Writing Tutorials That Actually Help Users

Communication for Learning

By Peter Vogel

The documentation for a user interface often consists of a detailed listing of what each component of the interface does. Unfortunately, that documentation doesn't provide what users need since most of the users' tasks require working with multiple components of the UI. What users actually need is end-to-end guidance that starts at the beginning of their task and finishes when they achieve their goals; in other words, a tutorial. Sadly, most tutorials are written in a way that prevents users from being able to take advantage of them.

Tutorials make the functionality provided through the UI more "real" by linking the UI to the user's goals and the actions that need to be performed. When users need to use a software application to accomplish a goal, they often begin by finding a tutorial that achieves that goal and then work through it, modifying the instructions in the tutorial to meet their needs. If you think about the last time that you used a set of written instructions (even if it wasn't called a tutorial) this probably reflects your own experience. In this respect, at least, you are a typical tutorial reader.

Selecting Tutorial Content

Users will only take time out of their lives to work through a tutorial when they need the tutorial to meet one of their own goals. There is no point in writing a tutorial that allows the user to "experience" some feature of the application or whose purpose is to demonstrate a piece of technology. That means your first step is determining the audience for your tutorial—who your users are. Your second step is to determine your user's goals. Only after you determine who the audience is can you determine what their goals are.

Be careful not to substitute your goals for your users' goals. It's unlikely that most of your users want to become experts in using your application. It's more likely that your users' goals fall into the categories of "getting my job done in a reasonable period of time," and "not looking (or feeling) stupid." If your users are going to interact with your application infrequently—only once a month, for instance—they're not going to be interested in learning or remembering any shortcut keys. Only if your users interact with your application frequently is it likely that they will be interested in becoming more efficient using it.

After determining your audience and their goals, your third step is to determine which of those goals will require your support, and for which of those goals the audience will seek out a tutorial. For tasks that seem "intuitive," the user may choose the "fumble around" strategy as the best way to achieve goals, rather than reaching for a tutorial. Experts may also choose to leverage existing knowledge rather than seek out instruction. (Experts would rather be caught dead than caught reading help information.) When a user is in an office, surrounded by other users, he or she will find out how to perform common activities by asking neighbors.

After you've determined your audience, its goals, and which goals can be supported by a tutorial, you're ready to start writing.

"If you think about the last time that you used a set of written instructions (even if it wasn't called a tutorial) this probably reflects your own experience. In this respect, at least, you are a typical tutorial reader."

Designing Tutorials

A tutorial falls into three sections: an introduction, a set of steps, and a short review. The introduction, which may be just a title, describes in the user's terms, the goal that is achieved by following the steps in the tutorial. It's the introduction that allows users to find the tutorial that they need. The review at the end of the tutorial, often a single line, reminds users of the goal that's been achieved and prevents them from forgetting as they fight off the alligators, why they started out to drain the swamp.

The individual steps are more complicated. Like a good UI, the steps in a tutorial provide feedback. At the very least, tutorial feedback allows users to orient themselves. For instance, many tutorials have steps like this one.

Open the wizard. Click Next. Click Next. Click Next.

By the second or third click, users won't be sure how far into the wizard they've come. Feedback, in the form of the titles on wizard steps, allows users to determine which step they're on.

More importantly, feedback allows users to determine if they are succeeding or failing. Feedback needs to provide users with easily recognizable signposts that allow them to determine if they're on the right track. Since users will be modifying the steps as they work through the tutorial, the signposts have to be invariant across all the modifications that a user may make to the steps. You may, for instance, write a tutorial for your application that makes use of some sample data that you've provided with your application. As you write your tutorial, you'll use that sample data in your steps:

To display the information for customer A123, select customer A123 from the list on the screen.

Users, however, will not be taking advantage of your sample data. Instead, they will be following your tutorial and using their own data. Providing the feedback, "Customer A123 will appear at the bottom of the form," therefore, won't be helpful. More useful feedback would be: "The information for the customer you selected will appear at the bottom of the form."

Writing Steps and Handling Errors

The steps themselves need to support users making modifications. While your tutorial may refer to specific data—creating a sales order for customer A123 for instance—your users will be creating a sales order with their own choices. To support the user, most steps must begin with a description of the purpose of the step: how this step moves users closer to their goals. With that information, users can make intelligent decisions about what modification they should (and can) make to the step.

For instance, the step "Click on customer A123 in the Customers window," doesn't support letting the user make modifications to the tutorial's step. The improved step:

Now you can choose the customer to assign to the sales order by selecting the customer from the Customers window. This examples uses customer A123.

As you probably know from your own experience, error happens. You're not a professional writer, so it's to be expected that some of your steps (however much sense they make to you) won't make sense to your readers. Even with the best writing in the world, readers will make mistakes; they will misread a step, fumble implementing a step, or misunderstand the feedback they get from the UI.

"…feedback allows users to determine if they are succeeding or failing. Feedback needs to provide users with easily recognizable signposts that allow them to determine if they're on the right track."

The solution to these errors is not to add more writing. In fact, more writing is more likely to hide the point of a step than to reveal it, and to create more errors than to eliminate existing ones. The correct solution is to diagnose the typical errors that users can be expected to make, and provide feedback that will allow the user to recognize the error. Following the diagnosis, you can provide the reader with instructions on how to correct the problem. Here's an example of a diagnosis followed by the corrective action.

I get the message, "Duplicate key in table OrderLines."

This message appears when you have added the same product to your order two or more times.

Scan the list of products on the order to see which product (or products) appears more than once. Delete any duplicate entries until each product appears only once. You can increase the quantity on the remaining line to order the correct number units of the product.

If you don't have the luxury of usability testing, you may find that you won't be able to determine what the typical errors are until your application has been released to the field. You may need to write an initial version of your tutorial where you'll guess at the typical errors that users will make and then revise your tutorial after release, once you determine what the typical errors for your users really are. Only at that point will you have finished writing a truly effective tutorial.

The secret to writing an effective tutorial is to recognize that users treat your tutorial as a cookbook, not a contract. And, like cooks, users will modify your instructions to take advantage of whatever ingredients are available to them and what they prefer to put on the table. To support that process effectively, first identify who you're writing for, what their goals are, and whether they'll even use a tutorial to meet those goals. Then write a recipe that supports user modifications and guides them through errors so that, using your UI, they can achieve their goals. **UX**

About the Author

Peter Vogel is a software developer, user interface designer, technical writer, and instructor. His company, PH&V Information Services, is entering its fifteenth year. Clients include Volvo, the Canadian Imperial Bank of Commerce, and Microsoft. Read his blog on technical writing at http://rtfmphvis.blogspot.com. In addition to helping clients empower users with effective user interface designs, Peter teaches UI design for Learning Tree International.