**WIZARDS**

**OTHER NAMES:** Stepping

---

**Figure 1- Proposed wizard in Horizon Plan of Care Editor module.**

**USE WHEN**

- The task is performed infrequently, is novel, and is complex (many steps)
- Users are unwilling or unable to learn the software
- The steps must be performed in a certain order.

Do not use a wizard:

- To teach a topic
- With creative applications (e.g. many ways to accomplish a task, unstructured task)
- With expert users
- When speed is important (the forced sequence and number of fields will slow them down)
- When there is no time to test the wizard

**USER PROBLEM**

The user needs help to complete the steps of a complex task in a particular order. It is important that the user makes very few mistakes.
SOLUTION DESCRIPTION

The wizard is launched in a separate window to clearly indicate that a separate process is happening.

- The wizard window is resizable but not closable (‘X’) from the Title Bar.
- The title bar of the wizard contains the overall goal of the wizard so the user can always see it.
- The number of the current step and the total number of steps is displayed in the upper-right corner of the title bar. Alternatively, a navigation graphic can be used at the top of the window (Figure 1).
- A ‘Main Instruction’ at the top of the page explains what to do in this step.
- The wizard has 2-6 steps.
- Three navigation buttons are located in the bottom right-hand corner of the wizard window.
- The navigation buttons (from left to right) are titled ‘< Back’, ‘Next >’, and ‘Cancel’.
- On the first step of the wizard, the ‘< Back’ button is inactive and greyed out.
- On each step, the ‘Next’ button is greyed out until all required fields have been entered.
- On the last ‘required’ step of the wizard (may not be the last step), the ‘Next >’ button changes to ‘Finish’.
- ‘Cancel’ closes the wizard window without saving any selections made in the wizard.
- ‘Cancel’ is active on every step of the wizard.
- No other screen elements are to the left of the navigation buttons.
WHY

A wizard breaks a complex task down into a series of simpler tasks, each determined by a decision that needs to be made. It is similar to a written checklist. The wizard is usually created as a series of screens in a separate pop-up window. The first screen lists the goal of the wizard and the steps that the wizard will take, and the last summarizes the completed task. Each screen contains a ‘previous’, ‘next’, and a ‘cancel’ button and is labeled consecutively along with the total number of screens in the wizard. Alternatively, wizard navigation can be done using a table-of-contents (Figure 6), tabs, or a drop-down menu at the top of the wizard panel. Users in Burton (1999) preferred the table of contents approach (liked, faster) over tabs or a drop-down menu approaches.

A wizard is used when it is very important that the user know where they are in the process of completing the task. The idea is to guide the user and prevent them from making any irreversible mistakes (hence the ability to go back a step, review at the end, informing them of the goal).

Do:
- Balance # steps (screens) with complexity of each step (screen)
- Keep the number of steps between 3 and 10
- Show current position and expected outcome at each step
- Provide a direct way to perform all the steps without the wizard (secondary approach)
- Include defaults for all fields in the wizard
- Finish the task within the wizard
- Consider a summary page as the last wizard step (screen)
- Group related choices on the same step (screen)
- Break task into steps based on decision points within the task
- Disable the ‘next’ (forward) button on each screen until the required fields have been entered
- Test the wizard early using paper prototypes
- Keep inputs very clear (or users will get stuck)
- Minimize screen text
- Allow the user to go back to previous, completed steps

Don’t:
- Default to ‘Next’ on every screen, or the user will not analyze each step taken
- Automatically advance steps in the wizard
- Use a ‘welcome page’ or a ‘summary page’ in the wizard.
- Use a wizard to cover up a poor interface design (e.g. overly-complex)

EXCEPTIONS

A wizard is usually considered a kind of ‘advanced help.’ If a lower level of help is needed or if the sequence of steps in the task can vary or some can be skipped, the developer should consider using a ‘skipping’ approach. This approach is like a wizard but allows the user to jump to (or skip) any step in the wizard as needed. If more help is needed, an agent or guide could be used. If less, a help file or even tooltips.
MORE EXAMPLES

Figure 3 – ‘Skipping’ wizard (buying airplane tickets online at KLM).

Figure 4 – Microsoft wizard.
Figure 5 – Completing the Microsoft wizard.

Figure 6 – Table-of-contents wizard (TurboTax 2005 for Mac).
REFERENCES


(Cooper, About Face 2.0)
Treats the user like a peripheral device… interrogation… Violates Axiom “asking questions isn’t the same as providing choices” Tendency for the user to just click ‘next’ without analyzing why. Better to let the program guess the responses and let user change those that don’t fit. Only use wizards for rarely used actions.

Not good for teaching a user how to do something (tutorials). Should be designed to hide as many steps as possible. Should involve real data. Reliance on wizards can be a sign of an overly complex (poor) interface design. Use wizard to supplement the direct way of performing the task. Include default settings for all controls. Use a larger number of simple pages than vice versa. Do not force the user to leave the wizard to finish the task. Don’t advance pages automatically. Keep the dialog text simple and short. Consider a summary page at the end of the wizard.

User must be willing to surrender control over the process, not good for creative applications or cases where the user does want to learn the software and its structure. Chunk related choices together. Does the order of the steps matter? Look for decision points (where current choices change the downstream choices). Allow the user to move completely backward and forward within the wizard. Show position in wizard if possible (i.e. a sequence map).

Tara Scanlon, When to Develop a Wizard, User Interface Engineering
http://www.uie.com/articles/wizard/)
Use wizards when: 1) task has many steps, 2) user lacks domain knowledge, 3) task steps must be done in a predetermined sequence. Don’t use wizards when: 1) users are too advanced, 2) a non-user problem is being addressed, 3) there is no time to test it. Users may resent the number of clicks a wizard requires. Paper prototyping works well for testing wizards. Let users re-run the wizard if a change is needed. Wizard inputs need to be very clear, or the users will get stuck. User need a way to understand the implications of every choice, so they don’t have to run the wizard over and over, exploring the choices.

Martijn van Welie, Wizard Design Pattern
Number of subtasks should be between 3 and 10. On the first screen, tell the user the goal that will be accomplished when the wizard is complete. Disable the forward mechanism until required fields are entered. Allow the user to exit the wizard whenever the required options are set. If forced to follow the task order, users will be slower but will commit fewer errors.

**REVISION HISTORY**

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/04/2006</td>
<td>Initial Creation</td>
<td>John Milanski</td>
</tr>
<tr>
<td>10/26/2006</td>
<td>Added specifications to the Solution Description</td>
<td>John Milanski</td>
</tr>
</tbody>
</table>