

Difficulties in Using Style Guides for Designing User Interfaces

Thomas Vogt

Department for Informatics, University of Hamburg
Vogt-Kölln-Strasse 30
22527 Hamburg, Germany
3vogt@informatik.uni-hamburg.de

Abstract. In [16] it was investigated how one can work with style guides called “UI guidelines” in this article when designing user interfaces in commercial software development. The document explained what aspects style guides deal with on an abstract and a specific level related to the development process and the software product. During this research various problems related to the use of style guides were encountered. These problems and possible solutions - or at least part solutions - shall be presented here.

1 Usability Problems with Style Guides

The usability problems that arise with using style guides are numerous. Some papers [2,8,9,10,13,14,15] deal with these problems. The problems can mainly be divided into two groups: Problems with the contents of a style guide and problems with handling the document. Especially [2] delivers many common different reasons why style guides fail. Also [15] contains a list of difficulties related to style guides. In general terms one of their difficulties is insufficiency, although their necessity is not questioned.

1.1 Problems with the Contents of Style Guides

These problems are related to the guidelines and design rules which are explained in a style guide. The prescriptions and suggestions that are given in a style guide can be problematic in different ways.

Consistency and Usability. All style guides say that ‘Consistency’ is the most or at least one of the most important principles. Of course many authors [2,8,14,15] mention that consistency alone is not sufficient and that usability cannot be reached by using style guides alone. Even interfaces with a consistent design and behavior can be very uncomfortable to use and therefore have a very low usability. “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.” [2]. [15] states that more than ergonomic sources are needed to guarantee UI usability. Style guides are one type of ergonomic source and they could be used together with “scenario- or use case-based design/evaluation user tests.”

Only [14] offers information about this shortcoming explicitly. In its introduction it describes its intention and the steps that must be taken separately because it “cannot substitute for knowledge of the information requirements associated with decision points embedded in the user’s tasks.”

Generalization. [8] doubts the fundamental base for style guides: According to this document the history of research about HCI showed that the generalization of UI design issues failed. If these issues cannot be generalized the sense of style guides for any generalized purpose must be questioned.

Focus and Level of Detail. Different authors describe problems with the various common aims that style guides have and the resulting details they contain. Often the focus is not clear enough. The document may be written for too many different types of applications or users. By addressing all these different goals the style guide does not cover many aspects sufficiently [2].

[9] uses the term “interface object catalog” for many style guides which just describe all kinds of objects that are available for an interface and their behavior. If a style guide is nothing more than an interface object catalog it lacks recommendations of where and where not to use the various objects or combinations of them and suggestions about design practices in general. An excellent example of such an interface object catalog is [11].

Style guides sometimes contain too many details. [5] states “Window behavior, such as resizing, dragging, minimizing, positioning, and layering, is controlled by the native operating system.” It seems that a lengthy explanation of these aspects is unnecessary because the average developer cannot or does not intend to influence this behavior in any way. However, within this style guide describing the item of windows exactly all these aspects and their characteristics are explained. The advantage that these things are controlled by the operating system and the developer doesn’t have to take care of them is not addressed by the document.

Conflicts. When using style guides many things can cause conflicts. For example, a contradictory or an inconsistent result [15]. Often usability questions have more than one valid answer which is only one example of how inconsistent results can be achieved. Examples for conflicts related to some specific aspects of the developed software product such as icons, dialog boxes and capitalization as an aspect of layout and visual alignment can be found in [16]. But conflicts can also arise when working with only one style guide. There may be more than one principle that can be used for a special design problem and they may suggest different solutions, but “No style guide can infallibly tell you how to strike a balance between two opposing high-level principles.” [13]

Conflicts may also arise when working with just one design rule, not comparing it to any other. For example, such a basic principle as consistency may not be appropriate in all circumstances. When it comes to dangerous operations it can be applicable to break with this principle. This can be done by not having a default selection for confirmation although this behavior will be different compared to the rest of the system [8].

Illustration. [15] says that style guides are illustrated insufficiently. This is obviously true for [3,4,12] since they are not illustrated in any way. For [4] this is excusable because of its abstract principles that are very difficult to illustrate without giving specific examples. Although [9] and [14] 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 cannot be generalized. [5,6,7] are good examples for the use of illustrations.

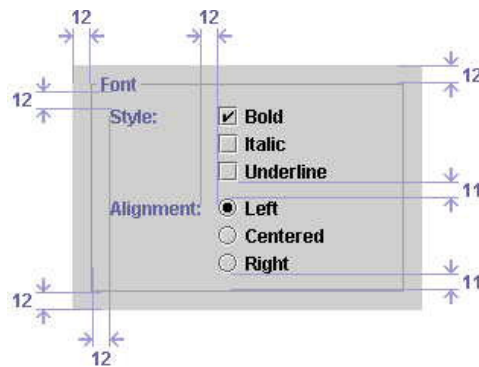


Fig. 1. Example of an illustration that is much more powerful than its textual counterpart

Correctness. It is also important to note that in the absence of conflicts the style guides cannot be trusted blindly. [15] says that some sources, although few, contain incorrect or contentious guidelines that are very difficult to detect at all. Some contradictions shown in [16] give a subtle hint of this incorrectness.

Transience. The rapid progress in the technical surroundings of software development is another problem. The technical environment (operating systems, programming languages etc.) is changing so fast that style guides become transient [13]. The need for a regularly update is also explained in [2]. The style guides have to adapt to the changes of technical conditions. This has to be done very quickly 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.” [10]

Lack of Bad Examples. A lot of style guides fail to give bad examples, so called “Don’ts”, e.g., [5,6,9,11]. 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 one 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.

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.” [5].

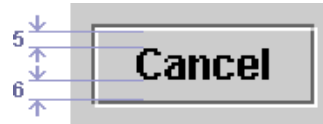


Fig. 2. Placement of button text

In the context of this document it is explained that a “flush” 3D appearance with a white line is the reason for non-equal distances. But why is the lower distance 6 pixel? Why not 5 or 7? Maybe it is related to the size of the font, but there isn't an 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.” [5] Why not 4 or 6 pixels? People have to learn rules like this like vocabulary. This is more difficult than learning something by understanding.

In [11] we can also find rules for spacing that are similar to these rules taken from [5]. But the spacing in [11] 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. It can be expected that he will ignore these rules or just work with some similar rules of thumb. As [8] says: “However, these style guides are usually hard to interpret and apply.”

Support by Development Tools. The common development tools do not support a UI design on the level of pixels easily. 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.

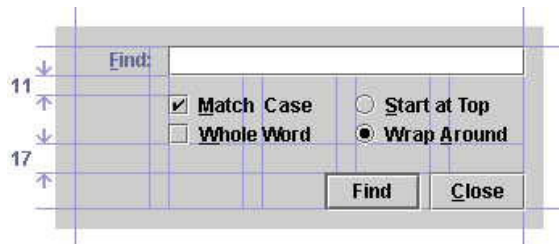


Fig. 3. Example of a design grid

Neither do they support the use of design grids as recommended in [5]. Design grids can be very useful to achieve a consistent and visually appealing layout but the average developer never has the possibility to use them.

Size. A general problem is that style guides are very big. The documents have an extent which make their handling uncomfortable. “It is ironic that many of the existing Style Guides, aimed at producing usable systems, are themselves difficult to use.” [2]

1.2 Problems with Developing Your Own Style Guide

The following two sections contain aspects that are related to the handling of a style guide. The aspects of creating a new style guide on your own are covered first. They all are also valid for the case of using a style guide from third parties (Sect. 1.3), but not the other way round as explained below.

Introducing the Document. The existence of a new style guide alone is not sufficient. The developers need to know how to work with it. [2] criticizes the lack of considerations about the style guides introduction and the absence of a responsible person to accompany the developers in this process.

Purpose. The effects and benefits offered by style guides are often overestimated. Such a document cannot ensure usability on its own but further actions are necessary to achieve this. “The production of a Style Guide is seen as a substitute for iterative design and user testing.” [2] which of course is inadequate.

Responsibility. Sometimes the responsibility for the style guide is not assigned properly. This means that other people than the development team and users are responsible for the creation of the style guide [2].

Incompleteness. Any collection of guidelines will be incomplete because of the three types of questions - unraised questions, raised but unsolved questions and raised but only partially solved questions - explained in [15].

1.3 Problems with Using a Style Guide From Third Parties

Now the problems will be explained that arise using an already finished document that was created by someone else. These aspects are more or less exclusive for this case as they are only of minor importance for style guides that are developed from scratch. Of course they become very important if one tries to use already published documents while one is creating his own style guide.

Number. The amount of style guides and related materials that are offered is enormous. Very fast one encounters a huge number of documents. Some of them are very large in themselves (problem of size) and it is very difficult to get an overview of the whole collection of documents.

Access. [15] mentions the difficulty to find guidelines in all the widespread sources. The problem becomes more severe for papers dealing with only one specific subject. Such articles are almost impossible to find amongst all the literature about HCI. Some questions therefore will remain unsolved in a special project because the answer cannot be found even though it exists.

Integration. This problem is similar to the problem of introducing a style guide that has newly been developed. Also when working with style guides from third parties the developers need to know how to use the document and how it can be integrated in their environment. Normally this subject and all such related problems are not covered in the existing style guides [2,15].

2 Ideas to Make Style Guides More Usable

As shown there are numerous problems that occur when working with style guides. It is hoped that the efficiency of style guides will rise, when these problems are solved. However, I will show now that only some of these problems can be solved. Some other problems are likely not to be solved and will remain for the future probably. Aside from these problems I will also show other ideas that could make style guides more usable.

2.1 Ideas to Solve the Mentioned Problems

Addressing problems that are unlikely to be solved [8] states that research showed that the problem of generalization would not be solved in the future. [15] says the same for incompleteness, because it is intrinsic to the style guides themselves. It will be explained shortly why also no solutions can be found for the problems of conflicts, correctness, transience, obviousness and the problems with using a style guide from third parties like number and access:

Conflicts and Correctness: These are also intrinsic problems.

Transience: It cannot be expected that the progress in developing new development tools will slow down in the near future, so this will remain a problem.

Obviousness: This aspect 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.

Number: With time the number of style guides will exceed and therefore also the problem related to this.

Access: It is also questionable if the access to these style guides, especially the web sources, will be easier as the web evolves in the near future.

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

Consistency and Usability. The main point was that consistency alone does not ensure usability at all. Therefore [2] suggests incorporating “HCI principles which help ensure ease of use and minimize learning of the application” to come up with a truly comprehensive style guide. Many more aspects can be added to style guides to support the realization of an easy to use interface.

[8] 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 style guides 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.

Focus and Level of Detail. A style guide must explain clearly what its purpose is and what it was not intended for. To write a style guide for a general purpose seems not to be practicable with good results. So style guides should always be specifically for a particular topic. This will also reduce their size. Examples for such topics are a platform dependency or a restriction to a special type of application.

Instead of explaining all the well known components of an interface separately and in detail, style guides 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. Style guides don't need to explain things the average developer can not or does not intend to influence in any way.

Illustration and Lack of Bad Examples. These problems are very easy to solve. The authors of style guides just need to be more aware of them and add more illustrations, graphic examples and bad examples to their documents. Although this will extend the style guide it is doubtful that this will increase the size-problem. An extensive use of illustrations will also increase readability and the suggestions of the style guide will be remembered more easily. The increased use of illustrations may even reduce the problem of size.

The illustrations must be designed very carefully because they must be compliant with the whole style guide and not only to the particular rule they are related to. Through this the users of the document can't make any wrong use of the illustrations.

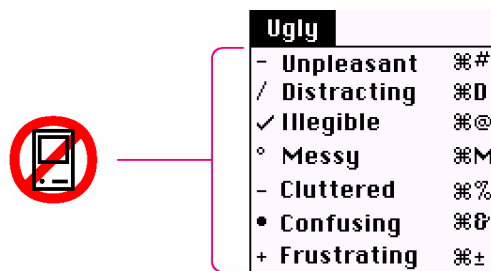


Fig. 4. Example of an illustration of a bad example

It is very important to convey what is an illustration of one or more specific guidelines and what is meant to be a bad example. [7] uses this technique very consequently making the difference between “Do’s” and “Don’ts” always very clear.

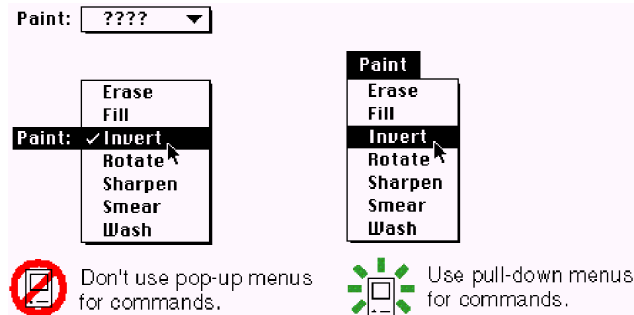


Fig. 5. Example of a “Do” with corresponding “Don’t”

Support by Development Tools. This problem is also easy to solve but it is uncertain if this will ever be done. The development tools must be adapted to support the use of style guides. For example, it will not be difficult to add a function to support the design grids mentioned in [5]. When labeling and placing buttons it should be easy to use standard settings for distances and gaps like suggested in a style guide. These standard settings must be easy to change.

Size. To make working with huge style guides easier there are the ‘Tools for Working with Guidelines (TFWWG)’. An excellent overview about the subject of Tools for Working with Guidelines is given in [1]. The introduction of this document contains a taxonomy of related programs and their various types of functionality. A short overview based on this document can be found in [16].

Problems with Developing Your Own Style Guide. The problems of introducing the document, its purpose and the responsibility will be discussed in a separate section (Sect. 2.2, “Creating a Style Guide”). [2] offers a collaborative approach to develop style guides to overcome these problems.

Integration. [15] suggests taking special care in creating style guides for the following roles to make the integration easier:

A task analyst. He takes the greatest benefit of style guides 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 style guide.

A UI evaluator. He prefers guidelines expressing UI usability in terms of measures.
(List principally taken from [15].)

2.2 Creating a Style Guide

This section is a summary of [2] and presents an approach to create a new style guide avoiding some common problems. The main idea is the involvement of developers and end users in the production of the new style guide (“collaborative approach”). This production is separated into 5 steps and is outlined here:

1. *"Raising awareness amongst developers and end users"*: The need for a document like a new style guide must be aware to all participants of the development process. This awareness cannot be stated as present but must be created explicitly.
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 style guide after revising the material.
3. *"Documenting the style guide"*: 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"*: A new style guide 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"*: A new style guide is not a static document. Working with the document will produce new good ideas that should be fixed by expanding the style guide. To do this effectively there must be an environment in which the document can grow over time.

The approach in [2] also contains some other aspects apart from these five steps.

Most of the currently published style guides do not support any kind of development process at all. It is, however, important for such a document to support 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.

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

Producing distinctive designs that are still compliant with the industry standards is a challenge. The corporate image of an organization is constantly evolving and this must also be reflected in the style guide.

The scope of a style guide needs to be restricted because it is impossible to produce a 'general-purpose' style guide. One document cannot cover all the aspects from back office to customer facing systems effectively.

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.

A style guide must not be just a large document that everyone has to read. To convey the contents of the guide more effectively [2] suggests to make use of different communication methods, such as interactive demonstrations, checklists to ensure conformity, reference tables for quick reference of critical data and competitions to spot the deliberate mistakes in a user interface design.

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.

The approach in [2] does not solve all the problems that come up with style guides but a document that is produced reflecting these steps and aspects at least produces less problems and will hopefully be more effective compared to documents not using this approach.

2.3 Other Ideas

The following are two ideas that are not meant to solve a particular problem but also can offer a better effect when working with style guides.

Guidelines for “Macro” Objects. [9] 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. When 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.

Checklists. This element of a style guide is not a new idea. For example, it is suggested in the template of a table of contents in [2]. But checklists are not integrated frequently to style guides and should be more readily available. [7] is one of the very few documents that contains such a checklist.

Example. Here is an excerpt from the checklist in [7] dealing with graphic design:

- Did you use graphics to illustrate commands, features, and parameters of the applications, as well as all of the user's data, whenever possible?

- Do the graphics resemble items that users are familiar with? Did you leave out insignificant detail (which unnecessarily complicates a graphic)?
- Does the screen look “clean” and free from clutter?
- Does the user have control over the design of the workplace, allowing him or her to individualize it?
- Is the information in windows organized so that the most important information can be read first?
- Do you use a consistent light source, one that always comes from the upper-left corner of the screen?
- Do you use white space and graphics to break up long pieces of text?

The big advantage of a checklist in the appendix of a style guide is that conformity with the guidelines can be checked very easily. Answering the questions brings the particulars of the guidelines to mind very easily.

3 Summary

It is not obvious that the use of style guides really will help developers of software. The style guides 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 style guides must learn (or be taught) how to work with them. There are a lot of pitfalls as mentioned in Sect. 1 and one must be aware of them before starting to work with style guides.

Benefits can be achieved but the environment must be adapted to the new method of using such documents. A company gets a better idea of what can be achieved with style guides if the document is used as a shared memory for conventions than to look at it as a list of prescriptions. One focus of future research should be how all the participants of a new team (designer, developer, user etc.) can build up such a condensed design memory more efficiently. The goal is not to get a huge amount of rules very quickly, but rather to transfer the knowledge about HCI issues as efficiently as possible from the experienced experts to all others.

Style guides indeed help although this will not be self-evident but it is something the developer has to spend time with to get comfortable with. Using style guides for designing user interfaces will remain a method that has to be learnt explicitly.

References

1. Dufour, Y., Rensonnet, D.: Requirements Proposal for a Generic Tool for Working with Guidelines. M.Sc. thesis. Facultés Universitaires Notre-Dame de la Paix, Namur (1997)
2. Gale, S.: A Collaborative Approach to Developing Style Guides. In: Proc. of ACM Conf. on Human Factors in Computing Systems CHI 96. Vol. 1 (1996) 362-367. Also accessible at http://www.acm.org/sigchi/chi96/proceedings/papers/Gale/srg_txt.htm
3. Harrison, J.: Designing the User Interface. Department of Computer Science and Electrical Engineering, The University of Queensland, Brisbane (September 1998). Accessible at <http://www.csee.uq.edu.au/~cs116/slides/ui/html/index.htm>

4. Human-Computer Interaction - UI Guidelines. IBM. Accessible at http://www.ibm.com/ibm/HCI/guidelines/design/ui_design.html
5. Java Look and Feel Design Guidelines. Sun Microsystems. Addison-Wesley, New York (1999)
6. Mac OS 8 Human Interface Guidelines. Apple Computer, Inc. Accessible at <http://developer.apple.com/techpubs/mac/pdf/HIGOS8Guidelines.pdf>
7. Macintosh Human Interface Guidelines. Apple Computer, Inc. Accessible at <http://developer.apple.com/techpubs/mac/pdf/HIGuidelines.pdf>
8. Myers, B.A.: Why are Human-Computer Interfaces Difficult to Design and Implement? Technical Report, no. CMU-CS-93-183. Carnegie Mellon University, School of Computer Science (July 1993). Accessible at <http://reports-archive.adm.cs.cmu.edu/anon/1993/CMU-CS-93-183.ps>
9. Schumacher, R.M.: Ameritech Graphical User Interface Standards and Design Guidelines. Accessible at <http://www.ameritech.com/corporate/testtown/library/standard/std-guix.html>
10. Stultz, D.C.: Programmer-responsibility. AskTog, Reader Mail. Accessible at <http://www.asktog.com/readerMail/1998-12ReaderMail.html>
11. The Windows Interface Guidelines for Software Design: An Application Design Guide. Microsoft Corp. Microsoft Press, Redmond (1995)
12. Tidwell, J.: Common ground: A Pattern Language for Human-Computer Interface Design. Accessible at http://www.mit.edu/~jtidwell/common_ground.html
13. Tidwell, J.: The Case for HCI Design Patterns. Accessible at http://www.mit.edu/~jtidwell/ui_patterns_essay.html
14. User-Interface Guidelines. NASA, Goddard Space Flight Center. Data Systems Technology Division/Code 520. Accessible at http://groucho.gsfc.nasa.gov/Code_520/Code_522/Documents/UG_96/newfrontmatter.html (RTF-copy accessible at http://groucho.gsfc.nasa.gov/Code_520/Code_522/Documents/UG_96/UGI.rtf)
15. Vanderdonckt, J.: Development Milestones Towards a Tool for Working with Guidelines. Interacting with Computers. Vol. 12, No. 2 (September 1999) 81–118
16. Vogt, T: The Use of Guidelines for Designing User Interfaces in Commercial Software Development. Department for Informatics, University of Hamburg (1999)