Presented at Workshop on Representations, QMW, July 1997

OSM: an ontology-based approach to usability evaluation

Ann Blandford & Thomas Green

School of Computing Science
Middlesex University
Bounds Green Road
London N11 2NQ
A.Blandford@mdx.ac.uk

Computer Based Learning Unit
University of Leeds
thomas.green@ndirect.co.uk

Abstract

An Ontological Sketch Model (OSM) is a structured but informal representation of the ontology — the essential underlying structure — of a system, forming a basis for usability assessment. Our primary aim is to develop an approach that is usable and that yields useful results.

We present a preliminary ontology for the OSM, based on descriptions of the entities in the domain of application, actions that the user can perform, and relationships (such as "constrains" and "affects") between entities. Initial studies into the usefulness and usability of OSMs indicate that the approach is promising, but that further work is needed on developing more comprehensive training materials.

Introduction

Usability is a central concern in the design of interactive systems. However, the most extensively used approaches to incorporating user concerns in the design lifecycle (e.g. use of guidelines (Smith & Mosier, 1986) or heuristic evaluation (Nielsen, 1994)) are informal and heavily reliant on craft skill. Such methods can be very effective in the hands of a skilled analyst, but they are highly dependent on the skill of the individual. In principle, theory-based methods offer approaches to usability evaluation that are less dependent on craft skill, that utilise established results from cognitive psychology, that can be applied earlier in the design process when fewer commitments have been made, and that can yield a deeper understanding of usability problems and appropriate solutions. However, while there is significant work on theory-based methods of usability assessment within the research community, there has been little take-up of these methods by industry (Bellotti, 1989). There are many reasons for this, including the difficulty of "technology transfer" (Buckingham Shum & Hammond, 1994) between the academic developers of such methods and the industrial users. The aim of the work reported here is to develop and test an approach to usability evaluation that draws on theoretical results from the academic community but can be applied by members of a design team at comparatively low cost, without depending too heavily on craft skill or expertise in cognitive science. The slogan is: "Useful and Usable Representation".

In the approach we propose, Ontological Sketch Models (OSM), the analyst generates a structured but informal representation of the ontology — the essential underlying structure — of the system. Unlike the great majority of HCI approaches, the focus is not on what the user needs to do, but on what the user needs to know, represented as a list of entities, actions, and relationships. A somewhat similar focus on structures rather than tasks can be found in ERMIA (entity-relationship modelling of information artefacts) (Green and Benyon, 1995), but ERMIA uses a more detailed, formalistic analysis, has less scope for describing differences between relationships, and has no representation of user actions. The OSM approach has drawn heavily on ERMIA but aims to be sketchier and more usable, offering a different type of usefulness.

OSM also draws on the knowledge analysis stage of Programmable User Modelling (PUM; Young, Green & Simon, 1989; Blandford & Young 1996). Knowledge analysis involves laying out the knowledge the user needs (in terms of conceptual objects they work with, and preconditions and effects of operations), and giving an account of where that knowledge might come from. Full PUM analysis then involves expressing that knowledge formally in an 'Instruction Language' and applying cognitive modelling techniques to reason about likely user behaviours when interacting with the device. OSM demands less rigour than a full PUM knowledge analysis, and allows a wider range of concerns to be expressed within the model.

We are aiming to develop an approach that can be used by non-specialists. Experience of training students in the PUM cognitive modelling technique (Blandford, Buckingham Shum & Young, in preparation) indicates that students can be trained to do knowledge analysis for surface phenomena (that do not require a deep understanding of the underlying cognitive model) within a matter of a few hours, whereas a much longer period of training and practice is needed for the analysis of phenomena that hinge on having a good understanding of the underlying cognitive theory. This deeper analysis can be justified in some circumstances — e.g. where the results have wide applicability (Bellotti et al, 1996) or where the interactive system has to be very reliable — but may be harder to justify in others. Further informal evidence of the value of non-specialist approaches comes from our subjective experience that the ideas behind Cognitive Dimensions (Green & Petre, 1996), which are expressed in a relatively informal way and are not reliant on a deep understanding of cognitive science, have been taken up by others (e.g. Lavery, 1996; Roast & Siddiqi, 1997) much more rapidly than those behind task-oriented approaches that are concerned with detailed behavioural predictions, such as GOMS (John & Kieras, 1996), CCT (Kieras and Polson, 1985) or PUM. However, we recognise that there is a trade-off between the power of a technique and the skill required of the analyst, which has to be addressed.

Aims of this work

The aim of this work is to develop and test a technique for usability evaluation that brings together many of the central ideas behind existing approaches, and that supports reasoning about various aspects of usability of a system. We can draw an analogy with task analysis. Within task analysis, complementary approaches have been developed, some focusing on procedural representations (e.g. Hierarchical Task Analysis (HTA; Annett & Duncan, 1967) or the goal-oriented sub-structure of Task Knowledge Structures (TKS; Johnson, 1992)) and others on declarative (e.g. the taxonomic sub-structure of TKS (Johnson, 1992)). Similarly, we see OSMs as a declarative approach to user-centred evaluation that can complement established procedural approaches such as GOMS (op. cit.) or Cognitive Walkthrough (Wharton *et al*, 1994). In particular, some recent work on domain modelling has focused on the use of explicit ontologies (e.g. Guarino, 1997) — a theme that we take up in our work on OSMs.

To have any long-term value, any approach to usability evaluation clearly needs, itself, to be both useful and usable.

A modelling technique is *useful* if it helps the analyst gain insights into usability issues that might well have been missed otherwise. No modelling technique can guarantee to identify problems, as it is a tool that depends on the analyst for effective use, but guidelines on how to reason with a model should help the analyst to identify particular usability concerns. Also, any modelling technique is of limited scope; what is needed is not a universal tool, but one whose usefulness is clearly scoped.

A modelling technique is *usable* if it is easy to learn to use and to understand. For any technique to be usable, the notation needs to have a clear syntax and semantics, and an accompanying methodology.

OSMs: ontology and methodology

Clearly, an approach intended to be usable by non-specialists cannot come anywhere near the depth and richness of representation afforded by full-scale knowledge representation languages. An OSM is therefore a sketchy description of the system at a fairly high level of abstraction. Usability issues are assessed by applying heuristics to the sketch. Construction and reasoning are typically iterative activities; in particular, it can be hard to identify an appropriate level of abstraction initially, so ‘construction’ can involve the production of gradually more abstract sketchy descriptions.

An OSM description covers three aspects of a system design, namely entities, actions, and relationships. For current purposes, we take brief examples from MS Word (version 5 for the Macintosh) to exemplify the modelling approach.

Entities

The analyst lists the 'things' that the user has to know about, such as ‘character’, ‘word’, ‘paragraph’, ‘column-width’. For each of these entities, the analyst notes down any of the following that are significant:

attributes

what attributes an entity has. For example, a character in a document has a font, a size, etc.

accessibility

whether the entity is *user-private* – an idea in the head of the user, such as the plot of a novel; *device-private* – something the user can't change easily, maybe not even see, but has to know about, such as the style hierarchy in Word; or *shared*, such as a word (in MS Word). User-private entities are typically components of a domain model that are not explicitly represented within the device; they may be things that users will want to manipulate, but which they cannot work with easily using this device.

relevance

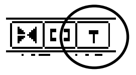
whether the entity is relevant to the domain of application, or just to the device. For example, a word is relevant to the domain of writing and typesetting (*domain-relevant*), but a scroll-bar is only relevant to the word-processor device (*device-relevant*). A user who is familiar with the domain of application will have to learn about the entities that are only device-relevant to be able to use this particular system effectively.

persistent visibility

whether an entity can be seen by the user. For example, a word is *persistent* (always visible), whereas the offspring of a style in the style hierarchy are *invisible* to the user. The symbols for newline, tab etc. that can be revealed in MS Word (if you know how) are *transient.*

disguise

whether an entity has a *meaningful* name, or a meaningful symbol, or an icon that would let you guess its meaning if you didn't already know; or whether it is disguised. In MS Word, the tool for changing column widths in tables is *disguised,* and new users have a lot of trouble finding it sometimes. (The icon originates in the markings on typewriters, now unfamiliar to today's users.)



Actions

The analyst lists the types of action users can do, such as selecting something (a paragraph, a word, etc.), and what the actions can be applied to. It is not necessary to list all the different actions, but to focus on the interesting actions that require new ideas, or that require users to know about some special tool or some other entity (such as the column width tool illustrated above). An action description consists of:

action

what the user does

entity

what they do it to

effect

what effect that action has

context

anything particular that has to be true about the state of the device for that action to have that effect

Relationships

Relationships connect two or more entities. We recognise three particular kinds of relationships: *consists-of, affects,* and *constrains.* In Word, a paragraph *consists of* sentences, sentences *consist of* words, words *consist of* characters, etc. Changing the style attribute of one of those entities will affect everything it consists of (by inheritance).

An affects relationship connects entities A and B when changing an attribute of A causes (or may cause) a change to an attribute of B, but they are not connected by consists-of. For example, the pagination algorithm in Word is such that adding to the amount of text preceding a table in a document may result in causing a new page to be started within the table (number of characters *affects* position of page-breaks).

A constraint relationship simply asserts that certain attributes of entities are in some way constrained; for instance, typographical convention requires all the lines of a table to lie on the same page if possible. We have made little exploration of constraint relationships as yet.

For each relationship, the analyst should record

type
consists-of, affects, constrains, or another kind

entities
the two (or more) entities that are related.

Reasoning with the representation

The purpose of writing out the OSM is to help the analyst to spot potential usability difficulties. While writing the OSM, or afterwards, the analyst can check for potential difficulties. These include:

- Is there a simple mapping between the underlying model and the surface presentation? If not, is this likely to cause the user difficulties?
- Are there actions that users can perform that cause irreversible changes to the system?
- How easy is it for the user to modify the things that matter to the user? [The analyst should focus particularly on how easy it is to change attributes or relations concerning user-private entities that have no direct device representation.] This may indicate “repetition viscosity” (Green 1990), in which changing an attribute that has a simple meaning to the user involves repetitive device actions.
- Is the real-world meaning of entities / attributes in the device representation always clear?
- Are there non-obvious dependencies between the context and the effect of a user action? (Does the same action have substantially different effects depending on the context? Will this always be obvious to the user?)
- Are there non-obvious relationships between the entities the user manipulates and the effects they have on other entities? (E.g. in a word-processor adding or deleting text can affect where page-breaks fall) This may indicate “knock-on viscosity” (Green 1990), meaning that changing one attribute may lead to the need to adjust other things to restore the values of others.
- Are there actions / entities that the user is unlikely to discover for any reason?
- Are there actions whose effects are unpredictable to the user for any reason?

Tool Support for Reasoning and Evaluation

We have been impressed by the potential offered by approaches that are explorable, executable or interactive, such as Monk’s Action Simulator (Monk, n.d.), which presents a simple ‘exploratorium’ for Olsen et al.’s (1995) propositional production system approach: for non-specialists, such environments may turn out to be far more usable than paper-based formalisms. Moreover, since an interactive environment requires a precisely specified semantics, some of the problems attending informal approaches are resolved. We are therefore intent on creating a demonstration version of an OSM environment. This has a long way to go and will not be further described here, but preliminary work has shown that it is easy to transcribe the OSM into Prolog assertions and then to use simple heuristics to automatically identify possible usability problems of repetition viscosity, knock-on viscosity, etc.

An Illustration: J-Sketch

To illustrate the use of the OSM notation, we apply it to a prototype sketching program, J-Sketch (Rieman, 1996). In brief, J-Sketch allows the user to produce sketchy drawings based on Bezier curves (See figure 1). The user has a choice of two methods of creating curves; in “click” mode, the user lays down individual points, and the most recent 4 points are used by the program to construct each line in the curve; in “drag” mode, the user drags the mouse to form the curve, and points are laid down automatically at 4 per second. The user has a choice of three colours (black, grey and white) and 3 pen-widths. There are 5 drawing layers; the layers can be re-ordered by clicking on tabs at the edge of each layer, and each layer can be made visible or invisible by clicking on the grey tab-edges.

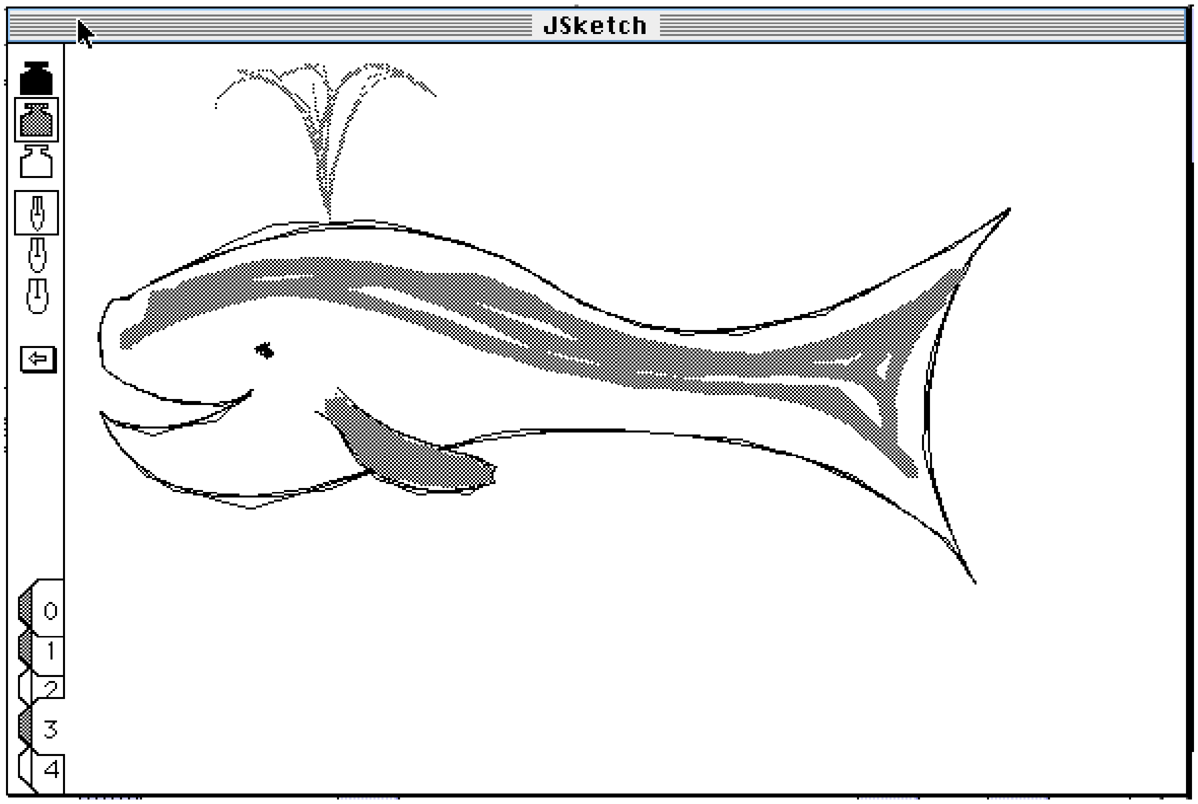


Figure 1: A J-Sketch screen.
Layer 3 is currently active; any marks on layers 0 & 1 will be visible, while those on layers 2 and 4 will not.

J-Sketch is distributed with a page of user notes. These have been used as the basis for constructing the OSM description.

J-Sketch Entities

The main entities and their properties are listed in the following table:

Entity	Attribute	Accessibility	Relevance	Persistence	Disguise
domain-object	shape	user	domain	persistent	
sketchy-line	shape, sketchiness, colour and width	shared	domain	persistent	
layer	current/other; visible/invisible	shared	device	persistent	tabs not very salient; tab edges disguised
mode	discrete / continuous	shared	device	transient	
point	position	device	device	invisible	
pen	width and colour	shared	domain	persistent	
undo icon		shared	device	persistent	disguised

The main points to note here are that the domain-objects (the “things” the user is sketching a picture of — e.g. the whale, eye, tail, fin, mouth, etc. in figure 1) have no explicit device representation; conversely, the individual points are not accessible to the user.

J-Sketch Actions

The main actions available to the user are listed in the following table:

Action	Object	Effect	Context	Notes
click	drawing area	lay down a point for a sketchy-line	discrete mode	
drag	drawing area	lay down shape for sketchy-line	continuous mode	speed of dragging affects sketchiness
press onscreen-backspace / select menu-undo / press keyboard-delete		undoes the most recent point	Works repeatedly through current curve (but cannot undo to a previous curve).	3 alternative actions all achieve the same effect
hide-layer		hides content of layer		layer re-appears if user starts drawing in it
clear-layer		deletes all drawing from layer		semantically confusable with “hide layer”.
select-pen		sets width of lines for next curve; starts a new curve		
select-ink		sets colour of lines for next curve; starts a new curve		
lift mouse button		finish-drag-curve	drawing in drag mode	
press space-bar, or click on pen or ink		finish-point-curve	drawing in click mode	no visible cue

The principal usability concerns in relation to actions are noted in the final column of this table. One additional point to note is that there is no facility to “redo” or to reverse the effect of clearing a layer.

J-Sketch Relationships

The final stage of constructing the OSM is to list important relations. In this case, the program is very simple, so we have only identified a few relations:

Type	Entity	Entity	Notes
consists-of	sketchy-line	points	(at least 4 to be visible)
consists-of	domain-object	sketchy-lines	
is on	curve	layer	can be difficult for users to find out this relationship if they forget it.

As discussed below, this set of relationships omits one important one — that a sketchy-line can consist of (or denote) several domain objects, as well as the converse being true.

Usability Notes

Overall, the process of constructing the OSM description has shown that in most respects the system is likely to be easy to use. In noting usability concerns of this prototype system, some issues arise from the process of generating the description; others arise from reasoning about aspects of it, such as the consequences of relationships holding. The main usability concerns identified through constructing this OSM description were as follows:

- We originally included a point as an entity when we tried to describe the effect of the undo action (which is that it undoes the last point). Undoing is meaningful (if limited) in click mode, but not at all in drag mode. This highlights the inappropriateness of “undo” in relation to the entities the user is working with (particularly in drag mode).
- There are three ways to undo, including pressing an on-screen button and selecting an item from a pull-down menu. This might cause confusion, since there is only one way to achieve most other effects.
- Since 4 points are needed to compute a line, the first three clicks in click mode have no visible effect. This might be disconcerting for the novice user.
- Curves are on layers. This relationship is always known to the device, but it has to be remembered by the user. It can only be found out by hiding then revealing a layer, to see what disappears and reappears.
- The user has to be aware of what mode the system is in; clicking in drag mode just makes lots of invisible points, while dragging in click mode only lays down the point at which the user presses the mouse button down. This system behaviour is likely to be confusing if the user ever changes mode.
- The drag action has property “speed” which affects the sketchiness of the curve (by affecting the spacing between points). Users need to understand this relationship if they wish to control the sketchiness of the curve.
- The tabs for making layers visible / invisible are not obvious to a novice user.

This OSM description and set of usability concerns was made just on the basis of reading the notes supplied with the program, and using the program sufficiently (usually interleaved with writing the description) to be able to produce an adequate description. (That is: the requirements of writing the description in OSM terms raised questions about how the program works, which could only be answered by running it and finding the answers. In a design context this would involve asking focused questions of the designer.) Below, we compare the results of that sketch-modelling with the results of an empirical study of the use of J-Sketch.

Testing OSMs for usability and usefulness

As outlined above, our aim in this work is to produce a technique that is both useful and usable. As a preliminary investigation into both of these concerns, we conducted an informal study, focusing on curve drawing, in which we assessed the usefulness and usability of OSMs.

Curve-drawing programs (J-Sketch and the curve-drawing facilities within ClarisWorks) were selected for this study for several reasons. One is that such programs are widely used, but comparatively few usability analyses of them have been conducted; another is that task-oriented approaches such as GOMS are poorly suited to analysing the usability of such systems.

The study, described below, was conducted with a group of 20 final-year undergraduate students who were enrolled on a module on HCI and Graphics. The results presented here are preliminary, and are being used to help highlight design modifications needed to the OSM approach.

Assessing usefulness

To assess the usefulness of the OSM approach, we did an OSM-based usability analysis of each of the software packages we were using, and compared our results against empirical data. The analysis of J-Sketch is reported above; space does not allow the ClarisWorks analysis to be presented here.

Students were put into pairs, matched as closely as possible for prior experience. Each pair was allocated to one of the two drawing programs, and while partner A worked, thinking aloud, partner B noted what A did and what difficulties were encountered. Audio and video recordings were taken of two pairs (one using each program). Students were asked to draw a whale (pictures were provided as a guide), then to modify it by moving its tail; the session lasted for 1.5 hours. The following week, the same procedure was repeated, with each pair of students using the other program. However, this session was preceded by a seminar in which students were asked to describe the program they had used to a pair of student who had used the other program, so that each pair approached the second program with more prior knowledge.

Every statement in the students' notes was categorised in terms of difficulties encountered and things they found easily. From this, an overall pattern of difficulties was established, to be compared against the OSM-based prediction.

In summary, for J-Sketch the OSM explicitly captured nearly all the usability problems. The exceptions are as follows:

1) Although "the domain object" was identified as a user-private entity, the analysis did not extend to consider ways the user might manipulate the entity — for example, by filling it (to make the whale grey). Subjects actually found filling tedious and annoying (since filling with a sketchy line causes the line to go outside boundary, like a small child's colouring in.)

2) The fact that there were 3 ways to undo and only 1 to clear was expected to cause confusion. This does not appear to have been the case. Most students understood the use of the "<=" on-screen undo button; few discovered the use of the delete key for undoing.

The results for ClarisWorks are less clear-cut. Of the nine specific difficulties encountered by subjects, 4 had been predicted, but 5 had not. Additional predictions were made about aspects of the program that subjects never got as far as exploring. Some of the unpredicted difficulties were ones that an OSM analysis would not be expected to highlight (e.g. students commented that they had difficulty getting the shape acceptable — a point that might emerge from several aspects of the system being difficult to work with, but not one that would emerge directly from an OSM analysis). Others were ones that could in principle have been detected, but were not. For example, some users tried to select part of a device object (e.g. the tail or the mouth, which have domain significance) and manipulate just that part. They could not do this; this difficulty did not emerge from the OSM analysis because it was not noted that a device object could represent several domain objects. Another example is that many students were confused about the effect of dragging the cursor in the drawing space (rather than clicking); this was a case where the effect of dragging was not documented in the user manual, so it was not included in the OSM.

These examples of failure to use OSM to predict usability problems that "in principle" it could have been used for serve to emphasise the value of expertise — both in the use of the system being analysed (in this case, ClarisWorks) and in the use of OSMs. As the technique is still under development, performance is likely to improve in future. (Nevertheless it would be unrealistic to expect 100% accuracy from any technique that is deliberately sketchy.)

Assessing usability

To assess the usability of OSMs in a broad sense, we asked the same 20 subjects to produce their own OSM descriptions, according to the following timetable:

session	duration	activity
1	1 hour	training session outlining the OSM approach and working through a brief example (based on MS Word),
2	1.5 hours	produce OSM descriptions of J-Sketch
3 (1 week later)	0.5 hour	short de-briefing on their J-Sketch analyses
4	2 hours	produce OSM descriptions of ClarisWorks
in their own time		produce a short usability report on the OSM methodology

Preliminary analysis indicates that:

· when describing J-Sketch all students except one described appropriate things as entities. (One student included some action descriptions, possibly because they appear as labels within pull-down menus.) However, most subjects only described shared entities, omitting either user-private or device-private ones. When describing ClarisWorks (after de-briefing on their J-Sketch analyses), some students did include user-private entities. All ClarisWorks descriptions were of appropriate entities, though some were at too fine a grain of detail to be really useful.

· similarly, all students (with one exception for J-Sketch) described actions in appropriate ways. For ClarisWorks , many of these were again at too fine a level of detail.

· the descriptions of relationships were highly variable, and many were inappropriate. In particular, many were reiterations of action information ("doing X affects entity Y").

· the subjects' usability notes did not clearly derive from their OSM descriptions, but from their experience. For example they included statements like "filling is tedious".

Their subsequent usability reports on OSM confirm the evidence from the data — that entities and actions were easily comprehended and described, but that relationships presented more difficulties, and that few of the subjects really understood how the modelling was meant to be used to derive usability assessments of the system.

The most important points that arise from this formative study are clear feedback on how training materials need to be developed:

· emphasising the ontology of the domain (from a user perspective — e.g. noting the significance of sentences in a document, or domain-objects in a drawing program) as well as that of the device,

· clarifying how to describe relationships appropriately,

· emphasising the ways in which the modelling is used to inform usability assessment, and

· clarifying how to establish an appropriate level of abstraction in the description.

These initial results are promising, since they represent modelling activity after a very short period of training and practice, and involve the production of models for systems that subjects had limited experience of. Further work aims to develop a fuller understanding of the optimal degree of granularity for our purposes, better teaching materials, and a demonstration of an interactive modelling environment.

Discussion

We have presented a novel approach to usability assessment that draws on ideas from ontology-based domain modelling, from PUMA knowledge analysis, and from ERMIA, and which has been motivated by aims of encapsulating a way of thinking about usability of a system, including capturing important cognitive dimensions. Initial studies indicate that the approach can be useful and usable. However, there is much further work to be done, including refining the method, developing more comprehensive training materials, scoping the approach (in terms of the kinds of systems it can be applied to, and the kinds of usability issues it can help raise), and developing tool support. This latter will help to clarify also to what extent usability insights result from the craft skill of the analyst, and to what extent they can be derived from the modelling.

Acknowledgement

This work has been supported by a grant from the Faculty of Technology at Middlesex University. We are grateful to Juliet Congreve and the students on her HCI and Graphics course for their co-operation and participation in the empirical study.

References

ANNETT, J. & DUNCAN, K.D. (1967) Task analysis and training design. *Occupational Psychology*. 41. 211-221.

BELLOTTI, V. (1989) 'Implications of current design practice for the use of HCI --in D. Jones & R. Winder (Eds.) *People and Computers IV, Proceedings of HCI'89,* 13-34. Cambridge University Press

BELLOTTI, V., BLANDFORD, A., DUKE, D., MACLEAN, A., MAY, J. & NIGAY, L. (1996) Controlling accessibility in computer mediated communications: a systematic analysis of the design space. *HCI Journal.* 11.4 pp.357-432.

BLANDFORD, A., BUCKINGHAM SHUM, S. & YOUNG, R.M. (in preparation) 'Training software engineers in a novel usability evaluation technique'.

BLANDFORD, A. E. & YOUNG, R. M. (1996) Specifying user knowledge for the design of interactive systems. *Software Engineering Journal.* 11.6, 323-333.

BUCKINGHAM SHUM, S. & HAMMOND, N. (1994) 'Transferring HCI modelling and design techniques to practitioners: A framework and empirical work' in G. Cockton, S.W. Draper & G. Weir (Eds.) *People and Computers IX, Proceedings of HCI'94,* 21-36. Cambridge University Press

GREEN, T. R. G. (1990) The cognitive dimension of viscosity: a sticky problem for HCI. In D. Diaper, D. Gilmore, G. Cockton and B. Shackel (Eds.) *Human-Computer Interaction – INTERACT '90.* Elsevier.

GREEN, T. R. G. & BENYON, D. (1996) The skull beneath the skin: entity-relationship models of information artifacts. *International Journal of Human-Computer Studies,* 44(6) 801-828

GREEN, T. R. G. & PETRE, M. (1996) Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *J. Visual Languages and Computing,* 7, 131-174.

GUARINO, N. (1997) Understanding, building and using ontologies. *International Journal of Human-Computer Studies* 46, 293-310.

JOHN, B. & KIERAS, D. E. (1996). Using GOMS for user interface design and evaluation: which technique? *ACM ToCHI.*1-30

JOHNSON, P. (1992) *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering.* London: McGraw-Hill.

JOHNSON-LAIRD, P.N. (1981) *Mental Models.* Cambridge: Cambridge University Press.

KIERAS, D. & POLSON, P. An approach to the formal analysis of user complexity. *Int. J. Man-Machine Studies,* 22 (1985), 356-394.

LAVERY, D. (1996) Specialising design principles and cognitive walkthroughs for software visualisations, In A. Blandford and H. Thimbleby (Eds.) *HCI 96 Adjunct Proceedings.*

MONK, A. F. (no date): The Action Simulator package. Available by ftp via URL `http://www.york.ac.uk/~am1/ftpable.html`

NIELSEN, J. (1994) Heuristic Evaluation. In J. Nielsen & R. Mack (Eds.), *Usability Inspection Methods* (pp. 25-62). New York: John Wiley.

NORMAN, D. (1986). Cognitive Engineering. in NORMAN, D.A. AND DRAPER, J.W., Eds. *User Centered System Design,* 31-62 Hillsdale NJ: Lawrence Erlbaum.

OLSEN, D., MONK, A.F. & CURRY, M.B. (1995) Algorithms for automatic dialogue analysis using propositional production systems. *Human Computer Interaction,* 10, 39-78.

ROAST, C.R. & SIDDIQI, J. (1997) Formally assessing software modifiability. In C.R. Roast & J. Siddiqi (Eds.) *Formal Aspects of the Human Computer Interface.* London: Springer.

SMITH, S. L & MOSIER, J. N. (1986) *Guidelines for Designing User Interface Software.* Mitre Corporation Report MTR-9420, Mitre Corporation.

WHARTON, C., RIEMAN, J., LEWIS, C., & POLSON, P. (1994). The cognitive walkthrough method: A practitioner's guide. In J. Nielsen & R. Mack (Eds.), *Usability inspection methods* (pp. 105-140). New York: John Wiley.

YOUNG, R. M., GREEN, T. R. G., & SIMON, T. (1989) Programmable user models for predictive evaluation of interface designs. In *Proceedings of CHI '89.* ACM, New York.