# Chapter 2: Task Analysis

## Objectives

Upon completion of this chapter, readers will be able to:

1. Distinguish between function-oriented and task-oriented documentation.
2. Perform a task analysis by studying a product’s user interface to identify key tasks.
3. Apply task-oriented phrasing to headings and subheadings to improve clarity for the user.
4. Construct step-by-step instructions using numbered lists to guide a user through a process.

## Task Analysis and Task-Oriented Documentation

When you write instructions, procedures, and "guide" or user-guide information (generally called documentation), you normally must use a task approach: providing steps and explanations for all the major tasks that users may need to perform.

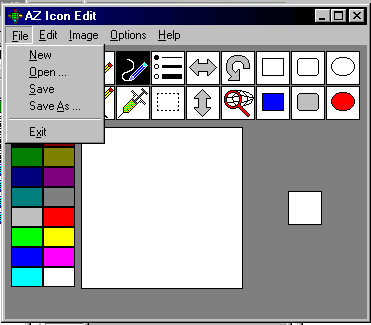
Of course, some instructions involve only one task--for example, changing the oil in a car--but we are concerned here with more complex procedures. While this chapter uses computer software as an example, these techniques can apply to any multi-task procedure—for example, operating a microwave oven.

## Identifying Tasks

To write in a task-oriented manner, you must first do some task analysis: studying how users use the product or do the task, interviewing them, and watching them. It can also mean interviewing marketing and product development people. If you can get your hands on the kinds of questions that help-desk people receive, that helps, too.

But sometimes, you may not be able to do a thorough task analysis. Typically, product developers don't think about documentation until rather late in the process. In these circumstances, it's often difficult to get marketing, development, engineering, and programming people to spend enough time with you to explain the product thoroughly. Therefore, you end up doing a certain amount of educated guesswork. The developer is more likely to review your draft and let you know if your guesswork is right.

To develop your own task analysis, you can study the user interface (buttons, menus, options, etc.) of the product. This process goes for both hardware and software. Consider the interface for an icon editing tool shown below:

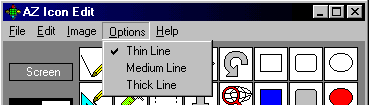
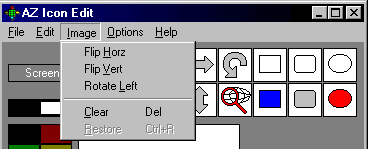
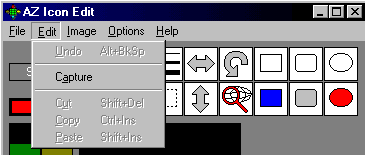


*Figure 1: Icon editing tool interface*

From just this snippet of the interface, you can identify several obvious tasks:

* Start a new icon project.
* Open an existing icon project for editing.
* Rename an icon project (Save As).
* Exit AZ Icon Edit.

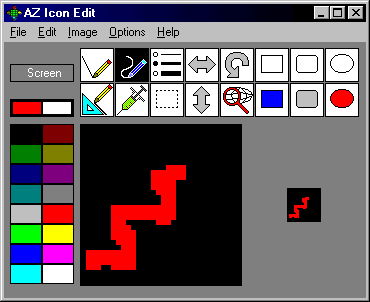
Now, look at the menu options for the next parts of the menu. You can see that when people are using this icon editor, they'll also most likely be doing these tasks:



*Figure 2: Icon editing tasks*

* Undo a mistake.
* Capture an image or some part of it.
* Cut something out of an icon project.
* Copy something out of an icon project.
* Paste something into an icon project.
* Flip the entire image horizontally or vertically.
* Rotate the image left.
* Clear the project, which probably means start over.
* Restore, which you'll have to ask around, experiment with, or dig into the programming spec to find out about.
* Draw with a thick, medium, or thin line.

Now look at the interface without the menu options hanging down. What additional tasks can you see? As with a lot of graphical user interfaces, some of the icons duplicate the menu options. For example, the bulleted-list icon enables you to select a thin, medium, or thick line the same way clicking on **Options** does. However, there are some new tools here, not available elsewhere in the interface:



*Figure 3: Icon editing tools*

* Draw straight lines (you'll have to experiment to see the difference between the two pencil icons).
* Draw freehand lines (the wavy-line icon).
* Draw unfilled rectangles (sharp edges) and unfilled rectangles (rounded edges).
* Draw unfilled ovals and filled ovals.
* Fill with color (the hypodermic needle).
* Select portions of the image to move, cut, or rotate (the dotted-line icon).
* Capture images—or parts of images— (the net, but how does it work?).
* Draw filled rectangles (sharp edges) and filled rectangles (rounded edges).
* Select background color (the Screen button).
* Select line or fill color (the double-box icon).

You may not know everything about this software, but you've already done some guesswork toward defining the major tasks. Group and consolidate things more tightly, like this:

* Creating, editing, renaming, and saving icons
* Selecting foreground and background colors
* Drawing lines, rectangles, and ovals
* Cutting, pasting, and copying objects
* Moving, flipping, and rotating objects

In this rough task list, there is no trace of tasks like filling an object with color, capturing images, clearing the workspace, undoing a mistake, or restoring. However, as you work, these details will begin to find their place in your scheme. Now, stand back from the details of the interface and put yourself in the place of an icon software user. Here are some questions an individual is likely to ask: How do I change the color of the background? How do I change the thickness of the lines I draw? How do I make the background transparent? The answer to these questions will take some research; anticipating the need for answers is the key to good task analysis.

## Different Approaches to Documentation

When writing for users, choose task orientation over function orientation. Function orientation lists buttons or icons with their functions, like “The save button saves your project.” This is helpful, but it doesn’t help users use the software effectively. Task-oriented documentation, created for specific user goals, allows users to quickly start using the product and achieve their goals, leading to higher satisfaction with the product and documentation.

### Writing with a Function Orientation

It should be obvious how to proceed after a task analysis, but oftentimes it is not. Computer publications and technical publications often stray into a non-task-oriented style of writing, which just doesn't work!

Another reason why some user guide instructions are not task oriented can be blamed on product specifications. Product specifications, which are written by and for programmers, engineers, developers, are written in terms of required function:

| **Button** | **Function** |
| --- | --- |
| File menu button | Enables user to create a new file, open an existing file, rename a file, etc. |
| Crop icon | Enables user to cut selected segment of image. |

This approach is also called function-oriented writing because it systematically explains each function, feature, or interface element of a product. Unfortunately, this approach shows up in user guides meant for nontechnical readers—perhaps because the writers are inexperienced, untrained, or untechnical; or else the writers have been called in too late to provide their expertise and instead, polish the developers' spec.

The function-oriented approach almost works sometimes. But "almost" and "sometimes" are not good enough. It almost works because the names of interface elements and functions sometimes match the tasks they support. For example, the **Open** menu option is intuitive: open an existing file. Others are not. For example, what do you suppose is restored by the **Restore** button in the AZ Icon Edit interface? Also, some interface elements don't accomplish tasks all by themselves. In Photoshop, for example, you can't crop text by pressing the **Crop** menu option. You must first click the text-selection button, then draw a selection box around the part of the image you want to keep, then press the **Crop** button.

### Writing with a Task Orientation

Instead of the function-oriented approach, use the task-oriented approach. Identify the tasks users will need to perform with the product, and then structure your document accordingly. Make your headings and subheadings task-oriented in their phrasing. Task-oriented phrasing means phrasing like "How to adjust the volume,” "Adjusting the volume," or "Adjust the volume." It does not mean phrasing like "Volume" or "Volume Adjustment." Here are some additional examples:

| **Problem phrasing** | **Task-oriented phrasing** |
| --- | --- |
| Capture | Capturing images |
| Screen button | Selecting background or foreground objects |
| Rectangles | Drawing rectangles |
| Oval icon | Drawing ovals and circles |

When you have defined user tasks, organized them into logical groups, and have defined task-oriented headings, you're ready to write! Here's an excerpt:

**Drawing Rectangles and Ovals**

You can use the icon editor to draw squares, rectangles, ovals, and circles.

**Draw a rectangle.** To draw a rectangle:

1. Ensure that you have selected the foreground color you want. (See “Selecting foreground color.”)
2. Click the rectangle icon.
3. Position the mouse pointer in the drawing area where you want the rectangle to appear, hold down the left mouse button, and draw to create the rectangle.

**Draw an oval.** To draw an oval:

1. Ensure that…

In this excerpt, you can see that an overall task-oriented approach is taken and that task-oriented phrasing is used for the headings (Drawing). Notice, too, that numbered lists are used to guide readers step by step through the actions involved in the task.

[View example documentation on creating alternative text.](https://alg.manifoldapp.org/read/alternative-text/section/5e9c2e43-2186-424c-82ac-416ed7dc60df)

Alternative text is added to images in documents to help readers using screen readers in accessing the information in a document.

## Attribution

This chapter is revised from the first edition of *Open Technical Communication*, Chapter 5.3: “[Task Analysis](https://alg.manifoldapp.org/read/open-technical-communication/section/164f489a-6e37-4792-96eb-6c266cb5fe20)” by David McMurrey and Tamara Powell, which is openly available under a Creative Commons Attribution license.

The content in Chapter 5.3 of the first edition of *Open TC* was originally sourced and revised from David McMurrey’s *Online Technical Writing*, section titled “[Task Analysis and Task-Oriented Documentation](https://mcmassociates.io/textbook/task_analysis.html),” which is openly available under a Creative Commons Attribution license.

## AI Assistance Notice

Some parts of this chapter were brainstormed, drafted, and/or revised in conversation with ChatGPT 4o and Google Gemini 2.5 Flash. All AI-generated content was reviewed and revised as needed by a human author.