

INTERACTION PATTERNS IN USER INTERFACES

Martijn van Welie
martijn@cs.vu.nl

Hallvard Trætteberg
hal@idi.ntnu.no

Abstract

This paper discusses and presents interaction patterns in user interfaces. These patterns are focused on solutions to problems *end-users* have when interacting with systems. The patterns take an end-user perspective which leads to a format where usability is the essential design quality. The format is discussed and presented along with twenty patterns that have been written in that format.

Introduction

The patterns presented in this paper are part of an ongoing effort to describe successful solutions that benefit the *users* of the system¹. Consequently, they are solutions most of us are acquainted with. Other collections such as Common Ground (Tidwell 1998) or the Web patterns collection (Perzel and Kane 1999) do not make the explicit distinction between the *user* perspective or the *designer* perspective. Although some of those patterns indeed benefit users, they lack the proper focus and rationale for it. For example, the problem statement in Tidwell's patterns is typically of the form "*How can the artifact best show ...*" which does not explicitly relate to a usage problem of users. Generally speaking, each pattern that focuses on the user perspective is *also* usable for designers but not vice versa. Therefore, our patterns should benefit both designers and end users.

Interest in patterns for user interface design (UID) goes back to 1994 (Rijken 1994, Bayle 1998) but although several pattern collections exist, an accepted set of such patterns i.e. pattern language, has not yet emerged. There appears to be a lack of consensus about the format and focus of patterns for UID. Consequently, a pattern language for UID has not been established since it is necessarily preceded by the development of a sufficiently large body of patterns written with the same focus or "view". It is our opinion that patterns in UID require a special format which is focused on *usability* (van Welie, van der Veer, and Eliëns 1999, Hartson 1998).

Taking the User Perspective

When taking the user perspective it becomes important to put emphasis on the argumentation for *how* and *why* the usability is improved. Without such a rationale it is impossible to see whether or why the solution is actually a *good* and *accepted* solution. Certain solutions in UID solve problems that designers or marketing people have but *not* necessarily problems users have. E.g. banners and splash screens are accepted designs, but hardly for usability reasons. It is not difficult to identify patterns in user interfaces but it is hard to identify those patterns that *really* benefit the user, and explain the usability aspects.

There are many examples of bad and reoccurring user interface designs, making it hard to identify good solutions. On the other hand, bad examples can be very useful to motivate designers to use patterns, similar to the idea of anti-patterns. The "Interface Hall of Shame"² is a nice commented collection of such bad examples. In other cases it is tempting to describe solutions that are *minimizing* usability related problems of user. For example, validating data after a user has entered it is not always the best solution, although frequently used; the user should not be allowed to enter syntactically incorrect data in the first place! The **Unambiguous Format** pattern can be used to achieve this goal.

Copyright © 2000, Martijn van Welie and Hallvard Trætteberg
Permission is granted to copy for the PLoP 2000 conference.
All other rights reserved.

¹ The Amsterdam Collection of Patterns in User Interface Design
<http://www.cs.vu.nl/~martijn/patterns/index.html>

² <http://www.iarchitect/mshame.html>

Categorizing User Problems

Our patterns are task related and in this collection we categorize them according to the kind of usage problems they address. In (Norman 1988) several user interface principles are suggested. The principles give an indication of the kind of problems and questions (McKay 1999) users may have when interacting with a system.

- **Visibility.** Gives the user the ability to figure out how to use something just by looking at it. *Hmm, I think this feature might do it...*
- **Affordance.** Involves the perceived and actual properties of an object that suggest how the object is to be used. *Now how does this object work? Oh, I get it...*
- **Natural mapping.** Creates a clear relationship between what the user wants to do and the mechanism for doing it. *To perform my task, I need to select this option, enter that information, and then press this button...*
- **Constraints.** Reduces the number of ways to perform a task and the amount of knowledge necessary to perform a task, making it easier to figure out. *Oh no, what do I have to enter here? Ok, I just have these choices...*
- **Conceptual models.** A good conceptual model is one in which the user's understanding of how something works corresponds to the way it actually works. This way the user can confidently predict the effects of his actions. *To perform the task, I provide the necessary information and gave this command ... and it seems to work as I expected it to...*
- **Feedback.** Indicates to the user that a task is being done and that the task is being doing correctly. *Great it worked!*

These principles assume that users always exhibit fully rational behavior. In practice, users make mistakes and do things they do not really wanted to do. Therefore, additional principles are:

- **Safety.** The user needs to be protected against unintended actions or mistakes. *Oops! I made a mistake and here is how I correct it. Now I understand and I'll try again.*
- **Flexibility.** Users may change their mind and each user may do thing differently. *Now that I think about it, that parameter should have been ... Cancel it, I want to change the order.*

As can be observed, the questions of the categories are quite general and in our patterns the context description is important to distinguish the situations when to use the pattern. Additionally, these problems can often be solved by several solutions which makes it even more important to be precise and concrete.

A Focus on Usability

If we focus on usability problems of user, we need to work out what the implications are for the way we write patterns. A pattern for UID should focus on solutions that improve the usability of the system in use, which must be measurable in usage indicators. Usability can be measured with the following usage indicators; learnability, memorability, speed of performance, error rate, satisfaction, and task completion (van Welie, van der Veer, and Eliëns 1999). Each pattern should therefore state the impact on these usage indicators. In short, if a "UID pattern" does not improve at least one usage indicator, it is *not* a UID pattern. We believe that UID patterns always use a certain ergonomic principle and the rationale section should explain how the ergonomic principle as used in the solution leads to an improvement of the usage indicators. Most of the common pattern elements can be used directly for UID patterns as well. However, it is important to write them down from the right point of "view". In our patterns we use the following elements:

- **Problem.** Problems are related to usage of the system and are relevant to the user or any other stakeholder that is interested in usability. In contrast to SE patterns, problems in UID patterns should not be focused on constructional problems designers are facing. Hence, problem descriptions should often be user task oriented and fall into one of the categories as defined by Norman.
- **Usability Principle.** Interaction patterns usually use a 'principle' on which the solutions are based. A complete set of principles is not known yet but an initial set can be given. The following list of usability principles is used grouped according to Norman's (Norman 1988) user problem categories:
 - **Visibility:** User guidance, Grouping, Incremental Revealing

- **Affordance:** Metaphors
- **Natural Mapping:** Compatibility
- **Constraints:** Minimizing actions, Self-explanation
- **Conceptual Models:** Significance of codes, Compatibility, Adaptability
- **Feedback:** Immediate Feedback, Legibility
- **Safety:** Error Prevention, Error Correction, Forgiveness
- **Flexibility:** Explicit control
- *Context.* The context is also focused on the user. What are the characteristics of the context of use, in terms of the tasks, users and environment for which the pattern can be applied?
- *Solution.* A solution must be described very concretely and must not impose new problems. However, a solution describes only the *core* of the solution and other patterns might be needed to solve sub-problems. Other patterns relevant to the solution should be referenced.
- *Rationale.* This section describes how the pattern actually works, why it works, and why it is good. The solutions section describes the visible structure and behavior of the system, while the rationale provides insight into the deep structure and key mechanisms that lie under the surface of the system. The rationale should provide a reasonable argumentation for the specified impact on usability when the pattern is applied. This section also describes the impact the pattern has on usability when it is applied. It describes what usability aspects have been improved, and which ones have become worse. It is usual that a pattern optimizes one or two aspects while other aspects have to suffer. Each solution tries to provide the right balance in the specified context. Usability literature has identified the following measurable aspects of usability:
 - **Performance Speed.** How fast users can work with the system.
 - **Learnability.** How easy the system is to learn.
 - **Memorability.** How well users remember how to use the system.
 - **Satisfaction.** The satisfaction users get when using the system.
 - **Task Completion.** How much of the task could be completed.
 - **Errors.** The number of errors users made.
- *Examples.* The example should show how the pattern has been used successfully in a system. An example can often be given using a screenshot and some additional text to explain the context of the particular solution. It is preferable to use examples from real-life systems so that the validity of the pattern is enforced. If a writer cannot find any real-life example, the pattern is not a pattern.

The fields and “view” needed to write UID patterns are important. For example if the view is taken wrongly, one might write patterns on “how to use tab controls”. This is very tempting to do especially when guidelines are rewritten into pattern format. However, such views take on the perspective of the designer and not the user. Moreover, the design knowledge about “how to use tab controls” depends on the context of when it is applied, the users and their task. In other words, it is looking at the problem from the point of the solution without knowing the problem. Another addition we use in several patterns is a "Counterexample" section. This is an example of a real application when the pattern should have been applied but was not applied. It creates a kind of anti-pattern effect as serves as an extra motivation for use of the pattern.

Using UI Model Fragments in UID Patterns

Similar to patterns in Software Engineering, for UID patterns there is need for both abstract and precise descriptions. A pattern *focuses* on the essence of a problem and its solution, and *abstracts* away aspects of the design that is coincidental and irrelevant to the addressed problem and solution. To find a pattern and then apply the identified and chosen solution, the pattern should also be *precise*. For SE patterns, the use of class and collaboration diagrams is well suited for this purpose.

However, UID patterns need different representations, and the user oriented design representations like snapshots, storyboards and sketches are problematic, since they are surface representations that lack the

abstraction and precision needed. User interface modeling languages on the other hand, support explicit, semi-formal representations of “every” aspect of the design throughout the process, and provide both abstraction and precision needed. Hence, UI model fragments can be very useful when formulating interaction patterns. For our collection of patterns we have started to develop a notation that is suitable for use in UID patterns, and believe its usage can have several benefits:

1. When initially *formulating* the pattern, it helps us as patterns writers to abstract away particular irrelevant aspects and focus on the core of the solution.
2. When *classifying* the pattern, it helps to see precisely what it addresses.
3. When *selecting* patterns for use it helps to see whether the pattern is applicable for the problems a designer is facing.
4. When *applying* the pattern solution, the precision of the fragments makes it easier to apply the solution in a particular context. It may even aid in the actual implementation of the user interface.

Structuring the Collection

We consider the pattern collection presented here a starting point for a pattern language. The patterns in this collection are linked and hence form a network of patterns. Besides the fact that patterns reference to other patterns, patterns could also be categorized. For patterns in UID several organizing principles/categories have already been proposed (Mahemoff and Johnston 1998). Mahemoff proposes the following categories: task related patterns, user related patterns, user interface element patterns and system based patterns. A categorization gives structure to a pattern collection and facilitates both *selection* and *understanding* of patterns themselves and the whole collection.

Patterns can be used to learn about design but also as reference material when there is a need for them. We consider the latter reason most relevant and it has determined the way we write down the patterns. We think the reason *why* a designer wants to *search* for a pattern, should actually determine the optimal structure of the pattern collection. For our collection, *several* alternatives are plausible. For example, the designer might want to address a particular problem or she might be looking for a pattern that closely matches a specified context. Other possible indices are the usability principle, the usage indicators that are involved, or patterns that only address presentational aspects. Table 1 shows an example of a structuring based on the type of end user problem. On our website, we use XML to describe patterns which facilitates several 'views' on the collection and the designer could use a view that is appropriate. Since not many interaction design patterns exist and they have hardly been used, we consider it premature to settle on one particular structure. Therefore, we present the patterns simply as a linked collection.

Figure 1 **Patterns and user problem categories**

Visibility	Command Area, Wizard, Contextual Menu
Affordance	Mode Cursor, Like in the real world, Setting Attributes
Natural Mapping	Like in the real world, List Browser, Continuous Filter, Navigating between spaces, Container Navigation
Constraints	Unambiguous Format, Focus
Conceptual Models	Grid Layout, Like in the real world
Feedback	Progress, Hinting, Contextual Menu
Safety	Warning, Shield
Flexibility	Preferences, Favourites

Problem The user wants to achieve a single goal but several decisions need to be made before the goal can be achieved completely, which may not be known to the user.

Usability Principle User Guidance (Visibility)

Context A non-expert user needs to perform an infrequent complex task consisting of several subtasks where decisions need to be made in each subtask. The number of subtasks must be small e.g. typically between ~3 and ~10.

- Forces**
- The user wants to reach the overall goal but may not be familiar or interested in the steps that need to be performed.
 - The subtask can be ordered but are not always independent of each other i.e. a certain task may need to be finished before the next task can be done.
 - To reach the goal, several steps need to be taken but the exact steps required may vary because of decisions made in previous steps.

Solution Take the user through the entire task one step at the time. Let the user step through the tasks and show which steps exist and which have been completed.

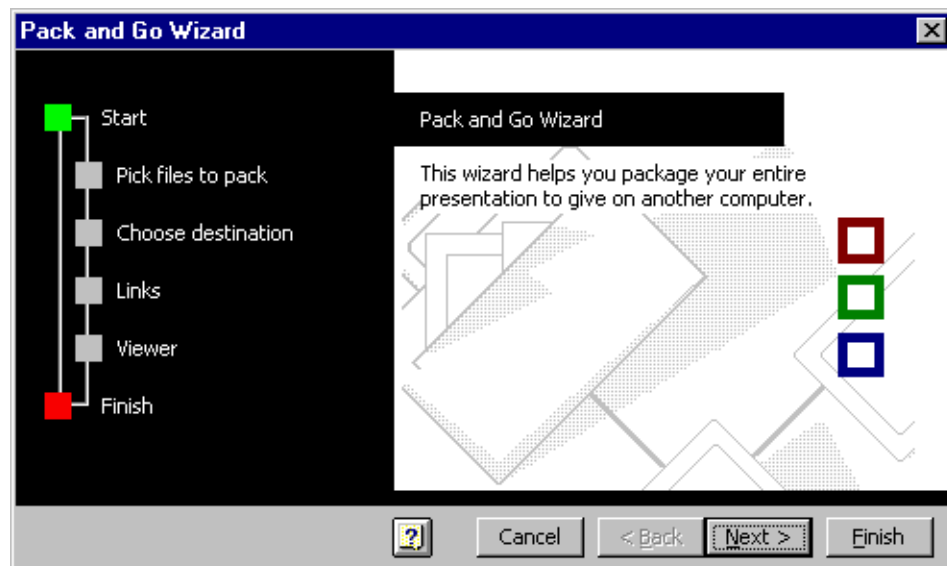
When the complex task is started, the user is informed about the goal that will be achieved and the fact that several decisions are needed. The user can go to the next task by using a navigation widget (for example a button). If the user cannot start the next task before completing the current one, feedback is provided indicating the user cannot proceed before completion (for example by disabling a navigation widget). The user is also able to revise a decision by navigating back to a previous task.

The users are given feedback about the purpose of each task and the users can see at all times where they are in the sequence and which steps are part of the sequence. When the complex task is completed, feedback is provided to show the user that the tasks have been completed and optionally results have been processed.

Users that know the default options can immediately use a shortcut that allows all the steps to be done in one action. At any point in the sequence it is possible to abort the task by choosing the visible exit.

Rationale The navigation buttons suggest the users that they are navigating a path with steps. Each task is presented in a consistent fashion enforcing the idea that several steps are taken. The task sequence informs the user at once which steps will need to be taken and where the user currently is. The learnability and memorability of the task are improved but it may have a negative effect of the performance time of the task. When users are forced to follow the order of tasks, users are less likely to miss important things and will hence make fewer errors.

Examples



This is the 'Pack 'n Go Wizard' from PowerPoint. The user wants to package a presentation so that the presentation can be given on another computer. Several relevant decisions need to be taken and the wizard helps the user take these decisions. The green box shows the current position in the sequence of tasks.

Known Uses Microsoft PowerPoint Pack and Go wizard; Installshield installation programs

Related Patterns NAVIGATING BETWEEN SPACES, LIST BROWSER

Problem The user needs to quickly understand information and take action depending on that information.

Usability Principle Consistency, Predictability (Conceptual Models)

Context Any circumstance where several information objects are presented and arranged spatially on a limited area. Typically in the design of dialog screens, forms and web pages.

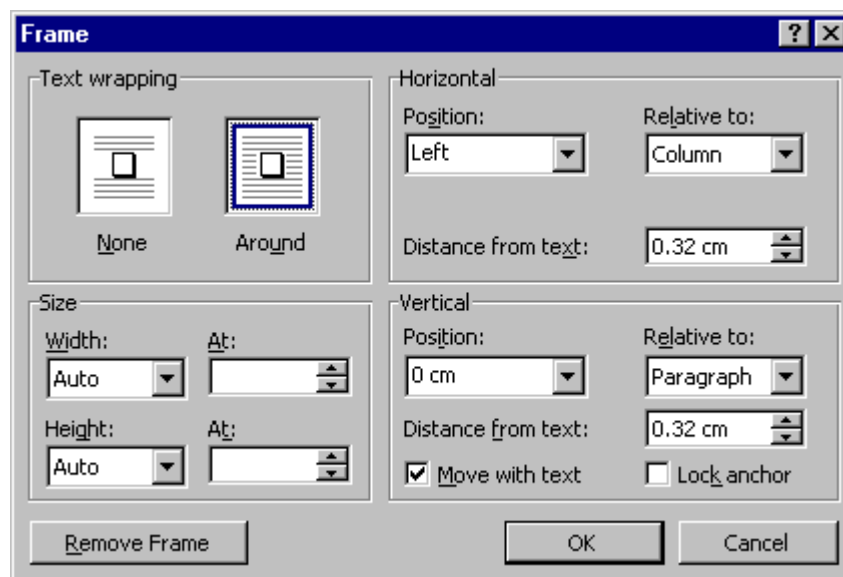
- Forces**
- The users need to see many objects but want to see them in a clear organized way.
 - The users want to minimize the time it takes to scan/read/view objects on the screen.
 - The objects are often related and can be grouped conceptually.
 - The presentation needs to be compact, but still clear, pleasant and readable.

Solutions Arrange all objects in a grid using the minimal number of rows and columns, making the cells as large as possible.

The objects are arranged in a matrix using the minimal number of rows and columns. Objects that are of the same type must be aligned and displayed in the same way. If several objects can be grouped, the grid applies on the level of groups as well. Short elements can be stretched, beginning and ending on grid boundaries. Long elements can span multiple grid cells. Certain objects may have a fixed size that increases the number of rows and columns in which case they should keep their fixed size. Standard response buttons may have predefined positions and can be regarded as being outside the grid.

Rationale Minimising the number of rows and columns improves the time needed to scan the information and to take the appropriate action (Fitts Law¹). Additionally, it causes a very consistent layout with minimal visual clutter and is perceived to be non-obtrusive to the user. The time needed to read the information is reduced which can increase the task performance time. The resulting layout is pleasant to see and increases the satisfaction.

Examples

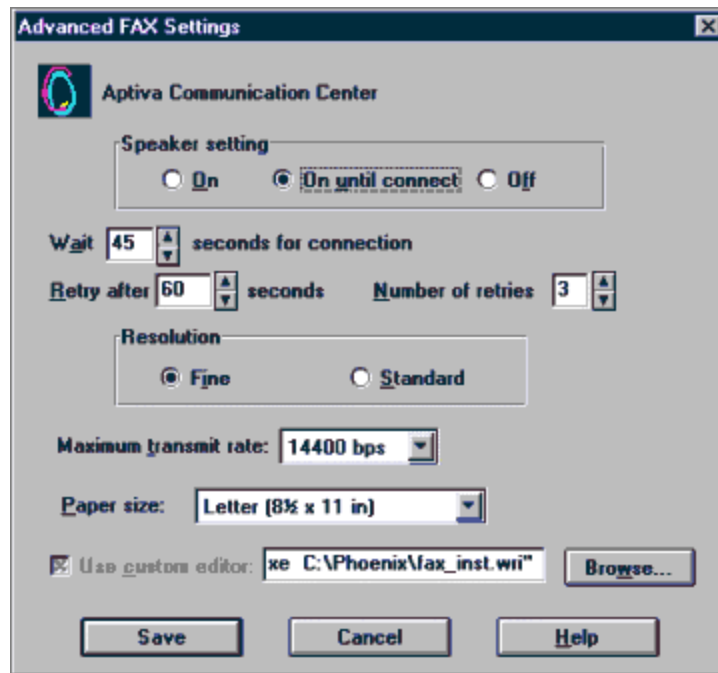


This screenshot is taken from Word 97. Several objects are placed in a dialog box. The elements have been arranged in a grid and objects have been aligned and sized evenly to reduce the number of rows and columns.

¹ FITTS, P.M. "The information capacity of the human motor system in controlling the amplitude of movement", Journal of Motor Behavior, 1954, vol. 47, pp. 381-391

Known Uses Microsoft Word Frame Options

Counter
Example



This image is taken from IBM's Aptiva Communication Center, and demonstrates that the developers simply wanted to get the settings on the screen, rather than make it easy for people to adjust the settings. There is no flow to the screen; your eyes just jump around from place to place as your brain tries to elicit some sort of order.

Problem The user wants to know whether or not the operation is still being performed as well as how much longer the user will need to wait.

Usability Principle Guidance (Feedback)

Context Systems tasks that take a long time (typically more than a few seconds) and must be completed before the next tasks can be started.

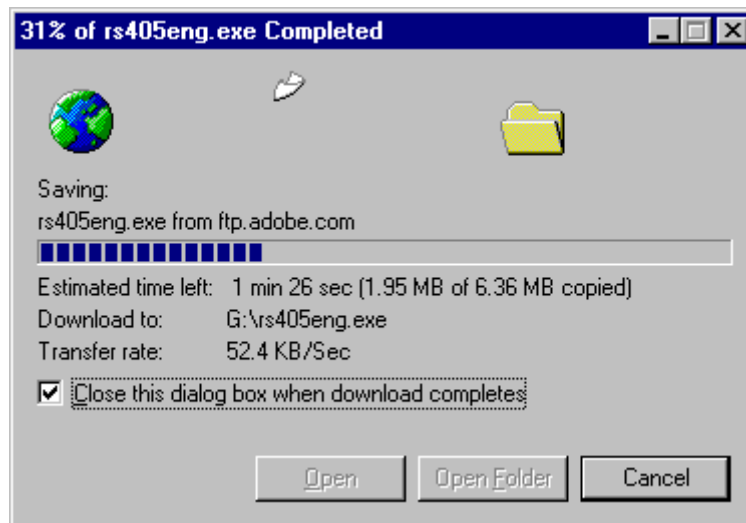
- Forces**
- The performance of the operation cannot always be controlled/avoided by the user (or designer), e.g. because it relies on an external system or hardware, which may fail, block or have low performance.
 - The users do not want to wait need clear feedback on the progress *and* estimated time to completion.
 - The users may not be familiar with the complexity of the task.
 - During the operation the user might decide to interrupt the operation because it will take too long.

Solutions Show that the application is still working and give an indication of the progress.

Provide feedback at a rate that gives the user the impression that the operation is still being performed e.g. every 2 seconds using animation. Additionally, provide a *valid* indication of the progress. Progress is typically the remaining time for completing, the number of units processed or the percentage of work done. The progress can be shown using a widget such as a progress bar. The progress bar must have a label stating the relative progress or the unit in which it is measured.

Rationale By providing new feedback at a rate around 1 or 2 seconds, the user can see whether the application is still processing and has not died. The progress indication gives feedback on how long the application will remain in this state. Combining these two aspects relieves the user's worries. Leaving one of the two out would not solve the user's problem. The solution increases satisfaction because the user knows what is going on and how much longer the user needs to wait. It increases the sense of control. The pattern also avoids additional system load by avoiding retries from users.

Examples



When downloading a file using Internet Explorer 5, the user is presented with this dialog. It shows the progress in percentage as well as the amount of kilobytes of received data. Additionally the estimated time left is shown and updated couple of seconds. An animation of a flying document shows that the download has not stalled.

Known Uses Netscape's Download box, Apple's file copy

Problem The user may accidentally select a function that has irreversible (side) effects.

Usability Principle Error Management (Safety)

Context Functions that have irreversible (side) effects or require a lot of resources to undo/reverse. The (side) effects may lead to unsafe or highly undesired situations. Often it is the combination of action arguments that makes it severe, and the user may not be aware of this, since it is normally safe.

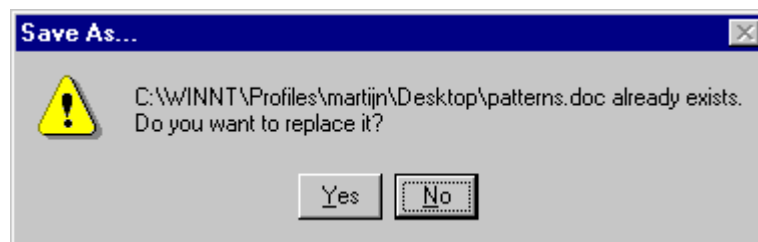
- Forces**
- The user needs to be protected but normal operation should not be changed.
 - The user is striving for speed while trying to avoid mistakes.
 - Some (side) effects may be more undesirable than others.

Solutions Protect the user by inserting a shield.

Add an extra protection layer to the function to protect the user from making mistakes. The user is asked to confirm her intent with the default answer being the safe option.

Rationale The extra layer causes the user to require 2 repetitive mistakes instead of 1. The safe default decreases the chances for a second mistake. The solution increases safety, reduces errors and increases satisfaction. However, it requires extra user action which leads to lower performance time.

Examples



A copy of the file already exists at the specified location. Overwriting it will result in loss of the copy. The default is "No" so that the speedy user has to take the effort of saying "Yes".



Here is another example, taken from a Sony' Mavica digital camera. The user has selected the format function and is asked to confirm the action, the safe option being default.

Known Uses Microsoft Explorer, Apple Finder

Related Patterns WARNING

Problem Each user is different and prefers to do things slightly different.

Usability Principle Adaptability (Flexibility)

Context The application is very complex and many of its functions can be tuned to the user's preference. Not enough is known about the user's preferences in order to assume defaults that will suit all users. The potential users can range from novice to expert.

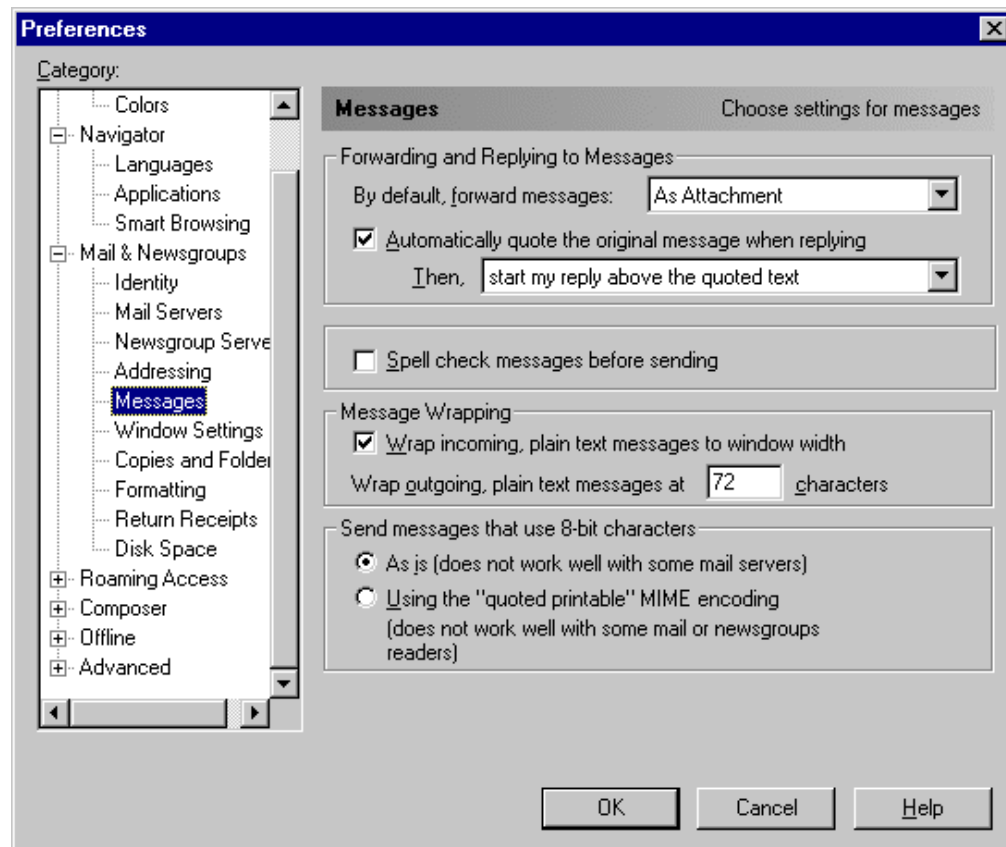
- Forces**
- Letting the user streamline usage can improve satisfaction but at the same time, it becomes more complex to use.
 - The user may not be familiar with the possible options for adaptation.
 - The user needs to know the consequences of the adaptations.
 - Many options are possible but users will only want to change a subset.

Solutions Allow the user to adjust their preferences.

Instead of forcing a single profile for each user, allow users to change their preferences. Provide choices for the user which will become the default for this user on further use. The options can often be grouped. If the number of groups is small, property pages can be used for each group but when the number of groups is high, use a tree. Each option must be explained so that the users know the consequences.

Rationale The navigational structure is used to choose the category of adaptations. Trees can accommodate more categories than tab controls. Showing all the settings together will give the user the impression of a "profile" that belongs to that user. Expert users can tweak the application for their particular purposes which increases satisfaction and possible performance. The solution increases satisfaction and performance times but decreases memorability and learnability.

Examples



This is the preferences dialog box from Netscape 4. In all kinds of areas, users can set their preferences.

Known Uses Netscape Preferences, IE5 Internet Options, Mac OS9 Multiple user feature.

Related Patterns GRID LAYOUT, LIST BROWSER

Problem At any point in time, users need to know what their possibilities are in order to decide what to do.

Usability Principle User Guidance (Visibility)

Context An application typically contains a lot of functionality and the user needs to know her possibilities at any point during use. The user is typically a novice or casual user and the functions are used infrequently.

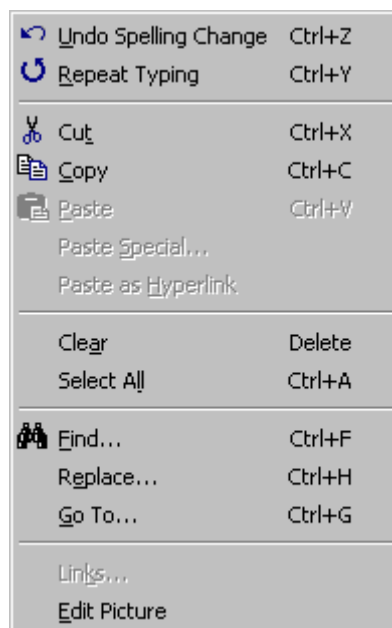
- Forces**
- The user may not be familiar with the meaning of a possibility.
 - The number of possibilities may be large and the user needs to locate the desired one.
 - Not all possibilities may be available in the current context, but the user may nevertheless want to know about the existence of the possibilities.
 - Integrating learning and using, while not hindering the expert.

Solutions Put the choices in a menu.

Show a list of all functions that are available in the current context. Make the functions accessible in one action. If the number of functions is high (> ~7), group the functions and make the groups distinguishable. If the total number of functions is low (in general or within a group), the list should show all functions and show which ones are possible to select in the current context. For users who are not familiar with the function label there should be a description available that explains the function. Functions should be ordered according to one or more of the following criteria; semantics, similarity, frequency of usage or alphabetically. If the list of possibilities does not differ much between contexts, the list should show all possibilities and highlight the possibilities of the current context.

Rationale The list of functions gives the user an immediate overview of all possibilities. Humans are familiar with such list (e.g. a dinner menu) and will quickly recognise its function. The solution improves memorability and satisfaction. It decreases performance speed because extra actions are needed.

Examples



This is the "Edit" submenu from Word97. This menu shows all the functionality related to editing things in a document. The functions are semantically grouped and only the possibilities of the current context are highlighted.

Known Uses Any application with menu's, Web site menu's

Related Patterns GRID LAYOUT

Problem Users want to quickly know information about an object they see and possibly modify the object.

Usability Principle Guidance (Constraints)

Context An application where several visual objects are manipulated, typically drawing packages or browsing tools.

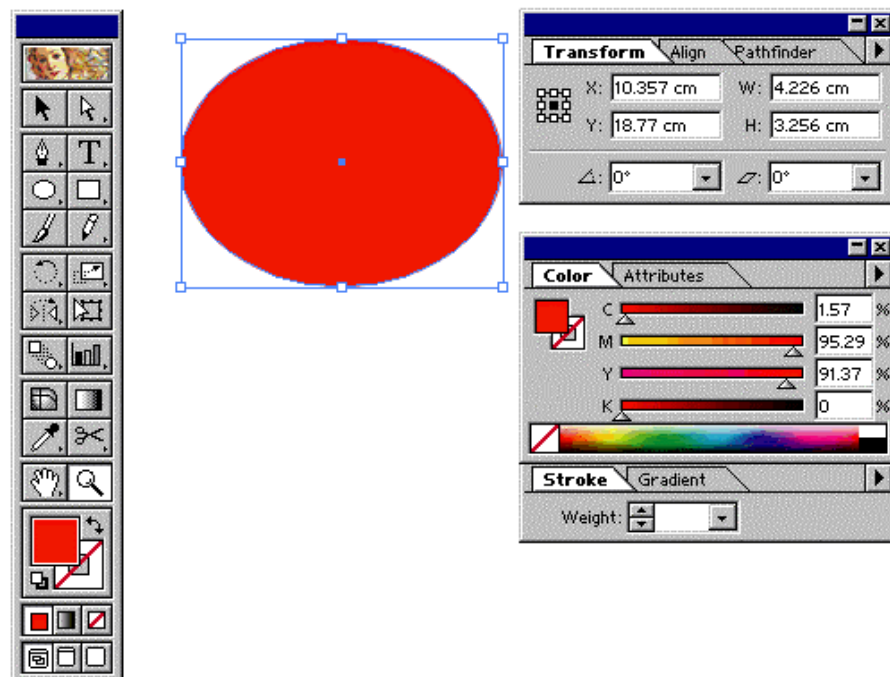
- Forces**
- Many objects can be visible but the user usually works on one object at a time.
 - The user wants both an overview of the set of objects and details on attributes and available functions on the objects.
 - The user may also want to apply a function to several objects.

Solutions Introduce a *focus* in the application.

The focus always belongs to an object present in the interface. The focus i.e. object the user is working on determines the context of the available functionality. The focus must be visually shown to the user for example by changing its colour or by drawing a rectangle around it. The user can change the focus by selecting another object. When an object has the focus, it becomes the target for all the functionality that is relevant for the object. Additionally, windows containing relevant functionality are activated when the focus changes.

Rationale Humans are used to working with objects. The focus is the equivalent of "grabbing an object". Hence it is natural that the functionality that belongs to the object is activated and presented to the users. This reduces the number of actions needed to select the function and execute it for a specified object. The solution improves the performance speed and memorability.

Examples



This screenshot is taken from Adobe Illustrator. With this drawing application the user can draw graphical objects. When the circle is selected the object specific windows display the state of the object. This example shows the colour and dimensions of the circle.

Additionally menu items that are not available for this object are disabled.

Known Uses Adobe Illustrator, Many other applications that use direct manipulation, Microsoft File Explorer, Forms.

Related Patterns SETTING ATTRIBUTES

Problem The user needs to supply the application with data but may be unfamiliar with which data is required or what syntax to use.

Usability Principle User Guidance (Constraints)

Context Any system where structured data must be entered. Data such as dates, room numbers, social security numbers or serial numbers are usually structured. The exact syntax used for such data may vary per country or product.

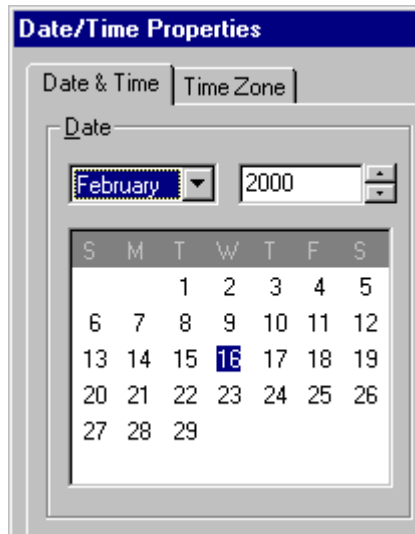
- Forces**
- When the data is entered using an unexpected syntax, the data cannot be used by the application.
 - The user may be familiar with the data but may not know the exact required syntax.
 - The user strives for entry speed but also wants it to be entered correctly.
 - Cultural conventions determine what the user expects the syntax to be. For example, day/month/year is usual in Europe while month/day/year is used in the United States.

Solutions Only allow the user to enter data in the correct syntax.

Present the user with fields for each data element of the structure. Label each field with the name of the data unit if there can be doubt about the semantics of the field. The field does not allow incorrect data to be entered. Avoid fields where users can type free text. Additionally, explain the syntax with an example or a description of the format. Provide sound defaults for required fields, fields that are not required should be avoided or otherwise marked as optional. When optional fields are used, the consequences for the user must be explained.

Rationale The main idea is avoid entering incorrect data by not making it possible to enter wrong data. By showing the required format the chances of errors are reduced because the user is given complete knowledge. However, because the user now has to give multiple data inputs instead of one, more time is needed to enter the data. The solution reduces the number of errors and increases satisfaction but the performance time may go down.

Examples

The image shows a screenshot of the 'Date/Time Properties' dialog box from the Windows operating system. The 'Date & Time' tab is active. Under the 'Date' section, there is a dropdown menu showing 'February' and a text box showing '2000'. Below these is a calendar grid. The grid has columns for days of the week (S, M, T, W, T, F, S) and rows for the months. The date '16' is highlighted in blue in the grid.

This snapshot is from the date and time control panel in MS Windows. Entering the date is split up in three input areas. Each of the input fields allows only valid entries. Entering an invalid date becomes impossible.

Known Uses MS Windows Date/Time control panel

Related Patterns GRID LAYOUT

Problem The user needs to access an amount of information which cannot be put on the available space.

Usability Principle User Guidance (Natural Mapping)

Context Systems with a lot of states, functionality and objects, that are relevant only in groups.

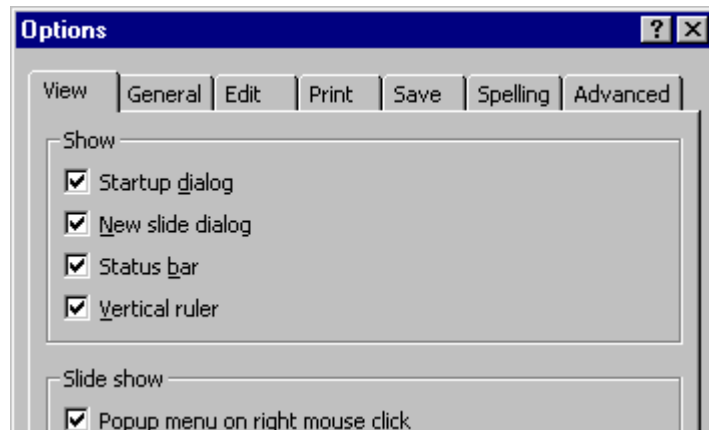
- Forces**
- Large amounts of data require a lot of space but the available display space is limited.
 - Large amounts of data are usually not unrelated and can be divided into categories that match the user's conceptual model of the data.

Solutions Show the information in several spaces and allow the user to navigate between them.

Group the information elements in separate spaces. Allow the user to select only one space at a time. Each space should be labelled with the name of the category. All the individual spaces should be accessible in one action from an area that is intended for navigating. Navigation areas should be placed at the top or left of the spaces and must be connected to the areas. If the number of spaces is low (e.g. <8), the navigation area should be placed at the top. When the number of spaces is large, the navigation area should be placed on the left side of the spaces using a tree structure.

Rationale Grouping of elements makes it easier for the user to find a particular element. Placing the navigation at the top or left reduces the needed screen space. The reason for this is that the labels are usually text which is wide and small. Additionally, in western society people read from left to right and from top to bottom. The solution improves the performance time.

Examples



This is the "options" window of MS PowerPoint. The users can navigate through all the options using the tab control. Each tab shows a category of options.

Known Uses MS PowerPoint, Netscape, Many web sites.

Related Patterns WIZARD

Problem The user needs to know how to control an object in the interface which resembles an object the user knows from the real world.

Usability Principle Analogy (Conceptual Models)

Context Applications that use real world metaphors and direct manipulation in the interfaces. The objects in the interfaces resemble real world objects and interaction is suggestion to resemble real world interaction.

Forces

- Interaction is done with input devices but each device has a limited set of manipulation possibilities such as movement, rotation, force feedback, and degrees of freedom.
- The way the object is manipulated in real life creates expectations about how the interaction will take place.

Solutions Match the input device and the widget used.

Use a combination of widget and input device that match in terms of the possible movements. Movements include the degrees of freedom and types of feedback (e.g. haptic or visual). For example, If the widget requires rotation around the z-axis, use an input device that supports it. If there is a mismatch, choose a different input device or change the widget.

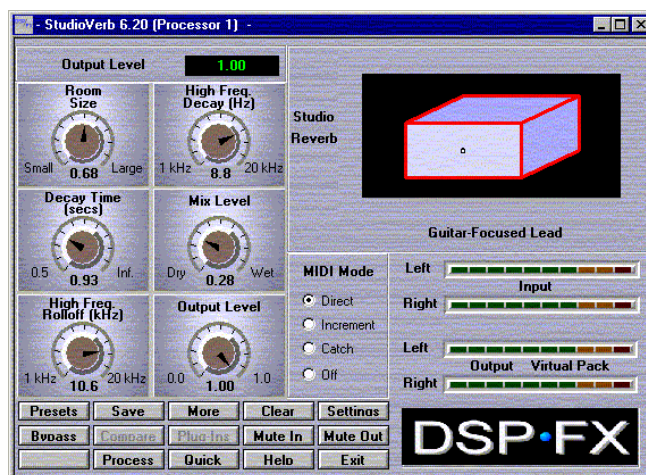
Rationale If real world metaphors are used it is important that the metaphor is used in the same way as in the real world. Otherwise, the user may initially recognise the widget but then finds unexpected behaviour. The solution decreases learning time, improves performance time and satisfaction.

Examples



This image shows how a special input device, called The Phantom, is used in a 3D modelling application. This resembles how a sculptor would create a statue in the real world. The phantom has many degrees of freedom combined with haptic feedback so that the sculptor can actually feel the surface of the statue that is being designed.

Counter example



This screenshot is taken from DSP-FX, a package with real-time audio effects. In the real world, physical devices with this functionality use turning knobs as well. One of the reasons for this is that they require less space than a slider, especially when the devices are 1 inch high. In this example, the same metaphor is used. However, the knobs require rotation which is not a problem for human hands but is very unnatural when using a mouse. A mouse is not capable of rotation and the user has to mimic rotation by moving the mouse in a circle. Effectively the knobs function as sliders and in this case they do not even save screen space.

Known Uses The Worldbeat system, Urban Planning workbench

Problem The user is looking for an item in a small set and tries to find the item by browsing the set.

Usability Principle Compatibility (Natural Mapping)

Context In many applications the user needs to find an item e.g. a file, a presentation, video clip, or an image, for which a visual or auditory search criterion is more effective but the index of the set is not audiovisual (e.g. a text label).

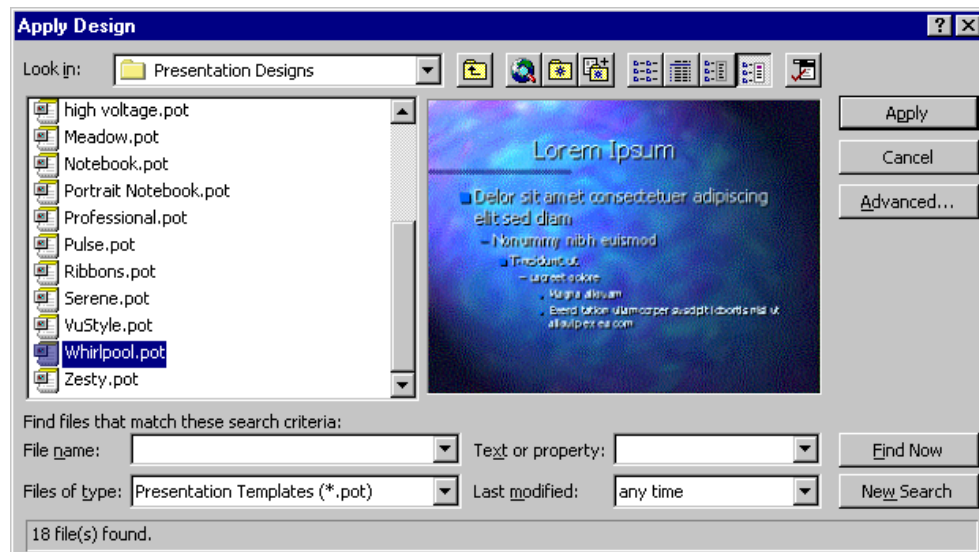
- Forces**
- Although the user just wants to see the real item, the resources needed to display the real item may be unavailable.
 - The user sees index name but might not be able to identify the item by just the index name.
 - The user is looking for an item but needs the index name as a search result for use in other tasks.

Solutions Allow the user to preview the item.

Show a representative in-place preview of one or more items. The preview can use fewer resources such as screen, loading time, or quality than the original. The preview should be sufficiently good for identifying it. If the number of items in the set is small, a preview may be shown for all items (if possible within constraints such as screen space etc). If the set contains many items or when the label is important to the user, show a preview of one item only alongside the label of the item. The preview should then be positioned near the selected item to enforce the link between them. When the selection changes the preview is also immediately updated.

Rationale The search time is decreased because the user can "see" or "hear" if the item was found. Otherwise, users would need to open the item first before they know if it is the right one. Because the preview uses less resources than the original it is much more efficient and searching becomes more effective as long as the preview is representative. The solution improves performance time and satisfaction.

Examples



This example is taken from Microsoft PowerPoint. The user can select designs and is presented with the filenames of the design definition files. What the user really is interested in is choosing the design on how it looks. The preview area shows a miniature example of a standard sheet with the design applied.

Another example is the "thumbnails" in a PDF document. The Acrobat viewer shows a thumbnail of each paper in order to "visually" search for a page in a document. Web pages often use thumbnails as well for browsing through images.

Known Uses PowerPoint, ACDSee, Acrobat, Adobe Photoshop, Web pages

Problem The user needs to find a regularly used item in a large set of items.

Usability Principle Minimizing Actions (Flexibility)

Context The user is looking for an item that is contained in a large set of items. The item is of importance and the user requires it regularly. Items are typically files, colours, web pages or database records and are part of large collections, respectively a file system, the web or a database.

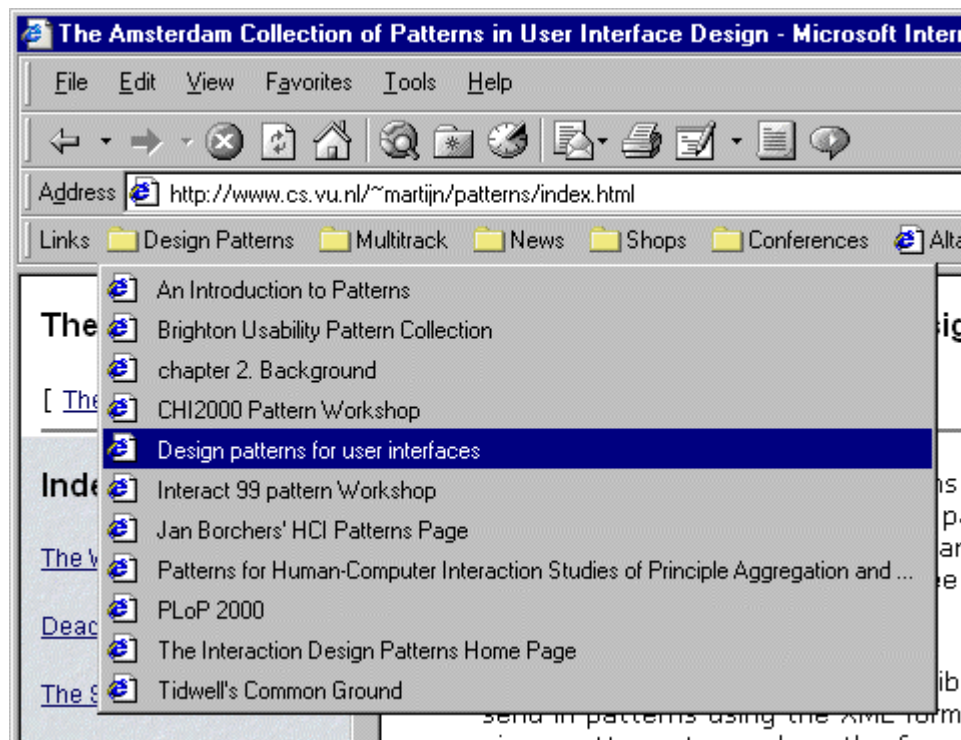
- Forces**
- The user knows of the item's existence and uses it regularly; hence the items need to be at hand.
 - The number of items contained in the space influences the time it takes to search for the item each time it is needed.
 - The user is interested in the item but may use the item only for a short period of time.

Solutions Allow the user to use favourites that point to the items of choice.

Introduce a possibility to create and remove favourites. A favourite is a label that points directly to the particular item. A label can be a textual description or something else that helps the user identify the item. Favourites should be made accessible in a minimum number of user actions, typically directly or from a menu. However, the number of favourites can become high. Therefore, it should be allowed that favourites are hierarchically ordered, if necessary.

Rationale The user is likely to know this concept from their experiences in reading books using bookmarks. The favourite allows the user to get the item without remembering the precise location of the item. Finding it again is reduced to searching in the set of favourites that is much smaller than the whole collection, hence searching time is reduced. The solution increases the performance speed and satisfaction.

Examples



This is the favourites menu from Internet Explorer 5. A web page is made accessible by selecting the label.

Another example is a colour chooser. In this case the item is a colour and the user can make "user defined" colours to access a colour quickly.

Known Uses Web browsers (Netscape, IE, Opera), Most recent used file section in many applications, Colour choosers.

Related Patterns HINTING

Problem The user needs to know where to find the possible commands and how to activate them.

Usability Principle Consistency, User Guidance (Visibility)

Context In every application the functionality needs to be made known to the user.

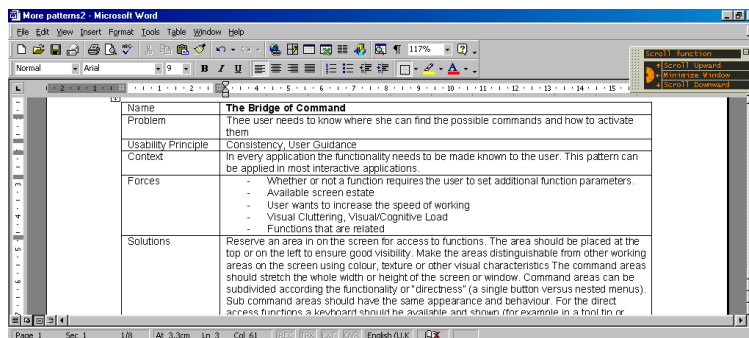
- Forces**
- Some functions need additional function parameters to be set by the user before it can be executed.
 - The available screen estate is limited and the main working area should be kept as large as possible.
 - Immediate function access using widgets increases the speed of interaction but consumes valuable screen estate.
 - Many different visual elements on a display clutter the overall image and increase the cognitive load to handle the different elements.
 - Some functions are more often used than other functions.

Solutions Put the commands in a specific recognizable area.

Reserve an area on the screen for access to functions. The area should be placed at the top or on the left to ensure good visibility. Make the areas distinguishable from other working areas on the screen using colour, texture or other visual characteristics. The command areas should stretch the whole width or height of the screen or window. If the number of functions is large they should be conceptually grouped. The command areas can then be subdivided to give access to a group of commands. Sub command areas should have the same appearance and behaviour. The commands should be accessible as direct as possible especially for often-used functions. The command area should not occupy more than 1/6 of the screen estate. The number of function entrances should not exceed 60 so that the user does not get a visual overload.

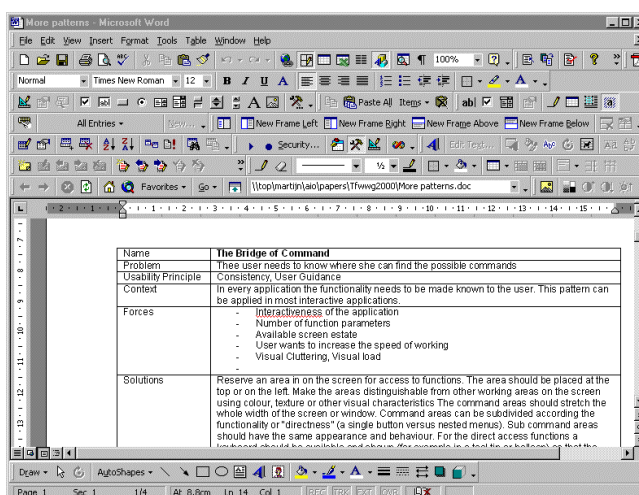
Rationale Providing a command area at a fixed location gives the user a consistent way to find functionality. Providing direct access areas provides direct access to often used function and facilitates quick interaction. The placement of the area is such that the area is always visible. The solution increases memorability and satisfaction but reduces speed of interaction.

Examples



This screenshot is taken from Word2000 and shows just two toolbars and a menu. The command area is clearly present but does not occupy too much screen space.

Counterexample



This screenshot is Microsoft Word and just about all the toolbars activated. The screen is highly cluttered and the user loses the overview. The command area occupies too much space and simply contains too many direct access functions. Even worse, the toolbars not only contain command shortcuts but they also contain status information or attribute settings. This difference is not visually distinguishable.

Known Uses Any windows based application or web page.

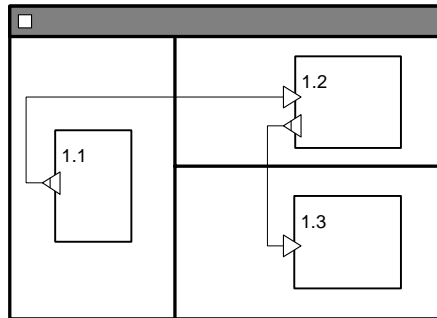
Problem The user needs to find an item in a collection of containers.

Usability Principle Grouping of Elements (Natural Mapping)

Context Many applications contain aggregated data, which the user must browse through. Quite often, the user wants to invoke a function taking one of the parts as input parameter.

- Forces**
- The importance for the task that the user sees all containers and an item at the same time.
 - The number of items is large but only a portion of the items needs to be visible at a particular moment.
 - The user may need to switch from one container to the other.

Solution

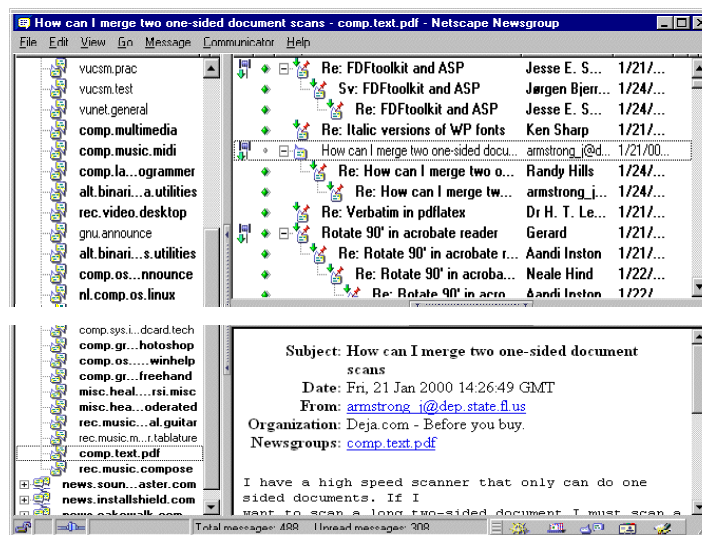


Split up the screen in three areas showing the containers and the final selection.

Split a window into three panes, one for the viewing a collection of containers, one for viewing a container, and one for viewing individual items. The selection of a container should determine the content of the container pane, and the selected item should determine the content of the item pane. The selections may be used as parameters to invoked functions. Each pane should be specialized to the type of content it presents. E.g. if the containers form a hierarchy a tree pane providing selection of leaf nodes could be used. The panes should be individually scrollable and resizable.

Rationale By configuring the panes according to the western way of reading (left to right, top to bottom), we support the causal relationship based on selection. By providing selection, other functionality besides navigation is supported. The layout should aid in understanding the causality among the panes. Each pane can be tailored to the domain and user. Individually resizable panes give user freedom.

Examples



This is Netscape's Mail/News viewer. In the left pane the set of containers (in this case newsgroups) is displayed. The other panes show the list of messages in the selected group and the selected message.

Known Uses Microsoft Outlook; Netscape Mail/News, Eudora Mail

Related Patterns GRID LAYOUT

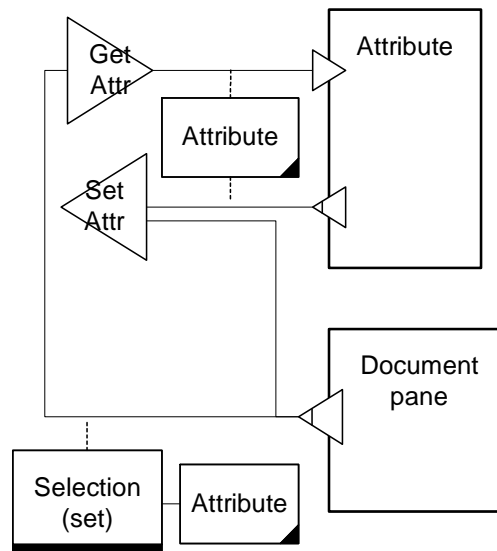
Problem Users want to see the attributes of the object they are working on and additionally they need to know how to modify them.

Usability Principle User Guidance (Visibility)

Context In many applications, a document can hold many different objects with many different attributes.

- Forces**
- Some of these attributes are partly visible in the document, but are difficult to modify by directly manipulating the objects in the context of the document.
 - Setting attributes requires different controls than when visualizing them.
 - Objects are usually of several types and have different sets of attributes.
 - Some attributes are unique for one type of object while other attributes might be shared by objects.

Solutions



Create special widgets that show the attribute values.

Frame the document pane with dialog elements for showing the various attribute types. Include only the most common/generic ones and let the user customize the available elements. Provide access through these to dialog elements for setting the same attribute types. Populate the dialog elements with the attribute values of the current selection, or in the case of multi-selection, the common attribute values. Dim the dialog elements holding attributes not supported by the selection. When accessing a dialog element and setting its value, set the corresponding attribute value of the selection.

Rationale Many attributes are partly visualized in the main document pane, but not with the same precision as a specific dialog element can provide. By providing specialized dialog elements for each attribute, precision is improved. Space can also be saved if a pop-up or pull-down style of dialog element is provided for input. By providing specialized dialog elements for each attribute type, the values should be easier to read and understand. The set of undimmed dialog elements for a particular selection, aids the learnability of the available attributes of each object type. Selection based population of elements, lets the user control the focus. By letting the user set the attributes using the same dialog elements, the user is provided with a sense of direct manipulation.

Examples Word processors and style (font, justification, ruler) attributes, drawing applications and graphic (font, color, pen) attributes, spreadsheets and cell (formatting, font, color) attributes.



Known Uses Word, Visio, Excel

Related Patterns COMMAND AREA

Problem The user may unintentionally cause a problem situation which needs to be resolved.

Usability Principle Error prevention (Safety)

Context Situations where the user performs an action that may unintentionally lead to a problem.

- Forces**
- Work may be lost if the action is fully completed.
 - The system can or should not automatically resolve this situation so the user needs to be consulted.
 - Frequency of occurrence.
 - The number of ways in which the problem can be resolved.
 - The likeliness that the user intentionally does the task, e.g. the user wants to do it.
 - Some actions are difficult or impossible to recover from.
 - Users may not understand why an action could be damaging.
 - Users may not understand the consequences or options.
 - The severity of the problem if it occurs i.e. how bad is it?

Solutions Warn the user before continuing the task and give the user the chance to abort the tasks.

The warning should contain the following elements:

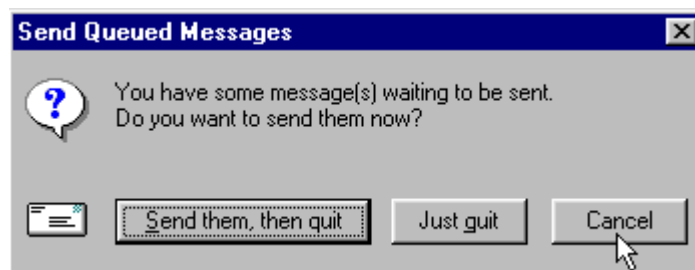
- A summary of the problem
- The condition that has triggered the warning
- A question asking the users whether to continue the action or take on other actions.
- Two main choices for the user, an affirmative choice and a choice to abort.

The warning might also include a more detailed description of the situation to help the user make the appropriate decision. The choices should be stated including a verb that refers to the action wanted. Do not use Yes/No as choices. The choices are answers to the question that is asked.

In some cases there may be more than two choices. Increasing the number of choices may be acceptable in some cases but strive to minimize the number of choices.

Rationale By stating the triggering condition the user can understand why the warning appears. Once that is understood the question leaves the user with only two options. By providing only two options the choice is made simple for the user: continue or abort. More options make it increasingly difficult for the user to make a decision. By using a verb in the options the user immediately knows what the user is choosing for whereas Yes/No choices require the user remember exactly what the question was. The solution decreases errors and increases satisfaction.

Examples



This screenshot comes from Eudora 4, if you try to exit the program. It shows that even three choices can be acceptable.

Known Uses Eudora, Installshield installers (when exiting)

Related Patterns SHIELD

Problem The user needs to know how to select functions.

Usability Principle Incremental Revealing (Visibility)

Context Applications where the functionality is accessible in more than one way, e.g. through menus, using keyboard shortcuts, or through toolbars. This pattern can be used to make the user aware of the other possibilities in a subtle and non-obtrusive way

- Forces**
- The available screen space may be limited so there is no space for extra visual hints.
 - The user needs some way of discovering and learning these alternatives and possibly more efficient ways, in a non-obtrusive way.
 - The user may or may not already know the other ways to access the function.
 - The number of ways to activate the function determines the number of possible hints.

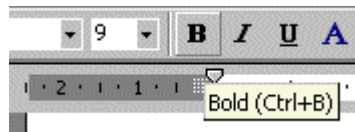
Solutions Give the user hints for other ways to access the same function.

When accessing a function in one way, provide hints for other ways to access the same function. One possibility is to use multiple labels; one label for each way the function can be accessed. For example, if the function has a keyboard shortcut, show the key combination. If there is an icon shortcut for the function, show the icon. Always show the main label and show other labels directly if possible within the constraints.

Other possibilities are to use helper agents or delayed messages that react on user actions. For example, a tool tip is displayed when the user holds the mouse over a widget for approximately two seconds.

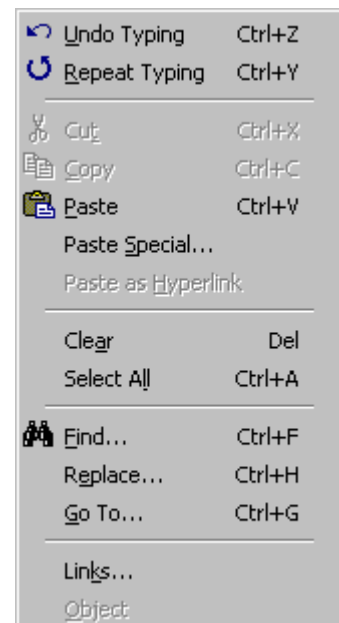
Rationale Chances are high that the user is familiar with at least one way to access a specific function. By showing other labels such as the key shortcut or an icon, the user will learn more associations for the function access. At some point the user may "see" the icon in a toolbar and use it instead of the menu. In the same way, the user may prefer keyboard access over mouse access. The solution increases learnability and memorability. When the user actually starts using other ways of selecting functions the performance speed may also increase.

Examples



These screenshots are taken from Word2000. They shown to possible instances of this pattern: one using tool tips and the other using menus with icons.

In the menu, there is space to include the icon and shortcut but the toolbar icon does not allow this. In that case the information is displayed in a tool tip that pops up after a short delay. That way advanced users are not bothered with windows that pop up all the time.



Known Uses Tool tips, Office2000 menus.

Related Patterns COMMAND AREA

Problem The user is creating or modifying an object and needs to know which edit function is selected.

Usability Principle Immediate Feedback (Feedback)

Context In many direct manipulation applications the users first selects a tool/function, thus entering a special mode/state, and then works on an object. Since such applications usually offer many functions to create or modify objects.

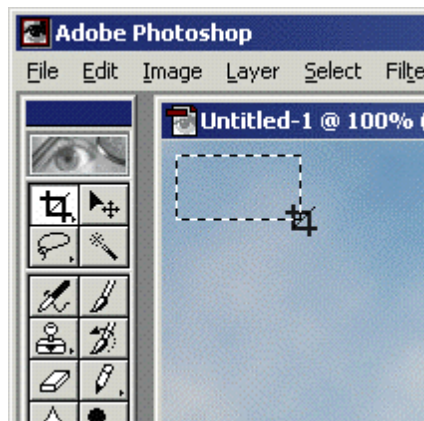
- Forces**
- Not every function may have an icon or shape.
 - Completing a function may cause several intermediate states which may also need to be shown.
 - The user needs immediate feedback on which function was selected, i.e. which mode/state the system is in.

Solutions Show the interface state in the cursor.

The interface state changes many times during interaction, for instance when a function is selected or when an action such as dragging is performed. Therefore, show the current state to the user by changing the cursor. The cursor can be changed to an icon or some other shape that gives feedback about the current interface state. Change the cursor back to a neutral cursor if the function is completed or deactivated.

Rationale The cursor gives extra feedback about the active function. The user watches the cursor when performing a function so it is the most appropriate place on the screen to give feedback i.e. the user does not need to look at another portion of the screen. The solution increases satisfaction and may decrease errors.

Examples



This screenshot is taken from Adobe Photoshop 5. The user has selected the Crop function as indicated on the left function panel. The cursor has changed and now has the same shape as the function icon.

Known Uses Photoshop, Illustrator, Powerpoint

List browser	by Martijn van Welie and Hallvard Trætteberg	Revision 4
---------------------	--	------------

Problem The user needs to browse or process several items out of a set or list.

Usability Principle Minimizing Actions (Natural Mapping)

Context In many applications the user needs to go through a list of items. For example, when reading news items on a news website or when browsing through the results of a database query. The user typically selects an initial item out of a list and then wants to move on to other articles, typically the next one or the previous one. In general, the user wants to read several items of the list in the order, backwards or forwards.

- Forces**
- The user wants to see an overview of the set/list and at least one item, but the screen space may not be sufficient to show both.
 - The user needs to be able to select an individual item as well as to process several items.


Solutions Find a natural ordering and allow the user to navigate directly from one item to the next and back.

Based on knowledge of the task, impose an ordering on the set/list. The ordering can be based on a ranking e.g. relevance, on field values e.g. date/time or a projection of an existing structure e.g. flattening/traversing a tree. Even an arbitrary ordering is useful.

For every item that is presented to the user, a navigation widget allows the user to choose the next or previous item in the list. The ordering criterion should be visible (should it be user configurable). To support orientation, the current item and a partial index list should be clearly visible. If increased complexity is not considered a problem, the current item number indicator could be a dialog that supports jumping to arbitrary items, dialog elements for jumping to the top and bottom could be added, as well as an element for switching to a full list/index/TOC view.

Rationale By giving the user a simple navigation model and allowing the user to go directly to the next or previous item, the user does not need to go back to the index to select the next item. The solution improves the performance speed and satisfaction.

Examples

 ATLAS F1 NEWS SERVICE

Ecclestone Wants Cheats Named

Thursday April 13th, 2000

Formula One supremo Bernie Ecclestone wants any teams found to be breaching the rules regarding traction control named.

Responding to last week's press conference by FIA president Max Mosley where Mosley claimed that a team may have cheated last season as they were using traction control, Ecclestone said it was essential any culprits were named to stop the whispers which have seen fingers pointed at every team along pit-lane.

"If they have the evidence then the FIA should name the team they believe is cheating," Ecclestone told Autosport magazine.

"At the moment a cloud of suspicion is hanging over everyone and the finger is being pointed all over the place. It is affecting everyone which is not fair."

Mosley said he would not reveal the name of the alleged culprits until FIA had 100 percent proof of a transgression.

[\[Previous \]](#) | [News Index](#) | [Next \]](#)

Latest Headlines:

- ▶ (04-13-2000): [Ecclestone Wants Cheats Named](#)
- ▶ (04-13-2000): [Speed Limiters Not Banned](#)
- ▶ (04-13-2000): [Paul Stewart Retires from Jaguar due to Cancer](#)
- ▶ (04-12-2000): [Prost Bids To Relive Formula One Glory](#)
- ▶ (04-12-2000): [Testing Ends at Fiorano](#)

[\[Homepage \]](#) | [Forums](#) | [Bet Your Nuts](#) | [Chat & Live Coverage](#) | [Bookstore](#) | [Search & Archive](#)]

Examples are news websites or the event viewer. One news item is shown and then you can go to the next one. This way the user doesn't need to go back to the index. Preferably the index is also there, at least partially (showing the context of the current item).

Known Uses Atlas F1 News Website, NT4 Event viewer

Related Patterns WIZARD, NAVIGATING BETWEEN SPACES

Problem The user needs to find an item in an ordered set.

Usability Principle Immediate feedback (Feedback)

Context This pattern allows the user to dynamically narrow the search depending on the immediate feedback given by the continuous filter.

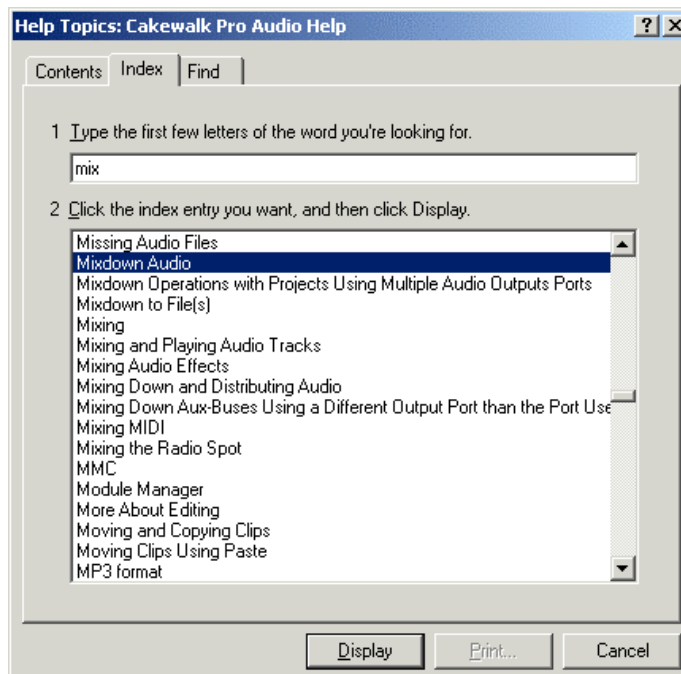
- Forces**
- The user is searching for an item in a large ordered set and may not be familiar with the exact item, nor is the user sure the item exists.
 - The user searches for an item but the search term may lead to multiple results.

Solutions Provide a filter component with which the user can in real time filter only the items in the data that are of his interest.

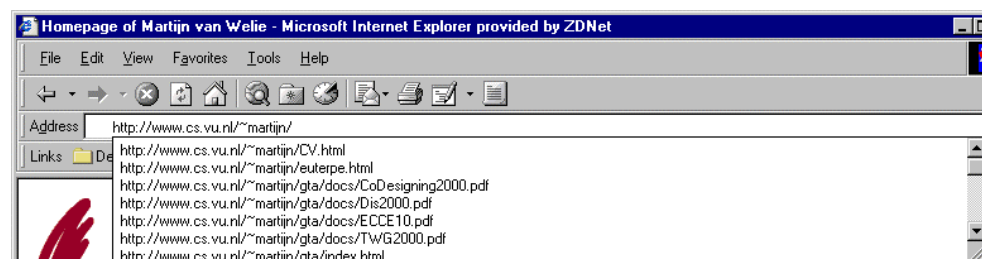
The filtered set is shown concurrently with the search term from the moment the user starts entering the search term. If relevant, the closest matches are highlighted while some previous and successive items might be shown as well.

Rationale Because the user gets immediate feedback on the search term, the user searches very efficiently and may even see other relevant items. Because the filtered items are shown the user can adjust the search term in real time or even bypass completing the search term and go directly to the preferred item. The solution improves the performance time and satisfaction.

Examples



This screenshot taken from Cakewalk 9 uses the common help functionality. In the index function the user is guided towards the item while typing.



This screenshot shows the URL history of Internet Explorer 5. As you type in a URL it shows the list of possible URLs that match the unfinished URL.

Known Uses Help systems (Cakewalk, MS Word 2000, Visual Studio 6), Internet Explorer 5, IntelliSense.

Related Patterns FAVOURITES

Status of the Patterns and the Collection

The patterns in this collection have mainly been developed by examining often-used applications and by going through guidelines. Most of our efforts have concentrated on writing down the patterns with a consistent focus as described in the previous sections. The patterns presented here are part of an ongoing collaborative effort; patterns were distributed by email and discussed by a group of three persons. Lately, we have started using a BSCW collaborative workspace to host a "virtual writers' workshop" for validating and improving the patterns. We hope that our work sensitises others so that we can together work on the development of a true pattern language.

Acknowledgements

Thanks to David Kane and Nicolò de Faveri Tron for discussing some of these patterns with us. Also many thanks to Jutta Eckstein our shepherd who gave a many insightful comments.

References

Bayle, E. (1998), Putting it All Together: Towards a Pattern Language for Interaction Design, *SIGCHI Bulletin*, vol 30, no. 1, pp.17-24.

Hartson, H.R. (1998), Human-computer interaction: Interdisciplinary roots and trends, *The Journal of Systems and Software*, vol 43, pp.103-118.

Mahemoff, M. J. and Johnston, L. J. (1998), Pattern Languages for Usability: An Investigation of Alternative Approaches, *Asia-Pacific Conference on Human Computer Interaction (APCHI) 98*.

McKay, E. N. (1999), *Developing User Interfaces for Microsoft Windows*, Microsoft Press,

Norman, D. (1988), *The Design of Everyday Things*, Basic Books,

Perzel, K. and Kane, D. (1999), Usability Patterns for Applications on the World Wide Web, *PloP '99*.

Rijken, D. (1994), The Timeless Way .. the design of meaning, *SIGCHI Bulletin*, vol 6, no. 3, pp.70-79.

Tidwell, J. (1998), Interaction Design Patterns, *PloP '98*.

van Welie, M., van der Veer, G. C., and Eliëns, A. (1999), Breaking down Usability, *Proceedings of Interact '99*, Edinburgh, Scotland.