FLEXIBILITY, COMPLETENESS AND SOUNDNESS OF USER INTERFACES

Towards a Framework for Logical Examination 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 precisely (Lund, 1997). How 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).

This paper initiates work with automated usability assessment. In order to implement model chcking software based on on the guidelines, they need to be operationalized. The goal is to reach a level of precision by which each principle may be specified using equational logic (Baader and Nipkow, 1998). 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 (Nielsen, 1994). This is the another long-term objective of the research described in this paper.

2 PREVIOUS RESEARCH

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 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 toolchain 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

Kristoffersen S. (2008).
FLEXIBILITY, COMPLETENESS AND SOUNDNESS OF USER INTERFACES - Towards a Framework for Logical Examination of Usability Design Principles.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - HCI*, pages 346-351
DOI: 10.5220/0001684803460351

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.

Ideally, any project should have some form of automatic usability evaluation support available, which is itself easy to use, stable and transparent. We may in some situation wish to be able to substitute dedicated expertise if that is not available, but most often it would be offered as a complementary 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 very difficult.

We wish 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. Hence, 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, 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 therefore 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 another paper.

3 RESULTS

In this section we will present our analysis 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.

3.1 Flexibility Principles

3.1.1 Dialog Initiative

This property indicates the difference between situations where the system initiates action, by prompting action from the user or itself acting autonomously, and situations where the user decides what and when to act. The former can be described as having internal and/or invisible actions implemented by the system. In order to determine the situation thus described, we need to look for and display the path leading ”through” states in which there is no user input required. It can then relatively easily be quantified by the length of such paths, the coverage of paths for the entire network of states which we envisage constituting the application, and we can even look for cycles or “dead-ends” of internal and/or invisible paths, which will tell us (eventually) by deadlocks in the interactive perspective of the user. This is a fairly operational category of usability design criteria, it seems.

3.1.2 Multi-threading

This is a property which we see as the capability of letting the user work with more than one task at a time. In other words, it is characterized by presence of states in which no other parallel non-terminating states exist, or the complementary, states which *have* other parallel non-terminating states. In the literature (Dix et al., 1997), it is common to differ between interleaving and concurrent multi-threading.

It still seems quite doable to formalize this notion, except that it places rather strict demands on the modeling effort, notwithstanding that the language used for modeling in itself needs to be well designed. For instance, we will have to be able to decide for every state, which of the other states that might be performing in parallel. One can expect, in any system, the search for such states to be very computationally demanding. This is clearly an issue for further research.

3.1.3 Task Migratability

Task migratability is a subtle, but important property, It designates whether a system or subset of the system will allow the user either to take control over a task in a step-by-step fashion, and vice versa if the user may request the system to perform a series of tasks as an aggregate. Otherwise, the system is less flexible in the sense that user interference in subtask sequences is not allowed. In terms of operationalizing it, we will have to search for sequences of states that cannot be executed one by one as a series of actions by the user.

From this discussion and tentative definition is seems to be fully viable to determine the number of states, which qualify as being of one category or the other. We will face two related problems, though, namely that any conceptual "state" can be arbitrarily modeled in terms of significance and granularity and the "number of" sub-states of what the designer perceives as a logical state, may as such not be particularly pertaining to user perception of multi-threading. This is an issue for further research, but for now we defer that to the modeling strategy. As will be further elaborated in the discussion, this is also the case for many other criteria.

3.1.4 Substitutivity

This is the property of allowing the user to decide upon a suitable format for representing input as well as output. For our purposes, it may, to keep it distinct from the two next categories and decidable in a formal analysis, be seen as characterized by the presence of input actions that (do not) have multiple translations of values to different representations. We can than search for states comprising output or input actions that have multiple presentations of the same values, including the difference between user inputting and computing a value.

Alan Dix writes about how "Equal opportunity blurs the distinction between input and output at the interface (Dix et al., 1997)" and this illustrates several problems of operationalizing the judgment of whether a system fulfills the *substitutivity*-criteria. Some types systems, archetypically the well-known spreadsheet, allows input as well as system-computed output, literally in the same cell. It is computationally exhaustive to search for states from which new values can either be computed or inputted, in a "simple" application like this, based on a simulation. The notion of typical user behavior may be necessary to constrain the search strategies. Other alternatives may be discovered as we extend our research into this domain.

3.1.5 Customizability

We see the criteria of customizability as (at least) twofold, but first and foremost as different from substitutivity by being related to variance of assemblages, rather than a translation or multi-modality of formats (particularly in the input/output regions). It dual nature can be associated with who mobilizes the customization, which is always on the user's behalf, of course. In the first instance, users modify the connection between components or script their behavior to make them fit their intentions. This is what we call *adaptability*. In the other instance, we are concerned with making the system "intelligent"enough to make the necessary adaptations, and we denote this *adaptivity*.

It turns out to be difficult in practice to make a system adaptive (or tailorable) in any sense of the word, but that is not our concern here. We offer a definition which can be operationalized in a formal analysis of the system. A search for adaptability can then be seen as looking for states from which the actions can be modified by the user, and still lead to the "same next state". On the opposite site of system preemptiveness, we have a search for adaptivity, which is when we look for states from where the actions are modified by the system, depending of user input, but not *caused* by it. And this reveals another important unresolved research problem in the domain of formal analysis of HCI, namely how to distinguished between causal chains of events which are intended by the user and those which are intended by the programmers. In runtime, they may be difficult to separate. Again, this is an issue for further research.

3.2 Task Conformance Principles

3.2.1 Task Adequacy

The criteria of *task adequacy* asks if tasks are mapped into action in a way that the user finds satisfying. We cannot decide this on a subjective level, of course. We need to relate somehow to requirements. Within the set of pre-suppositions thus given, we find that this is a criteria of great importance, since it asks if there are rules in the system which do not correspond to requirements, i.e., is the system *sound*. This gives us, at least in logical terms, a precise definition.

The problem of this criteria is, to begin with, the relationship between the requirement specifications and the system/user interface specification (model). Looking at it the way we do turns the usability research in a new conceptual direction, since it implies that it is (in logical terms) the requirement specifica-

tions which constitute the model (i.e., the world) and the user interface specification (executable), is a set of theorems. The research agenda is then set, to device a useful logical approach for evaluating theorems about a given set of requirements. This is something that we think will be very useful.

3.2.2 Task Completeness

As a consequence of the inverse perspective of the user interface versus the requirements, introduced above, we can re-interpret the idea of *task completeness* as well. The question becomes if there are relations in the model, i.e., requirements, which we have not mapped to theorems, i.e., rules which govern transitions between states. In logical lingo, we want to be able to ask if the system is *complete*.

The challenges of this criteria is of course close matching the ones which pertain to task adequacy.

4 DISCUSSION

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 that is reached.
- 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.

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 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 examplified or seem to detect simple instances of non-adherence.

When we know about the divergence caused by the evaluator effects (Hertzum and Jacobsen, 2003), 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, 1996). 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 multi-purpose specification would make it possible to conduct continuous evaluation of usability aspects. Indeed, it could be built into the software development environment.

LOGOS 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 LOGOS, 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.

5 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 *a 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 an issue for further research into 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

Baader, F. and Nipkow, T. (1998). *Term rewriting and all that*. Cambridge University Press, New York, NY, USA.

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.

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.

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.

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.

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.

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

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.

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.