uDig Usability Recommendations

January 11, 2005

Submitted To: Program Manager
GeoConnections
Victoria, BC, Canada

Submitted By: Jody Garnett
Refractions Research Inc.
Suite 400 – 1207 Douglas Street
Victoria, BC V8W 2E7
E-mail: jgarnett@refractions.net
Phone: (250) 383-3022
Fax: (250) 383-2140

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

This document is the result of a detailed examination of the uDig application with respect to the Eclipse User Interface Guidelines. The goal of this document is to provide guidance for the final stage of the development before release.

The Eclipse User interface guidelines focus on providing a consistent visual appearance. Contributions are expected to look and act in a similar manner. An explicit workflow is emphasized where the responsibilities of lifecycle of construct is clear.

The choices epitomized by the Eclipse User Interface Guidelines support a consistent workflow: editors are used in an open-edit -save lifecycle; views are used for direct manipulation; and dialogs are used for complex or external interactions.

This document provides an extension to the general Eclipse guidelines, allowing the use of Views in request/response situations.

We have been pleased with the use of the Eclipse User Interface Guidelines in evaluating our application. These guidelines have managed to address a wide variety of issues in an efficient manner.

2 USER INTERFACE GUIDELINES

To start out, let's review the elements of an application based on the Eclipse Rich Client Platform.

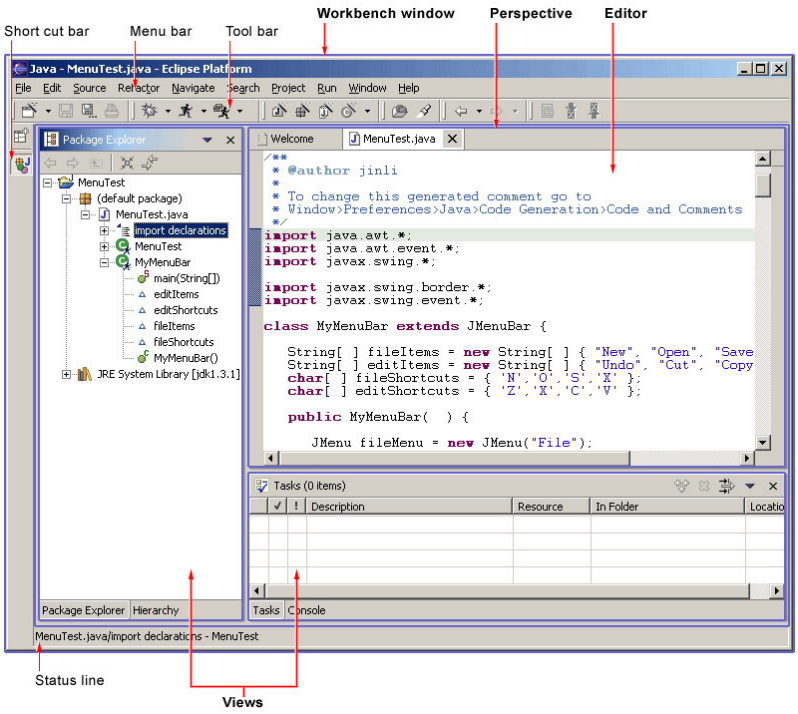

Figure 1 – The Workbench

uDig makes use of the following user interface components:

- A **Workbench window** including **Menu bar**, **Tool bar** and **Status line**.
- Various **Editors** including Map editor and Page editor. Editors make use of an open-edit-save lifecycle.
- Navigator **View** – such as Projects and Catalog. These views provide access to information.
- **Views** that provide support for the current activity. These views provide direct manipulation of an aspect of the current editor. Examples include the Layers, Style and Search view.
- **Dialogs** and **Wizards** that provide complex or external interaction.
- **Commands** are invoked by a user to carry out some specific functions and are available through **Menu bars** and **Tool bars**.
- **Perspectives** used for long-lived tasks.

2.1 Overview

You are encouraged to review the Eclipse User Interface Guidelines (available at http://www.eclipse.org/articles/Article-UI-Guidelines/Contents.html). We have a checklist from this guide in Appendix A if you would like to get a feel for the topics covered. The original document provides examples, and contra examples of difficult points.

The Eclipse User Interface Guidelines cover the following areas:

- **General UI Principles:** from capitalization to error handling
- **Visual Design:** focusing on consistent icons and imagery
- **Command:** invoked to carry out specific a function, should indicate the "behavior that occurs"
- **Dialog:** used for modal interaction such as feedback or asking for information
- **Wizard:** used for a series of steps, often import or export operations
- **Editor:** interaction with a data object, the editor represents the primary task, following an open-edit-save lifecycle
- **View:** supports role of the current editor, follows a direct manipulation lifecycle
- **Perspective:** arrangement of editors and views related to a specific task
- **Window:** provides support for menu bar, toolbar and status bar. Guidelines are given for standard menus such as navigate.
- **Properties:** shows properties that are not normally visible
- **Standard Components**: use of common Views such as Properties
- **Flat Look Design**: instructions for making use of web page style forms
- **Resource:** consistent use of the idea of a Resource (in our case a Layer)

2.2 Extended Guidelines

We have focused our extensions in two areas: providing a consistent set of imagery for spatial constructs, and allowing an apply/cancel workflow when working with external servers.

The introduction of an apply/cancel workflow is intended to address two design problems.

1. **Style View:** providing direct manipulation of style information quickly "floods" the system, and any referenced servers, with redraw requests. The recommended solution of using a Dialog would disrupt the visual nature of a GIS application.
2. **Forms**: GIS operations often provide a form to be filled out for a spatial operation. This is often accomplished in cooperation with the current Map.

We have adopted the following additional guidelines.

- 'Apply' and 'Cancel' buttons are the last entries in the local toolbar. This stable location represents a predictable location similar to the use of the "link" icon in navigation views.
- A toolbar may also support a 'link' button, allowing the option of direct manipulation (if the user can tolerate the delay).
- 'Apply' should be called when the view looses focus.

2.2.1 Style Dialog Alternative

The Eclipse User Interface Guidelines recommend the use of Dialog for operations such as style editing. This is reinforced by the contrast between the Properties Dialog (being used for complex operations) and the Properties View (being used for simple direct manipulation).

We feel that assigning style editing into these categories would disrupt the workflow of a GIS application.

2.2.2 Spatial Operation Form Alternative

Several of the existing Tools such as Select and Info can operate in a request response manner. The use of the tool initiates a request directly; when the response is available the associated view is brought forward to display it. This is similar in spirit to the Eclipse search facility.

The Eclipse user interface guidelines provide some guidance for using a Table Editor in the manner we request. The recommended functionality can be seen in both the Style and Select views where there is a text field to manipulate their current request. Pressing 'Enter' or changing focus away from the text field results in the request being sent off to the appropriate services.

3 USER INTERFACE RECOMMENDATIONS

This section is a compilation of recommendations generated using the Eclipse User Interface Guidelines to review each aspect of uDig. These recommendations will be used in Milestone 4 to improve application usability.

At this time we have identified several broad-reaching recommendations to be addressed via an ongoing QA process.

1. **User Interface Threading**, care needs to be taken to start a job for any action that involves a remote server (or even local file). This is particularly difficult with the use of *IGeoResource* in which most methods should be considered blocking. For long running operations, progress and the ability to cancel is provided by the Eclipse Job infrastructure.
2. **Wizard Progress**, the final step of all our import wizards needs to be threaded. This differs from our previous recommendation in that failure should return us to the wizard, a progress bar needs to be displayed in the wizard, and finally the newly created resource should be selected.
3. **Context Menus**, we need to ensure that the same context menu is provided regardless of where in the system a Layer is found.

Additional recommendations are provided for individual user interface elements in the following sections.

3.1 Editors

Editors are used to interact with primary content, usually a document or data object. The content is the primary focus of attention and a reflection of the primary task.

3.1.1 Map Editor

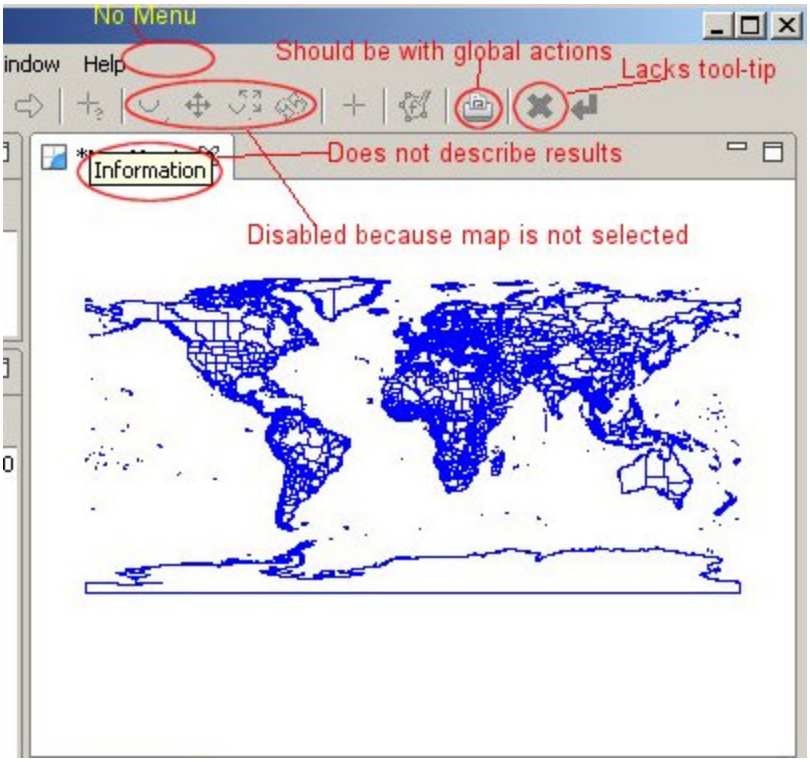

Figure 2 – Map Editor

- Toolbar buttons should be enabled when the map editor is not selected.
- Should have a localized version of the resources
- Print button on the toolbar should be plugged into the global action
- All toolbar buttons should have a tool tip
- The Information button and select features button should be more easily distinguished.
- Back and forward buttons should work properly
- Edit feature button should be disabled if features cannot be created
- Select features button should be disabled if there are no features to be selected
- All tool tips should describe their results
- Should use cut, copy and paste
- Should provide a main menu

3.1.2 Page Editor

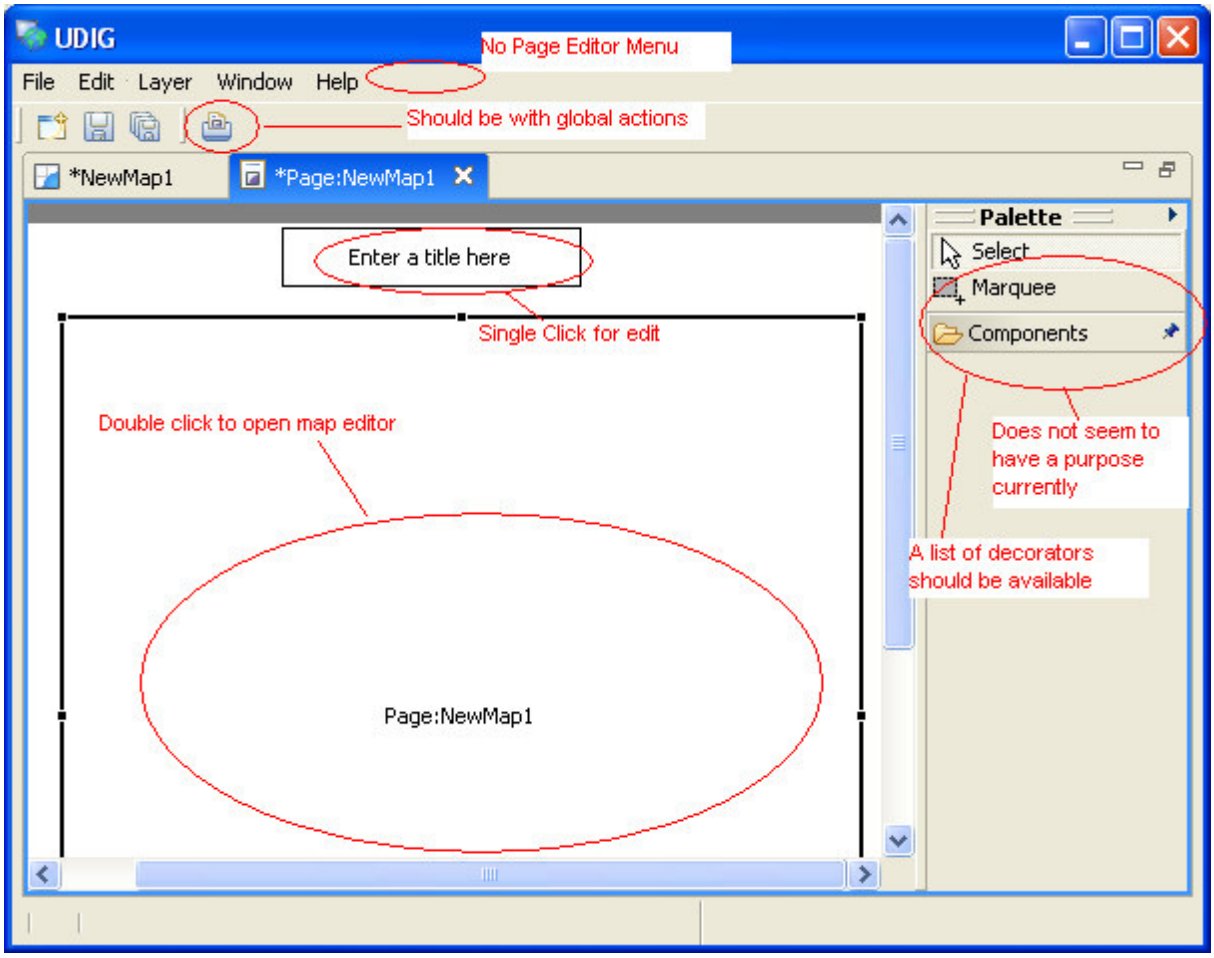

Figure 3 – Page Editor

- Page Editor should support saving changes
- Double clicking on map box in the page should open the map editor
- A single click on the title box should allow the title text to be edited
- It should be possible to set the zoom level of the page layout. For example, it should be possible to fit the entire page in the viewing area.
- Unable to determine the function of the Marquee tool.
- Unable to determine the purpose of the Components folder.
- Decorators should be added to a page by dragging in a decorator.
- Palette should have a list of possible decorators.

3.2 Views

A view is a visual component used in a supporting role for the primary task. They are typically used to navigate information, open an editor or view properties for the active editor.

3.2.1 Projects

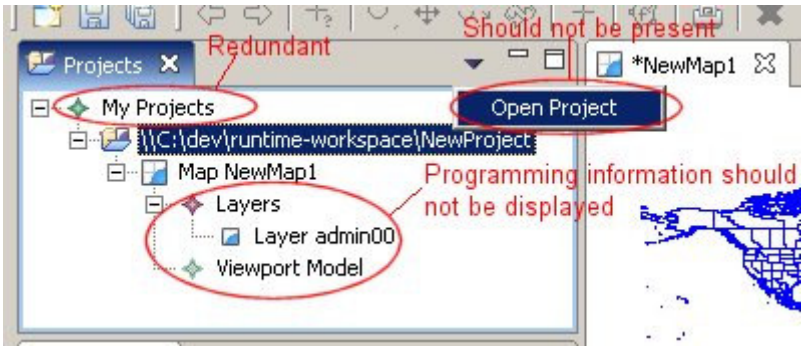

Figure 4 – Projects View

- Should provide localized version of the resources
- Properties dialog should contain a superset of items from the properties view
- Should provide keyboard shortcuts
- Should provide tool-tips (that should describe the result) or icons on the context menu
- Direct manipulation does not work – when you change a map's name, it is not updated in the editor area.
- The pull-down menu should not contain 'Open Project' – it is not a presentation command.
- The context menu should not contain 'Open Project' – it is not a selection command and seems out of place
- Context menu does not enable/disable actions based on selection type properly. The "My Projects" and project elements do not have PropertyPages and should not have that option in the menu.
- Does not hook into global cut, copy and paste.
- View does not persist properly across sessions
- It is a navigation style view but does not have a "link" button
- The map icon should display its modified status in the lower right corner

3.2.2 Layers

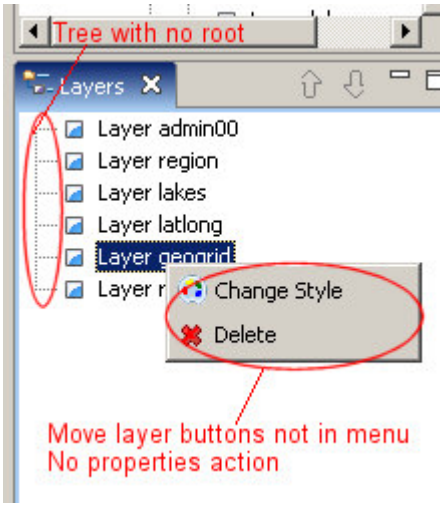

Figure 5 – Layers View

- *Move layer up/down* buttons should have headline style capitalization
- Should provide a localized version of the resources
- *Move layer up/down* buttons should be disabled when they can't be used
- Should use properties view and dialog
- Layers in this view should be treated the same as Layers in the Projects view
- *Move layer up/down* buttons should be available in the context menu
- The context menu of a Layer should be the same as in the Projects view.
- Should use cut, copy and paste
- Layer order should persist across sessions
- Elements in the pane should be part of a list, not a tree.

3.2.3 Layer Overlay Appearance

Layers appear in the Layer, Catalog, Search and Selection views. Here are some recommendations for their appearance.

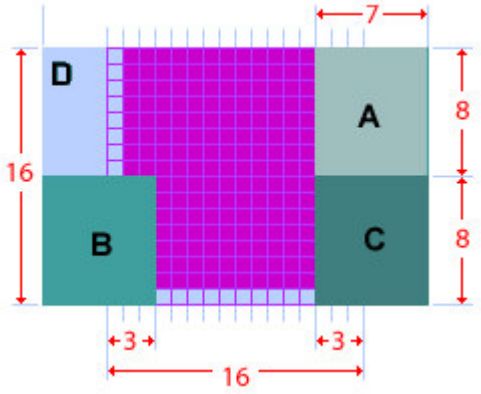

Figure 6: Layer Appearance

- A indicates in its top right corner if it has been modified
- B displays the status in the lower left corner
- C displays view specific information in the lower right corner
- D displays applicable attributes as a coloured underlay

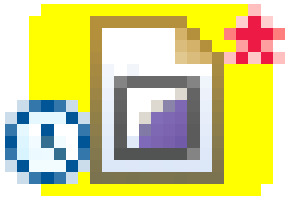

Figure 7 – Example of Layer Appearance

In Figure 7 the icon indicates we are working with a shapefile. The star indicates the layer has been modified. The clock indicates we are waiting on a process. The yellow underlay indicates the layer is applicable to the current tool.

3.2.4 Catalog

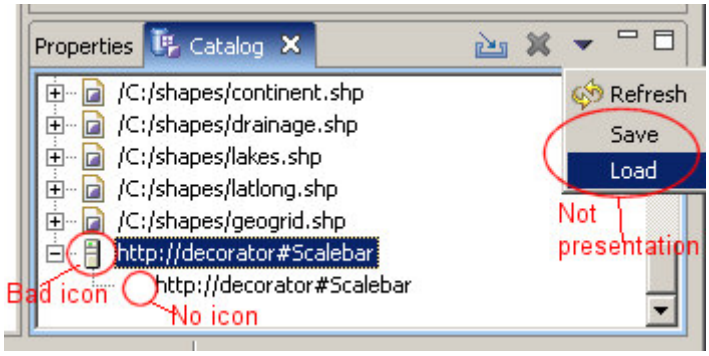

Figure 8 – Catalog View

- Should provide a localized version of the resources
- Decorator service shouldn't use the server icon
- Should use properties view and dialog
- *Add to current map* should be disabled if there is no current map
- *Save/load* buttons should not be on the pull down menu.
- Should use cut, copy and paste.
- Open objects should persist across sessions
- Service icons should display their connection status in lower left corner (See Figure 6, section B)
- Service and resource icons should display their modified status in the top right corner (See Figure 6, section A)
- Service and resource icons should display their cached status in the lower right corner (See Figure 6, section C)
- Resource icons should display their applicable attributes on the right edge

3.2.5 Service Overlay Appearance

Services appear in the Catalog view. Here are some recommendations for their appearance. See Figure 6 – Layer Appearance for the recommended layout.

- A indicates in its top right corner if it has been modified
- B displays the status in the lower left corner
- C displays view specific information in the lower right corner
- D displays applicable attributes as a coloured underlay

3.2.6 Search

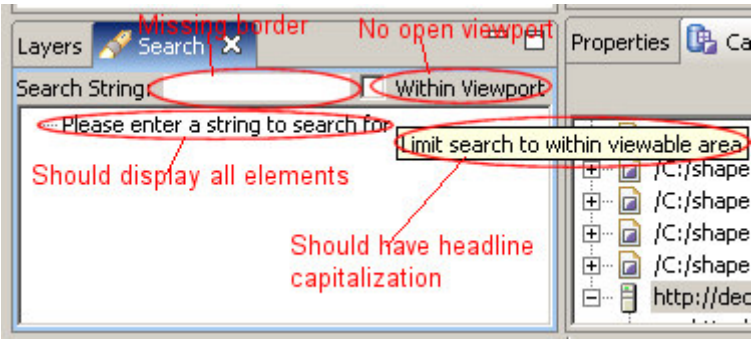

Figure 9 – Search View

- Search text box should have a border
- Should provide a localized version of the resources
- Decorator service should not use the server icon
- Should use properties view and dialog
- *Add to current map* should be disabled if there is no current map
- Should use cut, copy and paste
- Should persist across sessions
- Should display all elements when the search string is blank

3.2.7 Information

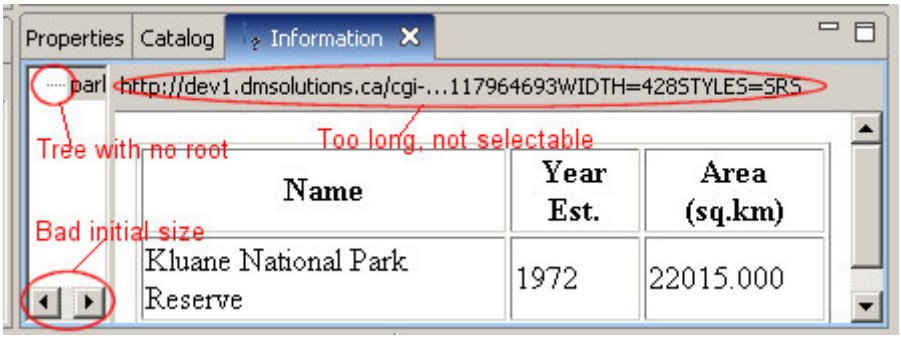

Figure 10 – Information View

- Should provide a localized version of the resources
- After initial opening, the left pane should be resized properly.
- The URL at the top of the right pane should be selectable. Since it can sometimes be too long, it should also be available in a properties dialog.
- Elements in the left pane should be in a list, not a tree.
- Should provide access to layer properties
- Should be able to perform common layer operations such as *add to current map* or *new map*.
- Layers object should have a context menu
- Should persist across sessions

3.2.8 Style

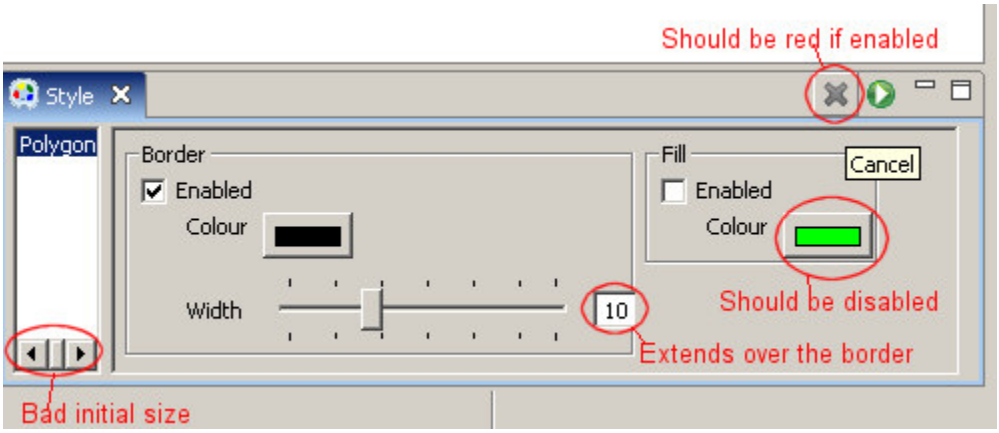

Figure 11 – Style View

- Should provide a localized version of the resources
- The cancel button should be red if it is enabled, or gray if it is disabled
- The initial size on the left pane should require no scrolling
- The line width text field should not extend over the group border
- Changing the state of the enabled fields should change the state of other widgets.
- Should provide tool-tips

3.3 Wizards

Wizards are used for the execution of any task involving a sequential series of steps. uDig uses wizards for creating projects, maps and layers, as well as for importing resources.

3.3.1 New Project

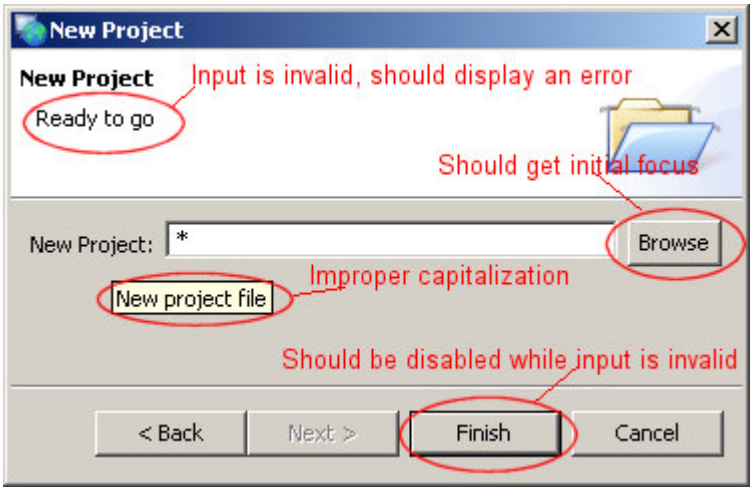

Figure 12 – New Project Wizard

- The initial focus should be set to the browse button. Alternatively, the wizard could use an explorer widget to provide project location.
- The initial message should indicate what the user should do.
- When the New Project field is empty, the message should refer to 'project', not a 'udig file'.
- When an invalid entry is placed into the 'New Project' field, the header should display an error
- The 'New Project' field should be renamed to 'Project location:"
- The finish button should not be enabled when 'New Project' is empty or invalid
- Should provide a localized version of the resources.

3.3.2 New Map

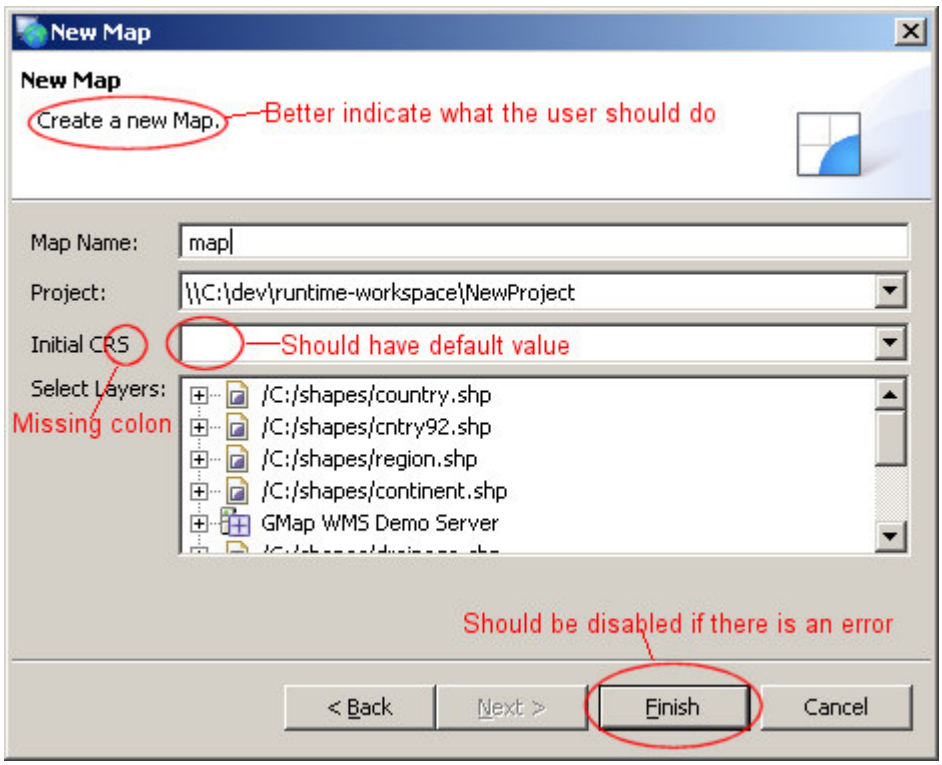

Figure 13 – New Map Wizard

- A project should be initially selected
- A default CRS should be initially selected
- Control labels should use sentence style capitalization
- Should provide a localized version of the resources.
- The initial message should indicate what the user should do.
- Messages should indicate the requirement of values.
- If initial CRS is blank then next and finish should be disabled.
- Decorator icon should not be set to the server icon
- Tool tips should follow headline style capitalization
- The Edit Styles page should hook into the style editor

3.3.3 New Layer

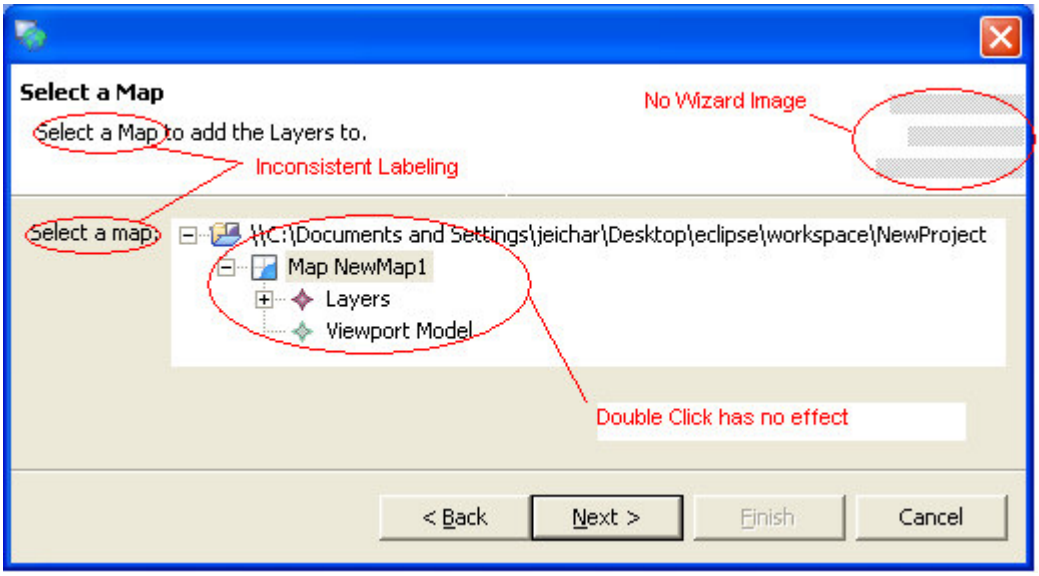

Figure 14 – New Layer Wizard

- Should provide a banner graphic
- Double clicking a selection, such as a map in the map selection page, should move to the next page (if legal), complete the wizard, or expand the branch of the tree.
- When the wizard completes, the map that the layer was added to should be displayed, regardless of whether the map was open previously.
- The user should be permitted to search for a new project that is not part of the workspace.

3.3.4 Web Map Server Import Wizard

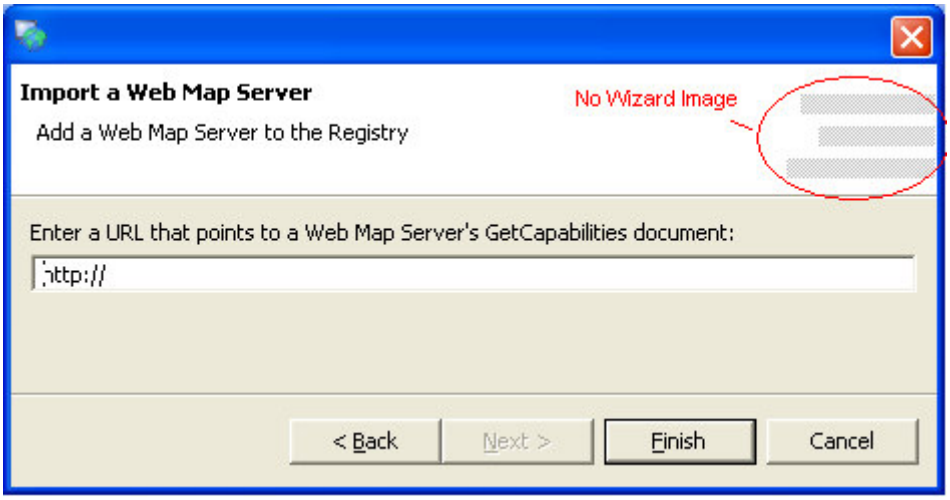

Figure 15 – Web Map Server Import Wizard

- Should provide a banner graphic
- The title should be Web Map Server Import, or something similar.
- The URL text field should have a history of previously imported Web Map Servers.
- The last URL entry should be kept between runs.
- An option to add multiple Web Map Servers should be available as part of the wizard.

3.3.5 Web Feature Server Import Wizard

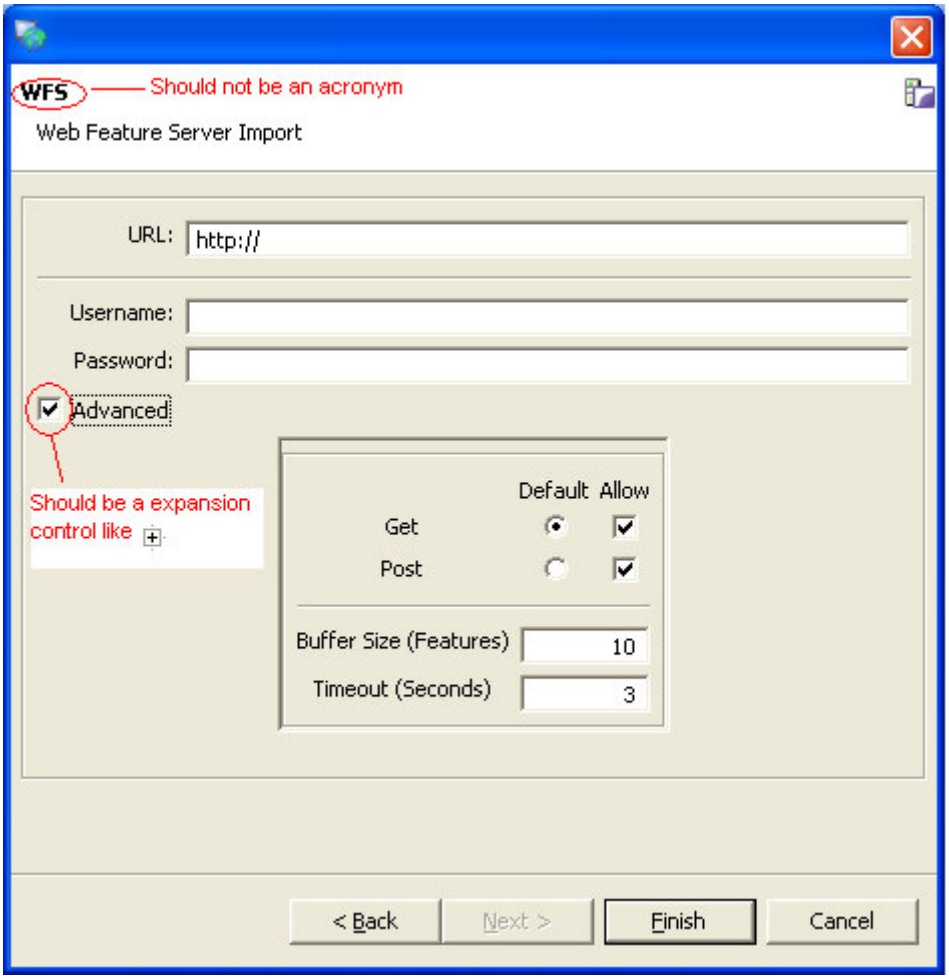

Figure 16 – Web Feature Server Import Wizard

- The URL text field should have a history of previously imported Web Feature Servers.
- The last URL entry should be kept between runs.
- An option to add multiple Web Feature Server should be available as part of the wizard rather than having to repeatedly open the wizard.
- The wizard title should not be an acronym.
- The Checkbox is used to hide and show the advanced options. A standard control such as the ⊞ or the ⊟ controls should be used instead.

3.3.6 World Image Format Grid Coverage Import Wizard

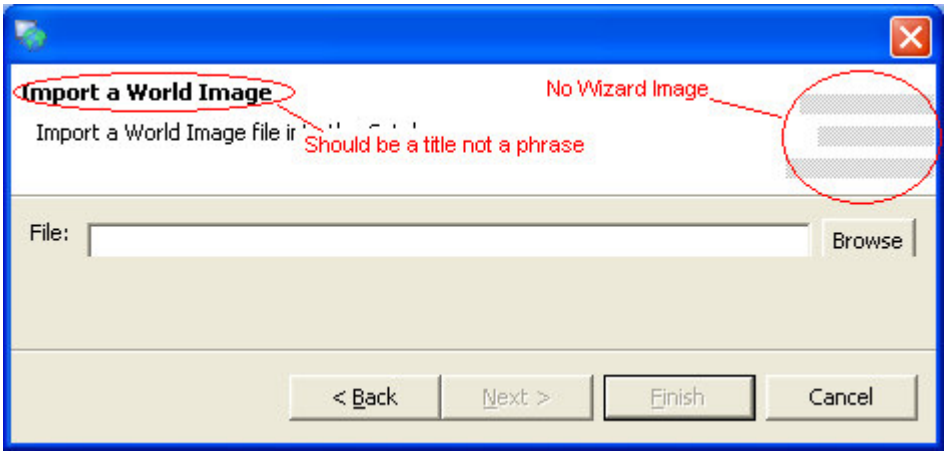

Figure 17 – World Image Format Grid Coverage Import Wizard

- Should provide a banner graphic
- File browser should permit multiple files to be selected and imported in one step.
- The title should be World Import Grid Coverage Format, or something similar.

3.3.7 ArcGrid Format Grid Coverage Import Wizard

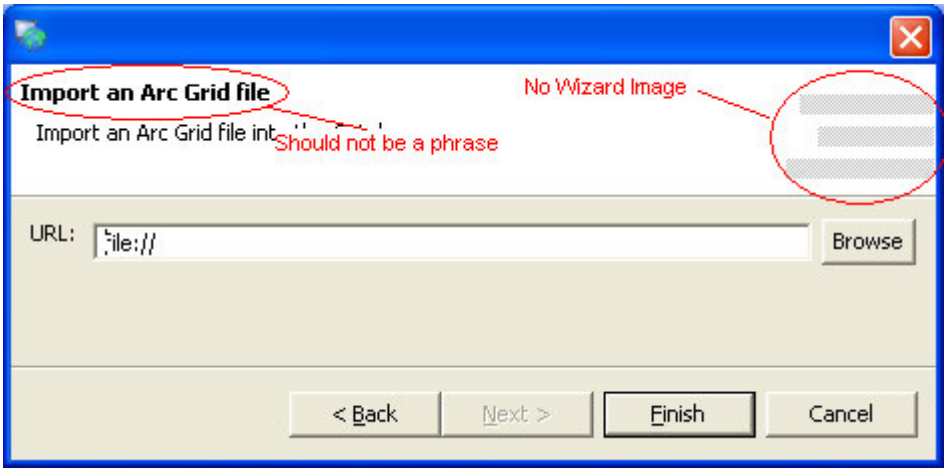

Figure 18 – ArcGrid Format Grid Coverage Import Wizard

- Should provide a banner graphic
- File browser should permit multiple files to be selected and imported in one step.
- The title should be ArcGrid Coverage Format, or something similar.

3.3.8 ArcSDE and MySQL

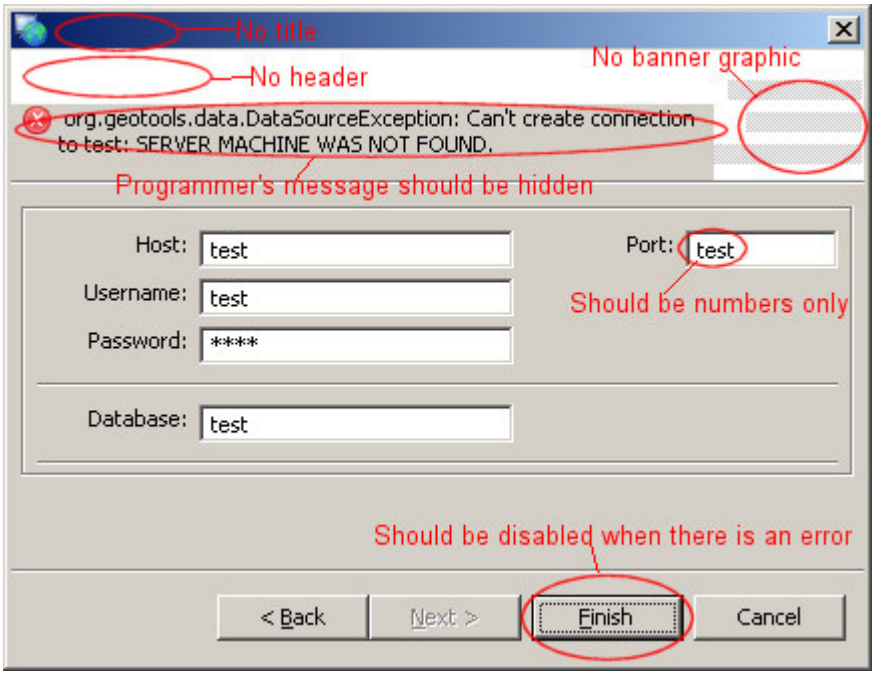

Figure 19 – ArcSDE Datastore Import Wizard

- Should provide a starting header and banner graphic
- Should provide a starting prompt
- Should remember previously entered values
- Should provide an error on invalid values
- Displays a Java exception when a connection is not made – it should display a proper error message
- Should have a window title
- Label tool-tips should use headline style capitalization
- MySQL wizard should enable its finish button
- Should display prompts or proper error messages
- Should open and select the object upon completion

3.3.9 Oracle Spatial

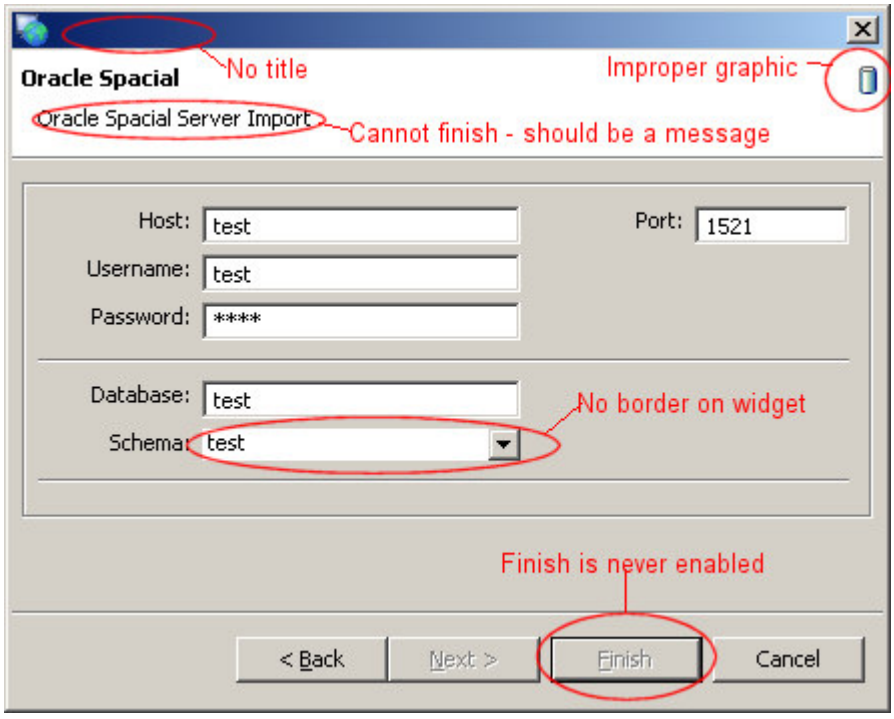

Figure 20 – Oracle Spatial Datastore Import Wizard

- Should provide a window title
- Schema field widget should have a border
- Should provide a proper banner graphic
- Finish button should be enabled once fields are valid
- Should display prompts or proper error messages
- Label tool-tips should use headline style capitalization
- Should open and select the object upon completion

3.3.10 PostGIS

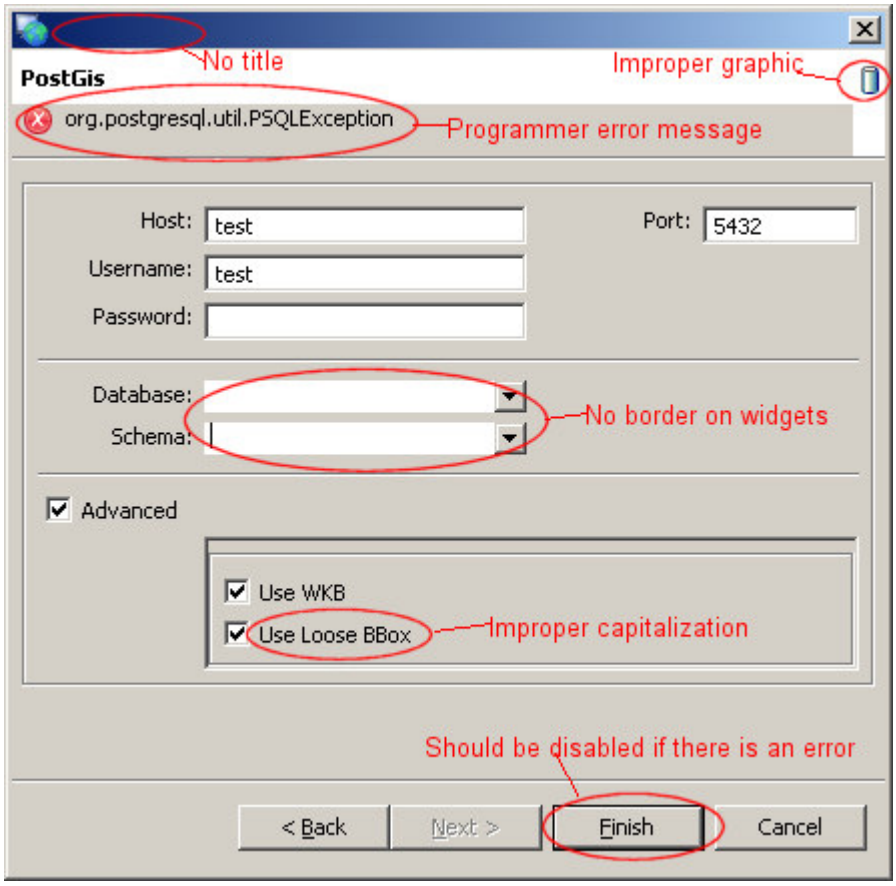

Figure 21 – PostGIS Datastore Import Wizard

- Should provide a proper header message and banner graphic
- Should provide a window title
- Label tool-tips should use headline style capitalization
- Database and schema field widgets should have a border
- When database or schema field is clicked, an error rather than an exception should be displayed in the header
- A connection attempt should only be made when the *next* or *finish* buttons are pushed.
- Finish should be disabled if there are required parameters missing
- Should open and select the object upon completion

3.3.11 Shapefile

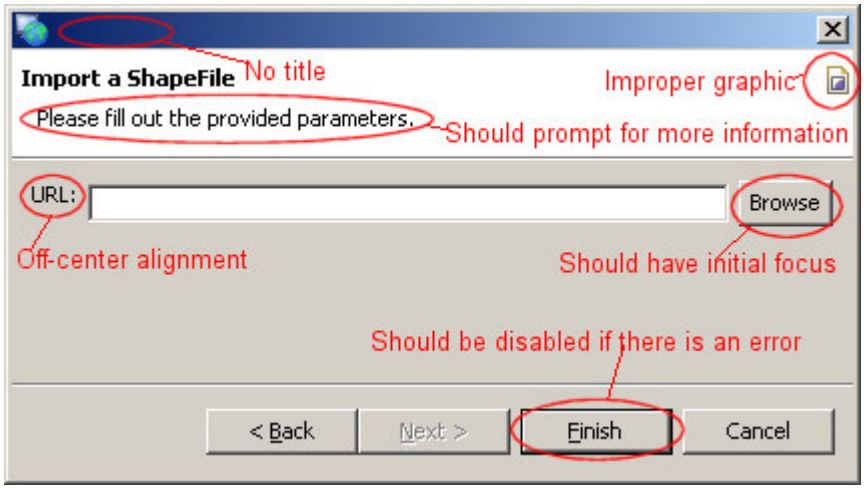

Figure 22 – Shapefile Import Wizard

- Should provide a proper header message (it should be more specific)
- Should provide a proper banner graphic
- Should provide a window title
- Should use a file browser or a similar widget.
- Should remember previously entered values.
- Should allow import of multiple shapefiles
- File browser should allow a *.* file filter.
- If an invalid file is selected, an error should be displayed
- If no file is entered, a prompt should be displayed
- When the wizard finishes, it should do something with the shapefile. Either add it to the current map or open the catalog and select it.

APPENDIX A: CHECKLIST

General	
	Do what eclipse does.
	Use Headline style capitalization for menus, tool-tip and all titles.
	Use Sentence style capitalization for all control labels.
	Localized version of the resources.
Visual	*Icons and imagery*
	Re-use the core visual concepts.
	Use the appropriate icon type in the location it is designed for.
	Cut the icons with attention to alignment, position, and size.
	Follow the specific size specifications for wizard graphics.
	Follow the predefined directory structure and naming convention.
	Use the enabled and disabled states provided.
Properties	
	Use Properties view for quick easy changes or switching between local objects.
	Use Properties Dialog to edit a remote or complex object.
	Properties Dialog contains a superset of items from the Properties view.
Widgets	*Such as Trees and Tables*
	Checkbox widgets should not have their state accidentally altered when the selection is changed.
Preferences	*Used for global options*
	Expose preferences for a view, editor or windows using a menu or toolbar.
	Start with a single preference page and then specialize with sub pages.
	Try and slot the pages into existing categories
Flat Look	*Use the flat look design for extensive property and configuration editing.*
	Have the core selections on the overview page expanded. Other pages should provide a Home icon to return to the overview.
	Have your tree in the outline correspond to the tabs in your content editor.

Tao	
	If your object is a special case, attempt to provide an adapter to make it as transparent as possible.

Accessibility	
	All of the features provided by a tool should be accessible using the mouse or keyboard.

Commands	*Invoked by a user to carry out some specific functions*
	Each command requires a label, tool-tip and icon.
	The tool-tip should describe the result of the command.
	Each command should follow the workbench example of New, Delete and Add.
	Enablement should be performed quickly.

Dialog	*Used for modal interaction with the user. It can be used to solicit information, or provide feedback.*
	When opened focus should be on the first control.

Wizard	*Used for tasks consisting of sequential steps.*
	Each wizard requires a header and banner graphic.
	Each wizard requires 'Back', 'Next', 'Finish', 'Cancel' buttons.
	Start with a prompt message and not with an error message.
	Fill out as much information as you can for the user.
	Validate in order. Prompt the user for more information. Indicate an error message for invalid information.
	Enable the 'Next' or 'Finish' buttons when the information is valid.
	Use a 'Browse' button where possible.
	Open or select the results of the wizard when finished. Change perspective if you have to.
	Use the most specific words possible, i.e. "WMS Layer" rather than "FeatureCollection".

Perspective	*Used for long lived tasks*
	The workbench layout (views, menus, etc.) should correspond to the primary task.
	Only open the perspective if the user agrees, as this is a large context switch for them.
	Limit the shortcuts in the New Perspective, Open Perspective and Show View menus to around seven entries.

Editor	*Used to edit or browse primary content*
	Use the open-edit-save lifecycle. An asterisk indicates that a save is needed.
	Should not be possible to open the same editor twice within a perspective.
	The editor should be labeled with name of its content.
	For multi-page editors, a tab control should be used.
	Hook into any global commands you can, such as cut, copy, paste, or delete.
	The toolbar should contain the most common items from the menu.
	The context menu should be based on the current selection.
	The context menu contents should set by selection type and enabled/disabled through the selection state.
	Support extension of context menu with MB_ADDITIONS and an IactionFilter.
	Use an outline view if the contents will not fit on one screen.
	Table cell editors should work with single click, with the changes being committed when the user clicks away. Enter button should commit while the escape button cancels.

Views	*Use views to navigate information, open editors or display properties of an object*
	Use a direct manipulation workflow.
	Only one view per perspective, but should be able to be opened by several perspectives though.
	Only commonly used commands should be on the toolbar. They should also be in a menu.
	Use the view's pull-down menu for presentation commands.
	Use the context menu for selection actions, in standard order, registered with the platform.
	Context menu should have a fixed set of commands by selection type, enabled or disabled based on the selection state.
	An object appearing in more then one view should have the same context menu.
	Support extension of context menu with MB_ADDITIONS and an IActionFilter.
	Hook into global commands like cut, copy, paste, delete.
	Persist view state between sessions.
	Navigation views should have a "link" button.

Window	
	Contribute ActionSets to the menu first, followed by the toolbar if they are to be frequently used.
	Each ActionSet should have a specific task in mind, such as "zoom" rather than "edit".
	Multiple smaller ActionSets are more desirable than a large one. This allows the sets to be shared between views and editors.
	Let the user control visible ActionSets.
	Display status related messages using the global status bar.