PUMA Footprints: linking theory and craft skill in usability evaluation

Ann Blandford, Richard Butterworth & Paul Curzon

Interaction Design Centre, School of Computing Science, Middlesex University, Bounds Green Road, London N11 2NQ, UK

A.Blandford@mdx.ac.uk

Abstract: 'Footprints' are marks or features of a design that alert the analyst to the possible existence of usability difficulties caused by violations of design principles. PUMA Footprints make an explicit link between the theory underlying a Programmable User Model and the design principles that can be derived from that theory. While principles are widely presented as being intuitively obvious, it is desirable that they should have a theoretical basis. However, working directly with theory tends to be time-consuming, and demands a high level of skill. PUMA footprints offer a theory-based justification for various usability principles, with guidelines on detecting violations of those principles.

Keywords: PUM, cognitive modelling, design principles, guidelines, usability evaluation, craft skill

1 Introduction

User modelling has had a small but important place in Human-Computer Interaction over many years, providing approaches that support rigorous, user-centred reasoning about user behaviour with interactive systems. Examples of such approaches include GOMS (Card, Moran & Newell, 1983), Cognitive Walkthrough (Wharton et al., 1994), Cognitive Reliability and Error Analysis Method (Hollnagel, 1998), Cognitive Work Analysis (Vicente, 1999) and Programmable User Modelling (PUM: Young, Green & Simon, 1989). There are arguments and counter-arguments about the costs and benefits of cognitive modelling in design. For example, advocates of GOMS, which has retained its explicit link with the underlying theory, make a point of arguing that the benefits of modelling outweigh the costs (e.g. Gray et al., 1993; John & Kieras, 1996). In contrast, the developments in Cognitive Walkthrough over time (Polson & Lewis, 1990; Lewis & Polson, 1991; Wharton et al., 1992; Wharton et al., 1994) can be interpreted as a process of seeking a balance between ease of learning, ease of application, and depth of understanding obtained through application. However, May and Barnard (1995) criticise Cognitive Walkthrough as having lost touch with its underlying theory, and hence of having lost its authority. Coming from the other direction, authors such as Nielsen (1994) propose lightweight approaches to usability evaluation that are quick and easy to apply, but that have little explicit link to any relevant theory. The work of Connell and Hammond (1999) indicates that heuristics are sufficient for identifying surface difficulties, but that more theoretically grounded usability principles enable expert evaluators to identify deeper difficulties with a design (though novices have difficulty applying principles effectively).

The tensions between theoretical grounding and ease of application have been experienced in the development of Programmable User Modelling from the early aspirations (Young et al., 1989) through the development of prototype tool support (Blandford & Young, 1993) and studies of learnability (Blandford, Buckingham Shum & Young, 1998) to work on lightweight PUM Analysis (Good & Blandford, 1999). In this paper, we provide a collection of (lightweight) usability principles and inspectable justifications for them based on a simple (theory-based) representation of the user as a rational problem solver. With this, we present simple

descriptions of the features of a design that indicate violations of the principles.

These principles share much in common with the principles presented by Dix et al. (1998) and Shneidernman's (1998) 'Golden Rules'. Indeed, some of the principles are identical, the difference being that we give a theory-based derivation for them while previous authors have tended to present them without particular justification beyond their 'obviousness'. The principles presented here can be used to identify 'PUMA footprints' in design. 'Footprints' are the marks in the design that alert the analyst to particular potential usability difficulties. The PUMA footprints do not encapsulate all possible errors, but those that are a consequence of breakdowns in the user's knowledge-based interactive behaviour, which is what PUM modelling focuses on.

2 PUM Analysis: an Overview

PUMA works from the position that the user is a rational agent, as discussed more fully by Butterworth & Blandford (1999). In particular, the problem-solving of the modelled user is based on mini-planning (Young et al., 1989).

In principle, a PUM analysis involves 'programming' a cognitive architecture that implements rational problem-solving behaviour with knowledge. This helps to identify difficulties in creating that 'user program': if it is difficult to define the user knowledge, then it is likely to be difficult for the user to acquire and apply the necessary knowledge. In addition, the resulting 'user model program' can be run with a device program to identify likely interactive behaviours, as discussed by Monk (1999). In practice, when conducting analyses of substantial designs it is more common to identify potential problem areas without conducting a full analysis, relying on a measure of craft skill; once the main problem areas are identified, then a fuller analysis may be conducted to investigate them further. PUMA footprints are a product of reflecting on that process: given an understanding of how rational users apply their knowledge in working with a design, how does that appear in a craft-based analysis? What is it that a PUM analyst is doing in the early stages of evaluating a system design? There are clearly at least two aspects to early analysis: one is gaining a deep understanding of the design; another is considering that design from a particular standpoint. Taking the PUM standpoint, the design is viewed from the perspective of users' mini-planning.

As illustrated in Figure 1, mini-planning goes, broadly speaking, through the following stages:

- Given a goal, the user identifies conceptual operations that will make progress towards the goal state. An operation comprises linked knowledge about actions, preconditions and effects, as shown in Table 1. Depending on the style of interaction, the identified operation is likely to be one of:
  - An immediately doable step (this is typical of display-based interaction); or
  - An operation that deals with the biggest difference between the current state and the goal state (e.g. most travellers will worry about booking their airline tickets before the relatively small problem of how to get to the airport when travelling abroad). This is standard means-ends analysis.
- If there are multiple candidate operations then the user has to select between them. The choice may involve additional knowledge, or it may be arbitrary (the user does not have knowledge to distinguish between the operations), in which case there is a space of possible behaviours.
- If the operation can be performed immediately, then it will be; the user has to perform the action(s) corresponding to the operation.
- If the operation cannot be performed immediately, the user adopts the preconditions as goals and aims to address them too.
- The user's knowledge of the state of the device is updated by observing visible changes, and by tracking known (predictable) changes.

The 'cycle' of rational interactive behaviour, which involves a mix of mini-planning and reacting, is more fully described elsewhere (e.g. Blandford, Buckingham Shum & Young, 1998); here we have outlined it just to derive a set of usability properties from it.

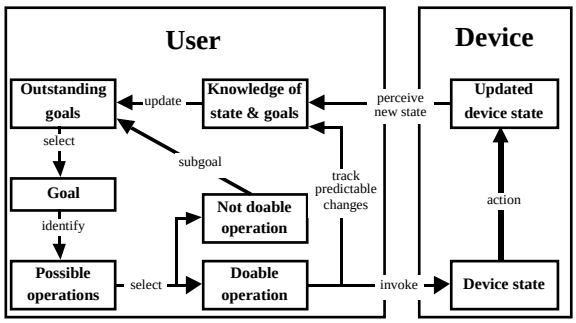

Figure 1: approximate cycle of interactive behaviour

PUM Analysis involves considering how the user exploits knowledge within the interaction, and hence what properties the device and task structure must have to support the user effectively. By

considering a range of tasks the device is intended to support, we can use this basic understanding of user cognition and user needs to identify possible breakdowns. Full PUM analysis involves laying out the user's knowledge using an 'Instruction Language' (IL). We can use this to structure our derivation of footprints and principles. Table 1 lays out the types of knowledge needed and the section of this paper in which each knowledge type is discussed. For each of these classes of knowledge, we can identify possible breakdowns that correspond to violations of design principles.

Knowledge type	Sec.
domain and device concepts the user has to, or does, work with	3.1
relationships between those concepts that the user has to understand	3.1
user's knowledge of operations, consisting of:	
the parameters of an operation (i.e. the concepts it manipulates)	3.2
the purpose of the operation (what goal does it address?)	3.3
tracked (predicted) effects	3.3
subgoaling preconditions (what does the user have to make true before this operation can be applied?)	3.4
actions to be performed to execute operation	3.5
the filtering conditions (what has to be true before this is a sensible operation to apply?)	3.6
user's initial knowledge of state	3.6
device commands and their effects, including what is displayed and the initial device state	3.6
user's task	3.7

Table 1: Summary of knowledge types.

3 Footprints

As outlined in the introduction, footprints are marks of the design that alert to the possible presence of a usability difficulty. If we start from the high-level design principle that users should be able to apply their knowledge within the interaction to achieve their goals, we can identify lower level design principles that relate to particular aspects of the goals and knowledge and hence to features of the device – for example, that features should be discoverable, and the state observable. If the device does not support the user well in applying knowledge to achieve goals in a particular way, then that is the footprint of a particular difficulty. Each of the following sections discusses one class of difficulties and their corresponding footprints.

3.1 Domain-Device Misfits

Many usability problems derive from a mismatch between the user's and the device's conceptualisation of the entities and operations available. In an ideal world, where a thorough task analysis has been completed, including an identification of all the important concepts users are working with, mismatches should pose few user problems. In practice, this is rarely achieved: the user is working with the device to achieve their domain goals, and has to be aware of device concepts and device commands as well as domain ones.

There may be essential concepts that the user has to work with to achieve their domain goals that are not clearly represented at the interface – i.e. that are not immediately visible to (and readily interpreted by) the user. This indicates a problem of discoverability. For example, the user of a drawing package may have to learn about layers, about 'handles' to manipulate objects and about how to indicate that an object is a 'special case' (e.g. a circle is a special case of an oval; a square is a special case of a rectangle). We can express this in terms of principles and footprints as follows:

Principle: Features should be discoverable.
Violation: Poor discoverability.
Footprint: Essential conceptual objects not clearly presented at the interface.

Discoverability applies mainly to device entities that have no real-world significance, where the designer's aim is to make the device learnable. However, it does not relate to the domain-relevant concepts the user is manipulating; there are often misfits between the concepts that users actually work with and the ones available at the interface. To take a simple example: the instructions for preparing a conference paper typically define the size of the text area, but the word processor may only allow the user to specify paper size (e.g. A4) and margin sizes; therefore many authors will have to get a ruler and manually calculate margin sizes, or find some other work-around, to achieve the desired domain goal.

Deeper misfits can result in problems such as viscosity (Green, 1990): the property that a simple

domain-level change may require multiple actions at the interface. For example, standard drawing packages are not well suited for drawing organisation charts that express relationships between people, as adding a new role in the organisation can involve creating space by moving many other people and links around on the page; this is typically very time-consuming unless the package has (and the user has used) an explicit representation of links between people so that these are automatically preserved.

In essence, if the user is manipulating domain concepts via this device, the mappings between them have to be clear and simple if the planning problem is to remain simple; if the relationships are unduly complex, this will cause user difficulties. In terms of the PUM Instruction Language, we can identify difficulties with the objects and relationships the user has to know about and work with. Such complex relationships are the footprint of mismatch in conceptual representation.

Principle: There should be a good fit between domain and device concepts.
Violation: Mismatch in conceptual representation.
Footprint: Conceptual objects include domain concepts that have no close device analogue; relationships are difficult to express clearly.

To work with the device, the user may have to make explicit data about the domain or device that would more naturally remain unstated. For example, the user of a database may be required to specify a maximum field size (information that is only device-relevant), while the user of an electronic personal organiser will usually have to specify an end time for every event entered even though this information may not be known at the time of entering the event. So, for example, the conceptual operation associated with creating a new event in the diary would have to include a parameter – end time – that is not important to the user.

Principle: The user should not be required to provide unnecessary information.
Violation: Enforced explicitness.
Footprint: Conceptual objects required include ones that would not naturally be considered important by the user.

3.2 Parameters to Operations and Communication Goals

If the user's task goal includes known parameters then the user will seek an opportunity within the interaction to communicate that information (Blandford & Young, 1998). For example, the amount of money to be withdrawn from an ATM, the number that calls from a telephone are to be diverted to, or the name of the person to whom an electronic mail message is to be sent are all such parameters. The user will experience difficulty if their expectations are not satisfied: if they cannot find the point at which to communicate the identified information, if they identify an apparently appropriate point that is actually incorrect, or if they are required to enter information that seems irrelevant to them. In addition to the mismatches in conceptual representation discussed in section 3.1, we identify a principle concerning such communication goals.

Principle: The user can easily identify the point at which to communicate information to computer system.
Violation: Breakdown in communication goals.
Footprint: There is a mismatch between the point in the interaction where a user would naturally communicate certain data and the point where the device demands it.

3.3 Purpose and Tracked Effects: Side Effects and Predictability

In order to maintain awareness of the state of the systems they are working with (so as to make informed choices about future actions), users need a means of updating their knowledge of the state. This may be by observing the state through the display (or other output device), or through predicting the effects of actions. Predictability depends on the user's knowledge of the current state and of the effects of operations. The effect of an action is predictable if the user knows all the factors that determine the effect, and is aware of the current state of all those factors. Thus, if all relevant state components are observable (see section 3.6) then predictability is unlikely to be a problem, but if some are not, or if the user is likely not to realise the significance of certain components relative to the action, then the device will not be predictable to that user. If user-significant aspects of the resulting state after any action are neither predictable nor observable then the device is not usable.

In many interactions, total predictability is not a requirement; for example, the fact that the user does not know what web page will be displayed when selecting a link, or that the results of submitting a database query cannot be anticipated, is not a problem: indeed, if such devices were completely predictable, they would be useless. Conversely, the user of a web form should be able to predict, and have confidence, that the form entry is being sent to

the intended destination when the 'submit' button is pressed.

Principle: The device should be predictable.
Violation: Breakdown of predictability.
Footprint: User may not have sufficient knowledge of current state of device or of effect of operation to appropriately predict effect of action.

One of the heuristics when producing an IL description is that users will generally track the main effects (the user purpose) of operations, whether or not the effect is visible. However, users are liable to miss, or forget about, side-effects unless they are very visually salient; even expert users are liable to forget about such effects occasionally (Blandford & Young, 1996). Therefore, a particular class of predictability problems is raised by side-effects. The footprint of unnoticed side effects is that there are effects of conceptual operations that are not part of the purpose of the operation and are either not visible to the user or are not particularly visually salient.

Principle: Side-effects should be avoided.
Violation: Unnoticed side effects.
Footprint: Action corresponding to a conceptual operation has effects that are not part of the user purpose, and are not visually salient.

3.4 Subgoaling Preconditions: Order Errors

Another common class of errors relates to the order of operations. For example, when using the particular word processor with which this text is being written, if a user creates a paragraph with the properties of a level two heading (as defined in the instructions for authors), then decides to specify a 'Heading2' style to look like that, all the details (12pt bold italic Times font) will be lost, whereas if the same user specifies first that the style is to be called 'Heading2' and then defines the details of the font etc., the intended effect will be achieved.

The 'footprint' of an order error is that there are circumstances in which doing *A* then *B* has a different effect from doing *B* then *A*, that the actions can be performed in either order, and that the user may not be aware of the order constraint.

Principle: The device should not provoke order errors.
Violation: Likelihood of order errors.
Footprint: There exist actions such that the effect depends on the order of application, but the user may have insufficient knowledge (of state or operations) to reliably choose the order.

Mode errors are a particular class of errors related to subgoaling, in that the user may not be aware that the device has alternative modes and that being in the correct mode is a precondition of the action achieving the intended effect. Similarly, the user may not be aware of the mode the device is currently in due to lack of observability. The footprint of a mode error is that the same device action has different effects depending on the mode – which the user may be unaware of (for whatever reason); they have been implicated in human errors in a range of situations, including aircraft accidents.

Principle: The risk of mode errors should be minimised.
Violation: Likelihood of mode errors.
Footprint: The effect of an action depends on a mode setting whose value the user may be unaware of.

3.5 Domain-Device Misfits: Actions

Just as users may have to learn about conceptual fit, so they also have to relate conceptual operations to device actions – that is, how to make domain-relevant changes using a particular device. This topic has been addressed by various researchers over the years; for example, Norman (1986) discusses the 'gulf of execution': the difficulty users may have in working out how to achieve their domain goals using a particular device; Payne & Green (1986) developed a Task Action Grammar that aimed to focus attention on consistency across a set of tasks (similar tasks should be achieved in similar ways). In PUMA terms, the question is: when the analyst specifies a conceptual operation, is it easy to define the actions that go with it?

Principle: The mapping between task and action should be simple.
Violation: Poor task-action mapping.
Footprint: It is difficult to specify the action sequence that corresponds to a conceptual operation.

One particular case of poor task-action mappings is found in label-following behaviour, where the user is expected to identify the label at the interface that corresponds to them making progress towards their goal.

Principle: All labels should be clear.
Violation: Labels may be confusable.
Footprint: Labels on actions have poor correspondence with domain-relevant conceptual operations.

3.6 Filtering Preconditions and Knowledge of State: Observability

One oft-stated requirement on a device (e.g. Dix *et al*, 1998) is that the state should be observable, without reference to the purpose of the interaction. A PUM analysis says that if the user has particular domain goals, and achieving those goals involves manipulating (or otherwise being aware of) particular concepts, then the state of the device as represented through those concepts should be observable. Therefore, a system fails the observability criterion if there are essential aspects of the state of the device (or domain) that are not observable at the time when they are needed. For example, users often experience difficulties with drawing packages if they need to manipulate objects on layers without being able to inspect those layers. More importantly, operators of safety-critical devices may have difficulty diagnosing faults if essential components of the system state are not directly accessible to them (e.g. Reason, 1990).

In the short term, there are cases where observability is not essential: the user may be able to predict the effects of actions without observing the state change (e.g. copying text to a hidden buffer; sending a document to a remote printer). However, users may be interrupted or be distracted from their work; the observability requirement dictates that non-observable goal-related state components should be easily restored or inspected on task resumption.

Conversely, there are cases where observability – or at least immediate feedback – is particularly important. For example, the user entering a password should not be able to see the characters entered, but may need to know that they have entered the correct number of characters. Norman (1986) discusses this in terms of the 'gulf of evaluation': that the user must be able to evaluate the current state of the system with respect to their goals.

Principle: Device state should be observable.
Violation: Breakdown of observability.
Footprint: Device does not display the current settings of all state components that the user needs to know to make an informed choice of operation, or to identify when a goal has been achieved.

3.7 Termination Errors

There are various sources of what Thimbleby (1990) terms 'termination errors' – referred to generally as errors that involve the user considering a task to be completed before it actually is. Some of these errors are knowledge-based and others are a consequence of the user's cognitive architecture.

Post-completion errors, which are persistent but intermittent errors that appear to derive from the user's cognitive architecture (Byrne & Bovair, 1997) can be viewed as arising because of a 'trailing subgoal'. That is: the main goal of the interaction can only be achieved by satisfying some precondition (subgoal), which in turn perturbs the state in some way (e.g. there is now an original on the photocopier glass, or a card in the ATM), and when the main goal has been completed the user may terminate the interaction without correcting the perturbation. In terms of PUMA footprints, post-completion errors are unusual, in that they appear to depend on features of the cognitive architecture that go beyond the simple rational problem-solver. However, they can be derived from the representation of the task goal (which leaves under-specified which other aspects of the initial state may be perturbed from their original values, and should be restored).

Principle: Avoid post-completion errors.
Violation: Design may provoke post-completion errors.
Footprint: There is a precondition to the conceptual operation that achieves the main goal, but satisfying the precondition perturbs the state, and a clean-up action is needed after achievement of the main goal.

Post-completion errors are the result of one class of 'implicit' goal (i.e. the total goal is not stated quite precisely). Other classes result from simplified representations of goals that result in error-prone approximations. Such simplified representations are typically provoked by the device design. For example Butterworth, Blandford & Duke (in press) describe the design of an electronic diary that allows the user to enter a regular series of events; the device representation of an event series can provoke the user into focusing on getting all the events from a paper diary entered into the electronic diary without noticing that this results in surplus 'ghost' events being entered (for instance, if the meetings are monthly but there is no meeting in August).

Principle: Avoid goal confusion.
Violation: Device may provoke incorrect task formulation.
Footprint: Precise statement of task goal is complex but device supports a similar task goal that can be clearly expressed and can be confused with task goal.

Incorrect termination can also be a result of a breakdown in predictability or observability, as discussed above.

4. Discussion

In section 2, we presented the knowledge components required for PUM analysis, and used that as a basis for identifying classes of usability problems that derive from breakdowns in the user's knowledge. In section 3, we laid out and discussed a list of design principles, relating them to knowledge requirements derived from PUM theory. We have also related them to the work of others so as to present an integrated set of principles that derive from user knowledge; few of the principles are new and none are surprising or inconsistent with past work in the area. For example, Dix et al. (1998) discuss observability and predictability, while Shneiderman's (1998) Golden Rules include 'Offer informative feedback'. Here, they have been brought together and presented in terms of rational user behaviour. In addition, we have presented PUMA footprints as a way of characterising the design features that denote violations of principles. While many of the principles are not new, they are explicitly justified and described in terms of a theory of human problem solving and interaction.

We have introduced the term 'footprint' to refer to the features of a design that alert the analyst to the possibility of a particular type of difficulty. Footprints are similar in spirit to Hollnagel's (1998) genotypes: Hollnagel refers to the manifestation of an erroneous action as a phenotype, and the underlying cause as the genotype; while he uses the term to refer principally to underlying cognitive causes, footprints focus more on features of design that provoke errors in the interaction. Footprints generally signify the failure to apply a corresponding design principle earlier in the design process. While there are 'common sense' understandings of many design principles, we have aimed to provide theory-based definitions of them that support reasoning.

A recent study of applying the PUMA approach within the early stages of design, working within the constraints (time and cost) imposed by the demands of the ongoing design process, led us to identify three 'knowledge questions' that could be used to guide a user-centred view on the design (Good & Blandford, 1999):

- 'What does the user need to know?',
- 'How does the user know?', and
- 'What are the consequences of the user not knowing?'

The footprints demand a deeper and more analytical approach than the three questions, but are also more directly related to the underlying theory.

The footprints do not cover all design principles. They take as their starting point a particular perspective on the design, which is assuming that the user is rational and applies their knowledge. They do not deal with the full richness of natural human behaviour; they do not, for example, deal with 'slips' (Reason, 1990), issues of interpretation of information (beyond simple semantic matching of labels to goals), or choice of colour or graphic design. Similarly, they do not aim to support reasoning about the complex interactions that are the focus of Cognitive Work Analysis (Vicente, 1999). However, they address one important set of requirements on design, relating to the user's goals and knowledge. Given this focus, footprints provide descriptions of design features that are likely to result in difficulties for rational users of a device. They thus provide a theory-grounded way that makes explicit the craft skill used in early stages of a PUM analysis. They also provide a point on the continuum of evaluation techniques in which ease of use is traded off against theoretical grounding of analysis.

Acknowledgements

This work was supported by EPSRC Grants GR/L00391 and GR/M45221. David Duke, Jason Good, Thomas Green, Sue Milner, Harold Thimbleby, Richard Young and anonymous reviewers have all helped clarify the ideas presented here.

References

Blandford, A. E., Buckingham Shum, S. & Young, R. M. (1998) Training software engineers in a novel usability evaluation technique. *International Journal of Human-Computer Studies* 45, 245-279

Blandford, A. E. & Young, R. M. (1993). Developing runnable user models: Separating the problem solving techniques from the domain knowledge. in J. Alty, D. Diaper and S. Guest, Eds. People and Computers VIII, 111-122 Cambridge: CUP.

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

Blandford, A. E. & Young, R. M. (1998) *The role of communication goals in interaction.* In Adjunct Proceedings of HCI'98.

Butterworth & Blandford (1999) The principle of rationality and models of highly interactive systems. In M. A. Sasse & C. Johnson (Eds.) *Human-Computer Interaction INTERACT'99*. 417-242. Amsterdam: IOS Press.

Butterworth, R., Blandford, A. & Duke, D. (in press) Demonstrating the cognitive plausibility of interactive system specifications. To appear in *Formal Aspects of Computing*.

Byrne, M. D. & Bovair, S. (1997) A working memory model of a common procedural error. *Cognitive Science*. 21.1, 31-61.

Card, S. K., Moran, T. P. and Newell, A. (1983). *The Psychology of Human Computer Interaction*, Hillsdale : Lawrence Erlbaum.

Connell, I. W. & Hammond, N. V. (1999) Comparing Usability Evaluation Principles with Heuristics: Problem Instances vs. Problem Types. In M. A. Sasse & C. Johnson (Eds.) *Human-Computer Interaction INTERACT'99*. 621-629. Amsterdam: IOS Press.

Dix, A., Finlay, J., Abowd, G. & Beale, R. (1998) *Human Computer Interaction*. Prentice Hall International. 2nd Edition.

Good, J. P. & Blandford, A. E. (1999) Incorporating Human Factors Concerns into the Design and Safety Engineering of Complex Control Systems. In J. Noyes & M. Bransby (Eds.) *People in Control*, IEE Conference Publication Number 463, IEE, London, 51 - 56.

Gray, W., John, B & Atwood, M. (1993) 'Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance', *Human-Computer Interaction*, 8. 237-309.

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.

Hollnagel, E. (1998). *Cognitive Reliability and Error Analysis Method (CREAM)*. Oxford : Elsevier.

John, B. & Kieras, D. E. (1996). Using GOMS for User Interface Design and Evaluation? Which Technique. *ACM ToCHI*, 3.4, 287-319.

Lewis, C. & Polson, P.G. (1991). Cognitive Walkthroughs: A method for theory based evaluation of user interfaces. *Tutorial presented at ACM CHI '91*.

May, J., and Barnard, P.J (1995) Towards supportive evaluation during design. *Interacting with Computer*s, 7, 115-143.

Monk, A. (1999) Modelling cyclic interaction. *Behaviour and Information Technology*. 18. 127-139.

Nielsen, J. (1994) Heuristic Evaluation. In J. Nielsen & R. Mack (Eds.), *Usability Inspection Methods*. 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.

Payne, S. J. and Green, T.R.G. (1986). Task-Action Grammars: a model of mental representation of task languages. *Human-Computer Interaction*, 2, 93-133.

Polson, P. & Lewis, C. (1990) Theory based design for easily learned interfaces. *Human Computer Interaction*, 5, 191-220.

Reason, J. (1990) *Human Error*. Cambridge : Cambridge University Press.

Shneiderman, B. (1998) Designing the User Interface : Strategies for Effective Human-Computer Interaction Addison Wesley Publishing Company

Thimbleby, H. (1990) *User Interface Design*, ACM Press Frontier Series, Addison-Wesley.

Vicente, K. (1999) *Cognitive Work Analysis*. Mahwah, NJ : Lawrence Erlbaum.

Wharton, C., Bradford, J., Jeffries, R. and Franzke, M. (1992). Applying cognitive walkthroughs to more complex user interfaces: experiences, issues and recommendations. In *Proc. CHI'92*, 381-388.

Wharton, C., Rieman, J., Lewis, C. and Polson, P. (1994) The Cognitive Walkthrough method: a practitioner's guide. In J. Nielsen and R. Mack, Eds. *Usability Inspection Methods*.105-140. Wiley : New York.

Young, R.M., Green, T.R.G. & Simon, T. (1989) 'Programmable user models for predictive evaluation of interface designs' in Bice, K. and Lewis, C. (eds.) *Proceedings of CHI '89*, 15-19, New York : ACM.