the Analysis phase are:
1. Identify current state and scope.
2. Develop user profiles.
3. Gather data.
5. Document problems and **opportunities.**
7. Develop usability specifications.
8. Develop use case scenarios.

### 1 Identify Current State and Scope
You are probably not the first person to do analysis work for a particular project. It is important, therefore, for you to know what has and has not been done, especially in the area of analysis. You do not want to redo work or be redundant with information that has already been gathered. On the other hand, be careful
about assuming that someone else has documented users' work thoroughly. Our experience shows that there is a lot of variability in both the quantity and quality of analysis work that is actually completed on a project.

1.1 Consider Work in Progress
You will want to study and/or review any materials on the project that are available. Has there been a feasibility study? A meeting to discuss the new system? Business process reengineering? Make sure you know what work has been done, and at what stage the project is. It is possible that some of the materials you need or even would be creating for analysis have already been completed, or at least started. If you find some already completed, verify that the information is still valid.

1.2 Decide on the Scope of the Analysis
Define the scope of the analysis in terms of the user activities the interface will support and portray. The goal is to identify the high-level task areas, not the flow or steps involved in performing the tasks. Detailed analysis of the current tasks will be performed later, as part of the Document Current Tasks step.

Consider the following as you decide on the scope of the analysis:

- Consult project documentation and project management to identify the general user activities that the new interface will support. User activities are high-level task areas. For example, for an electronic checkbook application these might be: Write Checks, Balance Checkbook, Record Deposits, and so on.
- List and describe briefly the user activities identified.
- Check the list of activities to make sure that the perspective and wording reflect user activities. There will be other system functions required to support the user in performing these activities, but the focus here is on what users do.
- If an activity is very broad, list user tasks that are performed as part of the activity. For example, for a personal information manager application, the activity Maintain Personal Calendar would include the tasks Enter Appointments and Move/Change Appointments.
- Check that the activity and task wording is independent of interface design solutions or assumptions, for instance Find A Check, not Scroll Through Register.
- Make sure that the wording doesn't refer to the current system design, for instance, Set Up New Client Service, not Enter an IN Order.

The list of activities and tasks represents starting assumptions about the scope of the interface and the analysis work. These assumptions and any outstanding issues should be reviewed with the rest of the project team.

1.3 Define Interface Design Constraints
In an ideal project, all decisions regarding the design and implementation of the interface would be driven by user needs. Unfortunately, in real-world projects, technology, business concerns, and other system issues place constraints on the interface design. Your objective is to identify all system hardware, software, and project features and requirements that might constrain the interface design:

- Consult work in progress to identify decisions that have already been made regarding hardware, software and project constraints.
- Identify the interface technology that will be employed, for example, GUI, a mixture of GUI and 3270 screens, and so on.
- Identify the platforms (for example, UNIX, DOS, OS/2) and machines (for example, UNIX workstations, X terminals, PCs) that will be used.
- Identify what user interface techniques will be available. For example, is this a multiple window environment? Will all users have display hardware that supports color? What are the available text fonts? Will all users have a keyboard with function keys, a mouse, a trackball?
- Identify any corporate or project interface design standards and style guides that are to be followed (such as IBM Common User Access, Microsoft Windows, Open Look).
- Identify the degree of distributive processing. How much functionality will be performed by the user interface (for example, edits, computations)? How will this affect user interaction (for example, system response time)?

Document these interface design constraints. They represent a set of starting interface design assumptions. Review the constraints and any outstanding issues regarding interface technology and implementation with the rest of the project team.

Figure 2.1 shows the interface design constraints for the Check-Ease application.
Interface Design Constraints for Check-Ease

IBM PC 486 or better (and clones). More and more, these will be laptop computers.
Modems with baud rates as low as 14400.
EGA and VGA monitors. Minimum resolution is 640 x 480.
Color and monochrome displays.
Microsoft Windows 95 compliance or better.

Figure 2.1 Design constraints for Check-Ease.

2 Develop User Profiles

A common mistake designers make is to assume that all users are like themselves. Then they design for themselves. When designing software for the nonprogrammer, most software designers/developers are not representative of their users. Two other reasons that interface designers are sometimes in the dark about users:

- They are not given the opportunity and resources to study users.
- Marketing or management doesn't want to get pinned down on the precise targets for the system.

It is not possible to design a user-friendly interface without designing for a user. It is not possible to design for no one nor to design for everyone. Building a description of target users is the first, and a critical, step in interface design.

To be useful for interface design, user profiles must go beyond simple characterizations like computer novices, computer experts, managers, or secretaries. The objective is to find out about those user characteristics that will be most important for interface design tradeoffs. These include:

- User experience -with the hardware and software environments that the project will use. What percent of target users are now using a mouse? What percent are keyboard touch typists familiar with standard keyboard layout? What percent are now using windowed interfaces? Are the interfaces these users are familiar with compliant with any standard styles like Microsoft Windows?
- User experience with the kind of software application the project will develop. Will this be their first application of this type (for example, first accounting application)? If not, which systems are they now using?
- User task experience/expertise. Have these users performed this kind of work before?
- Expected frequency of use and job turnover. If turnover is very high or frequency of use will be very low, the users will never become experts at the software. Therefore interface ease of learning will be important. If frequency of use will be very high and turnover will be low, ease of use will be more important than ease of learning.

A completed user profile is shown in Figure 2.2. Note that key interface design requirements are listed at the bottom of the profile.

How are user profiles different from marketing analyses? Marketing information and discussions with field representatives are often good places to start when building user profiles. However, they often contain information that is not critical to designing an interface, and may be missing some detailed information that is needed for interface design. You may need to supplement marketing information with user interviews and/or surveys in order to obtain and validate all the required data.

"But, we can't build a user profile, we have too many kinds of users." Designing for everyone is designing for no one. If the system will support multiple, very different groups of users, then user profiles must be developed for at least the key groups. Estimate what percentage of the total users each group represents. These numbers will be critical when trading off interface design approaches and features.

To develop user profiles:

- Work with project management and project marketing to identify targeted user groups for the system and its interface.
- Work with marketing, field representatives, and representative users to develop draft descriptions of targeted user groups.
- Identify user experience with the hardware environment that will be used (for example, IBM PC, SUN, networking) and input and output devices that will be used (for example, mouse experience, keyboard typing skills).
- Identify user experience with the software environment that will be used (such as windowed user interfaces in general, specific interface style such as Microsoft Windows 95, and so on.).
- Identify user experience with similar software applications (for example, other financial software, other accounting packages, and so on.).
- Identify user experience with the actual work and tasks that the application will support (such as accounting, financial planning, and so on.).
- Characterize targeted users on other dimensions that could be important for interface design, including frequency of use, rate of turnover, level of motivation in using the system, discretionary versus mandatory system usage, and multilinguism.
### User Profile

**Application**: Checkbook for IBM PCs and PC clones  
**Potential Users**: Adults who currently have an IBM PC or PC clone, Microsoft Windows, and a checking account at a bank.

**Hardware Experience**:
- 100% currently have an IBM PC or PC clone; that is, users will not be purchasing computer for this application.
- 100% will have a mouse or trackball.

**Software and Interface Experience**:
- Over 90% are using a Microsoft Windows 95 application now such as a word processing or a spreadsheet application; that is, users will not be purchasing and learning Windows 95 for this application.
- Less than 50% have used a modem before.

**Experience with Similar Applications**: Less than 15% are currently using another electronic checkbook application.

**Task Experience**:
- All currently have a checking account at a bank.
- Only 40% probably balance a checkbook now; users are not bankers or accountants.
- Less than 10% do any electronic banking from home now.

**Frequency of Use**: Frequency of use will be very low (every couple of months) to low (twice a month).

**Key Interface Design Requirements that Profile Suggests**:
- Conformance to Windows 95 interface style
- Ease of learning will be important
- Build on existing mental model of manual checking
- Provide an intuitive model for modem aspects
- Do not assume good understanding of procedures for balancing a checkbook

---

**Figure 2.2** Sample user profile form.

- Where needed, for each of these characteristics identify and estimate size of user subgroups in percentages (for example, 25% using interface now, 50% used interface a few times, 25% never used interface before).
- Validate these draft profiles through user questionnaire surveys, interviews, and site visits.
- Identify key interface design requirements that these profiles suggest (such as compatibility with existing interface standards, compatibility with current/existing applications, ease of use versus ease of learning).
- Everyone on the design team, in development, and in marketing needs to agree on who the interface is for before you start designing it (shared vision). Make sure there is good understanding and consensus.
- Document user profiles and other interface design requirements. Otherwise, how will you defend the design the first time a project manager (who is not a target user) tries the interface and declares it unfriendly?
- Use the user profiles to determine who to interview and observe during the rest of the Analysis phase, and who would be appropriate users for design reviews and usability testing in future phases.

### 3 Gather Data

Once you have identified who your users are, you need to gather data from them. The data you are gathering includes verifying your user profiles, as well as getting details on their current and future task flows. To provide the critical data needed for interface design, task descriptions must be built by talking to and observing users who do the work the system is targeted to support, not from talking to managers, marketing representatives, trainers, and so on. There are several methods you can use to gather data. Three are described below.

#### 3.1 Bringing Users In

One of the most common ways that information is gathered is by bringing users in to your work place and asking them questions about their work. This is usually a convenient and cost-effective way for you to gather data, and it can yield some valuable information, but it has some dangers and drawbacks that you want to watch out for:

- **Oversimplification**. If users feel you are not a domain expert, they will often try to be helpful by oversimplifying the description of their work. They may not realize that this is not helpful, but actually harmful to your data collection. And if you are, indeed, lacking in domain expertise, you may not realize at all that you don't have the full picture. These can lead to important tasks not...
being captured in enough detail later when you need to create a current task analysis or future task
description.

• **Missing information.** Have you ever tried to describe in words a task that you do out of context? It
can be difficult to remember or to describe everything in the right order. Yet this is what you are
asking users to do when you bring them in away from their work and then expect them to describe
what they do. You can also miss important information if you do not see their artifacts (artifacts are
the physical objects they manipulate in their real environment, for example, forms, cards, folders,
maps). These artifacts contain important information about how users do their work. When users
are out of their own environment and context they may forget to mention some artifacts, or you may
not understand exactly what they mean by just having them described.

• **Too much detail.** When users are interviewed out of context it is hard for you to know if you are
getting the appropriate level of detail. You might be getting more than you actually need in order to
complete your task analysis.

• **Misunderstanding.** Because the interview is out of context it is easy for you to misunderstand some
of the information you are getting. For example, your user may say that she gets calls from
customers on the phone. You may not realize that the user doesn't actually take the call, but gets a
summary message.

Since interviewing users in your office and out of context has these problems, you may want to avoid this
technique in favor of the contextual methods described later, or at least combine this method with others.

### 3.2 Role Playing

Another way to get information from users is to role play with them. If they are customer service
representatives, pretend you are a customer and role play what their interaction would be like. This role
playing will yield some additional and more realistic information than just asking them questions about how
they would interact with a real customer. It is still a far cry, however, from actually observing a real
interaction, and will therefore have the same drawbacks as bringing users in for interviews.

### 3.3 Field Studies

Field studies involve going to the users' workplace and observing real users. Field studies are the most
powerful way to gather useful and important data quickly. They also allow you to see the context of peo-
ple's work, for example, the physical environment, amount of stress or distraction, noise and light levels,
and so on. There are always surprises when you go out and watch users in the field, and because of this it
is almost always worth the extra time and/or expense that field studies might entail. There are many
interface designers that say they cannot design a usable system if they cannot conduct at least one field
study.

Here are some considerations in planning and conducting a field study:

• **Decide on the purpose of your visit.** Decide ahead of time what you want to capture. Are you
looking for details on specific tasks the users perform? Are you verifying the user profiles? Do you
want to collect artifacts from the user's real world? Are you just trying to get a feel for the users and
the work they do? All of these are valid reasons for conducting a field study, but you need to get
specific about the visit so that you can prepare any materials you need to bring with you or bring
back.

• **Plan your interviews and data collection.** Given your purpose in visiting, how will you collect your
data? Are you going to conduct a formal interview? Have users fill out surveys? Have an informal
discussion? Watch as much as possible without interrupting? Although you should build in
unplanned time just to see what the user's world is all about, make sure you do some planning so
you can collect the information you really need. What do you need to bring with you? What do you
need to bring back from the user's site?

• **Make sure you are expected and preapproved.** Don't just show up on site and expect to be taken
care of. Make sure you have set up your visit ahead of time, including arranging with the users'
manager and scheduling time that works for the users with minimal disruption of their work. Also
make sure the tasks you want to see are actually going to occur during the time you plan on visiting.
Obtain any necessary written permissions ahead of time, for example, being able to videotape, or
take materials with you. Make sure the users know you are coming, not just the manager. If you can,
speak ahead of time by phone to the actual users you are meeting with.

• **Consider a partner.** Consider working with a partner during site visits. That way one of you can be
taking notes while the other one observes or interviews. Or one can be operating the video camera
while the other one interviews.

• **Consider video- or audiotaping.** Videotaping can be a big plus if you are alone since you don't have
to worry about missing crucial information, or reading your notes afterwards. Remember, however,
that it can take a long time to pour through video afterwards. Consider videotaping just some of the meetings, or videotaping the physical environment. Consider audiotaping instead. It is not as powerful as video, but it is easier to do and your users will not feel as intimidated by an audiorecorder.

- **Leave time.** Leave plenty of time for questions when you are on site. Make sure you have time between users to write down your comments when you are alone. Schedule enough time with the users to ask them questions on what you observed.
- **Consider role playing during the visit.** If you can't observe an actual task while you are onsite (for example no one calls that day with the kind of problem you wanted to capture the user responding to), ask the user to role play the task with you.
- **Make the users comfortable.** People are often nervous about being watched. Build rapport before you begin the session. Introduce yourself and your partner, explain what you are doing and why. Make sure they know they can stop or take breaks whenever they need to.
- **Just watch.** Sometimes it is tempting to start designing the new interface while you are watching the user use the old one. Stifle this desire. Your job during a field study is to observe and collect data, not analyze and design. If you have an idea, jot it down and then forget about it until later.

4 Document the Current Tasks

Documenting the current tasks involves looking at how users do their work now in enough detail to provide the data you need to design a new interface. Task analysis provides important clues to what the interface organization and conceptual design should be. Of course, where possible, the new system and interface should improve on how users currently perform their tasks. This new vision is documented in later steps (Describe Future Tasks and Create Use Case Scenarios). But first, it is critical to clearly understand the mental models users will bring with them to the new system. Users have built up certain ways of thinking about their work and you need to study and document these mental models, and take them into account when you design.

4.1 Describe each task

For each user task, document the following information:

- The actual task performed
- Tasks that precede, follow, or interrupt the task (task flow)
- Interdependencies with other tasks, flexibility of task orders
- The frequency with which the task is performed
- Which users perform task
- Information display requirements (What information do users see when performing the task?)
- Input requirements (What do users enter?)
- Task support (What other documents and tools do they use when performing this task?)
- Task products and where they go
- Common task performance problems, errors
- Terminology and concepts users use when describing and performing the task
- Users' complaints about how the task is performed today and their ideas about how task performance could be improved
- Characteristics of the work environment where the task is performed (for example, small, cluttered, dirty workspace that would make mouse use difficult)

Task analysis is about learning what users do, not asking users to design the interface. Task analysis only asks users for what they know best: their expertise on performing their work. The interface designer doesn't ask, "Do you need windows?" Through task analysis, the interface designer translates a requirement like, "I need to see these three kinds of information at the same time," into an interface design requirement for multiwindowed display.

Task analysis describes user tasks from the user's perspective, not the system functional or architecture perspective, and independently of any interface screen design concepts. Documenting the current tasks means describing how the work is done now, before the new system is in place.

4.2 Document the Current Tasks

You have several choices for how to document current tasks: task charts, task detail tables, and task sketches.

Task charts show in great detail the tasks that the users currently follow. One of the main reasons that new interfaces fail is because they don't meet the needs of complex users performing complex, real-world tasks. A common lack in current analysis practices (where they exist at all) is to oversimplify the picture of how users perform current tasks, and/or to not document all the tasks or all the contingencies. Although it is
important during the next phase, Design, to design with common and frequent flow in mind, that does not mean that exceptions or less frequent tasks or branches can be ignored.

It is critical, therefore, that your current task description be detailed and thorough. Task charts are good tools for beginning to capture that detail.

Figure 2.3 Current Task Chart for Balancing a Checkbook.

Figure 2.3 shows part of the Task Chart for Balancing the Checkbook from our checkbook case study. Task charts show a lot of detail, but the use of diagrams and boxes limits the amount of text that can be included. Task detail tables allow
<table>
<thead>
<tr>
<th>Task#</th>
<th>Task</th>
<th>Frequency</th>
<th>Display Requirements</th>
<th>Input Requirements</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Acknowledge Checks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Locate check</td>
<td>High</td>
<td>Check number Date</td>
<td>Navigate to check</td>
<td>Statement ordered by check number, users may search on other parameters. Do all banks use format: date-no.-amount?</td>
</tr>
<tr>
<td>1.1.1</td>
<td>Enter missing</td>
<td>Low-moderate</td>
<td>Entry feedback</td>
<td>Check number Date</td>
<td>Users will want placed in numeric order</td>
</tr>
<tr>
<td>1.2</td>
<td>Verify</td>
<td>High</td>
<td>Same as 1.1</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Indicate check cashed</td>
<td>High</td>
<td>Same as 1.1. Cashed</td>
<td>Cashed state</td>
<td>Users will need to remove as well</td>
</tr>
<tr>
<td></td>
<td>(etc.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2.4** Part of the Current Task Detail table for Balancing a Checkbook.

Figure 2.4 shows part of the Task Detail Table for the Task Chart you saw above. The columns in the task detail table allow you to note how often users perform the step (Frequency), what the user needs to see to complete the step (Display Requirements), what the user enters (Input Requirements), and very importantly, a comments column where you can annotate problems with the current task flow and ideas for changes as the new interface is designed. Figure 2.5 shows part of the Task Detail Table for Paying Bills.

In contrast to task charts and task detail tables, task sketches show a high-level view of the current flow of work for a particular group of users performing a task or set of tasks. Figure 2.6 shows the Task Sketch that describes just one common path through the task of Paying Bills.

<table>
<thead>
<tr>
<th>Task#</th>
<th>Task</th>
<th>Frequency</th>
<th>Display Requirements</th>
<th>Input Requirements</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Get checkbook</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Get bills</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Add up bills and compare with current funds</td>
<td>High</td>
<td>View current balance and total of bills</td>
<td>Add each bill amount, subtract from current balance</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Decide if there are enough funds to pay all bills</td>
<td>High</td>
<td>View current balance minus total of bills</td>
<td>Add new amount of bills, subtract from current balance</td>
<td>Tedious — calculate and recalculate the balance</td>
</tr>
<tr>
<td>1.5</td>
<td>If there aren’t enough funds, decide how much to pay on each bill to balance with current funds</td>
<td>Low</td>
<td>View current balance and total of bills</td>
<td>Add new amount of bills, subtract from current balance</td>
<td>Tedious — calculate and recalculate the balance</td>
</tr>
<tr>
<td>1.6</td>
<td>Write checks</td>
<td>High</td>
<td>View checks and bills</td>
<td>Write date, payee, numerical dollar amount, text dollar amount, signature, and note</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>Enter checks in register</td>
<td>High</td>
<td>View checks and register</td>
<td>Write check number, date, payee, and amount</td>
<td>Redundancy — write out checks and write them in register</td>
</tr>
<tr>
<td>1.8</td>
<td>Calculate new balance</td>
<td>High</td>
<td>View register</td>
<td>Subtract each entry from current balance</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2.5** Current Task Detail table for part of Paying Bills.

Notice that these sketches are high-level. They lack the detail of the task charts and task detail tables. They are useful, however, in summarizing the task at a high level, and also have the added advan-
Figure 2.6 Current Task Sketch for Paying Bills.

The advantage of showing in a visual way the environment and location of the user when tasks are being performed. All of these techniques can be used on manual tasks that are to be computerized, tasks that are part manual/part computerized now, or tasks that are currently computerized, and will be updated with a new system.

During your data gathering you should have collected artifacts, or real-life pieces of people's work. For example, you could bring back lists, reports, memos, an example file, anything that helps you remember and describe the user's actual work.
For Check-Ease data gathering we brought back samples of bank statements, checks, checkbook registers, and bills to be paid.

5 Document Problems and Opportunities

After you have documented how work is currently done, you are ready to describe the changes to be made. There are many sources for the changes that you want to incorporate, for example, information from feasibility work or problems or opportunities you uncovered while you were documenting the current task analysis. Figure 2.7 shows possible sources for problems and opportunities. Gather all these problems and opportunities in one place. You may want to group them into categories and put labels on the categories (for example, Performance Issues, Frustrations, and so on). For example, we found the problems and opportunities shown in Figure 2.8 for the current method of Paying Bills.

You want to make sure to use all the valuable information that you have collected. It is critical that you use the current state as the starting point when making decisions on how the future should be. In order to do this, you will want to connect the current state with problems and opportunities for change so you know where to consider making changes. Consider creating a map to the future by putting each change or opportunity on a sticky note and attaching it to the place on the current task sketch or task chart where it belongs. One problem or opportunity might come up in more than one place. Put them wherever they fit, as many times as is appropriate.

Figure 2.9 shows the current Task Sketch for Paying Bills with the problems and opportunities attached where they belong.

![Figure 2.7 Problems and opportunities can come from many different sources.](image)

<table>
<thead>
<tr>
<th>Problems and Opportunities List for Current Task of Paying Bills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can't find calculator, must add/subtract in head or with paper and pencil.</td>
</tr>
<tr>
<td>Current balance might not be up to date, have to stop and calculate.</td>
</tr>
<tr>
<td>Error prone—relies on correct calculations.</td>
</tr>
<tr>
<td>Tedious to calculate and recalculate the running balance.</td>
</tr>
<tr>
<td>Handwriting is bad, people may read wrong amounts on checks.</td>
</tr>
<tr>
<td>Redundancy—write check and then repeat all the information in check register.</td>
</tr>
<tr>
<td>Would be nice to have tracking of what has been spent for different categories (such as food, medical, and so on).</td>
</tr>
<tr>
<td>More work to get an updated balance.</td>
</tr>
</tbody>
</table>

![Figure 2.8 List of problems and opportunities for task of paying monthly bills.](image)
Describe Future Tasks

Up to this point, your analysis activity has been aimed at understanding and documenting the users and their current work. Now, however, it is time to start focusing on the future. At this point you should gather everything you have so far about the users, the project, the constraints, the results of your task analysis, and the problems and opportunities map. You are ready to create new task flows that describe the future. One thing you will have to decide up front is how much of your future task description is going to be an ideal
world scenario, and how much is going to conform to given constraints. You should decide this point of view before you describe the future tasks so that you will not spend time on systems that cannot be implemented.

Describing the future tasks is similar to documenting current tasks. The best tools for describing future tasks are similar to some of the ones you used to describe current tasks. Task charts or task sketches can both be useful. Usually we do not use task detail tables to describe future tasks—by the time we are ready for that level of detail about the future we are ready to move on to use case scenarios (described below).

Figure 2.10 shows part of the future Task Sketch for the Check-Ease Paying Bills task.

How does documenting current and future tasks differ from business process reengineering? Business process reengineering (BPR) makes decisions about how the business will operate, for example, how an order will move through from inception to fulfillment. BPR often fails to take these decisions through to the level of how an individual or group of people in a certain job will carry out specific tasks. Task analysis takes BPR into account, but goes into more detail in describing exactly how users will do their work. If BPR is happening on your project, you will want to have all possible data and decisions from the BPR group before and during your task analysis work.

7 Develop Usability Specifications
Once you have analyzed current and future tasks, you are ready to specify what you mean by usable or user-friendly for this interface.
These are your usability specifications. Usability specifications provide a powerful means for pinning down and communicating what usability, ease of use, and user-friendly really mean for a system and its interface. For example, for a given project, is it more important that the interface provide ease of learning or ease of use? How important is it that the new interface be compatible with the user interface that the users are familiar with now? Usability specifications summarize key user, training, and documentation assumptions in a format that is objective, measurable, and testable.

Usability specifications should be defined at the start of every project. If you can't clearly define what user-friendly means for the system, how will you know when you have achieved it? And how will you
prioritize feature and interface design work?
Usability specifications can also provide a typically much needed common vision for the new system. When there are heated arguments about an interface design among project participants, often, underlying these arguments, are very different visions of the usability goals for the system. For example, marketing is concerned because informal testing shows users can't immediately perform some complex task using the system (usability goal = walk-up-and-use), but the interface designers assumed users would be trained and use the product all day, every day (usability goal = flexibility).
To develop usability specifications, first identify three to ten key usability attributes for the system. Consult with project management, marketing, and users to identify these attributes. Examples of usability attributes are:
- Ease of learning
- Rapid task performance
- Accurate task performance
- Perceived ease of use

Good questions to ask when looking for key usability attributes are: What problems is the new user interface supposed to solve? Is the interface suppose to improve speed or accuracy of user performance on key tasks? Or to reduce required time to train users? Or to improve user evaluations of system ease of use?

Once you have identified usability attributes, you can develop usability specifications. The three steps in developing specifications are detailed below.

1. For each usability attribute, identify how the interface could be measured on that attribute. Examples of usability measures are:
   - Time to train to a performance criterion
   - Time to complete a task
   - Number of errors in completing a task
   - Percentage of users that successfully complete the task
   - Time to relearn to criterion
   - Time spent correcting errors
   - Average ease-of-use rating given by users surveyed/interviewed

2. For each usability measure, specify what constitutes success. For example:
   - 20 minutes
   - Two errors
   - Fifty percent rate 8 or better on scale of 10

You may want to specify criteria for both the minimum acceptable level of usability and for the target level of usability.

Good places to look for usability criteria: What is clearly unacceptable performance? What is current performance that must be improved upon? How does the competition perform? Be careful not to select unrealistic usability performance criteria; for example, "One hundred percent of users can perform task A with 100% accuracy." Usability specifications are a tool to clarify real project goals and measure real system performance.

3. Include other key elements.
   - Make sure that the usability goals that are written include all assumptions about who users will be (user profiles) and how much preexisting knowledge and training they will be expected to have.
   - Be sure to get the participation of project management, development, training, and documentation in writing and reviewing usability goals.

Figure 2.11 shows some usability specifications for the Check-Ease interface.

Another way to document usability specifications is in a usability specifications table. Figure 2.12 shows a sample table for the Check-Ease application. The specifications still have the same components. If you are not familiar with writing usability specifications, this format may be useful.
Usability Specifications for Check-Ease

Ease of Learning
With no previous training and using only online help and documentation, 90% of adults who read and write English, currently have a checking account and currently use other MS Windows applications, can start the Check-Ease application, open a sample checkbook, and record a check in less than 15 minutes the first time.

Ease of Use
After having correctly started the Check-Ease application and recorded a check at least three times, 75% of adult users can do this task in five minutes or less (IBM 386 PC or PC clone).

Ease of Learning
After completing a short (20 minutes or less) tutorial, and using online help and documentation, 75% of adults who currently have a checking account, do or do not balance their checkbook, currently use other MS Windows applications, and read and write English can correctly balance a sample checkbook.

Ease of Learning
Seventy-five percent of adult IBM PC (or clone) users with MS Windows experience and who read and write English can successfully set up the Check-Ease modem features in 20 minutes or less. Ninety percent can do this in 40 minutes or less.

Ease of Learning
After a 20-minute tutorial, and using only online help and documentation, 90% of adults who read and write English, currently have a checking account and currently use other MS Windows applications, can pay bills in less than 30 minutes the first time.

Figure 2.11 Usability Specifications Narrative for Check-Ease.

\begin{table}
\begin{tabular}{|l|l|l|}
\hline
Measurable Behavior & Criteria & Key Elements & Users \\
\hline
Pay 10 bills & Within 30 minutes the first time and 20 minutes subsequent times & Online help and 20-minute tutorial are available, and tutorial is used & Read English, use Windows, have an account \\
\hline
\end{tabular}
\end{table}

Figure 2.12 Usability Specifications Table for Check-Ease.

8 Develop Use Case Scenarios
A use case scenario is an outline of tasks and subtasks that describes how users will do their work. The purpose of the use case scenario is to aid in conceptual design. It must describe how users will do their work with the new software so that the correct flow of screens can be developed.

How is a use case scenario different from future task descriptions? Parts of the future task description assume a new software solution, but not necessarily all of the future tasks involve an interface. Scenarios detail only the tasks that involve the interface you are going to design. They are the link between the future task description and the design of the interface.

The use case scenario is not a reengineering document, though it may reflect reengineered processes. The use case scenario describes how the users will do their work when the new software is in place—not how the work is done now, or what the problems are with how it's done now. Another major difference between business process reengineering documents and use case scenarios is that use case scenarios document users' tasks. Business process reengineering documents usually describe a business process, for example, how an order moves through the department, and don't necessarily describe it from the user's point of view.

Sample use case scenario for our Check-Ease system is shown in Figure 2.13.
To create effective scenarios, follow these guidelines:

- Write from the user's point of view, not the system's point of view. In order to match the way the users should be doing their work,
Use Case Scenario:

Paying Bills with Printed Checks

1. Enter bills (payee name and amount due) on a worksheet.
2. View the new balance (current balance less each payment amount).
3. Decide if there are enough funds to pay the bills.
   a. If yes (80% of the time), go to step 4.
   b. If no (20% of the time), mark the bills not to be paid (80%) or change payment amounts (20%), then go to step 4.
4. Tell system to pay the bills.
5. Set printing options for checks.
6. Put check stock in printer.
7. Print checks.
8. Look at checks.
9. Reprint checks if there is a problem (problem with checks 30% of the time).
10. View updated checking account register (viewed 50% of the time).
11. Run reports (50% of the time). Note: see separate scenario for detail on running reports.

Figure 2.13 Use Case Scenario for Paying Bills using Check-Ease.

the scenario must be a list of user tasks. There is a tendency to start to describe what the system is going
to do. If you really want a system description in your scenario, create a parallel scenario for the system.
Start first with the user's tasks and when the scenario is complete, go back and write what the system is
doing at each step.

- Make sure you start with the user's tasks. In order to be able to use the scenario to create a
  conceptual model and interface screens you must have a listing of user tasks.
- Include frequency information. In order to create the best flow of screens later on you must
document information on frequency of tasks in the scenario. If there are alternative paths or tasks,
or decision points, you need to decide how frequently each path is likely to be taken. Will users be
processing a new order most of the time or working with an existing order? This frequency infor-
modation is critical if the scenario is to be effective. What users do most should be the easiest to do.
Your design decisions and tradeoffs come in large measure from this frequency information. For
example, if your scenario indicates that working with an existing order occurs 80% of the time, then
during design you should start the screen flow for that task with a list of existing orders. If however,
your scenario indicates that starting a new order occurs 80% of the time, you should start the
screen flow for that task with a blank order, ready to be filled in.
- Make note of exceptions. If you are documenting frequency, then you will know as you go along
which parts of the scenario are describing exceptions. Exceptions should be noted, and eventually
need to be filled in, but make sure you have described all the common, frequent, and critical paths
before going back to document exceptions.
- Make note of critical tasks. Some tasks are frequent. Some are infrequent. But another aspect of
tasks is criticality. It is possible to have a task that is infrequent, but critical. All critical tasks should
be noted on the scenario, especially those that are infrequent, but critical.
- Write in words, not just diagrams. Diagramming tasks can be powerful, but if you are not familiar
with the notation or abbreviations they can be hard to understand. It is very important that everyone
on the scenario team be able to read and understand the scenario once it is completed. Also
important is that people not on the scenario team be able to easily read and understand the
scenario. If a person has the correct content matter expertise, he or she should be able to read your
scenario without much interpretation. This means the scenario will be more clear and easily
understood if it is written using words rather than notations. If you insist on using notation, you
should also prepare a version with words only in outline form.
- Describe the future. A scenario does not describe the user's tasks now, but the tasks they will
perform when the new system is in place. It is easy when creating a scenario to fall into describing
the current situation, but avoid doing that unless the new is exactly the same as the current. You
should instead be documenting in detail how the user will work with the new system.

8.1 Interface Scenarios Versus Other Forms of Use Case Scenarios

Use case scenarios are discussed a lot recently because of the use of the term in object-oriented
programming methodologies by Ivar Jacobson and others. In some ways we are using the term in
the same way as it is used within a development methodology, and in other ways we have modified the
definition.

Use case scenarios for interface design need to be focused on the user's actions as they interact with the
new software. Use case scenarios as we have seen them used by development groups sometimes contain
this information, but it is often woven into a description of what the software is doing behind the scenes.
Figure 2.14 shows an example of a scenario written from a systems point of view.

The description of software actions is critical for software developers, but is at best a distraction for
interface designers, and at worst, can cause new interface designers to veer away from true interface design, and move inappropriately into designing the underlying software. It is reasonable, perhaps critical, that both interface designers and the software development group be working from the same use case.

<table>
<thead>
<tr>
<th>System Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>The user enters each bill (payee name and amount) into the system. The system shows a running balance, starting with the current balance, less the amount of each bill as it is entered. The user can deselect a bill or change the amount to be paid. The system recalculates as changes are made. When the bills are processed for payment, the system prints out the checks, updates the checking account register, and recalculates the balance.</td>
</tr>
</tbody>
</table>

**Figure 2.14 System Scenario.**

**Figure 2.15 Shared Use Case Scenario.**

scenario. Figure 2.15 shows how both interface design and software design could come from a common scenario. To ensure that both sides have a usable scenario, you could create two separate, but matching scenarios, one for the user actions and one for system activities. That way, both groups (interface designers and developers) are getting the same scenario, but each group can get the information it needs. Each group also has the advantage of seeing (if desired) what the other group is working from. Figure 2.16 shows an example of separate but matching scenarios. Another approach is to first create the interface design version and then have the software group add what they need for the software development side. What is important is that those responsible for the next phase of interface design have a version to work from that has what they need (user tasks from the user's point of view), and that if the development team is also using scenarios, that the scenarios cover the same actions.

### 8.2 Holding Use Case Scenario Sessions

Before you create use case scenarios, the major business decisions need to be made. You don't want to be discussing business decisions.
Use Case Scenario: Paying Bills with Printed Checks

1. Enter bills (payee name and amount due) on a worksheet.
2. View the new balance (current balance less each payment amount).
3. Decide if there are enough funds to pay the bills.
   a. If yes (80% of the time), go to step 4.
   b. If no (20% of the time), mark the bills not to be paid (80%) or change payment amounts (20%), then go to step 4.
4. Tell system to pay the bills.
5. Set printing options for checks.
6. Put check stock in printer.
7. Print checks.
8. Look at checks.
9. Reprint checks if there is a problem (problem with checks 30% of the time).
10. View updated checking account register (viewed 50% of the time).
11. Run reports (50% of the time). Note: see separate scenario for detail on running reports.

System Scenario: Paying Bills with Printed Checks

7. System shows data for each bill: payee, due date, and amount due; system shows current balance in checking account.
2. System recalculates the balance as each bill is entered, if the bill is to be paid.
3. Show changed amount if user changes amount to pay. Mark as unselected if user chooses not to pay a bill.
4. System enters bill as an entry in the checking account register and recalculates the current balance.
5. Displays options for printing checks.
6. None.
7. System prints checks.
8. None.
9. Alert user about printing status. Allow user to enter which checks to reprint if necessary.
10. Show updated checking account register.
11. See separate scenario for running reports.

Figure 2.16 Separated Scenario.

during the use case scenario sessions, because this will slow you down, and the people able to discuss and decide on those issues may not be present. Discussions on such things as the new software's scope and performance should have occurred while documenting current and future tasks. Items that have not been discussed or decided upon will have to be dealt with before you can continue with your use case scenarios.

If you have not documented the tasks, your use case scenario sessions will be slow and difficult, because you will have to do the documenting of current and future tasks as you go along. Below are tips for running a useful and efficient session.

- Remember the definition of a use case scenario. Stay focused on the future and don't lapse into describing how the work is done now. Document the user's actions, not what the system is doing.
- Keep in mind the level of detail you need. You need enough detail to sketch screens later, but you don't need to decide exactly how many characters are in the name field.
- Know where to start and what to include. Document the most important and frequent tasks first. Make sure you capture the frequency and criticality of each task.
- Pay attention to the environment. Make sure you have enough white boards. Meet in a comfortable room with good lighting, ventilation, and noise control. Pick a place where people won't be interrupted or called away.
- Give the team two to four hours per session. Less than that, you can't accomplish enough. More than that, everyone gets too tired.
- Make sure everyone wants to be there. Having someone on the team that doesn't want to participate will make it difficult on the rest of the team and will slow you down.
- Be willing to give the facilitator power. The facilitator's job is difficult. Everyone on the team must be willing to be guided.
- Keep track of questions that arise and need clarification or answers. Decide before the session ends who is responsible for getting answers to the questions.

You will need members of your team to take the following roles:

1. Facilitator. The facilitator is a critical role. The facilitator keeps the group on track. The facilitator needs to be impartial: no hidden agendas, just a desire to accurately capture in detail what the user tasks are, and what they should be on the new system. Critical facilitator skills and actions include:
   - Knowing when to keep asking questions versus when to just write something down and come back to it later
   - Writing down enough detail to capture the information, but not so much as to slow down the process
• Being willing to erase and redo as new information emerges. Knowing when to put something on an "issues to be resolved" list and move on.
• Making sure that the group stays focused on capturing information and not sketching screens.
• Not letting any one person dominate conversations.
• Knowing when the group is tired and needs a break.
• Knowing when a particular task is done and it is time to work on another.

If someone on the team is experienced at group facilitation, you may want to pick that person for this role. The interface designer can be the facilitator. Content experts/users should not facilitate. The content experts/users need to be free to think and do not have time to facilitate. The facilitator does not need to be a content expert. Familiarity with the subject matter helps, in order to ask meaningful questions, but it is often better that the facilitator not be an expert, since lack of knowledge can lead to some good questions.

2. Scribe. Someone should be assigned the job of making sure the work gets captured and documented when complete. When the team agrees they are done with one task, the scribe should record the task so that the board can be erased for the next one.

3. Other members. The role of other members in the group is to listen and ask questions and clarify.

Technical people or interface designers are not the source of main information during analysis. They are there to listen, ask questions, clarify, and experience the discussions.

9 Test

The results from the Analysis phase are critical—they are the basis for the next phase where actual screens, dialog boxes, menus, and windows are created. Therefore, it is important that you test and verify the decisions you make during analysis.

There are many test activities that go on during the Analysis phase:

• The user profiles need to be validated by either going into the field and making sure they are accurate portrayals, and/or showing them to users and their managers for verification.
• It is critical that current and future task descriptions be verified with users, their managers, and any other key stakeholders.
• Usability specifications need to be verified and agreed to by users, their managers, and key stakeholders.
• Use case scenarios need to be verified with users, and their managers. If the software developers are using scenarios for their architecture design, then they need to verify and agree to the scenarios as well.

You may want each of these tests or verifications to be a formal step with sign-offs. If so, you will need to determine exactly who should be signing off on each one—it might vary slightly, for example, while the software developers might be helped by seeing user profiles, the use case scenarios are critical to their work.

Summary

Analysis ensures that your interface design decisions fit the users and their work. Figure 2.17 is a checklist to follow when performing analysis.