3-1-2016

26. Task Analysis and Task-oriented Documentation

David McMurray

Tamara Powell
Kennesaw State University, tpowel25@kennesaw.edu

Sexy Technical Communication Home

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. That means 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.

Chapter Objectives

At the end of this chapter, students will be able to

1. Define documentation

2. Identify and analyze tasks in order to create documentation

3. Differentiate between function and task orientation and explain the pros and cons of each approach

4. Explain how to begin writing documentation

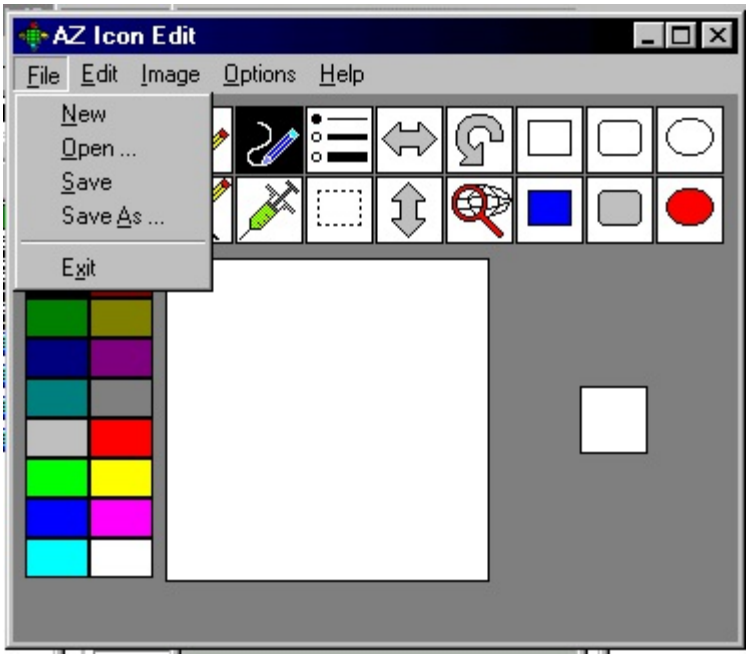
Identifying Tasks

To write in a task-oriented manner, you first have to do some task analysis. That means 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 in a position to do a thorough task analysis. Typically, product developers don't think about documentation until rather late. 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. And so 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.

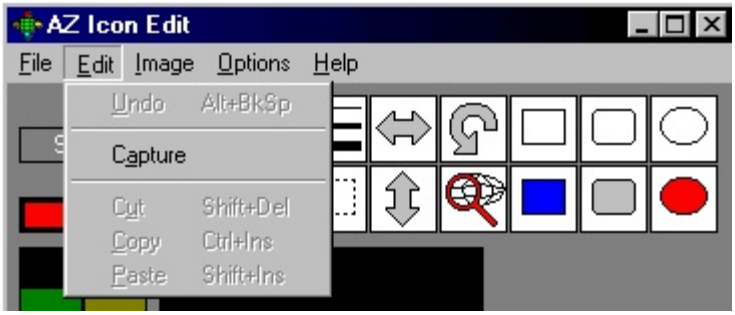
To develop your own task analysis, you can study the user interface (buttons, menus, options, etc.) of the product. This process goes for both hardware and software. Consider the interface for an icon editing tool shown below:

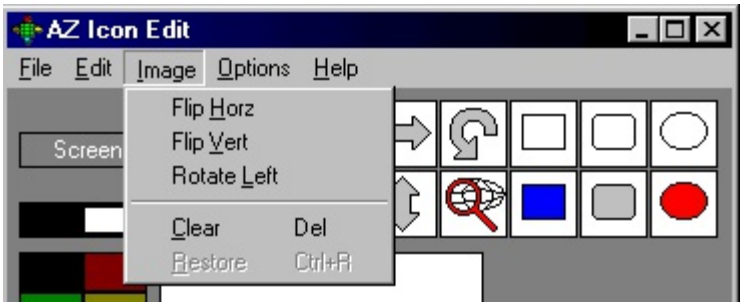


From just this snippet of the interface, you can identify several obvious tasks:

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

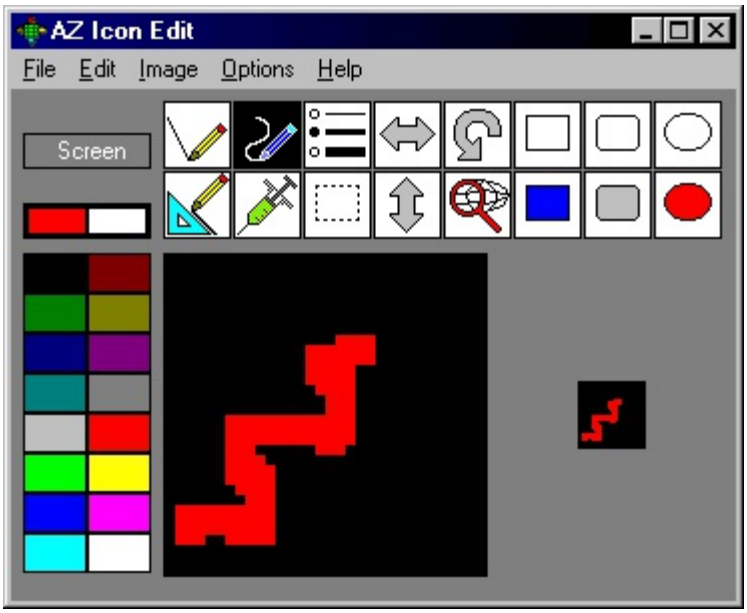
Now, look at the menu options for the next parts of the menu. You can see that when people are using this icon editor, they'll also most likely be doing these tasks:





- 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 horizontaly 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.

But 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:



- 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)

There's a lot you still don't know about this software, but you've already done a lot of guesswork toward defining the major tasks. You'd want to group and consolidate things much more tightly than above, perhaps like the following:

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

You can see that in this rough task list, there is no trace of tasks such as filling an object with color, capturing images, clearing the workspace, undoing a mistake, or restoring. But 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. What questions is that individual likely to ask? How do I change the color of the background? We've got that covered. How do I change the thickness of the lines I draw? Got that one covered too. How do I make the background transparent? Hmmm . . . that will be an issue

for the color section, but it will take some research.

Different Approaches to Documentation

When you write for users, you have a choice of two approaches, *function orientation* and *task orientation*, the latter of which is by far the better choice. A function orientation lists buttons or icons and then lets readers know what the function of each item is. For example, "The **save** button allows you to save your project for later use." This information is helpful for a user (although probably most users know what the save button does). While it is helpful to quickly list major buttons and what they do, it's not sufficient to help readers truly use the software or appliance. Task-oriented documentation, created for specific goals that you anticipate users will want to achieve (such as, "Capturing Images") allows users to begin using the product quickly and achieving their goals satisfactorily (which hopefully leads to a high level of satisfaction with the product and your documentation).

Writing with a function orientation.

It ought to be obvious how to proceed after a task analysis, but apparently not. Computer publications—if not technical publications in general—often seem to stray into a non-task-oriented style of writing. But, no! That 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:

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.

You might call this approach *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 do much else but 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 pretty 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 have to 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 draws 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 to the rectangle to appear, hold down the left mouse button, and drag 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.

Click here to view sample documentation on creating checklists with Desire2Learn (D2L). D2L is a learning management system that university faculty might use to share class

materials with students. The checklist function helps university faculty to let students know what tasks need to be completed within a course module or unit of time.