

**The use of guidelines
for designing user interfaces
in commercial software development**

Thomas Vogt

Studienarbeit

November 1999

**Department for Informatics
University of Hamburg**

Advisor: Prof. Dr. Horst Oberquelle

Table of contents

1	Introduction	1
1.1	Motivation for investigating this subject.....	1
1.2	Aspects to investigate in this paper.....	2
1.3	Steps to investigate these aspects.....	3
2	Which types of guidelines were part of the investigation?	4
2.1	Web resources	4
2.1.1	Long documents.....	4
2.1.2	Short papers.....	5
2.2	Comparison with paper-based books	5
2.3	Taxonomy of guidelines	6
3	Which aspects deal these guidelines with?.....	8
3.1	Long documents.....	8
3.1.1	Aspects of the development process.....	9
3.1.2	Aspects of the software product on an abstract level.....	9
3.1.2.1	Consistency	9
3.1.2.2	Metaphors.....	10
3.1.2.3	Visual design	11
3.1.2.4	Help.....	11
3.1.2.5	Error handling	12
3.1.3	Aspects of the software product on a specific level.....	12
3.1.3.1	Windows	12
3.1.3.2	Menus	13
3.1.3.3	Icons	13
3.1.3.4	Dialog boxes	14
3.1.3.5	Layout and visual alignment	15
3.1.3.6	Controls.....	15
3.1.3.7	Color	15
3.1.3.8	Font.....	16
3.1.3.9	Mouse and keyboard navigation.....	17
3.1.3.10	Help.....	17
3.1.3.11	Error messaging	18

3.2	Short papers.....	18
3.2.1	Aspects of the development process.....	19
3.2.2	Aspects of the software product.....	19
4	Usability problems with these guidelines.....	21
4.1	Problems with the contents of UI guidelines.....	21
4.1.1	Consistency and usability.....	21
4.1.2	Generalization.....	22
4.1.3	Focus and level of detail.....	22
4.1.4	Illustration.....	22
4.1.5	Conflicts.....	23
4.1.6	Correctness.....	23
4.1.7	Transience.....	23
4.1.8	Obviousness.....	24
4.1.9	Lack of bad examples.....	25
4.1.10	Support by development tools.....	25
4.1.11	Size.....	25
4.2	Problems with handling UI guidelines.....	26
4.2.1	Problems with developing your own UI guideline.....	26
4.2.1.1	Introducing the document.....	26
4.2.1.2	Updating.....	26
4.2.1.3	Purpose.....	26
4.2.1.4	Responsibility.....	26
4.2.2	Problems with using a UI guideline from third parties.....	27
4.2.2.1	Number.....	27
4.2.2.2	Incompleteness.....	27
4.2.2.3	Access.....	27
4.2.2.4	Integration.....	27
5	Ideas to make guidelines more usable.....	29
5.1	Ideas to solve the mentioned problems.....	29
5.1.1	Consistency and usability.....	30
5.1.2	Focus and level of detail.....	30
5.1.3	Illustration and Lack of bad examples.....	30
5.1.4	Support by development tools.....	31
5.1.5	Size.....	31
5.1.6	Problems with developing your own UI guideline.....	31
5.1.7	Integration.....	31

5.2	Creating a UI guideline	32
5.3	Tools for Working with Guidelines	34
5.4	Other ideas	36
5.4.1	Guidelines for “macro” objects	36
5.4.2	Checklists	36
6	Summary	37
7	References	39

1 Introduction

1.1 Motivation for investigating this subject

Undoubtedly, the majority of commercially developed software products have badly designed user interfaces. One reason for this may be the tremendous design freedom that is offered to developers who can use color displays, windowing systems, mice, icons and all these controls offered in modern software-products. “This provides software developers with an opportunity to create truly innovative and creative designs. However, this increased design freedom also provides a greater number of opportunities to produce poor, ineffective designs.” [Gale 95]

There are exceptions but most of the users have big or at least some problems with the handling of most of the programs. These products have interfaces that are unnecessarily difficult to understand, they are hard to learn and badly matched to the needs of the users' normal work. Often the main aspect during development is to present the necessary functionality without taking special care how to design the user interface (UI) by which the product will be used. If at all, this will be done at a very late stage in the project, more or less as an additional feature of the product. “Since usability issues are often ignored until late in a product’s life cycle, there is a misapprehension that improving usability will increase costs. The key to the design of better quality products is to put resources into improving usability when the cost-benefit ratio is highest – which usually means early in design.” [Bevan] But how can (also small) software companies “put resources into improving usability” without the need to rent an expert for Human Computer Interaction (HCI) for the personnel?

What I want to investigate in this paper is the use of UI guidelines during the development process.

Guidelines can be used in a lot of different phases during the development life cycle of a UI. According to [Vanderdonckt 99b] they can be used in the phases of specification, designing, prototyping, programming, evaluation, documentation, and certification. Meanwhile they are also used to teach UI design based on guidelines throughout the whole life cycle. The possible benefits using guidelines are numerous. “At design time, they can provide designers with some assistance by helping them to orient design options to obtain UIs which are more usable, more adapted, tailored to contextual needs (e.g., user population needs, physical environment constraints, and task needs). Guidelines can also serve as requirements to be achieved by developers at programming time. At evaluation time, a new UI or a previously designed one can be submitted to guidelines checking to guarantee a minimal threshold of usability.” [Vanderdonckt 99a]

[Gale 95] takes a look at possible benefits from different perspectives: From the end user perspective, the use of UI guidelines can result in reduced errors, which means less frustration, increased confidence in the system, reduced training requirements, increased morale, improved use of system functionality, improved productivity and reduced resistance to the use of new technology. From the IT developers' perspective it can mean to maintain control over look and feel, control third parties during the tendering and implementation phases, reduce arbitrary design decisions and re-invention, capitalize on learning and iteration, enable production of re-usable software and reduce development time. At last from the business perspective, the potential benefits could be to produce usable systems enhancing customer service and satisfaction, increase market and product awareness, reduce training costs, facilitate helpline support, improve staff retention and increase user acceptance of new systems.

Other motivating factors [Gale 95] for the use of UI guidelines can be to achieve consistent user interfaces for numerous applications in an integrated environment, especially when developing for different platforms or with different development tools. Also to communicate an organizations corporate image to its customers a UI guideline can be helpful because it makes it easier to present this image throughout all the visual representations of the product(s).

Last but not least a UI guideline is something quite tangible when a company starts to apply HCI methods in the development process, which itself is not tangible at all but "can be viewed as a change in perspective and possibly culture." [Gale 95]

There are a lot of good reasons to use guidelines and I want to question if they can keep what they promise.

1.2 Aspects to investigate in this paper

The main aspect to investigate in this paper will be to what extend UI guidelines can be used in the process of commercial software development. For this purpose this investigation is done in the context of the development of the CAS-tool (Computer Aided Sales) ELWIS. ELWIS is currently developed at the Universidad Católica Boliviana (UCB) in Cochabamba for pirAMide Informatik GmbH, Hamburg, Germany. First I want to investigate, which UI guidelines are currently available. Because the technical progress of the development environment is quite fast, I want to focus mainly on sources from the last five years. I also will take a look at documents that are dealing with UI guidelines themselves. This will give a short overview of the current state of research about this subject.

Reviewing these sources I want to answer the following questions: What will help the developers of software in their commercial environment? Can existing guidelines for the design of user interfaces help these developers? What improvements can be

expected when using UI guidelines? Are there big benefits to expect? Are there pitfalls? What can be suggested to solve problems of using UI guidelines, if there are any?

Are the existing collections of design suggestions for user interfaces useful in commercial software development with good results? Are they easy to understand? Are they specific enough to be used? Do they show the correct points to the alerted developer which he has to take care of during designing a user interface?

Do the experiences of the praxis show perhaps that also tips can be given for the usage of these design suggestions themselves? What else can be done to get the biggest benefit from working with UI guidelines?

1.3 Steps to investigate these aspects

“Everyone who want to apply knowledge recorded as guidelines must start by collecting guidelines.” [Vanderdonckt 99b] This will be also my first step. What material is offered and can be used in the development process? What are the results of the research about the usability of software and from professional interface designers? What kinds of UI guidelines can be found? In the context of this work I cannot review the whole literature and all the sources found about this subject. I have to focus on a small amount of the offer. An overview of the found sources will be presented in the next chapter (Chapter 2, "Which types of guidelines were part of the investigation?").

These documents will be presented comparing and evaluating (Chapter 3, "Which aspects deal these guidelines with?"). Which aspects do they deal with? Someone may find design suggestions of different levels of abstraction, which means that the offer extends from general tips someone has to think about to very specific “receipts”. In which way do these guidelines deal with the development process and the software product? Do they give positive suggestions and hints (Do’s) and also warnings, what to avoid, and bad examples (Don’ts)? What have the found sources in common and in which aspects are they different? What are characteristics of the different kinds of UI guidelines?

The next step will be a close look at usability problems with these guidelines (Chapter 4, "Usability problems with these guidelines"). Are there general problems when working with UI guidelines? Cause different kinds of guidelines different problems? What reasons for these problems can be identified?

What can be done to solve these problems? At last I want to try to give some ideas to make guidelines more usable (Chapter 5, "Ideas to make guidelines more usable"). Are some kinds of guidelines more effective than other ones? Are there already approaches to overcome the difficulties arising from working with guidelines? May there be a need for a completely or partly different kind of UI guidelines?

2 Which types of guidelines were part of the investigation?

2.1 Web resources

I have to divide the resources I found in the Internet mainly in two groups: Long documents that are mainly complete UI guidelines in the common sense and short papers which in general are dealing with only one aspect of UI design, e.g. Dialog-Boxes or Icon-Design. I included them as part of my investigation because often also these short papers contain suggestions, tips and hints how to create a good UI although they don't try to cover the whole subject.

2.1.1 Long documents

These are the UI guidelines offered from big software companies such as

- Apple ([Apple 95] and [Apple 97]),
- IBM ([IBM 97]),
- Microsoft ([Microsoft 97]) and
- Sun ([Sun 99]).

Also large software development departments have developed guidelines for designing a UI like those at

- Ameritech ([Schumacher 92]) and
- the Data Systems Technology Division/Code 520 at the Goddard Space Flight Center ([NASA 96]).

The characteristic these documents have in common is that they try to be complete UI guidelines. The guidelines from Apple, Microsoft and Sun want to support the development of their own environment, that is in detail the Macintosh operating system, MS-Windows or Java. The other two guidelines are not close related to a specific development environment. All these documents fulfill the definition for a style guide given in [Vanderdonckt 99b]. What these documents deal in detail with will be discussed in the next chapter (Chapter 3, "Which aspects deal these guidelines with").

Finally, there are long documents from researchers. These are not complete guidelines but also large documents that contain tips, suggestions and design rules intended to help designing a UI. These are from

- John Harrison ([Harrison 98]) and

-
- Jenifer Tidwell ([Tidwell 98a] and [Tidwell 98b]).

The size of the long documents varies from 142 pages [Apple 97] to 410 pages [Apple 95]. The guidelines from IBM and Microsoft were only used in HTML-format so it's difficult to tell their sizes in number of pages. But it can be estimated that they will fit in this range more or less although the IBM document will be very close to the lower and the Microsoft guideline will be far at the upper end. Exceptions of this range are [Tidwell 98a], which in fact is only a preface to [Tidwell 98b], and [Harrison 98], which is not written as continuous text but a set of 44 slides for a presentation.

2.1.2 Short papers

The short documents cover a lot of different kinds of documents. They are

- Conference proceedings (e.g. [Gale 95] and [Horton 97]),
- Articles from different journals (e.g. [Rizzo 96] and [Wildstrom 98]),
- Technical papers that are downloadable (e.g. [Bevan 96] and [Shackel 97]) and
- Publications in the Web (e.g. [Hawkes 98], [IIMRR 99], [TOG 99a]).

As mentioned above these papers mainly focus on only one subject of UI design. Their size varies from 2 pages [Wildstrom 98] up to 27 pages [Vanderdonckt 99b]. An exception is [Dufour 97], which contains 170 pages. This exceeds the size of a short paper but because it's not a UI guideline itself but a thesis dealing with the subject of tools for working with guidelines I mention it in this section.

As the form of the documents varies also their contents varies in a lot of ways. [Gale 95] is discussing how to prepare a UI guideline itself effectively. [Horton 96] and [Horton 97] cover both the subject of icon design. [Rizzo 96] is about the management of human errors. [IIMRR 99], [Schank 95] and [TOG 98h] are short introductions to the most important design principles. Other aspects of UI design these papers deal with are e.g. the difficulties of this process [Myers 93], text color and background color [Hawkes 98], psychological theory [Marchionini 91], accessibility [US DE 97], the Out-Of-Box-Experience [Berry], manuals [TOG 98g] and dialog boxes [VR b].

2.2 Comparison with paper-based books

The title of this section seems to be confusing, but I want to distinguish traditional printed, paper-based books from books, that are available in electronic form. Also some of the above mentioned Web resources are in fact books, but they were available in digital format, e.g. [Apple 95] and [Apple 97] were reviewed in PDF format and from [Sun 99] the HTML-version was used. Also some of the short documents are in fact paper-based. There are the ACM conference proceedings (e.g. [Gale 95] and [Horton 97]), which are also available in the Internet, and some technical papers (e.g. [Bevan], [Myers 93] and [Schank 95]), that are downloadable as Postscript, RTF or MS-Word

document. But I have chosen to make the difference between digital and paper documents.

Because the contents of paper-based documents does not differ from the digital documents, I limit the base for this investigation to digital documents.

But of course there are differences in the form between documents that are available online and paper-based documents. I want to mention these differences arising from different physical structures in this chapter shortly.

The following table containing the advantages (+) and shortcomings (-) of these two media is drawn principally from [Vanderdonckt 99a].

Paper-based documents	Online documents
+ Easiness	- Difficulty
+ Accessibility	- Limited accessibility
+ Legibility	- Reduced legibility
+ Timeliness	- Inconvenience
+ Hardware/software independence	- Hardware/software dependence
- Important size	+ Size independence
- Rigidity	+ Flexibility
- Fixed level of detail	+ Varying level of detail
- Static format	+ Dynamic format
- Limited reusability	+ Increased reusability

Table 1. Advantages and shortcomings of paper-based versus online documents.

Vanderdonckt evaluates the characteristics, which are considered as advantages for a paper-based document as shortcomings for an online document, and vice versa. Both versions are consequently seen as complementary to each other.

Preparing this paper in the described context the accessibility of the paper-based documents was in fact much more limited as the accessibility of the online documents. But this can be considered as an exceptional condition.

2.3 Taxonomy of guidelines

Before I describe the contents of the UI guidelines in detail in the next chapter I want to mention the different terms that are used when talking about guidelines. Of course the guidelines can have a different level of abstraction. In addition to the term ‘guideline’ someone can also find often the terms ‘design principle’, ‘design rule’, ‘style guide’ and ‘standard’ in the literature. Unfortunately the use of the previous terms is not consistent in the different sources. Although ironically consistency is mentioned all the time to be the most important principle.

In [NASA 96] the order ‘design principle’, ‘design guideline’ and ‘design rule’ describes a rising level of specification. Most abstract are the design principles and most specific are the design rules. A design rule is said to be a standard. This standard has

value for one system or project but may vary between different systems or projects and therefore cannot be stated generically.

Design rules and standards are also mentioned in [Vanderdonckt 99b]. While the meaning of ‘design rule’ is comparable the meaning of ‘standard’ is not at all. Here a standard is “a set of functional and/or operational specifications intended to standardise UI design”, which is the much more common use. These standards are developed from several national or international organizations, like e.g. the International Standard Organization (ISO).

In [Vanderdonckt 99b] a style guide is “a rule compendium, with recommendations for designing a UI satisfying a given specification.”

In contrast to this [Gale 95] defines a style guide not by its contents but only by its purpose:

“The objectives of a Style Guide are to:

- Promote visual and functional consistency within and between different applications;
- Promote good design practice;
- Reinforce company branding or an organisation's public image.”

This is not contradictory but there are possibilities for interpretation.

A totally different scheme is used in [Schumacher 92]. This document contains ‘requirements’, ‘guidelines’ and ‘good practices’. While ‘requirements’ are “must-be” advises and ‘guidelines’ are “should-be” advises the ‘good practices’ refer to situations where the advice is less specific or highly conditionalized.

The definition of a style guide in [Vanderdonckt 99b] describes what I call a guideline in my paper. I use this term in a more general meaning, which should not be confused with a design guideline like mentioned in [NASA 96]. Where it is appropriate and the meaning will be clear I will use the terms ‘principle’ or ‘design principle’ for abstract suggestions and ‘rule’ or ‘design rule’ as their more specific counterpart.

3 Which aspects deal these guidelines with?

3.1 Long documents

The guidelines are all very similar in form and contents. [Gale 95] suggests a template for a table of contents to give an example of an effective format for a UI guideline. The actual tables of contents of the different documents fit to this template quite well. Most of the suggested chapters are covered in the existing guidelines. Only the order of the sections varies a lot. This chapter contains some examples of covered aspects to convey what these documents have in common.

An exception is [IBM 97] whose table of contents is completely different. But this guideline differs from the other ones anyway. It doesn't contain any specific design rules at all but only principles on a quite abstract level. Therefore it cannot be used as a style guide as [Vanderdonckt 99b] defines this term. [IBM 97] more likely wants to give an overview about the important design principles. It focuses on motivation why these principles are important and contains some explanations of their effects on the designed product but nothing else.

Another exception is [Tidwell 98b]. This document isn't intended to be a guideline in the common sense. Here a pattern language for HCI design is presented. This causes inherently a more abstract level of the given suggestions. The author wants to overcome this difficulty by giving a lot of examples for each pattern. These examples are not only related to software development but more general to man-machine interfaces. The big advantage of the presented patterns is that they are independent of any hardware or software environmental conditions. They can be used independent of the technical context. The disadvantage is the abstract level of the patterns. Someone needs to interpret them in the current context, which is more difficult and takes longer than just applying a specific design rule.

The aspects these guidelines deal with are very common among them. Especially for the software product these long documents deal with very similar aspects. Therefore these aspects are really good documented.

The documents have also in common that they first present abstract principles to motivate the use of the rules presented at a more specific level.

Some documents contain warnings what has to be avoided. The best example for this can be found in [Apple 95]. It uses such "Don'ts" clear and very consequently. [Apple 97], [Microsoft 97] [Schumacher 92] and [Sun 99] do not contain "Don'ts". They fail to give these bad examples.

3.1.1 Aspects of the development process

The aspect of the process of developing software and designing a UI is only covered insufficiently in the long documents. Only [Schumacher 92] and [Apple 95] have short sections about this. They both deal with User Centered Design (UCD) but give only few and different suggestions.

[Schumacher 92] starts with a motivation for this subject and explains, the elements of UCD are

- understanding and specifying the context of use by listening to the users,
- prototyping,
- early and continual user testing and
- iterative redesign.

The elements of UCD in [Apple 95] (although this is not called 'User centered design' explicitly there) are similar:

- Definition of the future users,
- task-analization,
- prototyping and
- user-testing.

For this last element there is given a very useful list of 'Ten Steps for Conducting a User Observation'.

These suggestions are held abstract and apart from the mentioned list in [Apple 95] *nothing* specific can be found about the development process in the long documents. Maybe that is one reason, why UCD is not well done in commercial software development.

3.1.2 Aspects of the software product on an abstract level

As mentioned above the sections for these aspects are very similar in the long documents. On an abstract level [Harrison 98], [Microsoft 97], [NASA 96] and [Schumacher 92] all deal with the aspects of consistency, using metaphors, visual design, offering help and error handling.

3.1.2.1 Consistency

The importance of consistency is mentioned in all of the papers. First it is motivated why consistency is necessary for a good UI. In these parts of the documents the same things are said very often only with some different words. In [NASA 96] we can read this: "Consistent visual appearance and consistent response to user input are required throughout the user interface. [...] [I]nterface characteristics should be uniform and familiar, with consistent sequences of actions in similar situations. Terminology must be used consistently to avoid confusing the user. A user interface becomes intuitive by

consistently meeting users' expectations. [...] Allow the user to build-up expectations and predict system actions based on the system's performance of other actions. [...] Permit the user to take the general knowledge and skills learned in one system and transfer them to another like it, without requiring extensive learning and training exercises.”

Following are often some notes where someone has to take special care on consistency or how it can be achieved.

[Microsoft 97] differentiates between consistency within a product, within the operating environment and with metaphors. Consistency within a product means presenting common functions using a consistent set of commands and interfaces. Consistency within the operating environment can be achieved by maintaining a high level of consistency between the interaction and interface conventions provided by the operating environment so that the users are able to apply interaction skills they have already learned. Consistency with metaphors means to give special attention that the behavior of an object is reflected by its metaphor. This will make it easier for the user to associate that behavior with an object.

[NASA 96] suggests to maintain consistency in the design areas of the display (icon design and meaning, location of title fields, menu bars and messages, cursor shape and function, cursor home position, field delimiters, color meaning and data entry prompt), labeling terminology, system control (command terminology and meanings, editing procedures, function- and command keys), abbreviations, acronyms, mnemonics, visual coding, alarms and warnings.

In [Schumacher 92] we can find some tips how to support consistency:

“Similar interface components should have similar appearance and behavior. An action should always have the same result regardless of context (i.e., avoid modes). Standard functions should be reused across tasks and should be presented in the same way in each task. By definition, the behavior of standard functions should be consistent across different task contexts.”

3.1.2.2 Metaphors

This is handled surprisingly short in all the long documents. Although it is said that metaphors are very important a deeper discussion of them is missing. The developer cannot find something about the technique of creating successful metaphors for the interface. Therefore it is impossible to give here examples what the guidelines are proposing for creating and using metaphors.

Only already existing metaphors are explained in more detail. This is the desktop metaphor explained in [Schumacher 92] and the metaphor of objects used by the design model of the Windows UI as discussed in [Microsoft 97].

3.1.2.3 Visual design

This part is always introduced with an explanation of its goals. Although there are various terms used in the texts they all explain the importance of a visually appealing layout. “The objective is to achieve a screen layout that meets users’ information requirements while appearing well organized and uncluttered.” [NASA 96]

There are given a lot of tips of how to achieve this. Often the principle of consistency is mentioned here again (e.g., [Harrison 98] and [NASA 96]). What else to take care of is explained for example in [Schumacher 92]: “Make objects and groups of objects visually distinct. A number of factors impact a user’s comprehension of the display and efficiency of interaction with the display: total number of items, grouping of items, alignment of groups, length of items, labeling of groups, ordering of items and density of groups.” The same aspects of composition and organization can also be found in [Harrison 98] and [Microsoft 97].

Only [Schumacher 92] is giving more information how to group the elements on the screen. The heuristics mentioned there are function, frequency of use, sequence of use, location or related format, importance, habitual sequence and alphabetic, numerical or chronological ordering.

3.1.2.4 Help

On this abstract level the documents describe the goals of providing help to the user and how the goals can be achieved. “Provide users with online help that can be entered whenever needed. Tailor help to the task context and requirements. Presenting the entire user’s manual in response to a help query will only frustrate the user. Effective help can serve as a data-protection resource. [...] Provide multi-level help, beginning with summary information and providing more detailed explanations on request.” [NASA 96]

There are mainly two different kinds of help: The online-, or reference help and the context-sensitive help. Online help should contain information on executing a command, accessing and using an application’s capability, accomplishing a specific goal, and handling error conditions. Furthermore it should explain concepts associated with a task or interface component according to [Schumacher 92]. While online help should serve multiple purposes the goal of context-sensitive help is more tailored. According to [Microsoft 97] it should mainly indicate the action associated with the item. The explanation can contain information of how to perform an action and reasons why an action should be performed.

The different documents agree that the style of the help texts should be simple, concrete and natural.

3.1.2.5 Error handling

There is a common sense that an application should serve the principle of “Forgiveness”, which means to offer an undo-mechanism to give an easy opportunity to recover from potential dangerous situations.

But it is also well known that errors are not absolutely avoidable.

[Harrison 98] gives some hints for the authors of error messages: “Be as specific and precise as possible. Be constructive: indicate what needs to be done. Use a positive tone: avoid condemnation and choose user-centered phrasing.” This all is repeated also in [NASA 96] and [Schumacher 92].

3.1.3 Aspects of the software product on a specific level

On a more specific level the similarity is even bigger. [Apple 95], [Apple 97], [Harrison 98], [Microsoft 97], [NASA 96], [Schumacher 92] and [Sun 99] give all specific design rules mostly for the same elements. These elements are windows, menus, icons, dialog boxes, layout and visual alignment, controls such as buttons, check boxes, radio buttons, list boxes, combo boxes, text fields and others, color, font, mouse and keyboard navigation, help and error handling.

3.1.3.1 Windows

[Apple 95], [Microsoft 97], [NASA 96], [Schumacher 92] and [Sun 99] are all describing primary and secondary windows. “A **primary** window is a window in which the main interaction with the data or document takes place. An application uses any number of these windows, which are equivalent and can be opened, closed, minimized, or resized independently. Users do the majority of their viewing and editing in primary windows, such as document windows. A **secondary** window is a supportive window that is dependent on a primary window (or another secondary window) in which users can view and provide additional information about actions or objects in a primary window. Examples of secondary windows are dialog boxes and alert boxes. A **utility** window is a window whose contents affect an active primary window. Unlike secondary windows, utility windows remain open when any primary windows are closed or minimized. An example of a utility window is a tool palette that is used to choose a graphic tool. A **plain** window is a window with no title bar or window controls, typically used for splash screens.” [Sun 99]

All these documents enumerate the various elements like the title bar, scroll bars and the buttons for closing, minimizing, maximizing or restoring a window and explain that secondary windows can be modal or modeless and that the number of secondary windows should be limited to avoid creating navigation problems.

While [Apple 95], [Microsoft 97] and also [NASA 96] are describing the window behavior very detailed and explicitly [Sun 99] just says, “Window behavior, such as

resizing, dragging, minimizing, positioning, and layering, is controlled by the native operating system.”

3.1.3.2 Menus

The importance of single words as menu titles, separators for grouping, mnemonics for menu titles and menu items and keyboard shortcuts for the most important options is mentioned throughout all the documents.

An Ellipsis ("...") should be only used when the menu option evokes a window that requires more input from the user to complete this command. An option such as “Properties” should therefore not be followed by an ellipsis. Although it presents the user a window where his input may be necessary the command of showing the properties is already finished at this time.

It is also common practice not to disable a menu when all its items are disabled. The menu title may have a disabled appearance in this case but, however, users must still be able to pull down a menu and see the disabled menu items. This is the same for submenus in a hierarchical structure.

[Harrison 98] and [NASA 96] suggest limiting the number of items per menu to about 8. If this number is exceeded submenus should be used. Furthermore [Harrison 98] limits the number of levels of submenus to 3 and proposes to choose Breadth over Depth in large applications.

Some of the papers (e.g. [Microsoft 97] and [Sun 99]) additionally describe common menus like the File-, the Edit- and the Help menu.

3.1.3.3 Icons

The following guidelines are taken from [Sun 99] but are closely comparable to the corresponding sections in [NASA 96] and [Schumacher 92]:

“Design icons to identify clearly the objects or concepts they represent. Keep the drawing style symbolic, as opposed to photo-realistic. Too much detail can make it more difficult for users to recognize what the icon is intended to represent.

When designing large and small icons that represent the same object, make sure that they have similar shape, color, and detail.

Use a single style to create a “family” of icons that utilize common visual elements to reflect similar concepts, roles, and identity. Such icon families might use a similar palette, size, and style for different icons, such as folders and documents.

Don’t mix two- and three-dimensional styles in the same icon family.”

It is often recommended to limit the number of icons, for example to 20 in [NASA 96], and to test the acceptance and obviousness of icons with users.

The documents are contradictory about the usage of dark outlines for icons. [Apple 95] and [Sun 99] are recommending such a border “for satisfactory display on a wide range of background colors and textures” [Sun 99]. In [Microsoft 97] we can read “Previous

to Microsoft Windows 95, black outlines were recommended for icon design. This style recommendation has been dropped.”

Furthermore [Apple 95] suggests designing a black-and-white version of the icon first and adding color afterwards. [Microsoft 97] says it is sufficient just to check the monochrome appearance of an icon after the design because the “system automatically maps colors in your icon design for monochrome configurations.”

3.1.3.4 Dialog boxes

This element is always introduced as a special kind of secondary window. Sometimes the differences and effects of modal and modeless windows are explained again like in [Sun 99]. The other main aspects the long documents deal with are the title and placing of buttons in the dialog box.

The most detailed explanation of how to create a title can be found in [Microsoft 97]: “Because dialog boxes generally appear after choosing a particular menu item (including pop-up or cascading menu items) or a command button, define the title text for the dialog box window to be the name of the associated command. Do not include an ellipsis in the title text, even if the command menu name may have included one. Also, avoid including the command’s menu title unless necessary to compose a reasonable title for the dialog box. For example, for a Print command on the File menu, define the dialog box window’s title text as Print, not Print... or File Print. However, for an Object... command on an Insert menu, you can title the dialog box as Insert Object.” This is contradictory to the suggestion of [Sun 99]: “Use the form ‘ApplicationName: Title’ for the title of the dialog box.” For a search dialog box this would result in “Search” following the Microsoft rule and “MyApplication: Search” as title according to the rule of Sun.

For the placement of the buttons [Microsoft 97] and [Sun 99] give similar guidelines. “Align command buttons along the lower-right edge of the dialog box. Place the Help button last in a group of command buttons. Make the default button the first command button in the group. For languages that are read left to right, the default button is the leftmost button.” [Sun 99]

“Orient controls in dialog boxes in the direction people read. In countries where roman alphabets are used, this means left to right, top to bottom. Typically, define OK to be the default command button when the dialog box window opens. Place OK and Cancel buttons next to each other.” [Microsoft 97]

These both documents are contradictory to [Apple 95] where the principle “The Western reader’s eye tends to move from the upper-left corner of the dialog box to the lower right” results in the order Cancel and OK and the default button is the rightmost in this guideline, which is most comfortable for right-handed persons.

Also [Schumacher 92] contradicts the rules of [Microsoft 97] and [Sun 99]. Here it is recommended to “Make the default button the least destructive option.” This would mean to make always the Cancel button the default button.

3.1.3.5 Layout and visual alignment

Very often this subject covers how to place items on the screen. [Apple 95], [Microsoft 97], [Schumacher 92] and [Sun 99] agree to follow this principle: “In general, follow the layout conventions for how information is read. In western countries, this means left-to-right, top-to-bottom, with the most important information located in the upper left corner.” Only one of the short documents gives a different suggestion: “Where to place information on a screen: Provide an obvious starting point in upper left corner. Eyes move from upper left clockwise.” [Parkhurst 96]

Furthermore the different documents cover aspects like font and size, capitalization, grouping and spacing, alignment and some various other under this title.

[Microsoft 97] proposes for capitalization: “When displaying text in menus, command buttons, and tabs, use conventional book title capitalization. [...] [C]apitalize the first letter in each word unless it is an article or preposition not occurring at the beginning or end of the name, or unless the word’s conventional usage is not capitalized.

Field labels, such as those used for option buttons, check boxes, text boxes, group boxes, and page tabs, should use sentence-style capitalization.”

[Sun 99] contradicts these suggestions. Sentence capitalization is recommended only for lengthy text messages while headline capitalization is also recommended for “most names, titles, labels, and short text” which includes “text in buttons, sliders, and combo boxes”.

[Schumacher 92] gives some general tips for grouping and spacing: “Place no more than 20 organized groups, and usually far fewer, of information on a single window or dialog box. [...] Any particular window or dialog box should have no more than 40% of it filled with words or graphics; it should contain at least 60% white space. White space is defined as the number of display locations that do not have any characters or graphics.”

3.1.3.6 Controls

Every long document has a section where it explains the proper usage and behavior of the standard controls like pushbuttons, radio buttons, check boxes, text fields, list-boxes, combo-boxes, sliders and many more. There are no differences worth to mention in the explanations of these components.

3.1.3.7 Color

[Apple 95], [Harrison 98], [Microsoft 97], [NASA 96] and [Schumacher 92] give all the same suggestions for working with colors. However, this is done most explicitly in [NASA 96]:

“When to use: A powerful attention getter is needed.

When not to use: Several colors are already used on the display.

Limit the number of colors to be used. Preferably, use four or less.

Use color to support visual-search tasks and symbol-identification tasks.

Use no more than six distinct colors or three shades of gray if the user must recall the meanings of colors or shades.

Use no more than six distinct colors if the user must perform rapid visual searching based on color discrimination.

If functional requirements dictate the use of more than the recommended number of colors or shades of gray, display a legend of color/shade meanings.

Avoid the color combinations listed in [the following table:]

Saturated Red and Blue	Saturated Red and Green
Saturated Blue and Green	Saturated Yellow and Green
Yellow on Purple	Green on White
Yellow on Green	Blue on Black
Magenta on Green	Red on Black
Magenta on Black	Yellow on White

Use a medium achromatic background (e.g., dark or medium gray) to maximize the visibility of foreground colors.

Consider using complementary colors (yellow on dark blue; magenta on green) to maximize color contrast, if appropriate for the user's task environment.

Consult [the following table] for colors to make thin lines easily perceivable:

# of Colors	White Background	Black Background
1	Red or Green	Yellow, Cyan or Green
2	Red and Green Magenta and Cyan Red and Blue	Green and Magenta Yellow and Magenta Cyan and Magenta
3	Red, Blue and Green	Cyan, Magenta and Yellow

Be sure that the information coded by color is coded in some other manner for the sake of color-deficient users.

Before adding color to a display, design for its use in monochrome. Then, use color as a redundant code to enhance an otherwise logical, well-organized design.

If the user community has previously established meanings for various colors, retain those meanings. Do not use a color to signify a different condition than it signified in the previous system.”

3.1.3.8 Font

This subject is handled quite short in most of the guidelines. Like for color the most explicit description can be found in [NASA 96]:

“When to use: Text items should stand out.

When not to use: There are more than 2 to 4 different character types.

Improve legibility by the use of well-formed letter shapes and type sizes that are appropriate for viewing distances.

Where alphanumeric characters are displayed, select font styles to allow discrimination of similar characters, such as letter l and number 1, letter Z and number 2.

Do not use varying sizes or styles of fonts for any reason other than coding (for example, text as labels, text as data, text as command input).

Use selected fonts in a consistent fashion throughout the interface, and provide upper and lower case with full descenders.

Avoid type faces that have extended serifs, internal patterns, or that are striped, italicized, stenciled, shadowed or 3-dimensional, or fonts that appear like handwritten script or like Old English script, and fonts that are distorted to look tall and thin or wide and fat.”

3.1.3.9 Mouse and keyboard navigation

Most of the long documents [Apple 95], [Microsoft 97], [Schumacher 92] and [Sun 99] describe explicitly the mouse behavior for clicking, double-clicking, pressing and dragging. For the keyboard they all tell about the importance to provide shortcuts for the most used options and to offer the possibility to navigate through the interface without the mouse, normally by tab-navigation.

3.1.3.10 Help

Although specific tips for writing help cannot be found often [NASA 96] and [Schumacher 92] give identical rules for this.

“Users must be able to initiate a help request whenever they want, get help on any topic they desire, control the type of help information, and exit help at any time. [...] It is strongly recommended that on-line help remain visible until the user explicitly removes it. [...] Provide information that focuses on the task -- include all needed information, be sure all information is correct, and exclude anything that is not needed. The best way to ensure that help is helpful is to review and test help with target users.” [Schumacher 92] Additionally [Microsoft 97] gives some specific rules for context-sensitive help:

“In English versions, begin the description with a verb [...]. For command buttons, you may use an imperative form [...]. When describing a function or object, use words that explain the function or object in common terms instead of technical terminology or jargon.”

3.1.3.11 Error messaging

[Harrison 98], [NASA 96] and [Schumacher 92] deal with specific aspects shortly and give various but not contradictory suggestions. The most complete collection of specific rules for error messaging contains [Microsoft 97]. Even in the guideline itself the rule to avoid the term “error” is followed very strictly. Instead of the common term “Error messages” the more unspecific term “Messages” is used there.

“Never use the word ‘error’ in the title text. [...]

- State the problem, its probable cause (if possible), and what the user can do about it -- no matter how obvious the solution may seem to be. [...]
- Consider making the solution an option offered in the message. [...]
- Avoid using unnecessary technical terminology and overly complex sentences. [...]
- Avoid phrasing that blames the user or implies user error. [...]
- Make messages as specific as possible. Avoid mapping more than two or three conditions to a single message. For example, there may be several reasons why a file cannot be opened; provide a specific message for each condition.
- Avoid relying on default system-supplied messages [...]; instead, supply your own specific messages wherever possible.
- Be brief, but complete. Provide only as much background information as necessary. A good rule of thumb is to limit the message to two or three lines. If further explanation is necessary, provide this through a command button that opens a Help window.

You may also include a message identification number as part of the message text for each message for support purposes. However, to avoid interrupting the user’s ability to quickly read a message, place such a designation at the end of the message text and not in the title bar text.” [Microsoft 97]

3.2 Short papers

As explained in chapter 2 ("Which types of guidelines were part of the investigation?") these papers cover a lot of different kinds of documents. This causes a lot of differences in form and content from the various papers. Here I can just present a short overview.

Because each of the short papers deals with only one aspect of UI design they cover many different aspects altogether. They are not intended to serve general purposes. Therefore they have almost nothing in common and it is very difficult to give examples and to discuss them as it was done for the long papers.

Unlike the long documents from the Web they mainly contain abstract design principles and fewer specific design rules. Comparable with the long documents is the fact that they also deal more with aspects of the software product and less with the development process. Therefore they also contain almost no specific suggestions for this process.

As some of the long documents they only tell positive suggestions what someone should do. They almost contain no warnings mentioned before as “Don’ts”.

3.2.1 Aspects of the development process

This subject is covered mainly on an abstract level. Like in the long documents the focus is mainly UCD although it is to say that this is done much more detailed in the short documents because they deal only with this one subject.

[Bevan 96], [Inuse 97] and [Shackel 97] are dealing with UCD in general. More special [Bevan] is talking about usability, [TOG 98a] mentions iterative design and [TOG 98c] gives important notes about prototyping and user feedback.

Like in [Bevan 96] UCD is introduced and presented more exhaustively compared to the long documents. [Bevan 96] explains, that it is most important to involve users at all stages of design and to obtain early feedback from informal and rapid prototypes. The principles and the activities of UCD are the same in [Bevan 96] and [Inuse 97]:

Principles of user-centered design:

- finding an appropriate allocation of function between the users and the system
- iteration of design solutions
- active involvement of users
- employment of multi-disciplinary design teams

User-centered design activities:

- understanding and specifying the context of use
- specifying the user and organizational requirements
- producing designs and prototypes
- carrying out user-based assessment

The few papers that handle this subject on a specific level deal with the development of a style guide [Gale 95], myths the developers have about the users [UIE 96a] and testing of programs in close couples [TOG 98i]. The last two aspects can be taken as an element of UCD.

3.2.2 Aspects of the software product

Like for the development process the short papers cover the contents of the long documents and often deliver other aspects additionally.

The abstract principles like consistency ([Gale 95]), using metaphors ([TOG 98d]), offering help, error handling ([Rizzo 96] and [TOG 98b]) and visual design ([IIMRR 99]) are all covered. Other aspects that only can be found in these short papers are the out-of-box experience (OOBE in [Berry]), the concept of “User in control” ([Schank

95] and [Wildstrom 98]), Interfaces and psychological theory ([Marchionini 91]), differences of subjective and objective time-experience ([TOG 99b]) and many others. They also give some abstract principles for elements that are covered in the long documents only on a much more specific level like icons ([Horton 96] and [Horton 97]), color ([HIMRR 99]) and context-sensitive menus ([TOG 99c]).

On the specific level of design rules the short papers offer fewer aspects than on the abstract one. On this level all the aspects can also be found in the long documents like color ([Hawkes 98]), menus ([VR a] and [VR b]), toolbars ([Mandel 99]), error messages ([TOG 98f]) and accessibility ([US DE 97]).

Where the short papers and the long documents deal with the same aspects the short papers confirm the statements of the long documents. This is especially true for the aspect of metaphors, where also [TOG 98d] lacks any hints of how to create and work with good metaphors. [Rizzo 96] is an example where a short paper deals with a subject - here it is 'Error handling' - much more detailed than it is done in the long documents.

4 Usability problems with these guidelines

The usability problems that arise with using UI guidelines are numerous. Some of the short papers ([Gale 95], [Myers 93], [Stultz 98], [Vanderdonckt 99b]) and some of the long documents' introductions ([NASA 96], [Schumacher 92], [Tidwell 98a]) deal with these problems. The problems can mainly be divided into two groups: Problems with the contents of a UI guideline and problems with handling the document.

Especially [Gale 95] delivers many common different reasons why guidelines can fail. Also [Vanderdonckt 99b] contains a large list of difficulties related to guidelines. In general terms spoken one of their difficulties is insufficiency although their necessity is not questioned. Why guidelines are insufficient will be explained now.

4.1 Problems with the contents of UI guidelines

4.1.1 Consistency and usability

In all the guidelines it's said, that 'Consistency' is the most or at least one of the most important principles. I don't want to doubt the importance of consistency but a lot of authors ([Gale 95], [Myers 93], [NASA 96] and [Vanderdonckt 99b]) mention that consistency alone is not sufficient and that usability can not be reached by using UI guidelines alone.

"However, ensuring consistency does not ensure usability. Although an application may be highly consistent, it does not automatically follow that the application will be useful. Compliance with a low level Style Guide will ensure consistency. However, it will not ensure that the applications themselves perform the tasks required either by the end users or the business." [Gale 95]

Other problems with ensuring usability can be found in [Myers 93]: "However, these style guides are usually hard to interpret and apply. Furthermore, the standards will only cover a small part of the user interface design, and will not insure that even this part has high usability." [Myers 93]

[Vanderdonckt 99b] states that more than ergonomic sources - guidelines are one type of ergonomic sources - are needed to guarantee UI usability, e.g. "scenario- or use case-based design/evaluation user tests."

Only [NASA 96] is informing about this shortcoming in its introduction explicitly: "The guidelines are intended to help software developers generate user interfaces that support the end user's tasks and functional requirements. [...] The guidelines cannot substitute

for knowledge of the information requirements associated with decision points embedded in the users' tasks. This document assumes that developers have detailed user requirements available to them through other sources, such as cognitive task analyses and frequent, direct interaction with end users."

4.1.2 Generalization

[Myers 93] doubts the fundamental base for guidelines: The generalization of UI design issues. "Whereas early papers in HCI were full of experimental laboratory studies of small issues in user interface design, such as the proper menu organization, you rarely see any of these now because the results have failed to generalize." [Myers 93]

If these issues cannot be generalized the sense of UI guidelines for any generalized purpose must be questioned.

4.1.3 Focus and level of detail

Different authors describe problems with the various common aims that guidelines have and the resulting details they contain. "The guide attempts to be too prescriptive and tackle too many application domains or different types of end users. The result is that it is not appropriate for a number of target applications." [Gale 95]

"Most style guides are interface object catalogs. They provide a description of the object and tell something about how it works. However, style guides offer the reader little in the way of good design practices and when certain objects should and should not be used. Furthermore, style guides rarely discuss how to use combinations of different types of objects. Style guides are still necessary, but they do not in themselves mean that good, usable interfaces will necessarily be developed (i.e., they are under-specified)." [Schumacher 92] An excellent example of such an interface object catalog is [Microsoft 97].

On the other hand it is to say that guidelines sometimes contain too much details. "Window behavior, such as resizing, dragging, minimizing, positioning, and layering, is controlled by the native operating system." [Sun 99] But these are exactly the aspects that are explained very often in the guidelines when describing the item of windows. These explanations seem to be unnecessary, at least at this level of detail. The average developer cannot or doesn't want to influence this behavior in any way. The advantage is that these things are controlled by the operating system so the developer doesn't have to take care of them.

4.1.4 Illustration

[Vanderdonckt 99b] says that guidelines are illustrated insufficiently. This is obviously true for [Harrison 98], [IBM 97] and [Tidwell 98b] since they are not illustrated in any

way. For [IBM 97] this is excusable because of its abstract principles that are very difficult to illustrate without giving specific examples. Although [NASA 96] and [Schumacher 92] have some graphical examples they also fit to this critic. The illustrations are sometimes sporadic and there are many possibilities for a more extensive use of them.

But this critic can not be generalized. [Apple 95], [Apple 97] and [Sun 99] are good examples for the use of illustrations.

4.1.5 Conflicts

When using UI guidelines there are plenty of possibilities that can cause a conflict. This can be an inconsistent or a contradictory result. “Collecting guidelines inevitably induces conflicts between various sources. For the same usability question, ergonomic sources can lead to inconsistent results.” [Vanderdonckt 99b] This was shown earlier in chapter 3 (“Which aspects deal these guidelines with?”) for the specific aspects of the software product in long documents such as icons, dialog boxes and capitalization as an aspect of layout and visual alignment.

But conflicts can also arise when working with only one UI guideline. There may be more than one principle that can be used for a special design problem and they may suggest different solutions. “And no style guide can infallibly tell you how to strike a balance between two opposing high-level principles.” [Tidwell 98a]

At least conflicts may also arise when working with just one design rule not comparing it to any other. “In fact, there are important counter-examples to even the most basic guidelines. For instance, most sources put consistency at the top of lists of guidelines, but Grudin discusses many cases where consistency is not appropriate. For example, menu systems might have the default selection be the more recent or most likely selection, but still might not use this rule for questions confirming dangerous operations.” [Myers 93]

4.1.6 Correctness

Also in the absence of conflicts the guidelines cannot be trusted blindly. [Vanderdonckt 99b] says that some sources, although few, contain incorrect or contentious guidelines that are very difficult to detect if it’s possible at all. The contradictions shown in chapter 3 give a subtle hint for this incorrectness.

4.1.7 Transience

The rapid progress in the technical surroundings of software development is another problem. “[Guidelines] are transient - toolkits, trends, and operating systems come and

go, and as soon as the world gets comfortable with one, another arises to take its place. Remember the transition from Windows 3.1 to Windows 95?" [Tidwell 98a]

The guidelines have to adapt to these changes of the technical conditions. This has to be done very fast because new methods and tools are always used by the developers almost instantly. "If you give a kid a hammer, everything will look like a nail. Give a programmer new tools and programming languages and he'll use them whether a more simple solution is more appropriate or not." [Stultz 98]

4.1.8 Obviousness

Some specific guidelines are not obvious. The developer cannot understand them. Instead he has to learn them. An example for this is the following rule: "Insert five pixels of space between the top of the button text and the inside white border. Insert six pixels between the text's baseline (not the descenders) and the bottom button border." [Sun 99].



In the context of this document it is explained why both measures are not the same. The "flush" 3D appearance with the white line is the reason. Measuring the distance from text to the black line the distance is in both cases 6 pixel.

But why is the distance 6 pixel? Why not 5 or 7? Maybe it's related to the size of the font, but there's no explicit reason given why it must be exactly 6 pixels.

Another example is a rule for the distance between buttons: "Space buttons in a group five pixels apart." [Sun 99] Why not 4 or 6 pixels? Someone has to learn rules like this like vocabulary. This is more difficult than to learn something by understanding.

In [Microsoft 97] we can find also rules for spacing that are similar to these rules taken from [Sun 99]. But the spacing in [Microsoft 97] is measured in 'dialog box units': "One horizontal dialog box unit is equal to one-fourth of the average character width for the current system font. One vertical dialog box unit is equal to one-eighth of an average character height for the current system font." The distances given in dialog box units are not obvious and even very hard to interpret. It is clear that the distances are related to the size of the system font but how should a developer work with these rules? As a first step he has to find out what the average character width and height are. Someone can expect that he will ignore these rules or just work with some similar rules of thumb.

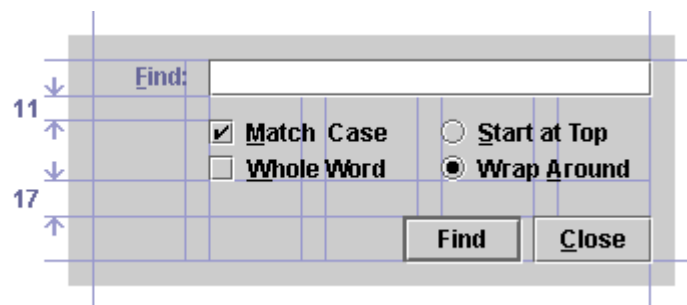
4.1.9 Lack of bad examples

As it was mentioned before in chapter 3.1 ("Long documents") a lot of UI guidelines fail to give bad examples, so called "Don'ts", e.g. [Apple 97], [Microsoft 97] [Schumacher 92] and [Sun 99]. The advantage of showing "Don'ts" is that bad examples are remembered very easily and they are easy to detect. They serve the principle that someone can learn from errors, which are shown as common mistakes. It is always much easier to detect some mistakes as it is to find out what things were done in a good way.

4.1.10 Support by development tools

The common development tools do not support such a UI design on the level of pixels. So it's very difficult to fulfill the recommendations even if you know them all. The example given for the problem of obviousness is also valuable for this problem. Most of the development tools place the text in the buttons automatically. They don't offer a function to move the label of the button one pixel up or down.

Neither do they support the use of design grids as recommended in [Sun 99].



Design grids can be very useful to achieve a consistent and visually appealing layout but the average developer has never the possibility to use them.

4.1.11 Size

A general problem is that guidelines are very large and have an almost unhandable size. "Often the documentation runs into hundreds of pages. The material becomes difficult to use and read. It is ironic that many of the existing Style Guides, aimed at producing usable systems, are themselves difficult to use." [Gale 95]

The long documents contain such a huge number of specific design rules that it is impossible to remember them all.

4.2 Problems with handling UI guidelines

4.2.1 Problems with developing your own UI guideline

4.2.1.1 Introducing the document

The existence of a new guideline alone is not sufficient. The developers need to know how to work with it. “Little consideration is given to how the Style Guide is introduced into the organisation - there is no supporting material and no one is there to encourage developers to use it. When there are problems there is no one around to listen and amend the guide (if necessary).” [Gale 95]

4.2.1.2 Updating

This problem is related to the problem of transience. The guideline needs to reflect a change and progress in the technical surroundings where it is used in. “The document is never updated - either because no one realises that it needs to be updated or because no one is sure about who should update it.” [Gale 95]

4.2.1.3 Purpose

The effect and benefit offered by guidelines are often overestimated. “Often the use of the Style Guide alone is perceived to solve the ‘usability problem’. For example, many organisations seem to believe that once a Style Guide is in place no further action is required to ensure usability. The production of a Style Guide is seen as a substitute for iterative design and user testing.” [Gale 95]

4.2.1.4 Responsibility

Sometimes the responsibility for the guideline is not given properly. “Developers and users are not involved in the content and development of the Style Guide. [...] Another group, or person, is perceived to be responsible for the system’s usability and the development team are [sic!] no longer responsible for the usability of their own system.” [Gale 95]

4.2.2 Problems with using a UI guideline from third parties

4.2.2.1 Number

As mentioned in chapter 2 the amount of guidelines that are offered is enormous. “A wide range of literature should be systematically investigated to find sources of guidelines. [...] An enormous quantity of researchers’ papers, internal reports, manuals has been published on guidelines. These are unfortunately widespread in both time and space, as well as in both format and access. This access to guidelines remains mostly difficult, even for skilled, experienced persons.” [Vanderdonckt 99b]

Very fast someone has encountered a huge number of documents. Some of them are very large themselves (problem of size) and it is very difficult to get an overview of the whole collection of documents.

4.2.2.2 Incompleteness

“[C]ollections will remain perpetually incomplete with respect to UI usability for the following reasons [...]:

- unraised questions: e.g., what is the usability of novel interaction techniques uncovered in existing guidelines?
- raised but unsolved questions: e.g., what is the most usable metaphor for a given context?
- raised but only partially solved questions: e.g., what are the possible metaphors for a given context?” [Vanderdonckt 99b]

4.2.2.3 Access

“Initial guidelines are scattered throughout sources that are sometimes hardly accessible. [...] A person in a commercial environment does not generally have the time nor capability to explore such a distribution in a limited amount of time.” [Vanderdonckt 99b]

This is especially true for the short papers, which are not easy to detect between all the literature about HCI. When looking after an article about one specific subject it’s almost impossible to find. So some questions will remain unsolved because the answer cannot be found although it exists.

4.2.2.4 Integration

This problem is similar to the problem of introducing a guideline that was newly developed. Also when working with guidelines from third parties the developers need to

know how they can use the document. Normally this subject is not covered in the existing guidelines.

“The guide does not address how it should be used in practice or how it should be integrated with development methods.” [Gale 95]

“[The guidelines] do not answer questions such as: where should this guideline be used? When should it be used? How? Under what conditions?” [Vanderdonckt 99b]

5 Ideas to make guidelines more usable

In the last chapter I showed that there are numerous problems that occur when working with UI guidelines. These problems cause various difficulties when working with UI guidelines as shown above. Someone can hope that the efficiency of UI guidelines will be higher, when these problems are solved.

However, I will show in this chapter that only some of these problems can be solved. Some other problems can not be solved and will remain for the future.

Aside from these problems I will also show other ideas that could make guidelines more useable.

5.1 Ideas to solve the mentioned problems

Not all of the mentioned problems can be solved. [Myers 93] states that research showed that the problem of generalization would not be solved in the future. [Vanderdonckt 99b] says the same for incompleteness, because it is intrinsic to the guidelines themselves. It can be added that also no solutions can be found for the problems of conflicts, correctness, transience, obviousness and the problems with using a UI guideline from third parties like number and access:

- Conflicts and correctness are also intrinsic problems.
- It cannot be expected that the progress in developing new development tools will slow down in the near future, so transience will remain a problem.
- Obviousness seems also be intrinsic in some way. Some measures like some gaps and distances must be defined in one way, also some other ways are possible.
- With time the number of guidelines will exceed and therefore also the problem related to this.
- It is also questionable if the access to these guidelines, especially the web sources, will be easier while the web is evolving in the next time.

Some of the problems (consistency, level of detail, updating and integration) arise because the guidelines lack of something. It can be assumed that adding the missing parts to a guideline will solve the problem. But it must not be forgotten that adding more to UI guidelines will even increase the problem of size. Therefore extending guidelines must be done very carefully.

5.1.1 Consistency and usability

The main subject in this point was that consistency alone does not ensure usability at all. Therefore [Gale 95] suggests to incorporate “HCI principles which help ensure ease of use and minimise learning of the application” to come up with a truly comprehensive guideline. Much more aspects can be added to guidelines to support the realization of an easy to use interface.

[Myers 93] mentions, “there are important counter-examples to even the most basic guidelines” and gave the example of the principle of consistency that collides with design for dangerous decisions. It should be no problem to extend guidelines by such well-known exceptions. This will give the developer a better impression of the sense of a specific design rule and the context in which to use it.

5.1.2 Focus and level of detail

A guideline must explain clearly what its purpose is and what it was not intended for. To write a UI guideline for an in any sense general purpose seems not to be practicable with good results. So UI guidelines should always be specific for any topic. This will also reduce their size. Examples for such topics is a platform dependency or a restriction to a special type of application.

Instead of explaining all the well known components of an interface detailed and separated from each other, guidelines should focus more on the common use of components and their combinations to convey a more practical point of view. This will also reduce the size of the documents and increase their readability.

The average developer doesn’t need a lengthy explanation of which controls must be added to a window so the user can close, minimize, restore and maximize a window. These functions are generally offered automatically by the current development tools or the underlying operating system. Guidelines don’t need to explain things the average developer can not or doesn't want to influence in any way.

5.1.3 Illustration and Lack of bad examples

These problems are very easy to solve. The authors of UI guidelines just need to be aware of them and add more illustrations, graphic examples and bad examples to their documents. Although this will extend the guideline it is doubly that this will increase the size-problem. Probably an extensive use of illustrations will increase readability and the suggestions of the guideline will be remembered easier. By this more illustrations may even reduce the problem of size.

5.1.4 Support by development tools

This problem is also easy to solve but it is not sure at all if this will be done. The development tools must be adapted to support the use of UI guidelines. I.e., it will not be difficult to add a function to support the design grids mentioned in [Sun 99]. When labeling and placing buttons it should be easy to use standard settings for distances and gaps like suggested in a UI guideline. These standard settings must be easy to change.

5.1.5 Size

To make working with huge UI guidelines easier there are so called ‘Tools for Working with Guidelines (TFWWG)’. These will be discussed more detailed in an own section (Chapter 5.3, "Tools for Working with Guidelines").

5.1.6 Problems with developing your own UI guideline

The problems of introducing the document, updating, its purpose and the responsibility will also be discussed in a separate section (Chapter 5.2, "Creating a UI guideline"). [Gale 95] offers a collaborative approach to develop guidelines to overcome these problems.

5.1.7 Integration

[Vanderdonckt 99b] suggests taking special care in creating guidelines for the following roles to make the integration easier:

- A task analyst. He takes the greatest benefit of guidelines that are organized according to a taxonomy characterized by task attributes, and not according to interaction styles (e.g., information retrieval, decision taking, form fill-in, multi-criteria search, problem solving).
- A project leader. He only pays attention to high level guidelines (e.g., selecting an appropriate metaphor).
- A human factors expert. He is experienced in cognitive principles and may assume that guidelines are sorted by them.
- A designer. Most important for him is to instantly isolate specific guidelines and solve conflicts between them by ranking them into ordered sequences.
- A programmer. This part is mainly covered by the responsibility for the widgets of the program, which should be reflected in the organization of the guidelines.
- A UI evaluator. He prefers guidelines expressing UI usability in terms of measures.

(List principally taken from [Vanderdonckt 99b].)

5.2 Creating a UI guideline

This section is a summary of [Gale 95] and presents an approach to create a new UI guideline avoiding some common problems. The main idea is the involvement of developers and end users in the production of the new UI guideline ("collaborative approach"). This production is separated in 5 steps and will be explained in the following.

1. *"Raising awareness amongst developers and end users"*

The need for a document like a new UI guideline must be aware to the developers, end users and to the management. This awareness cannot be stated as present but must be created among all participants.

2. *"Building consensus"*

So called "Working parties" that consist of key developers and end users from each of the teams within an organization should work on a weekly basis over a period of time. These parties review and comment some specific topics (e.g. Keyboard Interaction, Navigation, Terminology) to form a chapter in the new UI guideline after revising the material.

3. *"Documenting the UI guideline"*

This documenting starts after a certain level of consensus has been reached to ensure that the commitment and understanding of the developers is gained.

4. *"Providing training and support material"*

Introducing the new document into the organization is a very critical part. A new UI guideline should be supported by the use of training and support materials, which are provided by the members of the working parties.

5. *"Establishing an environment which enables the guide to evolve"*

It is also very important to make clear that a new UI guideline is not a static document. Working with the document will produce new good ideas that should be fixed by expanding the guideline. To do this effectively there must be an environment in which the document can grow over time.

The approach in [Gale 95] contains apart from these five steps other aspects.

"Integrated with the development process"

Most of the currently published UI guidelines do not support any kind of development process at all. But it is important for such a document that it supports the development methods being used in the project. If it's intended to give this support the authors of the document must be aware that there are different ways in which the document could be used.

"Getting it right in the first place"

UI guidelines are often used to check a user interface after its implementation. Although this is a valid use, it is more productive to use a UI guideline right from the beginning of a project, e.g. to teach new programmers.

"Communicate strong visual and functional style"

Producing distinctive designs that are still conform with the industry standards is a challenge. The corporate image of an organization is constantly evolving which must be also reflected in the UI guideline.

"Encompass relevant platforms and functionality"

This means to restrict the scope of a UI guideline because it is impossible to produce a 'general-purpose' guideline. One document cannot cover all the aspects from back office to customer facing systems effectively.

"Developed alongside a development team"

Developing the document alongside the development team ensures that the designs are practical and implementable. Commitment to the use of the guide is gained quick and directly. The developers like to use the guide because they were involved in its production.

"Not just a document"

A UI guideline must not be just a large document that everyone has to read. To convey the contents of the guide more effectively [Gale 95] suggests to make also use of the following communication methods:

- "Interactive demonstrations to illustrate the look and feel. Some of these demonstrations have a mode which allow developers to browse the layout of the screen to pixel level;
- Checklists to ensure conformity;
- Reference tables for quick reference of critical data. These have been produced as "bumper stickers" which developers have attached to their monitors;
- Competitions to spot the deliberate mistakes in a user interface design [...];
- Source code, custom controls and property tables which promote sharing user interface code."

"Ensuring ownership"

The collaborative approach presented here helps to produce a feeling of ownership by itself, but a group or individual in the organization must be identified explicitly as being responsible for the maintenance and updating of the guide. These owners often become the focal point for the guide and end up providing developers with ad hoc advice on generic user interface design issues.

This approach in [Gale 95] does not solve all the problems that come up with UI guidelines but a document that is produced reflecting these steps and aspects produces at least less problems and will hopefully be more effectively compared to documents not using this approach.

5.3 Tools for Working with Guidelines

The SIG (Special Interest Group) “Tools for Working with Guidelines (TFWWG)” is a SIG in SIGCHI (Special Interest Group on Computer-Human Interaction) at ACM (Association for Computing Machinery) itself. The first time it has been launched was during the CHI’94 conference in Boston.

TFWWG is a term to describe a group of programs that give software-support to work with a set of guidelines. There are basically two types of TFWWG: Passive tools and active tools.

- *Passive tools* present just a new interface to look at the set of guidelines. The guidelines are in a form that cannot be applied, evaluated automatically or in a computer-aided way. Examples in this category are hypertexts or other kinds of hypermedia, database tools, advice-giving systems and critique tools. This kind of tool is primarily intended to support design and evaluation phases, so their main goal is to provide assistance.
- *Active tools* are programs helping the designer developing a new UI and are based on guidelines. These guidelines can be applied, evaluated automatically or in a computer-aided way. In this category there are design aid tools, expert systems and tools for automated UI generation or evaluation.

A not up to date overview of TFWWG is presented in [Vanderdonckt 94]. A more current overview of the state on research about TFWWG and the programs themselves can be found in [Dufour 97]. In this section I will summarize the introduction to TFWWG from [Dufour 97].

[Dufour 97] divides TFWWG into four groups:

1. Software that supports designing of an interface
2. Software that helps evaluating an interface
3. Learning software on the subject of interface design that may be specific for a platform such as MS-Windows or Apple-Macintosh
4. Prototyping software

The boundaries between these groups are not sharp as for example products of group 2 can also be used in group 1, but at least the opposite is not always true.

To give an impression what TFWWG provide I cite the "non-exhaustive" list of functions that were faced in the review-process in [Dufour 97]:

- *Browsing* provides the capability to jump from a guideline to the next one (or to another one) by means of a button or key. The guideline order is given hierarchically.
- *Navigation* provides the capability to jump directly to a particular guideline by means of a displayed hierarchy for example. Backtracking is included in this definition.
- *History* provides the capability to create a new window with the path covered by the user and to jump to a previous state.
- *Gathering* provides the capability to collect relevant guidelines for the user and to put them in a separated window or file.
- *Keyword searching* provides the capability to look for guidelines containing a specific word. This may contain searching for a keyword in guideline titles, searching for a conjunction or disjunction of keywords in the development of guidelines or searching for any type of word in the development of guidelines.
- *Index* is a list of keywords with links to guidelines concerning these words.
- *Glossary* is a definition of important words.
- *Sorting* provides the capability to sort (selected) guidelines according to their importance, their hierarchic order or their usage frequency for example.
- *Rating* provides the capability to give a guideline a score or evaluation (rating compliance or usage frequency for example).
- *Reporting* provides the capability to print a report.
- *Annotating* provides the capability to add comments about a guideline."

It was found that the most important functions for software for designing an interface are Navigation, Keyword searching and an Index. Characteristics of software for evaluating an interface are Gathering, Sorting, Rating, Reporting and Annotating. Key functions of learning software is Browsing and a Glossary. Prototyping software was not analyzed in this document.

For a deeper introduction and a better overview it is recommended to look at [Dufour 97]. How effective TFWWG in fact are or if there are other problems that arise with them cannot be examined in this investigation. A small hint can be found in [Vanderdonck 95], where it is said that TFWWG "replicate usability problems from one implementation to another". For example it is questionable if TFWWG can solve problems such as Consistency and usability, Generalization, Illustration, Correctness or Obviousness mentioned in chapter 4 ("Usability problems with these guidelines").

5.4 Other ideas

Here are two ideas that are not meant to solve a particular problem but also can offer a better effect when working with UI guidelines.

5.4.1 Guidelines for “macro” objects

[Schumacher 92] introduced in his document the idea of creating guidelines for “macro” user interface objects. Candidates for such macro-objects are standard dialog boxes, message boxes or sub-components like button-panels, edit areas, toolbars, menus and many others.

If a developer could work with such macro-objects the progress of the development would be faster. Designing such common elements he wouldn't need to take care of the low-level design rules.

Such macro-objects could also give good examples for a correct implementation of these rules.

5.4.2 Checklists

This element of a UI guideline is not a new idea. E.g., it is suggested in the template of a table of contents in [Gale 95]. But [Apple 95] is the only document that contains such a checklist. The big advantage of a checklist in the appendix of a UI guideline is that conformity with the guidelines can be checked very easily. Answering the questions brings very easily to mind the particulars of the guidelines. Such checklists should be available much more.

6 Summary

After explaining what aspects UI guidelines that are available as long and short documents deal with on an abstract and a specific level related to the development process and the software product I encountered numerous problems when using these guidelines. These problems are related directly to the contents of the guidelines or to the process of handling the documents. I gave ideas to increase the usability of guidelines but it is hard to say to what extent these improvements will lead.

I can now answer some of the questions raised in chapter 1.2 ("Aspects to investigate in this paper"). It is not sure that using UI guidelines really will help developers of software. These guidelines are hardly able to ensure more than a consistent interface. Larger improvements cannot be expected. Whether consistency can be achieved at all is questionable. At least the persons that should adapt guidelines must learn (or be taught) how to work with them. There are a lot of pitfalls as mentioned in chapter 4 ("Usability problems with these guidelines") and someone must be aware of them before starting to work with guidelines.

Benefits can be achieved but the environment must be adapted to the new method of using such documents. A company is getting a better idea of what can be achieved with UI guidelines if the document is used as a shared memory for conventions than to look at it as a list of prescriptions.

“There are many methodologies, theories and guidelines for how to produce a good user interface (each CHI conference proceedings is likely to have a few). [...]. Although there are a number of reports of successful systems created using various methodologies, evidence suggests that the skill of the designers was the primary contributor to the quality of the interface, rather than the method or theory.” [Myers 93] This is also true for UI guidelines. If a software-team wants to improve their work by using guidelines they need a *person* who is more or less an expert on this subject. The guidelines themselves will be useless. Just to have the document at the office available to everyone will help nothing.

To conclude I want to mention doubts if any method, theory or tool can improve the results of UI design in an amount worth mentioning. [Myers 93] contains a list of problems that are inherent to the design of a UI:

- "Designers have difficulty learning the user's tasks.
- The tasks and domains are complex.

-
- There are many different aspects to the design which must all be balanced, such as standards, graphic design, technical writing, internationalization, performance, multiple levels of detail, social factors, legal issues, and implementation time.
 - The existing theories and guidelines are not sufficient.
 - Iterative design is difficult.

[...]

- The software must be especially robust while supporting aborting and undoing of all actions."

How can these problems be solved? Can they be solved at all? The result would be tremendous, but it is more likely that HCI experts for user interfaces will be the ones that can help to produce good UIs.

“Most articles about design of human-computer interfaces (HCI) start off with a comment like ‘Because user interfaces are hard to design...’ and then propose a method or tool to help.” [Myers 93] I did this too but I'm sure that UI guidelines indeed can help although this will not be self-evident but something the developer has to spend time for to get comfortable with. I expect using guidelines for designing user interfaces will stay a method someone has to learn explicitly. Although it is unlikely that UI guidelines can solve the problems mentioned in [Myers 93] they will help to get better results.

Acknowledgements: I want to thank Prof. Jean Vanderdonckt from the Université catholique de Louvain (Belgium) whose support about the subject “Tools for working with Guidelines” was very helpful in preparing this paper.

I want to thank Dr. Julio Aspiazu and Mr. Dirk Möller from pirAMide Informatik GmbH for offering me the opportunity to prepare this paper in the context of the development of ELWIS.

I want to thank Prof. Dr. Horst Oberquelle from the Department for Informatics of the University of Hamburg who was willing to give his advice from Hamburg while I was working in Cochabamba.

7 References

[Apple 95]

Apple Computer, Inc.

Macintosh Human Interface Guidelines

<http://developer.apple.com/techpubs/mac/pdf/HIGuidelines.pdf> (19.04.1999)

[Apple 97]

Apple Computer, Inc.

Mac OS 8 Human Interface Guidelines

<http://developer.apple.com/techpubs/mac/pdf/HIGOS8Guidelines.pdf> (19.04.1999)

[Berry]

Dick Berry

Designing the Out-Of-Box User Experience

IBM, Ease of Use

<http://www.ibm.com/ibm/easy/design/lower/p050000.html> (21.04.1999)

[Bevan]

Nigel Bevan

User-centred design of man-machine interfaces

<ftp://ftp.npl.co.uk/pub/hci/papers/design.rtf> (15.04.1999)

[Bevan 96]

Nigel Bevan, Dr. Jurek Kirakowski

User-centred design

<http://info.lboro.ac.uk/research/husat/eusc/ucd.doc> (15.04.1999)

[Crow 94]

Daniel Crow

Designing for people: The HalClon project

E-mail from Jean Vanderdonckt <vanderdonckt@qant.ucl.ac.be> (18.6.1999)

[Dufour 97]

Requirements Proposal for a Generic Tool for Working with Guidelines

Yves Dufour, Didier Rensonnet, 1997

E-mail from Jean Vanderdonckt <vanderdonckt@qant.ucl.ac.be> (24.6.1999)

[Gale 95]

Stephen Gale

A Collaborative Approach to Developing Style Guides

CHI 96 - Electronic Proceedings, December 1995

http://www.acm.org/sigchi/chi96/proceedings/papers/Gale/srg_txt.htm (16.04.1999)

[Harrison 98]

John Harrison

Designing the User Interface

Department of Computer Science and Electrical Engineering,

The University of Queensland, Brisbane, September 1998

<http://www.csee.uq.edu.au/~cs116/slides/ui/html/index.htm> (13.04.1999)

[Hawkes 98]

Adriene Hawkes

Human-Friendly Computer Instructional Systems: Guidelines for text color, background color and the reduction of eyestrain

NOVA Southeastern University, May 1998

<http://www.cybernw.com/~clear/2colort1.html> (14.04.1999)

[Horton 96]

William Horton

Designing icons and visual symbols

CHI 96 - Electronic Proceedings

http://www.acm.org/sigchi/chi96/proceedings/tutorial/Horton/wh_txt.htm (16.04.1999)

[Horton 97]

William Horton

Designing icons and visual symbols

CHI 97 Electronic Publications

<http://www.acm.org/sigchi/chi97/proceedings/tutorial/wh.htm> (16.04.1999)

[Iannella 94]

Renato Iannella

“BRUITASAM – A HyperCard Tool for Browsing Large User Interface Guidelines Sets
E-mail from Jean Vanderdonckt <vanderdonckt@qant.ucl.ac.be> (18.6.1999)

[IBM 97]

IBM

Human-Computer Interaction – UI Guidelines

http://www.ibm.com/ibm/HCI/guidelines/design/ui_design.html (15.04.1999)

[IIMRR 99]

Interactive Instructional Material Research and Resources

The Human-Computer Interface General Principles

<http://www.und.ac.za/users/murrell/classrm/genprin.html> (13.04.1999)

[Inuse 97]

Inuse

User-centred design

http://www.lboro.ac.uk/research/husat/eusc/g_user_centred_design.html (15.04.1999)

[Mandel 99]

Theo Mandel

what's new - ui café - Tips & Hints, March 1999

Interface Design and Development

<http://www.interface-design.net/ui-tips.htm> (30.03.1999)

[Marchionini 91]

Gary Marchionini

Psychological Dimensions of User-Computer Interfaces

http://www.ed.gov/databases/ERIC_Digests/ed337203.html (07.04.1999)

[Microsoft 97]

Microsoft Corporation

The Windows Interface Guidelines for Software Design

<http://www.microsoft.com/win32dev/uiguide/> (20.04.1999)

[Myers 93]

Brad A. Myers

Why are Human-Computer Interfaces Difficult to Design and Implement?

Carnegie Mellon University School of Computer Science, July 1993

Technical Report, no. CMU-CS-93-183

<http://reports-archive.adm.cs.cmu.edu/anon/1993/CMU-CS-93-183.ps> (14.04.1999)

[NASA 96]

NASA, Goddard Space Flight Center

User-Interface Guidelines

Data Systems Technology Division/Code 520

http://groucho.gsfc.nasa.gov/Code_520/Code_522/Documents/UG_96/newfrontmatter.html

(RTF-copy on http://groucho.gsfc.nasa.gov/Code_520/Code_522/Documents/UG_96/UG1.rtf)

(07.04.1999)

[Ogawa 94]

Katsuhiko Ogawa, Kaori Ueno

Guidebook: Design guidelines database in assisting interface design task

E-mail from Jean Vanderdonckt <vanderdonckt@qant.ucl.ac.be> (18.6.1999)

[Parkhurst 96]

Carol A. Parkhurst

Principles of Human-Computer Interface Design

University of Nevada, Reno, October 1996

<http://www.library.unr.edu/~carolp/hciweb.html> (09.04.1999)

[Raskin 94]

Jef Raskin

Intuitive equals familiar

Communications of the ACM. 37:9, September 1994, pg. 17.

<http://www.asktog.com/papers/raskinintuit.html> (12.04.1999)

[Rizzo 96]

A. Rizzo, O. Parlangeli, E. Marchigiani and S. Bagnara

HCI in Italy: The Management of Human Errors in User-Centered Design

SIGCHI Bulletin, Vol.28 No.3, July 1996

<http://www.acm.org/sigchi/bulletin/1996.3/rizzo.html> (16.04.1999)

[Schank 95]

Roger C. Schank, Lawrence Birnbaum

ILS Interface Design: Principles and Design Guidelines

Institute for the Learning Sciences, Northwestern University, February 1995

ftp://techreport.ils.nwu.edu/reports/word/tr-62-interface_design.rtf (14.04.1999)

[Schumacher 92]

Robert M. Schumacher, Jr.

Ameritech Graphical User Interface Standards and Design Guidelines

<http://www.ameritech.com/corporate/testtown/library/standard/std-guix.html> (09.04.1999)

[Shackel 97]

B. Shackel

Aspects of usability

Material drawn principally from:

"Usability - context, framework, definition, design and framework", B. Shackel

In: Human factors for Informatics in Usability,

Ed. B. Shackel and S. Richardson, Cambridge University Press, 1991.

http://info.lut.ac.uk/research/husat/inuse/f_usability_aspects.html (15.04.1999)

[Stultz 98]

DC Stultz

Programmer-responsibility

AskTog, Reader Mail

<http://www.asktog.com/readerMail/1998-12ReaderMail.html> (12.04.1999)

[Sun 99]

Sun Microsystems, Inc.

Java(TM) Look and Feel Design Guidelines

Early Access Release, April 1999

<http://java.sun.com/products/jlff/guidelines.html> (29.04.1999)

[Tidwell 98a]

Jenifer Tidwell

The case for HCI design patterns

http://www.mit.edu/~jtidwell/ui_patterns_essay.html (16.04.1999)

[Tidwell 98b]

Jenifer Tidwell

Common ground: A Pattern Language for Human-Computer Interface Design

http://www.mit.edu/~jtidwell/common_ground.html (16.04.1999)**[TOG 98a]**

Bruce Tognazzini

Maximizing Windows

<http://www.asktog.com/columns/000maxscrns.html> (12.04.1999)**[TOG 98b]**

Bruce Tognazzini

Designers as Architects

<http://www.asktog.com/columns/004designerasarch.html> (12.04.1999)**[TOG 98c]**

Bruce Tognazzini

Justifying Rough Sketches

<http://www.asktog.com/columns/005roughsketches.html> (12.04.1999)**[TOG 98d]**

Bruce Tognazzini

Intuitive vs. Familiar

<http://www.asktog.com/columns/006intuitvsfamiliar.html> (12.04.1999)**[TOG 98e]**

Bruce Tognazzini

Designing Single-Use Applications

<http://www.asktog.com/columns/010minaturegolf.html> (12.04.1999)**[TOG 98f]**

Bruce Tognazzini

The Polite Interface or Guidelines for Dialogs

<http://www.asktog.com/columns/012dialogGuidelines.html> (12.04.1999)**[TOG 98g]**

Bruce Tognazzini

How to Publish a Great User Manual

<http://www.asktog.com/columns/017ManualWriting.html> (12.04.1999)

[TOG 98h]

Bruce Tognazzini

First Principles

<http://www.asktog.com/basics/firstPrinciples.html> (12.04.1999)**[TOG 98i]**

Bruce Tognazzini

\$1.98, Close-coupled Usability Testing

<http://www.asktog.com/columns/001closecoupleddtesting.html> (12.04.1999)**[TOG 99a]**

Bruce Tognazzini

A Quiz Designed to Give You Fitts

<http://www.asktog.com/columns/022DesignedToGiveFitts.html> (12.04.1999)**[TOG 99b]**

Bruce Tognazzini

Maximizing Human Performance

<http://www.asktog.com/columns/023MaxHumanPerf.html> (12.04.1999)**[TOG 99c]**

Bruce Tognazzini

Context-(In)Sensitive Menus

AskTog, Reader Mail

<http://www.asktog.com/readerMail/1999-01ReaderMail.html> (12.04.1999)**[UIE 96a]**

UI-Engineering

Harnessing the Power of Myth

<http://world.std.com/~uieweb/myth.htm> (12.04.1999)**[UIE 96b]**

UI-Engineering

Making Tips Work

<http://world.std.com/~uieweb/tips.htm> (12.04.1999)

[US DE 97]

US Department of Education

Requirements for accessible software design

Version 1.1, March 1997

<http://gcs.ed.gov/coninfo/clibrary/software.htm> (07.04.1999)

[Vanderdonckt 94]

Jean Vanderdonckt et al.

Tools for Working with Guidelines

A CHI'94 Special Interest Group, April 27th, 1994, Boston

[Vanderdonckt 95]

Jean Vanderdonckt

Accessing guidelines information with SIERRA

In Proceedings of Fifth IFIP TC 13 International Conference on Human-Computer Interaction INTERACT'95 (Lillehammer, 27-29 June 1995), K. Nordbyn, P. H. Helmersen, D.J. Gilmore and S.A. Arnesen (Eds.), Chapman & Hall, London, 1995, pp. 311-316.

[Vanderdonckt 99a]

Jean Vanderdonckt

Automated Generation of an On-Line Guidelines Repository

To appear in "Proceedings of 8th International Conference On Human-Computer Interaction of HCI International '99", Munich, 1999

[Vanderdonckt 99b]

Jean Vanderdonckt

Development milestones towards a tool for working with guidelines

To appear in "Interacting with Computers", Vol. 11 (No. 4), 1999

[VR a]

Vertical Research

Designing a UI in a Win95 world

<http://www.vrix.com/design/design.htm> (12.04.1999)

[VR b]

Vertical Research

A page from our Desktop Quick Reference Style Guide

<http://www.vrix.com/style/style.htm> (12.04.1999)

[Wildstrom 98]

Stephen H. Wildstrom

A computer user's manifesto

Businessweek September 28, 1998

<http://www.businessweek.com/1998/39/b3597037.htm> (12.04.1999)