PUMA
PROGRAMMABLE USER MODELLING APPLICATIONS

PUMA Footprints: linking theory and craft skill in usability evaluation

A. Blandford
2000
WP26

Principal Investigator
Prof Ann Blandford
Middlesex University

Research Fellow
Dr Richard Butterworth
Middlesex University

Academic Collaborator
Dr David Duke
University of York

Research Fellow
Jason Good
Middlesex University

Industrial Collaborator
Sue Milner
Praxis Critical Systems Ltd

Academic Collaborator
Prof Richard Young
University of Hertfordshire

http://www.cs.mdx.ac.uk/puma/

Contact: Dr Ann Blandford
School of Computing Science
Middlesex University
Bounds Green Road
London, N11 2NQ, UK

tel: +44 (0)181 362 6163
fax: +44 (0)181 362 6411
email: a.blandford@mdx.ac.uk

Project funded by
EPSRC grant number GR/L00391
PUMA Footprints: linking theory and craft skill in usability evaluation

Ann Blandford

Abstract

‘Footprints’ are marks or features of a design that alert the analyst to the possible existence of usability difficulties caused by violation of design principles. PUMA Footprints make an explicit link between the cognitive 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, cognitive theory tends to be presented in the form of techniques to be applied directly in design, which are time-consuming and demand a high level of skill. Footprints offer a way of balancing the costs and benefits of theory-based modelling with those of a heuristic approach to usability evaluation.

1 Introduction

Cognitive 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, Cognitive Walkthrough and Programmable User Modelling (PUM). 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 an explicit point (e.g. Gray et al., 1993; John & Kieras, 1996) of arguing that the benefits of modelling outweigh the costs. In contrast, the developments in Cognitive Walkthrough over time (Polson & Lewis, 1990; Lewis et al., 1990; Lewis & Polson, 1991; Rieman et al., 1991; Polson et al., 1992; 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. The same tensions 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 et al., 1998a) to work on lightweight PUM Analysis (PUMA; Good & Blandford, 1999).
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 that we have a particular theory-based 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 theoretical standpoint.

In this paper, we derive a set of ‘usability principles’ from the PUMA method, thereby offering a way of obtaining many of the insights of conducting PUM analysis without generating full models. These principles share much in common with the principles presented by Dix et al. (1998) and Shneiderman’s (1998) ‘Golden Rules’. Indeed, some of the principles are identical, the difference being that we give a particular derivation for them while previous authors have tended to present them without particular justification beyond their ‘obviousness’. They are also very similar in spirit to the ‘Cognitive Dimensions’ of Green (1989), in that they derive from a particular theoretical position about cognition. These principles can be used to identify ‘PUMA footprints’ in design. That is: by understanding the principles, and their definitions, it becomes relatively straightforward to spot potential difficulties in a 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 planning behaviour, which is what PUM modelling focuses on.

2 PUMA Theory: a brief 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).

Mini-planning goes, broadly speaking, through the following stages:
1. Given a goal, the user identifies conceptual operations that will take her nearer to the goal state. Depending on the style of interaction, this is likely to be one of:
   • An immediately doable step (this is typical of display-based interaction);
   or
   • Dealing with the biggest difference between the current state and the goal state and identifying appropriate operations to achieve that (big) goal (e.g.
most travellers will worry about booking their airline tickets before the relatively small problem of how to get to the correct airport when travelling abroad). This is standard means-ends analysis.

2. If there are multiple candidate operations then the user has to select between them. The choice may be arbitrary (the user does not have knowledge to distinguish between the operations), in which case there is a space of possible behaviours, or it may involve additional knowledge.

3. If the operation can be performed immediately, then it will be; the user has to perform the action(s) corresponding to the operation.

4. If the operation cannot be performed immediately, the user will adopt the preconditions as goals and aim to address them too.

5. The user updates their knowledge of the state of the device 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 (Blandford et al., 1998a; Blandford et al., 1998b; Blandford & Good, 1999); here we have outlined it just to derive a set of usability properties from it.

PUM Analysis involves considering how the user exploits their knowledge to form plans 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, the IL terms for those types, and the section of this paper in which each knowledge type is discussed.
Knowledge type | IL term | Section
--- | --- | ---
domain and device concepts the user has to, or does, work with | Object | 3.1
relationships between those concepts that the user has to understand | Relationship | 3.1

user’s knowledge of operations, consisting of:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>the parameters of an operation (i.e. the concepts it manipulates)</td>
<td>Arguments</td>
<td>3.2</td>
</tr>
<tr>
<td>the purpose of the operation (what goal does it address?)</td>
<td>User purpose</td>
<td>3.3</td>
</tr>
<tr>
<td>the filtering conditions (what has to be true before this is a sensible operation to apply?)</td>
<td>Filtering precondition</td>
<td>3.4</td>
</tr>
<tr>
<td>subgoaling preconditions (what does the user have to make true before this operation can be applied?)</td>
<td>Subgoaling precondition</td>
<td>3.5</td>
</tr>
<tr>
<td>tracked (predicted) effects</td>
<td>Tracked effect</td>
<td>3.3</td>
</tr>
<tr>
<td>actions to be performed to execute operation</td>
<td>Device action</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Table 1: summary of Instruction Language

For each of these classes of knowledge, we can identify possible breakdowns that correspond to violations of design principles.

3 Footprints

As outlined in the introduction, footprints are marks of the design that alert to the possible presence of a usability difficulty. In this section, we relate various classes of difficulties to the user’s knowledge and the device representation. Each of the following sections discusses one general class of footprints; a following section summarises the footprints and their corresponding usability difficulties (or violations of design principles).

3.1 Objects and relationships: 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 (e.g. Johnson, 1992), including an identification of all the important concepts users are working with, mismatches should not pose 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.
Norman (1986) discusses the importance of fit between the user’s conceptualisation of the device and the designer’s; he places this in the context of mental models, and argues that it is a responsibility of the designer to help the user to acquire an appropriate mental model of the device, by representing it in a suitable way at the interface. This approach is appropriate for device entities that have no real-world significance, where the designer’s aim is to make the device learnable. However, it does not go far enough for the domain-relevant concepts the user is manipulating; there are frequently 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 this paper state that the text area is to be 118mm horizontally by 192mm vertically, but the word processor being used only allows the user to specify paper size (e.g. A4) and margin sizes; therefore many authors submitting to this conference will have to get a ruler and manually calculate margin sizes, or find some other work-around, to achieve the desired domain goal. In terms of the PUMA Instruction Language, we can identify difficulties with the objects and relationships the user has to know about and work with.

There may be important domain concepts the user has to work with that are not directly represented in the device, so that the user has to manipulate them indirectly; this results in a mismatch in conceptual representation. As well as the kinds of difficulty sketched out in the page size example presented above, these can also lead to 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, which 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.

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. Again, we can find examples in the domain of drawing: 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).

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. Green (1989) considers such problems of enforced explicitness under the heading of ‘premature commitment’ – that the user has to specify information earlier than is natural; in practice, it can
require the user knowing about and defining relationships that they might choose never to define explicitly.

In practice, identifying domain concepts and relationships may involve the application of knowledge-acquisition techniques; the matching between domain and device is the subject of more detailed work in ongoing work on misfit analysis (Blandford & Green, 1998). In essence, if the user is (effectively) 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.

### 3.2 Arguments to operations and communication goals

Blandford and Young (1998) discuss one specific idea about communication between user and device: that if the user’s task goal includes some parameters – e.g. 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 – then the user will seek an opportunity within the interaction to communicate that information. 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 (see discussion on mismatch in conceptual representation above).

### 3.3 Tracked effects and user purpose: side effects and predictability

In order to maintain awareness of the state of the systems they are working with (in order 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 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 below) 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 the resulting state after any action is neither predictable nor observable then the device is not usable.

In data-centred interactions, total predictability is often 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 that the form entry is being sent to the intended destination when the ‘submit’ button is pressed.
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. Thus, if there are effects of conceptual operations that are not part of the purpose of the operation and are not visible to the user, they are likely to be ignored, resulting in side effects, which may have knock-on consequences.

3.4 Filtering preconditions and recognising the goal 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.

3.5 Subgoaling preconditions: order errors

Another common class of errors relate to the order of operations. For example, when using a traditional fixed telephone, a user has to lift the receiver before dialling the number; this sequence will not generally work for a mobile phone (as the analogous actions have to be conceptualised differently: as specifying a number and making a connection). Similarly, 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.

Mode errors are a particular class of errors, related to order errors in that the user may either not be aware that the device has alternative modes, or may not be aware of the mode the device is currently in due to lack of observability (see below). Mode errors occur when 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.

3.6 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); Kieras and Polson (1985) proposed the idea of ‘cognitive complexity’ in relating domain and device tasks; Moran (1982) addressed the quality of fit between domain and device tasks though his ‘External Task – Internal Task’ (ETIT) analysis; and Payne et al. (1990) developed the same idea in terms of ‘Yoked State Spaces’. In PUMA terms, the question is: when the analyst specifies a conceptual operation, is it easy to define the actions that go with it?

When the user is aiming to achieve some small thing in the domain, there may be cases where the corresponding device actions are unclear. That is: the user has to learn non-obvious task-action mappings. For example, a certain mobile telephone has a complex menu structure that the user has to navigate to set various parameters on the ’phone. However, setting the keyboard lock (so that the user cannot dial a number by accidentally pressing the ’phone keys while it is in their pocket) involves pressing a non-obvious key sequence without any support from the device, and does not involve the menu hierarchy at all.

In terms of the things the user is trying to achieve in the domain, using a graphical user interface, a particular difficulty is caused by unclear menu labels. Blandford et al. (1997) discuss the example of a mobile telephone menu hierarchy that offers options ‘call related features’, ‘messages’ and ‘phone setup’: a novice user may have difficulty identifying which option to select for tasks such as ‘setting up the ’phone to divert all incoming messages or calls to another number’. This issue is also addressed explicitly in the application of Cognitive Walkthrough.
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 are derived (within the Instruction Language outlined in section 2) 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 which should be restored).

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 (submitted) 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).

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 PUM Instruction Language, and used that as a basis for identifying classes of usability problems that derive from different possible failure points relative to the user’s knowledge. In section 3, we have laid out and discussed a list of design principles, relating them to the PUMA approach to usability evaluation. We have also sought to relate them to the work of others; by doing so, we have shown that 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’; PUMA footprints offer a link between the theory and the possible usability problems, as summarised in Table 2.
Table 2: relating footprints to design principles

<table>
<thead>
<tr>
<th>Footprint</th>
<th>Usability alert (violation of design principle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual objects include domain concepts that have no close device analogue; relationships are difficult to express clearly</td>
<td>Mismatch in conceptual representation</td>
</tr>
<tr>
<td>Essential conceptual objects not clearly presented at the interface</td>
<td>Poor discoverability</td>
</tr>
<tr>
<td>Conceptual objects required include ones that would not naturally be considered important by the user</td>
<td>Enforced explicitness</td>
</tr>
<tr>
<td>Mismatch between the point in the interaction where a user would naturally communicate certain data and the point where the device demands it</td>
<td>Breakdown in communication goals</td>
</tr>
<tr>
<td>User may not have sufficient knowledge of current state of device or of effect of operation to appropriately predict effect of action.</td>
<td>Breakdown of predictability</td>
</tr>
<tr>
<td>Action corresponding to a conceptual operation has effects that are not part of the user purpose, and are not visually salient</td>
<td>Unnoticed side effects</td>
</tr>
<tr>
<td>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</td>
<td>Breakdown of observability</td>
</tr>
<tr>
<td>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 desired order</td>
<td>Likelihood of order errors</td>
</tr>
<tr>
<td>Effect of action depends on a mode setting whose value the user may be unaware of</td>
<td>Likelihood of mode errors</td>
</tr>
<tr>
<td>It is difficult to specify the action sequence that corresponds to a conceptual operation</td>
<td>Poor task-action mapping</td>
</tr>
<tr>
<td>Labels on actions have poor correspondence with domain-relevant conceptual operations</td>
<td>Labels may be confusable</td>
</tr>
<tr>
<td>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.</td>
<td>Design may provoke post-completion errors</td>
</tr>
<tr>
<td>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.</td>
<td>Device may provoke incorrect task formulation</td>
</tr>
</tbody>
</table>

The footprints do not form an orthogonal set, but are interdependent, drawn from the same theoretical base. While the principles, and their corresponding footprints, are neither new nor surprising, 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
generally signify the failure to apply a corresponding design principle earlier in design. While the terms have a ‘common sense’ meaning, we have aimed to provide theory-based definitions of them that support reasoning.

We believe that this provides an account of the origins of ‘craft skill’ in PUM Analysis. In our previous work, we have typically worked in one of two ways. While developing theory, we have generated examples that tested that theory and then worked them through in detail. When we have chosen examples to test theory, it has sometimes been suggested to us that we were proving something we already knew the answer to, the implicit suggestion being that we were somehow cheating: identifying problems first then formally proving that the problems that we already knew about were really there. This approach can be regarded as working from the underlying theory to the footprint, and then generating an example that has that footprint for illustration or testing purposes.

Conversely, when analysing substantial designs, we have generally done a ‘craft-based’ analysis and then focused on particular problems, once identified, for more detailed modelling. In this case, we would identify the ‘footprints’ that are predicted by the underlying modelling theory, and then do a relatively superficial analysis of the design, spotting footprints. More formal modelling is then appropriate if it is likely to yield additional results. In this case, we work from footprints to more theory-based modelling.

Footprints provide one approach to making cognitive modelling accessible within design. 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. In addition, like design principles, they should be intuitive to reason with.

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 what Reason (1990) terms ‘slips’, issues of interpretation of information (beyond the simple level of semantic matching of labels to goals), choice of colour or graphic design, etc. Rather, they focus on aspects of design that relate to the user’s goals and knowledge. Given this focus, they provide explicit support for identifying difficulties the rational user is likely to experience when working with a device.
5 Acknowledgements

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

6 References


