**Published in: Vanderdonckt J., Farenc C. (eds.): Tools for Working with Guidelines, Springer, London, 2000, pp.225-236**

# Tools for Working with Guidelines in Different Interface Design Approaches

Harald Reiterer

University of Konstanz, Department of Computer and Information Science
D-78457 Konstanz, Germany
Harald.Reiterer@uni-konstanz.de

**Abstract.** The use of Tools for Working with Guidelines (TFWWG) during the design and development of user interfaces (UIs) is one possible approach to design usable UI. This paper addresses the following questions: What have been the typical contributions of different UI design approaches to the TFWWG approach and which roles TFWWG have played in these different UI design approaches? To answer these questions, typical contributions of different UI design approaches will be presented. For each approach, benefits and shortcomings and the role of TFWWG will be discussed. A final outlook shows possible directions for further research activities in the field of TFWWG.

## 1 Introduction

The increasing importance of human-computer interaction (HCI) for software has been recognised by industry, academia, and government. A good example is the US-report "Information Technology Research: Investing in Our Future" 27 by the President's Information Technology Advisory Committee. In the report software is one of the four areas where the major part of a strategic initiative in long-term research and development should be focused in the next five years. The report says "special emphasis should be placed on developing software for managing large amounts of information, for making computers easier to use, for making software easier to create and maintain, and for improving the ways *humans interact with computers*". One of the four main goals in the area of software research should be the support of fundamental research in human-computer interfaces and interaction.

The benefits of good user interfaces (UI) are obvious, but there is surprisingly little attention to why UI are difficult to design and implement. It seems that UI are often more difficult to engineer than other parts of the system. UI are critical to the success of software products. However, designing UI is difficult and time-consuming. There are many reasons why a focus on the UI is important, and why UI design and implementation are inherently difficult tasks. Myers 22 gives a quite comprehensive overview about typical problems designing and implementing UI.

Many research activities have been undertaken during the last years to overcome the problems of designing good UI. This paper focuses on one possible approach to design usable UI: the use of guidelines and Tools for Working with Guidelines (TFWWG) during the design and development of UI (Section 2). This paper addresses the following questions: *What have been typical contributions of different UI design*

*approaches to the TFWWG approach and which roles TFWWG have played in these different UI design approaches?* To answer these questions typical contributions of different UI design approaches will be presented. Benefits and shortcomings of the approaches will be shown. For each approach the role of TFWWG will be discussed, with an emphasis on tools developed by the author in different research projects (Section 3). A final outlook shows possible directions for further research activities in this field (Section 4).

## 2 Tools for Working with Guidelines

The fundamental goal of the TFWWG approach is to build UI that optimise human performance, including error reduction, increased throughput, user satisfaction, and user comfort. To reach this goal, human factors play a major role because a main focus of usability is to improve user performance of tasks with an interactive computer system 13. Most of the human factors knowledge is nowadays available as guidelines. Vanderdonckt 37 defines a guideline by a design and/or evaluation principle to be observed in order to get and/or guarantee the usability of a UI for a given interactive task to be carried out by a given user population in a given context.

Typical sources for guidelines are *guideline sets*, *standards*, *style guides*, *design rules* and *ergonomic algorithms*. A detailed definition of each type of source can be found in 37. Today we have plenty of guidelines. One good example of a comprehensive summary is the project SIERRA with a detailed computer-based description of 3700 guidelines 37.

One of the goals of the TFWWG approach is to formalize guideline sets, standards, and style guides with the help of design rules and ergonomic algorithms. This is an important precondition for developing special TFWWG (Section 3.3). Empirical studies (e.g., 35) have shown the importance of standards, guidelines, and style guides for the design of usable UI, but also that most of the software developers have no or only very limited knowledge about guidelines. Therefore most of the software developers are not able to apply guidelines to the design process. Developers were asked what kind of support they prefer to overcome their lack of guideline knowledge. Most of them prefer computer based design aid tools that should be integrated in their common software development tools. What they dislike is “paperware”, like manuals or technical reports with a great amount of written style guides or guidelines.

An important research goal of the TFWWG approach is to discover helpful, unobtrusive, structured, and organised ways to integrate the use of guidelines into the design process without stifling creativity. These research issues include methods and tools for offering the developer assistance in understanding, searching, and applying design principles, guidelines, and standards during the design process 618.

## 3 Approaches to User Interface Design

During the last years a lot of different research approaches to UI development have been established 11. An approach can be seen as a way to solve a problem. It can be characterised by different criteria and figures as a framework for a discipline. Table 1

shows four approaches to UI design. The table is inspired by Wallace and Anderson 37. The criteria on the left side characterise the approaches. The main distinguishing criteria are *philosophy of development process* and *sources of quality*.

**Table 1.** Different approaches to user interface design

| **Approach Criteria** | **Craft Approach** | **Cognitive Psychology Approach** | **Usability Engineering Approach** | **Technologist Approach** |
|---|---|---|---|---|
| Philosophy | Craft: Design through skill and experience | Applied science: Apply theory of human information processing | Engineering: Incorporate HCI into software engineering | Engineering: Automate or support the software engineering process |
| Source of Quality | Talent | Theory | Methods | Tools |
| Character | Monolithic, Evolutionary | Structured transformation | Structured transformation | Black box generation or design aid tools |
| Focus | „Analysis - Implementation - Evaluation“ | Evaluation, Analysis | Analysis | Implementation |
| Role of Practitioner | „Craftsman“, „Artist“ | Psychologist, Ergonomist, HCI specialist | Software developer, Usability engineer | Software tools developer |
| Methods and Tools | Brainstorming, Prototyping | Evaluation methods and tools, Task analysis methods (e.g. GOMS) | Enhancement of traditional software engineering methods | Formal grammars, KI, Generators, UIMS, Style Guides, Class Libraries, Design Agents |

All approaches have one common goal: to improve the quality (especially the usability) of the UI. So the focus is on the *product*. What makes them different is the way in which the UI development *process* is approached**:** Should the development process be craft-oriented, an applied science or an engineering discipline? Another important difference is the view on where the improvements in UI quality will come from. Four sources of quality can be distinguished: talent, theory, methodology or technology. The *talent* view is that great designs come from great developers. This perspective underlies much of the *craft approach* to UI design. The second source implied is *theory*. This approach sets out to use scientific knowledge to support scientific methods for solving the problems of UI development. This perspective underlies much of the *applied science approach* to UI design. The third and fourth sources of quality

are *methodology and technology*. The methods and tools used to develop UI are the main determinants of UI quality. Methods and tools are not an immediate substitute for talent, but by reducing dependence on talent it can greatly increase the chance of successful design. This perspective underlies much of the *engineering discipline approach* to UI design.

This paper will be restricted to the last three approaches. One reason is that it is obvious that crafted failed to mass-produce manufactured goods in the last century and for the same reasons a strictly craft approach to UI design will fail to meet the demands for software. Another reason is that only the last three approaches have important impacts on the research field TFWWG. For each of the three approaches the role and possible contributions of TFWWG will be shown.

### 3.1 Cognitive Psychologist Approach

Cognitive psychologists apply theories of human information processing and problem solving –developed in the previous two decades– to UI development. Numerous methods emerged, and they can be divided into four categories: *cognitive metric methods* (e.g., keystroke-level model), *grammar models* (e.g., formal grammatical notations for the description of mental models), *knowledge methods* to make explicit the mental processes of the user when performing tasks (e.g., Cognitive Complexity Theory, GOMS), and *user modelling methods* to get a qualitative understanding of what might be going on in a user's head (e.g., Programmable User Models).
Much of this work has expanded the cognitive psychology knowledge base, but it has done little to meet the needs of UI developers. The strong influence of the *Cognitive Psychology* approach - especially during the 80`s - is responsible for the following important improvements in the field of UI development:

- a strong user and task orientation in UI development (e.g. in the standard ISO 9241 Part 11 the "context of use" plays an important role),
- definition of usability principles, guidelines and requirements (e.g. ergonomic requirements in ISO 9241 Parts 10-17),
- development of a broad set of evaluation methods and tools (e.g. ISO 9241-Evaluator).

Today we have a broad range of evaluation tools supporting different kinds of evaluation approaches. A very comprehensive overview can be found in 17. A good example that shows the impact of the cognitive psychologist approach on TFWWG is the evaluation tool *IS0 9241-Evaluator* 25. The author has been involved in the development of this evaluation tool for many years. The ISO 9241-Evaluator is a guideline oriented expert-based evaluation method that prepares the requirements of the multi-party standard ISO 9241 to be tested in about 500 test items. The test items are structured in a two-dimensional space defined by technical components and software-ergonomic criteria. The dimension of the technical components is inspired by an IFIP model for UI, extended and adapted to the structure of the multi-party ISO standard. The second dimension consists of the software-ergonomic principles, based on the dialogue principles of ISO 9241, part 10 and extended by requirements for the included I/O interface of a system. Each test item checks a particular aspect of ergonomic re-

quirements specific for the given component and criteria. The tool includes an interactive questionnaire, which allows an automatic calculation of the requirements achievement. The interactive questionnaire can be generated with the help of a database of about 500 test items. The database allows the definition of different evaluation views (e.g. help system, menus). The tool support contains also an editing facility of technical components, software ergonomic criteria and test items. Due to the ongoing development of research and development in UI, as well as in the standards, there is a need for maintenance of the instruments. In particular, the support contains the evaluation process of a given application and the writing of an evaluation report. To facilitate the application of the multi-party standard, explanations are provided to be used in understanding its requirements for the completion of conformance testing. Several possible methods for applicability and adherence are defined and a checklist for the results of the testing is presented. With the ISO 9241-Evaluator the support for the testing, the documentation of the testing, the evaluation, and the report of the results is provided.

The main problem of the Cognitive Psychology approach is that most of the methods (e.g., task analysis, evaluation) and tools (e.g., ISO 9241 Evaluator) require detailed HCI knowledge and therefore are not easy to use for typical software developers. Another important shortcoming is that the dominance of evaluation driven methods leads to a corrective UI development approach: the evaluation in the development process takes place too late and the software developers get no support to avoid ergonomic deficiencies. To solve these principal problems was one of the motivations for the Usability Engineering approach.

### 3.2 Usability Engineering Approach

The idea behind this approach is that specific methods based on HCI research –mainly inspired from the cognitive psychology approach– have to be integrated or combined with typical software engineering methods, if they are to have any impact on UI design in the real world. UI design should not be left alone to the hands of human factor specialists; it should be mainly the job of the software developers (e.g. system analysts, programmers). Therefore it is necessary to use notations that are compatible with the original software engineering methods and familiar to the developers. To improve UI development by enhancing existing software engineering methods is the most pragmatic strand of HCI research. If HCI wants to make significant impact on UI development it will have to be packaged in a way that is attractive to developers.

The Usability Engineering approach - especially during the 90`s - is responsible for the following important improvements in the field of UI development:

- leads to a prospective UI development approach,
- incorporation of measurable usability principles in the software development process,
- a prototyping oriented process model for UI development which has been integrated in the traditional software life cycle,
- enhancement of established software engineering methods (e.g. extension of E/R- or OO-models),

- incorporation of methods from the cognitive psychology approach (e.g. evaluation, task and user analysis).

All these improvements are also of great importance for the TFWWG approach. To be really successful the use of guidelines must be incorporated in all stages of the development life cycle. An early attempt by the author to this approach was the integration of usability principles, HCI-specific methods and tools in all phases of an evolutionary and prototyping oriented software development lifecycle, using common software engineering methods and tools as much as possible 19. Today there are a lot of different Usability Engineering approaches available. Good examples are the approaches called "Graphical User Interface Design and Evaluation" by Redmond-Pyle and Moore 29, "Usability Engineering Lifecycle" by Mayhew 21, and "Contextual Design" by Beyer and Holtzblatt 4. There is also an ISO standard available (ISO 13407), defining "human-centred design processes for interactive systems" 14. The standard offers the project manager a framework how to introduce usability engineering activities, methods and principles in the software development process.

Examples of special attempts of the TFWWG approach to the Usability approach are the detailed guidance for incorporating UI guidelines and standards into the software development process presented in 30 and the concept of "Usability First" 5. "Usability First" can be used with other software development approaches and methodologies. It involves continual evaluations throughout the lifecycle (e.g. evaluating the usability of methods for developers and the usability of applications, designs and developed systems for users). It also involves a number of activities throughout the development lifecycle that help developers to use task analysis methods, to identify guidelines, to develop use models that transform requirements into design and to evaluate designs for usability and conformance to applicable guidelines.
All the new or enhanced methods of the Usability Engineering approach require additional work for the developers and they also establish a need of new knowledge for the developers: e.g., knowledge about new or enhanced methods (e.g. how to use task scenarios or evaluation methods) or knowledge about usability requirements (e.g. guidelines). The main shortcoming of the usability engineering approach is the lack of tools supporting the developers with the necessary knowledge.

### 3.3 Technologist Approach

The Technologist Approach can be seen as a complementary approach to the Usability Engineering approach focusing on the development of tools. This approach is based on the assumption that developers produce poor UI, not because of a lack of understanding of requirements, but because of a lack of time and resources during development. The answer to this problem, from the technologist perspective, is to provide development tools that will free programmers from mundane and time-consuming tasks and leave them more time to spend on design.

The ideas behind the TFWWG approach are very closely related to this approach. Most of the nowadays available TFWWG can be seen as a contribution to this approach. In the following, only typical representatives of this approach could be presented to show the wide variety of possibilities to support developers with the help of tools.

One important category of tools of this approach are **UI software tools**. The emergence of these tools has been greatly aided by the adoption of graphics-based UI style guides (e.g., Windows, Motif, CUA) and guidelines. They are extremely difficult to apply from written specifications. Today, UI software tools are at least a US$ 100 million per year business and are widely used for commercial software development 24. A comprehensive overview of user interface software tools can be found at Myers's web site 23. An important benefit of these tools is that they encourage rapid prototyping as a means of requirements capture for UI development. One important contribution of the TFWWG approach is the integration of features in the UI software tools that *automates* or *supports* the consideration of established guidelines.

The two mainstreams in the Technologist approach are based on two very different assumptions of the developer's role in the design process. The automatic generation tools approach regards the developer as a passive part in the design process. As many as possible of the design decisions are devoted to the development tool. Generators automate most of the design activities. The role of the developer is to specify the necessary meta-information for the generation process and to improve the result of the generation process. The developer is under control of the development tool and most of the necessary design knowledge is devoted to the system. The support approach regards the developer as an active part in the design process making all important design decisions. Powerful development and design aid tools enable the developer to make all necessary decisions. The developer has everything under control and learns the necessary design knowledge on demand.

In the following, some examples of this two different tool approaches will be shown. The first category of tools considering guidelines are **generators** that allow an automatic transformation of the formal specification to a preliminary UI. You can call this the CASE-approach for UI development. There are a lot of research systems available, e.g., GENIUS 16, UIDE 1. The JANUS system is a good example for the automatic generation of the UI out of an object-oriented data model with the explicit consideration of guidelines 2. The JANUS system assumes that the problem domain has been modelled using the OOA method UML with the help of the CASE tool Rational Rose 28. The existing OOA-model will be analysed by JANUS. JANUS uses different knowledge bases to transform the OOA-model into a GUI. A knowledge-based inference engine using rules for selecting and placing GUI controls generates GUIs. The knowledge base of the inference engine also includes knowledge about guidelines (e.g., layout, dialogue, conventions, etc.). The generated UI description is visualised with the help of a UIMS. Now a commercial version of JANUS is available which shows the practical success of this approach 26.

A complementary approach to the automatic generation approach is the use of **design aid tools** support learning and designing. You can call this the learning on demand approach for UI development. There are a lot of research systems available, e.g. FRAMER 8, KRI/AG 20, EXPOSE 9, and SIERRA 36. Another typical example for this approach is a research system called User **I**nterface **D**esign **A**ssistant (IDA) 31,32,33] developed by the author. The primary idea of IDA was the explicit incorporation of guidelines into a UIMS with the help of design aid tools to empower UI developers. This means that development tools and developers are bringing complementary strengths and weaknesses to the job. To shape the tools into a truly usable

and useful medium, the tools should let the developers work directly on their problems and their tasks. The following design aid tools assisting the UI developers during the design, implementation, and evaluation process are part of IDA.

The IDA *advice-giving tool* supports design and implementation activities presenting guidelines for GUIs with the help of an online style guide. The access to the advice-giving tool could be context-sensitive or global. When the developer activates the advice-giving tool in a context-sensitive way, he gets the related guidelines of the selected interaction object (e.g., notebook) in the working area of the UIMS. When the developer activates the advice-giving tool in a *global way*, a graphical content browser of the online style guide appears. The online style guide offers a variety of possibilities to present the guidelines, e.g., text, pictures, screen shots, animations, spoken explanations.

The IDA *construction tool* supports implementation activities considering human factors with the help of a library of reusable ergonomic GUI software (e.g., controls, dialogues, windows). The library is not restricted to generic UI objects (e.g., push button, list box, table). It includes domain-specific window dialogues (e.g., primary window with different secondary windows and dialogue boxes) based on common window architectures. Using the library the developer generates an instance from each predefined software component.

The IDA *quality assurance tool* evaluates the ergonomic quality of GUIs with the help of an expert system. When the developer demands an evaluation, the passive quality assurance (critique) is explicitly invoked. Using the rules of the knowledge base and controlled by an inference mechanism the expert system analyses the conformance of the UI with the human factors knowledge base (analytic critique). The results of the analytic critiquing process are presented in a special window. The window contains a short description of all discovered ergonomic deficiencies, the source of the short description and the identifier of the evaluated object that contains some ergonomic deficiency. The multifaceted architecture of IDA derives its power from the integration of its design aid tools. Therefore from each design aid tool, the other tools could be started in a context sensitive way.

A new interesting contribution to the design aid tools approach is a system called Sherlock 10. There are some similarities with the IDA system. There is also a strong integration with a commercial UI development tool (Visual Basic 5.0 IDE). Like the IDA quality assurance tool the systems offers the UI designer, developer, or Usability Analyst a kind of checker (like the spell checker of a text editor) integrated in the UI development tool. During the development process the user of the system can press the “Evaluate” Button (integrated in the VB 5.0 IDE) and then the evaluation results are shown in an inspection results window. For this purpose a parser generates a textual description of the UI. This description will be analysed with help of a rule-based system. The rules represent a formal description of different guidelines (mainly from ISO 9241). A wise decision was the distinction of the rules in different inspection types: automatic, semi-automatic, by the user. This offers a great flexibility of use of the system, because task- and user-specific guidelines can be integrated as well as guidelines about the interactive behaviour of an application. A rules help module (similar to the IDA advice-giving tool) is a customised web browser that presents rule-related information to the user. The help module is especially important for semi-

automatic rules and rules that could only be accomplished by the user. It offers the user some advice how to solve detected or potential design deficiencies.

Another very interesting new contribution to the design aid tools approach is a system called GUIDE (Guidelines for Usability through Interface Development Experiences) 12. The GUIDE system uses case-based decision support techniques in an organizational memory and learning structure to offer UI developers a context-specific level of guidelines that can directly be applied to interface design problems. Using a Web-based UI the UI developer can specify the actual design context with the help of a question-answer dialog. Rules are used to match the current design context with existing guidelines and cases (a case is assigned to a guideline for each project that is required to conform to the guideline). Answering the questions the user steps down a decision tree until guidelines are assigned to the project (building new cases that could be used in further projects). The guideline hierarchy follows the Smith and Mosier standard 34 and breaks guidelines into increasingly more detailed and context-specific information that has been included following a specific GUIDE development process. All guidelines, rules, and cases are stored in a repository. This repository allows developers to draw on the collective experience of the organization. Therefore a reuse of UI design rationale will be possible and the problem of interfaces guidelines often being too abstract to be directly applicable could be solved with the GUIDE approach.

As the above description shows, a number of attempts to automate or support the incorporation of guidelines into design have been reported in the research literature over the years, but as a matter of fact very few generally applicable, validated, and truly usable commercial tools have yet emerged. This is a big challenge for the TFWWG approach in the future!

## 4 Summary and Outlook

There have been a lot of useful contributions of the different UI design approaches to the TFWWG approach in the last fifteen years and vice versa.

The ***Cognitive Psychology*** approach is responsible for a comprehensive amount of usability principles, guidelines, and requirements. There is now a sound basis of guidelines available for different kinds of “traditional” UI (e.g., GUIs for desktop computers). An important contribution of the TFWWG approach was the development of special tools supporting UI evaluation methods (e.g., ISO 9241-Evaluator). There is now a sound basis of usable evaluation tools available and I think further research effort in TFWWG field should be spent in the following domains.

An important future challenge for the TFWWG approach is that UI design is poised for a radical change in the near future, primarily brought on by the rise of ubiquitous computing, recognition-based UI, 3D and other new technologies. Therefore, there is a real need for research on UI software tools like TFWWG to support the *new UI styles* and *considering new guidelines* for new devices (e.g. information appliances with small displays, high-quality speech input, gesture and handwriting recognition 3) because today TFWWG are primarily focused on traditional GUI guidelines for desktop computers.

The ***Usability Engineering*** approach established a methodology framework to integrate TFWWG in different stages of the software development life cycle. Based on this methodology framework different TFWWG approaches have been developed when and how UI guidelines have to be integrated in the software development lifecycle (e.g., "Usability First"). An important new contribution of the TFWWG approach to this field could be a kind of *usability engineering environment* supporting usability engineers during the whole usability lifecycle. A first attempt to this direction (but still a research prototype) is a tool called MMI-HyperBase 7. It summarizes and presents ergonomic design knowledge and usability methods for the whole usability engineering lifecycle with the help of a knowledge base and make the knowledge available via Intranet and Web-Browser. The ergonomic design knowledge and the usability methods have been attached to the different phases of a typical usability life cycle (project management, requirement analysis, UI design, evaluation and test, use of the system). The MMI-HyperBase supports the software development department with a special focus on UI developers to select the right methods and tools and to define the roles of the project team members responsible for the UI. The tool also supports the cooperation between the different members of the UI development project with the help of a groupware tool (Lotus Domino). The aim is to reuse UI design rationale and to support the management of the UI development process.

Most of the research results of the TFWWG approach can be seen as a specific contribution to the ***Technologist*** approach with a main focus on incorporating guidelines knowledge in tools. As this paper shows, there are now a lot of very promising research systems, but we have only very few commercially available. This circumstance should encourage researchers in this field to come to co-operations with commercial tool developers to see their ideas influencing the daily work of UI developers!

What we also need in the future are TFWWG based on a *combination* of the two different Technologist approaches: *automation and support*. The real benefit of the automatic generation approach lies in the necessity of a formal specification of the software architecture and the UI requirements. For example, without a clear defined UML class diagram and the help of the CASE tool (like Rational Rose) a system like JANUS won't work. If such a conceptual model exists, an automatic generation of the UI helps the developer to focus his work on the application specific details of the UI. It is obvious that the automatic generated UI has to be refined. This is the place where the support approach could come in. Here design aid tools could help the UI developers to improve the automatic generated prototype and to focus their work on the parts of the UI that could not be generated automatically considering design rules (e.g. task and user related aspects of the UI). UI tools based on a combination of automation and support features would allow a tight integration of UI design in the whole software development process, would offer good support for a rapid prototyping approach of the UI considering design rules (formalised guidelines), and finally would assure that knowledge about guidelines could be used on demand by the developers during the UI development process.

## References

1. de Baar, D., Foley, J., Mullet, K.: Coupling Application Design and User Interface Design.

In: Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'92 (Monterey, May 3-7, 1992). Addison-Wesley, Reading (1992) 259–266
2. Balzert, H.: From OOA to GUIs – the JANUS System. In: Proc. of IFIP Conf. on Human-Computer Interaction Interact'95 (Lillehammer, June 27-29, 1995). Chapman & Hall, London (1995) 319–324
3. Bergmann, E.: Information Appliances and Beyond. Morgan Kaufmann, San Francisco (2000)
4. Beyer, H., Holzblatt K.: Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann, San Francisco (1998)
5. Carter, J.: Incorporating standards and guidelines in an approach that balances usability concerns for developers and end users. Interacting with Computers 12, 2 (1999) 179–206
6. Cohen, A., Crow, D., Dilli, I., Gorny, P., Hoffman, H.-J., Iannella, R., Ogawa, K., Reiterer, H., Ueno K., Vanderdonckt, J.: Tools for Working with Guidelines. SIGCHI Bulletin 27, 2 (1995) 30–51
7. DaimlerChrysler, TU Illmenau, DASA, Siemens ElectroCom, Carl Zeiss: Objektorientierte Softwarewiederverwendung in verteilten Architekturen. Abschlußbericht, BMBF, Förderkennzeichen 01IS605A4 (1999). See Part III: Chapter 8.2 Experienced based Usability Engineering, 253–293
8. Fischer, G., Lemke, A., McCall, R.: Making Argumentation Serve Design. Human-Computer Interaction 6 (1991) 393–419
9. Gorny, P.: EXPOSE. HCI-counselling for user interface design. In: Proc. of IFIP Conf. on Human-Computer Interaction Interact'95 (Lillehammer, June 27-29, 1995). Chapman & Hall, London (1995).297–304
10. Grammenos, D., Akoumianakis, D., Stephanidis, C.: Integrated support for working with guidelines: the Sherlock guidelines management system. Interacting with Computers 12, 3 (2000) 281–311
11. Hartson, H., Boehm-Davis, D.: User interface development processes and methodologies. Behaviour & Information Technology 12, 2 (1993) 98–114
12. Henninger, S.: A methodology and tools for applying context-specific usability guidelines to interface design. Interacting with Computers 12, 3 (2000) 225–243
13. Hix, D., Hartson, R.: Developing User Interfaces, Ensuring Usability Through Product & Process.John Wiley & Sons, New York (1993)
14. ISO 13407: Human-centred design processes for interactive systems
15. ISO 9241: Ergonomic Requirements for Office Work with Visual Display Terminals
16. Janssen, C., Weisbecker, A., Ziegler, J.: Generation User Interfaces form Data Models and Dialogue Net Specifications. In Proc. of ACM Conf. on Human Aspects in Computing Systems INTERCHI'93. Addison-Wesley, Reading (1993) 418–423
17. Behaviour Information Technology, 16, 4/5 (1997). Special Issue: Usability Evaluation
18. Interacting with Computers, 12 (November 1999 and January 2000). Special Issue: Tools for Working with Guidelines
19. Koch, M., Reiterer, H., Tjoa, A.: Software-Ergonomie, Gestaltung von EDV-Systemen - Kriterien, Methoden und Werkzeuge. Springer-Verlag, Vienna (1991)
20. Löwgren, J., Nordquist, T.: Knowledge-Based Evaluation as Design Support for Graphical User Interfaces. In: Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'92 (Monterey, May 3-7, 1992). Addison-Wesley, Reading (1992) 181–188
21. Mayhew, D.: The Usability Engineering Lifecycle. Prentice Hall, Englewood Cliffs (1999)
22. Myers, B.: Challenges of HCI Design and Implementation. Interactions (Jan. 1994) 73–83
23. Myers, B.: http://www.cs.cmu.edu/afs/cs/user/bam/www/toolnames.html
24. Myers, B., Hudson, S., and Pausch R.: Past, Present and Future of User Interface Software Tools. ACM Transactions on Computer-Human Interaction (2000, to appear). Available at http://www.cs.cmu.edu/~amulet/papers/futureofhci.pdf

25. Oppermann R., Reiterer, H.: Software evaluation using the 9241 evaluator. Behaviour & Information Technology 16, 4/5 (1997).232–245
26. OTRIs: http://www.otris.de
27. PITAC: President's Information Technology Advisory Committee Report to the President, Information Technology Research: Investing in our Future. National Coordination Office for Computing, Information, and Communications. Arlington (February 1999). Available at http://www.ccic.gov
28. Rational; http://www.rational.com/products/rose/index.jtmpl
29. Redmond-Pyle, D., Moore, A.: Graphical User Interface Design and Evaluation. Prentice Hall, London (1995)
30. Reed, P., Holdaway, K., Isensee, S., Buie, E., Fox, J., Williams, J., Lund, A.: User Interface Guidelines and Standards: Progress, Issues, and Prospects. Interacting with Computers 12, 2 (1999) 119–142
31. Reiterer, H.: User Interface Evaluation and Design. Oldenbourg, München (1994)
32. Reiterer, H.: IDA – a design environment for ergonomic user interfaces. In: Proc. of IFIP Conf. on Human-Computer Interaction Interact'95 (Lillehammer, June 27-29, 1995). Chapman & Hall, London (1995) 305–310
33. Reiterer, H.: Die IDA-Entwicklungsumgebung: Einsatz von objekt-orientierten, multimedialen und wissensbasierten Unterstützungswerkzeugen zur ergonomischen Gestaltung von Benutzungsoberflächen. Informatik Forschung und Entwicklung, Springer, München, Heft 4/1995, S.180-196
34. Smith, S.L., Mosier, J.N.: Guidelines for Designing User Interface Software. Technical Report ESD-TR-86-278. The MITRE Corporation (1986)
35. Tetzlaff, L., Schwartz, D.: The Use of Guidelines in Interface Design. In: Proc. of ACM Conf. on Human Aspects in Computing Systems CHI'91 (New Orleans, April 28-May 2, 1991). Addison-Wesley, Reading (1991) 329–333
36. Vanderdonckt, J.: Accessing Guidelines Information with Sierra. In: Proc. of IFIP Conf. on Human-Computer Interaction Interact'95 (Lillehammer, June 27-29, 1995). Chapman & Hall, London (1995) 311–316. Available at http://www.info.fundp.ac.be/cgi-publi/pub-spec-paper?RP-95-020
37. Vanderdonckt J.: Development Milestones toward a Tool For Working With Guidelines. Interacting with Computers 12, 2 (1999).81–118
38. Wallace, M., Anderson, T.: Approaches to Interface Design. Interacting with Computers 5, 3 (1993) 259–278