Software Engineering in User Interface Design with Guidelines – from traditional applications to the Web sphere –

Hans-Jürgen Hoffmann

Darmstadt Univ. of Technology, Darmstadt, Germany
Chair on Programming Languages and Compilers
Alexanderstr. 10, D-64283 Darmstadt
HJHoffmann@ACM.org

Abstract. Software Engineering (SE) knows a set of techniques for improving and guaranteeing quality of programs. In User Interface Design (UID) rules for achieving good usability results are available in what is called Design Guidelines or Style Guides. In both fields supporting methods and tools are in use. Combined methodology and tool support is a challenge for researchers, implementors, and designers in both fields. Findings extend to the Web sphere.

The article gives a survey from the point of view of a computer scientist (in Germany) and reports on successful and unsuccessful approaches to achieve the goal of combination.

1 Introduction

Construction of interactive application systems and design of user interfaces thereof are *interdisciplinary work*. One person with qualified knowledge in both fields or a team of co-operating computer scientists/implementors familiar with SE techniques and interface designers trained in UID are (to be) engaged in realization of *system requirements* initially agreed upon between commissioning client organizations and software implementation agencies. The paper argues that to aim for a combined methodology and tool support is an actual challenge for SE/UID researchers, software implementors, and user interface designers.

2 Software Engineering

In software engineering (SE) for what is called P*rogramming-in-the-large*[1] (DeRemer and Kron 1975) waterfall and spiral process models (e.g., Royce 1970/1987, Boehm 1986/1988, Boehm et al. 1997,. 1998)[2] guide the program/software system construction process. Several document forms are in use. State-of-the-art is to apply the *Universal Modeling Language*, UML (Rumbaugh et al. 1998), for *specifications* demanding the implementor to search for a thorough understanding of objects relevant for the application and for functional relations between them before starting to *code* in the programming language chosen. In object-oriented programming style *patterns* (e.g., Gamma et al. 1995) and/or *frameworks* (e.g., Fayad et al. 1999a, 1999b) are concepts to achieve re-use of previously implemented (and tested/validated) program parts. *Test* and *maintenance plans* are additional documents to be prepared in advance and derived from the system requirements, based on formal verification, walk-throughs etc. An overview of techniques/guidelines recommended for program/software system construction (with contributions of the present author; Hoffmann 1993) can be found in (VDI-GIS 1993), a follow-on book is (DGQ/ITG/VDI 1998); similar compendia of program/software system construction guidelines exist, it´s not the place to give a detailed listing.

A very critical point of view about how and to what extent software engineers really follow the research findings sketched above in program/software system construction has been taken by Parnas (1998). Considering SE in relation to UID there are only a small number of serious investigations publicized (e.g., Draper and Norman 1985, Taylor and Coutaz 1995, MSWE 1998).

There are more or less integrated tools available and in (limited) use, covering the construction phases listed above, known under the umbrella term CASE-tools (*Computer-Assisted-Software-Engineering)*, viz., focused to a specific programming language (*Programming Environment)*, and/or focused to the engineering

[1] *Programming in-the-small* is not of interest in the SE context of the paper at hand. However, see [14].
[2] There are more such models proposed and in use, see (e.g.) Balzert (1998).

approach chosen (*Systems Environment)*, resp. In any case, the goal is to achieve and to guarantee a high quality of the software product finally delivered to the client organization. Programming guidelines covering the know-how of high-quality program/software system construction (should) have been followed when implementing the tools mentioned above. In so far, no specific consideration has to be given to interactivity in an application although the formalized techniques usually allow and demand interaction specification (from a point of view considering the technical interface between program/system kernel and user interface software component[3]).

3 User Interface Design

This field, UID, adresses an *interactive software product* from a distinct point of view: Not the functionality (and quality achieved) of the application kernel is of importance; paramount emphasis is given to how a user of the product can reliably work with the software product to gain desired application results[4]; *usability aspects* are the criteria guiding the design process[5]. Requirement analysis identifies *user tasks* to be achieved and relations between them, e.g., relations for sequencing, what is called *task analysis*. Starting point is to know the *user population* and their specific requirements[6].

To-day we find interactive software products in closed environments, to name three:

(i) for business applications, e.g, document processing in text editors, simulation/planning in spreadsheets etc.;
(ii) for information retrieval in, e.g., management information systems, library access and the like; and
(iii) for engineering applications, e.g., system control, laboratory automation, CAD-applications, etc.

And there are open, distributed environments exemplified in the World Wide Web supporting not only simple user applications but also co-operative situations with open user populations.

May we expect to find, to experimentally validate, and to be able to formulate guidelines in a consolidated way accepted by both, implementors of programs/software systems and designers of user interfaces, covering this wide scope of scenarios, including the Web sphere? May we build useful, robust and reliable construction and design tools to improve the situation?

For the following discussion[7] I want to point out an observation: You may find three categories of User Interface Design Guidelines characterized below in sections 3.1 to 3.3.

Similar quality goals as those described above related to Software Engineering stand behind all these forms of User Interface Design Guidelines, many of them covered by terms *„ease of…“* (you name it). Are they easier and better applicable in tool design? What was/still is the impact of a tutorial of CHI´95 (Perlman 1995b) directed towards software engineers?

It should be mentioned that, in a growing scale, there are authoritative agencies (like TÜV in Germany, e.g. *„Software-Ergonomie geprüft“*, a label of TÜV Rheinland, see Rudlof and Dzida 1997) providing public services for checking interactive application systems applying guidelines (in most cases based upon national or international standards like ISO, DIN, etc.).

It is not the place here to discuss techniques widely introduced and in (successful) use in UID to improve and guarantee usability. The topic of the paper is to discuss possible combinations in a SE methodology and tool. Constructors and designers should both be supported in their work by a combined tool integrating quality requirements of both fields (section 4).

---

[3] In English the term *user interface* is commonly used for the system component and the system appearance seen and manipulated by the user, e.g., by a WIMP (*window, icon, menue, pointing*) realization. In German a distinction is made (e.g., by contributors to Schneider 1997) between *Benutzungsschnittstelle*, the system component, and *Benutzungsoberfläche*, the system appearance.

[4] Not considered here is the recently advocated argument an interactive system (Web included) should be fun.

[5] The reader should observe my (restricted) use of the word *design* (of the user interface realising a certain system appearance) in contrast to the use of the word *construction* (of the software realising the application kernel and the supporting user interface system component).

[6] The earliest user interface design guidelines I am aware of (Hansen 1971) already puts the rule of thumb *Know-the-user* at the first position; others (e.g., Shneiderman 1997) follow this sequence of emphasis.

[7] You may find a compendium (status of 1995) of references to guidelines covered in the discussion in (Perlman 1995a).

3.1 General User Interface design Guidelines

They express generally accepted rules of thumb which can not be objected. One example already given is *Know-the-user*, Hansen´s paper (1971) lists in four categories of *User Engineering Principles* 13 more; other *„buzzwords"*[8] of user interface design are *„user-centred design", „participatory design"*, *modeless user interfaces* etc. Standardization[9], e.g., the ISO standards (1992) or, in Germany, the DIN-Norm (1988), contribute in this category of guidelines.

I will attribute a *high-level granularity* to this form of guidelines, a too high level of granularity. Is there a chance to detail and to consolidate them so far as program/software system constructors and user interface designers whole-heartedly follow them and tools enforce them?

3.2 Detailed User Interface Design Guidelines

This is a (too) *fine grained approach* of *„Design Guidelines"* without appropriate metrics for computerized knowledge representation, covering all aspects of user interface techniques in present and forthcoming technical spheres. Well-known examples are compendia as those of Smith and Mosier (1986), 944 guidelines listed in about 400 pages, of Vanderdonckt (1994), over 1200 and more actual guidelines, both with a strong research basis. It has to be mentioned that many of these guidelines are heavily backed by research findings in psychology and cognitive sciences. This criteria differentiates them from *„Company Style Guides"* published (e.g.) by IBM, SUN, MicroSoft, in a less formalized, prosa style and oriented towards specific company platforms.

An interesting tool in the scope of the Smith and Mosier guidelines (Mosier and Smith 1985) was *NaviText SAM* (Perlman 1987), a hypertext-based guideline collection system to select applicable guidelines for a design task[10]. In the late 80ties, in my opinion, it was too advanced for combination with a construction tool, when a multi-window system platform was not available, one window for SAM and another window for software coding. Vanderdonckt (1995) provided the SIERRA tool with a similar goal based on his compendium; and in his paper (1999) he gives a systematization of milestones for achieving such goals.

3.3 Topical User Interface Design Guidelines

In between these two extreme cases you may find what I want to call *Topical User Interface Design Guidelines* exemplified especially in the Web sphere by *„gurus"*: J. Nielsen´s *Alertbox* (1995a) covering at the time when writing this article (May 2000) about 105 topics[11] and his books on *Multimedia and Hypertext* (1995b), on *International User Interfaces* (Nielsen and Galdo1996) as well as on *Designing Web Usability* (Nielsen 2000); D. Norman (1986), B. Laurel (1990), B. Shneiderman (1997) to mention some more.

Topical User Interface Design Guidelines are also discussed on varying levels of competence and originality in *„Web journals/reviews/..."* easily accessible in the Web by surfing/searching the keyword phrase *„Web usability"*. One finds here a *medium-level granularity*, picking interesting aspects of the just actual technical sphere, not intented to achieve total covering of all circumstances. Here, statistics and *„fashion of application"/"fashion of technical realizability"* play an important role. It is only a small step to even less serious *„Topical Style Guides"* as discussed, e.g., in the CHI-WEB (199x), a very active international E-mail discussion chat forum,

4 Combination and some previous „ general and tiny attacks" to achieve it

Realization of a combined, integrated methodology *and* tool to support program/software system construction and user interface design assessment is, as already said, an interdisciplinary work, a *hard* interdisciplinary work. It is the purpose of this paper to push research in this direction. Not all approaches I am aware of have been successful. So, allow me to use the term „attack" in this context. I will consider the topic mainly from the (personal) point of view of a software engineer involved in UID.

---

[8] By intention I re-use this word remembering its use in Software Engineering (Parnas 1974).

[9] Reed et al. (1999) discusses the present state in standardization.

[10] For similar tools also see (Perlman 1995a).

[11] As an example take (Nielsen 1995c).

4.1 Combination

UID with guidelines is not simply achieved by having an interdisciplinary team working together in one project. Such a group may (and has very often in the past) agree(d) ad hoc to follow a set of rules for program/software system construction and user interface design („style guides“ as mentioned above). They did it in many cases with success, also re-used the agreement, may be locally modified in details, in another project (thanks a lot!).

What I mean by *„combination“* is

(i) a successful attack and finding of a method to achieve an integrated set of guidelines for both fields, integrated in the sense of mutual validity for guaranteeing the quality goals set in both fields and with solved technical interdependence for both processes, program/software system construction and user interface design; and

(ii) successful finding of a technique to realize a tool which supports both, the constructors and the designers, in adhering to combined guidelines in both processes.

4.2 Achievements in „general attacks“

I am sorry to say that I personaly don´t see successful general attacks in a broad scope. One reason I see is twofold:

(i) The present inadequateness and unavailability of guidelines in both fields prepared for combination. To give some in my opinion outstanding technical deficiencies: missing metrics, missing adequate granularity level, missing hyperlink functionality.

(ii) The uselessness in tool realization with combined guidelines (if they were available) due to mutual misunderstandings between the fields involved, SE *and* UID.

A second reason is, to my opinion, even more serious: How to achieve combination not only with the manifold populations of users addressed (second argument above), but in the unlimited applications spheres requiring widely differing user tasks to be realized, and with diverging demands for construction/design efforts, for technical platform resources supporting traditional applications as well as the Web sphere, for security measures, also for paying attention to cultural differences.

Hence, stop here! Stop here?

4.3 „Tiny attacks“ in SE

No, there are some tiny attacks which may be considered to be successful. I beg your pardon for seeing them (only) in the scope of applications coded in Smalltalk (Goldberg 1995) or realized in the object-oriented style of Smalltalk coding (Skublics et al. 1996).

At the very beginning the so-called MVC paradigm of Smalltalk has to be mentioned. It contributes on a very low level of granularity[12] a mechanism for cooperation of an application kernel, the *model*, with a *view* and a *controller* component. From a SE point of view it offers a secure mechanism to handle presentation (i.e., the *view*) and interaction (the *control*) as well as their interrelations, interrelations also with the application kernel. The Smalltalk platforms embed model, view, and control classes for re-use in object instantiations providing the necessary registration and broadcasting messages in a quality-proven way. The paradigm is widely and successfully applied in Smalltalk applications (and, by scholarly work of constructors, in Java applications, too) avoiding problems in interrelation and synchronisation of user interactions.

To-day, MVC is considered to be an *interface design pattern* (e.g., Buschmann et al. 1996). In the patterns community search for such patterns is going on (one contribution resulting from research under my supervision is Wu 1999). There are already a lot of application frameworks in construction/use[13] in many fields, e.g., evolutionary programming, learning support, simulation and planning, electronic commerce in the Web (Hoffmann 1999), banking, information systems, plant control, where such patterns are integrated guaranteeing a required level of usability in re-use if once checked and validated. Accepted that in most cases the UID aspect is based on platform design guides, not on serious interface design guidelines.

---

[12] It falls outside my previous categorisation in section 3; one may list it in the first, general category realizing the rule of thumb to have appropriate linking between application and interaction, or one may list it in the third category belonging to the very low granularity level (however without explicit ergonomic foundation).

[13] Even coded in Java if the experience in and techniques of Smalltalk implementations are followed, for an overview see the STJA Proceedings (1995 ÷ 1999).

4.4 ISO 9000

In SE ISO 9000 certification (ISO 1993) of a company involved in (interactive) software construction is a very hot topic. Emphasis, however, is not given to UID (why not?). *Software documentation* in general and means to guarantee software quality from the point of view of the specific application are well to the fore. In the contract between a commissioner and the company doing the construction and design work, of course, adherance to a set of UID guidelines (in most cases of the company style introduced above) would be wise; one may not expect tool support to achieve the combination in an integrated manner.

4.5 Web sphere

Construction and design of Web sites[14] is construction and design of interactive systems. There may be a shift in importance of some aspects, e.g., not-knowing the (international) user population beforehand at all, the *re-born* dependency on performance[15], necessity of platform independence, searching instead of direct accessing, the manifold navigation capabilities needed. Accepted, in many situations real interactivity in a page of a site is pronounced by the interactivity of the browser window – hopefully professionally well constructed and designed, who will question it! –, not of the page.

Are available tools for construction/design of (Web pages and) Web sites going beyond ad hoc adherence to topical UID guidelines, the serious ones? They help and support constructors and designers to master HTML coding, following the techniques of Graphic Interface Builders (*GIB*s), also in use for interactive non-Web applications (very often only with programming-in-the-small capability). As a booming field, educated constructors and designers in the Web sphere covering both SE and UID are rare. Research publications (e.g., Ratner et al. 1996) are thinly scattered. CHI-WEB is often beat by commercial orientation. And last but not least, have all constructors/designers studied the relevant topical UID guidelines?

History of Web „presentation“ from the beginning to now, say 8 years later, in my opinion, doesn´t show much quality improvement. Browser versions added features and capabilities, true, but did they add provisions to enforce better quality? The evolution of the Web sphere is still technology-driven. What will Web access by mobile phones contribute?

Combined methodology and tools are missing, not seen to be available soon. Web construction and design has, in my opinion, just reached the first development milestone in Vanderdonckt´s (1999) quality race.

5 Some combined approaches/tools

In the work of the Special Interest Group on Tools for Working with Guidelines, since its first meeting in 1994, some approaches to combine SE construction and UID design with quality assessment in *one* tool have been discussed. I will not bring up the discussions again (except about one tool in which I was personally involved, DIADES). A second work I was involved in, EPK-fix, not known to the UID community will be presented in short below. However, four more developments outside my personal involvement must be mentioned in addition where I see efforts with similar goals: EXPOSE (Gorny 1995), another German development and counseling tool with some similarities (and differences) compared to DIADES; the JANUS system of Fischer et al (1991); CritiGUI (Nitsche-Ruhland and Zimmermann 1995); and Sherlock (Grammenos et al 2000). There are certainly more where I am not aware of.

5.1 DIADES

The DIADES project (Hoffmann 1984, 1987a, Hoffmann et al. 1987b, Hoffmann 1993, Dilli and Hoffmann1995, Vogt 1995) aimed to research a powerful construction tool for interactive applications with combined, integrated UID assessment and identification of design decisions considerd to be *weak* from the point of view of sound, qualityassuring (detailed) UID guidelines, accepted by the human factors community as such. Visually-supported *do-it-by-example* was planned to be the selection mechanism for decisions, related towards construction and design (in the sense as I use these two terms), about *entities* finally combined to become the intented user interface program. Features of advanced user interface techniques (e.g., a menu structure) were planned to be modelled by a *knowledge-based description* technique following the semantic net approach in KL-

[14] Following the distinctive terminology of SE a Web page is considered to be *construction/design-in-the-small* as part of *construction/design-in-the-large* of Web sites.

[15] To work with a 500 MHz computer doesn´t guarantee timely work if the network delay is in the range of seconds or even minutes!

ONE (Brachman 1977). An entity description (Hoffmann 1988) combined four parts used in entity instantiation, viz.,

- the *roles* of the entitiy in relation to other entities and its parts,
- the VISUALIZE_AS-component supporting visual construction by interacting with an example,
- the ASSESS_AS-component for contributing the score of the selected entity from the point of view of UID guidelines (of the detailed level as introduced above) as an assessment metrics rule (all together to be finally combined in a PROLOG evaluator), and
- the GENERATE_AS-component providing code for the interface construction process.

One reason for the nonsuccess of the DIADES project was (in the 1983 – 1995 time frame) the incompatibility of state-of-the-art SE techniques in modelling all the features of advanced user interface techniques on one side and the nonavailability of a covering, sound set of UID guidelines in adequate granularity with a metric adequate for identification of weak decisions about entities and their interrelations.

5.2 EPK-fix

Electronic product catalogues, EPKs, are an important service device of electronic commerce applications on the WWW. EPK-fix, a project (Lutze et al. 1996, Schneeberger et al. 1997) to support rapid (*fix*) production and maintenance of public catalogues, realized an interactive approach with four components,

- RASSI for interviewing a businessman commissioning the catalogue to gather his cataloguing requirements, data and files,
- SASSI, a specification editor[16] to transform the informal interviewing results into an objectoriented specification document (Hoffmann and Closhen 1997),
- GASSI, the (humansupported mechanical) construction component for producing the catalogue as a HTML site, and
- TASSI for testing/validating its usability (Fritzsche et al. 1997, Fritzsche and Michel 2000)

Features to be mentioned in the context of the paper at hand are the *feedback loop* between TASSI and RASSI/SASSI to discuss bottlenecks detected by the TASSI inspection[17] with the commissioner and to consistently adapt the specification documents to changes and, secondly, the *strict, versioning identification* of all construction/design decisions by *marking tags* flowing around in the loop together with the textual and formal objects describing the catalogue and its construction state.

6 Conclusions

SE and UID based upon combined guidelines and used to guide realization in a flexible, general construction and design tool seems to be a challenge, unrealistic in general (DIADES), realized/realizable with success only in limited scope and limited use (if any; EPK-fix, EXPOSE, JANUS, CritiGUI, Sherlock). Much more work has to be done in this interdisciplinary field of research and realization. Carter (1999) comes to a similar result.

The booming Web sphere, in my opinion, is in the long range a danger to attain a high usability quality (e.g., consistency, conformity, robustness, learnability and teachability) everywhere, for all user groups addressed/reached in the Web. There are too much untrained constructors/designers involved looking only for attractiveness in a questionable sense, in the best case following the very often weak/confusing/contradicting topical UID guidelines.

Literature references

Balzert H (1998) Organisation – Prozeß-Modelle. In H. Balzert (1998) Lehrbuch der Software-Technik. Spektrum Akademischer Verlag, II LE 4, pp 97 - 138

Boehm BW (1988) A spiral model of software development and enhancement. IEEE Computer, vol 21, no 5, pp 61 – 72 (re-print of a 1986 paper)

Boehm B et al. (1997) Developing multimedia applications with the WinWin Spiral Model. ACM Software Engineering Notes,22/6: 21 - 39

Boehm B et al. (1998) Using the WinWin Spiral Model: A case study. IEEE Computer 31/7: 33 - 44

[16] The project contribution realized under my supervision, coded in Smalltalk.

[17] Not specifically controlled by UID guidelines – sorry –, but with some metrics on important technical details relevant for usability, e.g., accessibility of files, density of text, look-and-feel of buttons and input fields.

Brachman RJ (1977) What´s in a concept, structural foundations for semantic networks. Intl. J. Man-Machine Studies 9: 127 - 152
Buschmann F et al. (1996) Pattern-oriented software architecture, a system of patterns. John Wiley & Sons, pp 125 – 143
Carter J (1999) Incorporating standards and guidelines in an approach that balances usabiltiy concerns for developers and end users. Interacting with Computers 12/2: 179 - 206
CHI-WEB (199x onwards, x unknown) Electronic discussion list. CHI-WEB@ACM.ORG
DeRemer F, Kron H (1975) Programming-in-the-large versus Programming-in-the-small. ACM SIGPLAN Notices 10/6: 114 – 121
DGQ/ITG/VDI (1998) Zuverlässigkeit komplexer Systeme aus Hardware und Software. Beuth-Verlag, Berlin
Dilli I, Hoffmann HJ (1995) Diades-II, a multi-agent user interface design approach with an integrated assessment component. ACM SIGCHI Bulletin 27/2: 33 - 34
DIN (1988) DIN 66234: Grundsätze ergonomischer Dialoggestaltung. Beuth-Verlag, Berlin
Draper S, Norman D (1985) Software engineering for user interfaces. IEEE Trans Software Engineering, SE-11/3: 252 – 258
Fayad M et al. (1999a) Building application frameworks: Object-oriented foundations of framework design. John Wiley & Sons
Fayad M et al. (1999b) Implementing application frameworks: Object-oriented frameworks at work. John Wiley & Sons
Fischer G et al. (1991) CRITICS, an emerging approach to knowledge-based human-computer interaction. Intl. J. Man-Machine Studies 35/5: 695 – 721
Fritzsche H et al. (1997) Test-Assistenz für formal spezifizierte elektronische Produktkataloge. In: Schneeberger J (ed) Software-Engineering für Multimedia Systeme. Workshop GI´97, pp 19 - 32
Fritzsche H, Michel T. (2000) Formalization and proof of design guidelines within the scope of testing formally specified electronic product catalogues. Interacting with Computers 12/3: 209 - 223
Gamma E et al. (1995) Design patterns, elements of reusable object-oriented software. Addison Wesley
Goldberg A (1995) Why Smalltalk?. Comm. ACM 38/10: 105 - 107
Gorny P (1995) EXPOSE, An HCI-counseling tool for user interface designers. ACM SIGCHI Bulletin 27/2: 35 – 37
Grammenos D et al (2000) Integrated support for working with guidelines: the Sherlock guideline management system. Interacting with Computers 12/3: 281 - 311
Hansen W (1971) User engineering principles for interactive systems. Proc. Fall Joint Computer Conference, pp 523 - 532
Hoffmann HJ (1984) DIADES, ein Entwurfssystem für die Mensch-Maschine-Schnittstelle interaktionsfähiger Systeme. Notizen zu Interaktiven Systemen, no 12, pp 59 – 69
Hoffmann HJ (1987a) DIADES: A design tool for interactive programs with provisions to assess design decisions about the man-machine interface. In: Zunde P, Agrawal JC (eds.). Empirical Foundations of Information and Software Science IV – Empirical Methods of Evaluation of Man-Machine Interfaces, Plenum Press New York London, pp 163 - 175
Hoffmann HJ et al. (1987b) Entwurf und Güteeinschätzung von Menü-Netzen: Ablauf einer Entwurfssitzung mit DIADES. Proc. GI-17. Jahrestagung Computerintegrierter Arbeitsplatz im Büro, IFB 156, Springer-Verlag Berlin Heidelberg, pp 337 - 352
Hoffmann HJ (1988) A sample conceptualisation for DIADES: Menu systems. Techn. Report PU1R1/88, Darmstadt University of Technology, Chair on Programming Languages and Compilers
Hoffmann HJ (1993) How to improve overall system quality by a sensible design of man-machine interaction - views of a software engineer -. In: Kafka P, Wolf J (ed) Proc ESREL´93, Elsevier Amsterdam et al, pp 939 - 947
Hoffmann HJ, Closhen P (1997) Spezifikation elektronischer Produktkataloge mit Hilfe der SASSI-Komponente von EPK-fix. In: Schneeberger J (ed) Software-Engineering für Multimedia-Systeme. Workshop GI´97, pp 5 - 6
Hoffmann HJ (1999) Multimedia features in the correspondents´ interface of MALL2000 systems. In Proc IEEE Intl Conf Multimedia Computing and Systems, IEEE Computer Society, vol 2, pp 1065 – 1067
ISO (1992 onwards) ISO 9241-xx, Ergonomic requirements for office work with visual display terminals
ISO (1993 onwards) ISO 9000-x: Quality management and quality assurance standards
Laurel B (1990) Art of human-computer interface design. Addison-Wesley
Lutze R et al. (1996) EPK-fix, Methoden und Werkzeuge zur effizienten Erstellung elektronischer Produktkataloge. Statusseminar Softwaretechnologie, BMBF, pp 299 - 318
Mosier JN, Smith SL (1985) Application of guidelines for designing user interface software. Proc Human Factors Society, 29th Annual Meeting, pp 946 - 949

MSWE 613, Univ. of Maryland, Usability Engineering Class (Fall 1998) Guide to usability for software engineers (GUSE). http://otal.umd.edu/guse/...
Nielsen J (1995a onwards) Bi-weekly alertbox. http://www.useit.com/alertbox
Nielsen J (1995b) Multimedia and hypertext: The Internet and beyond. AP Professional
Nielsen J (1995c) Guidelines for multimedia on the Web. In (Nielsen 1995a), Dec. 1995
Nielsen J, Galdo EM (1996) International user interfaces. John Wiley & Sons
Nielsen J (2000 announced) Designing Web usability: The practice of simplicity.
Nitsche-Ruhland D, Zimmermann G (1995) CritiGUI – Knowledge-based support for the user interface design process in Smalltalk. In: Blumenthal B, Gornostaev J (eds) Human-Computer Interaction. Springer Berlin Heidelberg, pp 179 – 188
Norman DA (1986) User centered system design. Lawrence Erlbaum Assoc
Parnas DL (1974) On a „buzzword": hierarchical structure. Information Processing 74, vol 2 Software, pp 336 - 339
Parnas DL (1998) Design and documentation of program structures. Lecture notes *„randell.slides"*, 20/9/1998
Perlman G (1987) An overview of SAM: A hypertext interface of Smith & Mosier´s guidelines for designing user interface software. Washington Inst of Graduate Studies, WI-TR-87-09
Perlman G (1995a) User interface guidelines and standards. ACM interactions 2/1: 5 – 7
Perlman G (1995b) Teaching user interface development to software engineers. ACM CH´95 Conf Companion, pp 375 - 376
Ratner J et al. (1996) Characterization and assessment of HTML style guides. ACM CHI 96 Electronic proceedings, Interactive Posters,
http:/www.acm.org/sigchi/chi96/proceedings/intpost/ratner/rj_txt.htm
Reed P et al. (1999) User interface guidelines and standards: progress, issues, and prospects. Interacting with Computers 12 72: 119 - 142
Royce WW (1987) Managing the development of large software systems. 9th Intl. Conf. on Software Engineering, pp 328 – 338 (re-print of a 1970 paper)
Rudlof C, Dzida W (1997) Die Rolle von Fehlern und Mängeln in der Qualitätssicherung von Software. http://selab24.informatik.uni-bremen.de/sw-ergo-news/sw-ergo160.html
Rumbaugh J, et al. (1998) The Unified Modeling Language, Reference Manual / User Guide.ACM Press and Addison-Wesley
Schneeberger J et al. (1997) EPK-fix: Software-Engineering und Werkzeuge für elektronische Produktkataloge. In: Jarke M et al. (eds) Informatik´97, Informatik als Innovationsmotor, Springer-Verlag, pp 446 - 455
Schneider HJ (1997) Lexikon Informatik und Datenverarbeitung, Version 4.0.R. Oldenburg Verlag München Wien, pp 102 & 103
Shneiderman B (1997) Designing the User Interface: Strategies for effective human-computer interaction. Addison-Wesley, 3rd edition
Skublics S et al. (1996) Smalltalk with style. Prentice-Hall Englewood Cliffs
Smith SL, Mosier JN (1986) Guidelines for designing user interface software. TheMITRE Corp., ESD-TR-86-278
STJA (1995 ÷ 1999) Smalltalk und Java in Industrie und Ausbildung. Förderkreis STJA e.V.
Taylor RN, Coutaz J (Eds., 1995) Software engineering and human-computer interaction. Springer, Berlin Heidelberg
Vanderdonckt J (1994) Guide ergonomique des interfaces homme-machine. Presses Universitaires de Namur, Namur
Vanderdonckt J (1995) SIERRA: An interactive system for ergonomic realization of applications. ACM SIGCHI Bulletin 27/2: 50 - 51
Vanderdonckt J (1999) Development milestones towards a tool for working with guidelines. Interacting with Computers 12/2: 81 - 118
VDI-GIS (1993) Software-Zuverlässigkeit: Grundlagen, konstruktive Maßnahmen, Nachweisverfahren. VDI Verlag
Vogt G (1994) Über Struktur und Aufbau eines Entwurfssystems für Benutzungsschnittstellen. Doctoral dissertation, Darmstadt University of Technology, Dept. of Computer Science
Wu Y (1999) Acquisition-Computation-Execution-Expression (ACEE): A software architecture pattern for computer supported interactive automation and control systems. PLoP99