LEARNABILITY AND ROBUSTNESS OF USER INTERFACES
Towards a Formal Analysis of Usability Design Principles

Steinar Kristoffersen
Østfold University College, Halden, Norway

Keywords: Logical modeling, precise analysis of usability evaluation, model checking.

Abstract: The paper is concerned with automatic usability assessment, based on heuristic principles. The objective is to lay the ground, albeit still rather informally, of a program of assessing the usability of an interactive system using formal methods. Further research can then extend this into an algebra of interactive systems.

1 INTRODUCTION

We know that the effect of poor usability is difficult to measure (Lund, 1997). Usability itself is difficult to define, at least at any level of precision deeper than by example. Researchers agree that improving usability may save considerable time and resources (Myers, 1994). Few in industry will say that usability has little value, although perceived user-friendliness is not a significant determinant of adoption in the first place (Davis, 1989). Vredenburg et al. found that ca. 20% of a project's resources will be spent on activities related to usability, but that the effectiveness of user-centered design activities are not usually measured (Vredenburg et al., 2002).

When can we tell that our usability goals have been reached? Many forms of usability assessment exist (Holzinger, 2005), but for recurring reasons such as lack of resources, time and technology, the most widely encompassing and precise methods are often ignored (Mulligan et al., 1991).

Various forms of *heuristic evaluation* based on experts or users benchmarking of a specification or prototype again pre-defined usability design principles, have proven to be quite efficient (Nielsen, 1992), compared to other techniques (Jeffries et al., 1991). Nevertheless, it may require as many as 15 evaluators for an optimal result (Nielsen, 1993), which is well beyond the reach of most projects.

Information technology is becoming commoditized and the time of stake-holders is a relatively scarce resource. Many project have to be developed for the web or with mobile clients, which tends to limit the degrees of freedom even more. Most users are experienced with similar applications already, and want to be able to apply the skills that they already have. Less software is built "from scratch". These trends towards standardization may lead to a reduction in the resources available for an expert-based, manual evaluation of usability.

Moreover, it has been proven that the *evaluator effects* in heuristic evaluation are significant and that they may invalidate the results of using many evaluators (Hertzum and Jacobsen, 2003). In order to make better interfaces, and iin the next instance even assess the usability process itself, as well as communicate clearly the results and intentions coming out of an evaluation, a stable set of usability criteria and design principles needs to be observed.

It is, however, difficult to say if and to which respect, one set of usability guidelines or design principles is going to perform better than another (Nielsen, 1994). The sets of guidelines and design principles that we have today are arguably the result of experience and well-founded theoretical reasoning, but they have not themselves been subjected to scientific testing. This is the long-term objective of the research described in this paper. In order to get there, some groundwork entailing the operationalization of the underlying heuristics is an absolute prerequisite. A stricter approach to expertise-based evaluation, based on well-defined methods embedding a fixed set of principles is necessary to develop a stable "best practice" of usability principles, since evaluation of the approach is otherwise impossible.

Experience indicates that human actors, however, will not meticulously follow à priori rules (Carlshamre and Rantzer, 2001). Moreover, it is difficult to

Kristoffersen S. (2008).
LEARNABILITY AND ROBUSTNESS OF USER INTERFACES - Towards a Formal Analysis of Usability Design Principles.
In *Proceedings of the Third International Conference on Software and Data Technologies - SE/GSDCA/MUSE*, pages 261-268
DOI: 10.5220/0001869402610268
Copyright © SciTePress

isolate the usability concerns from everyday development activities (Gentner and Grudin, 1990). Ivory and Hearst found many studies confirming that designers find it difficult to adhere to guidelines, and that they are biased towards an esthetically pleasing design regardless (Ivory and Hearst, 2001). The development of automatic usability evaluation techniques is therefore essential to advance the research in this area.

2 PERTAINING SYSTEMS FOR USABILITY ASSESSMENT

Many guidelines and standards turn out to be poorly formulated and difficult to use, upon closer inspection (Thovtrup and Nielsen, 1991). The tools that exist for automating the assessment of usability correctness criteria are often not sufficiently oriented towards efficient software development. Many tools require dedicated mark-up or tool-chain facilitation. Others are directed towards post-hoc evaluation. This redundancy aspect may explain why they have had relatively limited industrial success. Some tools have simply been too cumbersome for designers and developers to be able to adopt (Ivory and Hearst, 2001).

Some progress has been made in the realms of HCI (Human-Computer Interaction) research, however. The Catchit project has taken the need for efficient integration in the software development tool-chain seriously. It addresses evaluation in predictive terms as well as development-oriented modeling of user behavior. Its software automatically instruments the source code with a monitor which detects deviation from the expected work-flow (Calvary and Coutaz, 2002). Most previous approaches needed such instrumentation to be carried out manually (Coutaz et al., 1996), which is error-prone in itself. Catchit represents an improvement, since it does this automatically. It needs, however, a running codebase to instrument. This limits the usefulness of the tool to stages in which a running version of the system has been implemented.

Another example of an integrated user interface design and evaluation environment, is AIDE. It supports the assessment of a precisely defined set of metrics related to efficiency on the keystroke-level, the alignment and balance of elements on the screen, and eventual violations of constraints given by the designer (Sears, 1995). It can generate layouts, but is perhaps limited in spite of its precision by this strong focus exactly on the layout of a single dialogue. It leaves the implementation of more broadly scoped usability principles almost as an “exercise for product developers”. We believe that it is exactly in the formal modeling of deeper relations between multiple elements (rather than widgets) that an improvement of user experiences can be mostly improved.

One early example of an automatic tool aiming at improving the usability of projects, based on guidelines which are then compared to the actual performance of the implemented system or at least its specification, is the KRI/AG (Löwgren and Nordqvist, 1992). The project builds, like ours, on a selected number of operationalized user interface design guidelines. The interface is encoded and then subjected to automatic analysis. It does not, on the other hand, see this as an instance of a more general model-driven approach to software engineering, and in spite of encouraging results from its initial trials, it is now a near-forgotten endeavor.

3 RESEARCH OBJECTIVES

Our paper aims to pick up where the KRI/AG project left off. A novel contribution of our work compared to Löwgren and Nordqvist’s work are the improvements that we suggest to the modeling approach, in order for this type of effort to succeed. This is crucial if we want to see it as part of a more ambitious, general approach, and even if we look only at the more widespread interest lately in automatic evaluation of the usability of web-pages (Chevalier and Ivory, 2003).

Ideally, any project should have some form of automatic usability evaluation support available, which is itself easy to use, stable and transparent to complement. We may in some situation wish to be able to substitute dedicated expertise if that is not available, but ideally of course, we would be offering it as a complement tool.

The research projects that we outlined above have not spawned into successful commercial products. Looking at the formulations of usability principles themselves, in research papers (Gould and Lewis, 1985) as well as in the HCI curriculum (Dix et al., 1997), this should come as no surprise. They are usually left rather vague, even for a human student of the principles, and implementing automatic tool support on the current conceptual platform is thus impossible.

In order to be able to develop an improved and partly (at least) automatic checking of usability guidelines adherence as an integrated element in the development tool-chain, so that it will be able to detect failure of invariants based on well-known usability principles in an efficient manner, these principles need to be properly operationalized. It is therefore necessary to take a step back and look at the specification of the

criteria themselves.

In order to prepare the ground for an even more ambitious theoretical endeavor, which we outlined above, a walk-through and discussion of the "user requirements" is warranted. In order to study the correlation of perceived usability and formal assessment, the guidelines need to be stable, well understood and operationalizable. This is what we do next. The principles are ordered within categories of broader usability concerns, which are learnability, flexibility, robustness and task conformance. We deal only with two of the categories in this paper, due to page number limitations. We are going to deal with the remaining two categories in a completely analogous fashion, in future work.

4 RESULTS

In this section we will discuss and then summarize the analysis that we did with respect to the operationalizability of the design principles. The purpose is to uncover at least one possible precise (albeit still informal, in terms of the notation that we apply) formulation of the principles, and sketch the research problems that we believe must be solved in order to implement a fully automatic check of the principles.

4.1 Learnability Principles

4.1.1 Predictability

The user ought to be able to judge what system response is going to be as a response to the next user action, and which state it will lead to. An informal specification of this principle, as a "theorem of usability" might be that it ought to be impossible to get from any to state to a state that is invisible, or to apply (inadvertedly) a rule which has not made itself known to the user.

The problem is that the requirement takes to its logical consequence, instructs that all states in the path of action that may be performed without user interruption (or some other definition of "closure"), from the current state, need to be visible. This is usually neither possible nor desirable and it is an empirical question if it really does encourage learnability much. It certainly may leave the impression of a messy interface. Some of the published actions in the next sequence of possible steps, may now be "out of context" for the user, and therefore diffcult to comprehend.

4.1.2 Synthesizability

The user should be able to understand which user actions have lead to the current state, and what the system did to get there. We need to find out if the there are somehow invisible states that lead to the current state. Thus, this is the criteria representing the inverse of predictability.

The problem is similar to the one above, and the trivial solutions equally unproductive. Unless the user can learn to remember any possible path leading up to an identifiable state, the publishing of possible paths (at least a small handful of steps back) seems to be necessary. Unfortunately, that will totally clutter the interface.

4.1.3 Consistency

The system should offer the same or similar functionality from comparable situations, and in a familiar fashion. The same or similar actions should yield the same response. This means that we expect the same or similar components to look alike and to respond similarly on user input. We summarize it as the extent to which similar appearances offer the same functionality. We think that this colloquially resembles the second of Grudin's consistency types: "External consistency of interface features with features of other interfaces familiar to the users (Grudin, 1989)."

The principle of consistence is, to be fair, not uncontested. Grudin's paper is one strong voice in this respect (Grudin, 1989). One can imagine situations in which consistency does not encourage learnability. It is outside the scope of this paper to enter that discussion, however, at least from a theoretical angle only. Our ambition is to prepare the grounds in this paper to do this empirically at a later stage.

4.1.4 Generalizability

Generalizability is sometimes described as "a form of consistency," except that it applies more broadly to situations, rather than just operations. It is a state where existing knowledge can be successfully applied; as such it digs even deeper into the question of what is "the existing knowledge". We summarize this as being the extent to which related functionality can be grouped, or a sequence of actions can be seen as coming to some form of "closure." Thus, it is aligned with Grudin's first type of consistency, which he calls "Internal consistency of an interface design (Grudin, 1989)."

In terms of operationalizing this principle, it is, to start with, difficult to know exactly what to match, and

certainly there is no useful ontological or etymological answers at the surface anywhere. We have already pinned down *consistency* as a criteria which stipulates yielding the same effect from similar actions. Relying on the abstraction mechanisms well-described in object-oriented programming, we now define *generalizability* as the property of categorizing sensibly, so that similar action-effect pairs can be grouped together under more abstract headings, which seen from outside the group behave in a coherent manner.

4.1.5 Familiarity

This is an externally oriented criteria, which capture the extent to which the user experiences a real-world parallel to the system. Can we match the actions that we work with to similar activities outside the system, so that lessons learned can be exploited either directly or metaphorically? This criteria attempts to measure the correlation of users' knowledge with the skills needed for effective interaction. We summarize it as the extent to which functionality offered by the system is similar to "a priori" or at least widely held, experiences. It overlaps nicely to Grudin's third consistency definition; which is "correspondence of interface features to familiar features of the world beyond computing. (Grudin, 1989)."

The biggest problem here is of course to be able to match anything within the system with a theoretically infinite and unspecifieable universe without. The character and number of "experiences" held by the users will be vast, and even if human beings can be expected to, to some extent, make their mind up about what constitutes a priori knowledge, we cannot expect to make an *algorithm* which does.

4.2 Robustness Principles

4.2.1 Observability

The question about observability asks if it is possible for the users to decide which state the system is in, from what they are observing. Are there states which cannot be assessed from the interface?

We find an unresolved problem right away, when we try to operationalize this principle, namely identifying which states we shall judge as being significant. It is likely to clutter the interface and overwhelm the user if too many such states are "listed" at the interface, so the majority of states will and should be hidden from the user. But the user still need be able to find out what is "going on," in order to diagnose and repair the interaction if it does not proceed according to the intention. The next principle, of browsability, is partly an answer to this question.

4.2.2 Browsability

The principle of browsability concerns whether there there states which cannot be assessed from other states, i.e., can the user cycle through all the states once one is presented at the interface, to assess all others.

This is an equally tricky claim to respond to, theoretically as well as technically. The systems which we are interested in is often going to have infinitely many, even uncountable states and the interactivity of the system as such makes it non-deterministic. Heuristics to constrain and limit the search strategies and algorithms that act out the user options fairly, will be a prerequisite to establish fulfillment of this criteria.

4.2.3 Defaults

Next, a simpler criteria is often listen, namely if all input states have default value suggestions. We might want to allow an empty input be the default, but is is important that this is at the designers discretion and a conscious choice. Pragmatically, this might simply mean that some explicit choice needs to have been made.

Given that a modeling language is available which makes implicit that an element is of type *input field*, we can quite straightforwardly implement a check for default values and alert the designer is one is not given (and preferably *explicitly* not given). This is not difficult in a static description. Given that we are interested in the dynamic behavior of an applications, some problem may arise in which it is difficult to separate a default value ("output") from user input.

4.2.4 Reachability

The main notion of **reachability** concerns whether the user can navigate from any given state to any other state. This criteria is easily operationalized, although it may not be practical to check since the number of states is usually very large, and potentially infinite. Thus, we expect to have to devise clever search strategies in order to be able to check our model. It may even be necessary to built assumptions about which subset of states that are relevant from any given "position" in the application. This is perhaps ideologically unfortunate. Often one tends to assert that the underlying models, and the actions and analysis which is furnishes ought to be independent. In extrapolation from this failure to separate concerns, errors may be introduced into the model. Moreover, maintenance and extension of the entire framework of formal analysis may be made more difficult. This is also a con-

cern which we need to dig deeper into in further research.

4.2.5 Persistence

The idea is that system communication to the user at any state needs to be easily retrievable, unless (and perhaps in spite of, sometimes) having been explicitly deleted. This criteria overlaps nicely in abstract terms with the previous one, by a fortunate side-effect of our choice of checking not the static model, but the dynamic trace of entire state universe from the application. Communication with the user is going to be captured by states, and the question can then be translated into one of reachability of these states from those which follow. It highlights the general question of ordering of states, of which chronological ordering is but one of the simplest of orderings. Logical and semantic ordering of (communication) states is also an area of which our perspective invites further research.

4.2.6 Recoverability

The challenge of making sure that is is possible to recognize and repair errors when they are detected is a much harder one. The literature distinguishes between several interesting aspects.

- "Forward error recovery," i.e., is it possible to move forward to a state without errors.
- "Backward error recovery," in other words, are there states from which an action is irreversible?

Both of the above may be seen as instances of reachability, since they stipulate an extent to which other states (bearing in mind that we are investigating properties of the total dynamically generated state-space) can be reached.

Additionally, the "commensurate efforts" aspect is associated with recoverability. It denotes the extent to which the length of the path of actions that lead from one state p to another state q, is equal to the one which goes back from q to the first state p. In other words, if something is difficult to repair, it should be difficult to break in the first place.

In terms of our research, the biggest challenge that we have recognized so far is to be able to recognize the start and stop of a user action (in some meaningful sense of "closure"). This is also related to the "calculus" of user interface behavior that we implicitly prepare the ground for now, similarly to ordering, since it concerns the capability of "grouping" state transitions.

4.2.7 Responsiveness

Finally, in the *robustness* category we find the criteria that if re-action is not immediate, there needs to be mechanisms in place that indicate to the user how long it is going to take before it is finished. A historical measure is also. according to Dix, useful, in the sense that "time stability" is desirable (Dix et al., 1997). It means that the time it takes to execute actions needs to be approximately the same every time.

We see this criteria as bringing to the fore a set of research questions related to the dimensions discussed above, concerning grouping and ordering of state transitions. In addition, it demonstrates the need to introduce real-time aspects into the modeling approach. This has so far been under-emphasized in formal research on interaction design. It is by now evidently clear that we need to separate between concepts more clearly, and give some of them a more exact interpretation in relation to the domain of user interaction. The most important are listed here:

- Significant State. This is a state that is modeled. A "real" program during execution will have a vast number of states that we do not need to model, since they have no bearing on the properties that we wish to analyze. Our theorems only describe significant states, by definition.
- Visible State. This is a state that makes itself known to the user as the state or a previous one, is entered.
- Published Rule. This is a rule that is visible at the state from which the rule can applied, or earlier.
- Published state. This is a state that is visible at a state from which a rule can be applied, which will lead to the state, or earlier.
- Aggregate Action. A "macro" of state-changes, which are in themselves atomic in the sense that they can be performed individually or in other aggregate actions.
- Compound Action. A "procedure" of state-changes, in which individual statements are not independently meaningful.

5 DISCUSSION

At the very first point of reflection, it becomes clear that the usability guidelines and principles that one aims to assert in some formal fashion, will need to be operationalizable at an entirely different level from what we know today. This turns out to be very hard, though, as noted by Farenc et al. (Farenc et al., 1999).

An effort such as the one described in his paper contributes to advance this situation, by attempting to respecify usability design principles in a form that may be decidable "even by" computers. One should be careful not to expect to be able to capture every aspect of each rule in this way, however. Our attempts at formalizing design ambitions may make them more trivial. Clearly, we do not expect such an effort to eliminate the competencies of a human evaluator and see it instead as a complement and a first stab at tool support for usability engineering. Lack of precision is not, of course, an advantage, on the other hand(Doubleday et al., 1997), and one should arguably be doubtful of design principles that cannot at least be exemplified or seem to detect simple instances of non-adherence.

When we know about the divergence caused by the evaluator effect and the time and resources needed to do robust usability testing, tool support for investigation the the HCI (Human-Computer Interaction) aspects is clearly warranted. Some systems for usability testing exist, relying on guidelines and standards, which turn out to be hard to use even on their own (Thovtrup and Nielsen, 1991). The tools for automating the assessment of usability correctness criteria is often not efficiently integrated with software development, or facilitate only post-hoc evaluation (Ivory and Hearst, 2001). This introduces redundancy aspects, which may explain why they have had relatively limited industrial success.

Usability engineering is often limited to informal user testing and ad-hoc observations (Holzinger, 2005), which, apart from the problems of divergence and user/expert involvement needed, suffer from the lack of generalizability and predictive force. Thus, a "theory of usability" is needed. Many such attempts to make a formal argument of usability is related to GOMS (Goals-Operators-Methods-Selection rules) or similar rational or at least goal-oriented models (Gray et al., 1992). There has been reasonable correlation of GOMS-based predictions with experiments established in the literature (Gray et al., 1992). Unfortunately, creating such models is labor-intensive and error-prone. Using it for evaluation requires a low-level specification or finished software which can be run to elicit a task model which is sufficiently high-fidelity, since the GOMS family of model represents user actions down to the level of singular keystrokes (John and Kieras, 1996a).

As a dedicated medium, theoretical representation of the users' interaction with the system can be seen as facilitating the job of evaluating usability. On the other hand, it is not usually viable as a modeling approach that is going to drive the development of the interface in the first place, although this is a possibility (John and Kieras, 1996b). Too often, it relies in the first instance on an existing computer system or an implementation level specification (Card et al., 1980), which arguably is exactly what one wants it for in the first place.

Ideally, the formal modeling of user interfaces, which are input to the evaluation, should be exactly the same specification as the one used for the design in the first place. It makes it more likely that it will actually be used, since it does not create redundant specification work. More importantly, however, a multipurpose specification would make it possible to conduct continuous evaluation of usability aspects. Indeed, it could be built into the software development environment.

LOTOS is one alternative specification language for interactive user interfaces (Paternó and Faconti, 1993), which could be seen as aiming to fill this role. It is more akin to a general design language than the syntax of the keystroke-level GOMS. This could also be seen as its biggest downside also, since it becomes almost as complex to make the specification as the actual programming of the user interface. It is similar enough to a full-blown programming language for an even more overlapping representation in the form of running code to be a tempting alternative in many projects, and the advantages of formal specification is lost if it is not robustly simple and abstract enough for the designer to be able to verify that the model is accurate. Still, it is not an executable specification, so the work entailed by making the test aid is easily perceived as redundant. We believe that a declarative approach is needed, and preferably one that can rely on model-checking of the logical properties of the specification.

Approaches used in formal research in HCI, such as GOMS and LOTOS, are not widely used in the industry. Some argue they have fundamental problems, which means they only succeed in narrow domains and will not realistically be useful in actual design projects (Bannon and Bødker, 1991). In this paper, instead, we see the problem as being the lack of proper operationalization of the underlying usability design principles. As a first step toward resolving that, we have offered a re-specification of a subset of the most commonly taught usability principles (Dix et al., 1997). Additionally, we think for further work that creating or extending a formal modeling language so that it is not only suited for describing interactive user interfaces in a platform-independent fashion, but also testing its logical properties in a precise way, is absolutely necessary.

6 CONCLUSIONS

It is important to remind ourselves that we do not take for granted that implementing the principles in accordance with any of the definitions above, is *à priori*, in itself, virtuous or necessary (although it seems reasonable) to achieve usability. We see this as an empirical question, which needs to be assessed independently. It will, however, be a much more doable assessment in the first place, if as we have suggested, a precise and formal definition of what each principle entails. Moreover, the availability of a tool which can identify fulfillment or breakage of consistency criteria will be necessary for any quantitative assessment of the correlation between practice and theory. A subjective or example-dependent qualification of each principle may be sufficient for teaching the notion comprised by each principle, but it will not do as a point of departure for experiments of a more quantitative nature. We believe that the latter will be a strong supplement to the existing body of work.

Another equally important contribution of the research presented in this paper is that it documents shortcomings of modeling techniques when their objective has not been taken properly into account. We know that many of the more generic frameworks for describing user interfaces are not suitable for the dual task of development and formal analysis. In many respects that we have touched upon in this paper, they are not suited for formal validation of usability design principles. There are many drawbacks. The volume and verbose nature of the specifications make them hard to write and understand for the "human model checker,"who at least has to be able to check the model checker. We are of course aware of the irony in this, but improvement of practice must be seen as desirable even if it is stepwise rather than total, in our opinion.

It will, as we see it, be a great advantage compared to most other automatic usability evaluation methods based on models, if one can devise an approach which does not need be "made to match" an existing artifact, i.e. a dedicated format or tool. These approaches suffer from an "impedance mismatch" problem, by which we mean that the representation of the artifact intended for checking may itself be an inaccurate image (or it may not be one-to-one). By definition, using a declarative product from the software life-cycle product chain itself, will make our "substrate" correspond more accurately with the manifest artifact that one aims to implement in the next instance, namely the dynamic user interface. Th result may still not be exactly what the users wanted, but at least we can check it properly and know that it represents correctly the artifact, since the relationship between them *is* one-to-one. On the other hand, this may represent a problem for the specification of the search strategies that perform the model checking.

Finally, we need to state that in our opinion the possibility of a nice framework and associated toolkit for logical and precise analysis of usability principles in an interactive application, does not pre-empt the need to work closely with users. Notwithstanding the internal validity of our contribution, which is to some extent only depending on our efforts to formulate an abstract world, the usefulness of such a framework depends wholly on the "real world". Thus, we look forward to being able to compare the predictions of a formal analysis with traditional usability evaluation of the same systems. Only when correlation on this level has been established, of course, one may conclude that this type of approach is really viable.

REFERENCES

Bannon, L. J. and Bødker, S. (1991). *Beyond the interface: encountering artifacts in use*, pages 227–253. Cambridge University Press, New York, NY, USA.

Calvary, G. and Coutaz, J. (2002). Catchit, a development environment for transparent usability testing. In *TAMODIA '02: Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design*, pages 151–160. INFOREC Publishing House Bucharest.

Card, S. K., Moran, T. P., and Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Commun. ACM*, 23(7):396–410.

Carlshamre, P. and Rantzer, M. (2001). Dissemination of usability: Failure of a success story. *interactions*, 8(1).

Chevalier, A. and Ivory, M. Y. (2003). Web site designs: influences of designer's expertise and design constraints. *Int. J. Hum.-Comput. Stud.*, 58(1):57–87.

Coutaz, J., Salber, D., Carraux, E., and Portolan, N. (1996). Neimo, a multiworkstation usability lab for observing and analyzing multimodal interaction. In *CHI '96: Conference companion on Human factors in computing systems*, pages 402–403, New York, NY, USA. ACM.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):319–340.

Dix, A., Finlay, J., Abowd, G., and Beale, R. (1997). *Human-computer interaction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Doubleday, A., Ryan, M., Springett, M., and Sutcliffe, A. (1997). A comparison of usability techniques for evaluating design. In *DIS '97: Proceedings of the 2nd conference on Designing interactive systems*, pages 101–110, New York, NY, USA. ACM.

Farenc, C., Liberati, V., and Barthet, M.-F. (1999). Automatic ergonomic evaluation: What are the limits? In *Proceedings of the Third International Conference on Computer-Aided Design of User Interfaces*, Dordrecht, The Netherlands. Kluwer Academic Publishers.

Gentner, D. R. and Grudin, J. (1990). Why good engineers (sometimes) create bad interfaces. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 277–282, New York, NY, USA. ACM.

Gould, J. D. and Lewis, C. (1985). Designing for usability: key principles and what designers think. *Commun. ACM*, 28(3):300–311.

Gray, W. D., John, B. E., and Atwood, M. E. (1992). The precis of project ernestine or an overview of a validation of goms. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 307–312, New York, NY, USA. ACM.

Grudin, J. (1989). The case against user interface consistency. *Commun. ACM*, 32(10):1164–1173.

Hertzum, M. and Jacobsen, N. E. (2003). The evaluator effect: A chilling fact about usability evaluation methods. *International Journal of Human-Computer Interaction*, 15(1):183–204.

Holzinger, A. (2005). Usability engineering methods for software developers. *Commun. ACM*, 48(1):71–74.

Ivory, M. Y. and Hearst, M. A. (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv.*, 33(4):470–516.

Jeffries, R., Miller, J. R., Wharton, C., and Uyeda, K. (1991). User interface evaluation in the real world: a comparison of four techniques. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 119–124, New York, NY, USA. ACM.

John, B. E. and Kieras, D. E. (1996a). The goms family of user interface analysis techniques: comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351.

John, B. E. and Kieras, D. E. (1996b). Using goms for user interface design and evaluation: which technique? *ACM Trans. Comput.-Hum. Interact.*, 3(4):287–319.

Löwgren, J. and Nordqvist, T. (1992). Knowledge-based evaluation as design support for graphical user interfaces. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 181–188, New York, NY, USA. ACM.

Lund, A. M. (1997). Another approach to justifying the cost of usability. *interactions*, 4(3):48–56.

Mulligan, R. M., Altom, M. W., and Simkin, D. K. (1991). User interface design in the trenches: some tips on shooting from the hip. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 232–236, New York, NY, USA. ACM.

Myers, B. (1994). Challenges of hci design and implementation. *interactions*, 1(1):73–83.

Nielsen, J. (1992). Finding usability problems through heuristic evaluation. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 373–380, New York, NY, USA. ACM.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 152–158, New York, NY, USA. ACM.

Paternó, F. and Faconti, G. (1993). On the use of lotos to describe graphical interaction. In *HCI'92: Proceedings of the conference on People and computers VII*, pages 155–173, New York, NY, USA. Cambridge University Press.

Sears, A. (1995). Aide: a step toward metric-based interface development tools. In *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, pages 101–110, New York, NY, USA. ACM.

Thovtrup, H. and Nielsen, J. (1991). Assessing the usability of a user interface standard. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 335–341, New York, NY, USA. ACM.

Vredenburg, K., Mao, J.-Y., Smith, P., and Carey, T. (2002). A survey of user-centered design practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page 471478, New York. ACM Press.