



The University of Sydney

**Customised Hypertext as an
Individualised Learning Environment**

Technical Report Number 498

June, 1995

Judy Kay and R J Kummerfeld

ISBN 0 86758 978 7

**Basser Department of Computer Science
University of Sydney NSW 2006**

Customised hypertext as an individualised learning environment

J Kay and R J Kummerfeld
Department of Computer Science
University of Sydney

ABSTRACT

This paper describes a mechanism for providing hypertext documents that accommodate individual learner's different needs. The customisation of the text can take account of a variety of factors, including the learner's background, interests and preferred learning style.

Central to the customisation is the *user model* that represents the learner's preferences and knowledge. We describe the way that it is used to create individualised hypertext spaces for the learner and the issues involved in eliciting and acquiring the user modelling knowledge.

We describe the way we have implemented this and applied it to customising a hypertext course for the C programming language.

1. Introduction

An important use for hypertext is in the delivery of teaching material. Many computer aided learning systems use hypertext in one form or another allowing the learner to pick their own path through the material, request further elaboration or descriptions of terms (glossaries). In general, however, there is only one description of each point and only one explanation of each term. If these don't suit the learner, for reasons of background, preknowledge or learning style, then the hypertext will not be as effective as it could be.

Current hypertext systems can give considerable support for individual learner's needs. For example, the provision of a glossary enables those unfamiliar with terms to follow the links to the glossary explanations. However, they require the hypertext author to decide on a *typical* learner audience and then to write the text of the glossary so that it is a reasonable match to the needs of the expected audience, with differences accommodated by allowing the learner to select the links to hypertext pages that interest them.

Much of the appeal of hypertext is in the fact that it allows the learner flexibility in what they choose to explore and read in the hyperspace. However, in a hypertext that allows complex forms of customisation this is inadequate. For example, in a hypertext that

describes the programming language C, a learner who likes mathematical examples could be offered a set of links to mathematical examples of each concept. This poses problems in that either the learner must repetitively find and select the mathematical example for each aspect or the hypertext author must duplicate the common text in what are essentially separate hypertexts for each class of examples available.

This approach may get very complex and unwieldy if even a few learning styles are accommodated. In the case of exercises designed to test understanding of the work it may be very difficult to guide the learner to the one they are most suited to since it may depend on many factors in their pre-knowledge. Even simple variations in language level may become tedious for the learner to select at each point.

Our approach is to extend hypertext, customising it on the basis of a *user model*. The material presented to the learner is constructed dynamically from hypertext that is augmented to indicate text to be presented depending on elements of a model of the learner.

We currently construct the model largely from a dialogue with the learner before the beginning of the learning task.

There is a growing move towards helping the learner find what they want in a hyperspace. For example, Encarnacion and Boyle (Encarnacion 1991, Boyle 1994) and Lithgo (Lithgo 1991) customised Unix hypertext manual documentation on the basis of a user model, with the learner's knowledge defining what is presented in each manual entry.

At the same time, there is the recognition that different learners need different classes of text. For example, Paris (Paris 1985, 1988, 1989) found the need to provide quite different explanations of concepts for learners with different levels of expertise.

Another important influence on our work is the trend towards cooperative teaching systems, such as advocated by Cumming and Self (Cumming 1989) . There is also a growing number of researchers exploring the creation of systems that allow a range of teaching strategies, for example (London 1992) .

We support a co-operative model of interaction between the user and the system in two main ways. Firstly, we ensure that the way the customisation operates can be made available to the user. Secondly, we take in care in defining the customisation to allow the user to control the customisation by their answers to questions.

2. User models to customise hypertext

A user or student model is information that enables the system to model the learner. In teaching systems that attempt to customise their interaction to match the individual, or where the system has been designed to match models of learning, this 'user model' is critical. This term (and 'student model') are used to refer to three main levels of model:

1. the domain representation level;
2. representations of important learner groups and
3. the individual model of this learner, in terms of the above two.

In a hypertext for teaching, the user model should contain the learner's preknowledge and preferences.

Here the relevant domain representation (model level 1) defines the aspects of the system's knowledge of the domain that will be used in modelling the user. It makes sense to model only what can be used by the system (Self 1992) to manage the interaction. It will also commonly represent the meta-domain of the system that teaches. This would include aspects like the styles of examples that the system can offer.

The user model at level 2 could usefully define some stereotypes over this domain. For example, it is common to define a model for the 'typical novice' and the 'typical expert' and possibly some other intermediate classes of users. Each of these would be intended to capture a useful set of assumptions about each class of users. So, for example, the 'typical novice' would be represented as knowing some things and not others and having preferences for particular styles of explanation, favouring the concrete over the abstract.

The third level of the user model is used to represent an individual. This models the user's membership of various stereotype groups as well as their individual differences from these.

As an example, the authors have constructed a hypertext course for the C programming language. Part of a level-1 user model in this system is shown in Table 1.

Knowledge

competent in Pascal
understands run time structures
knows about functions
knows about function arguments
understands call-by-value
understands call-by-reference
understands call-by-name

Preferences

abstract or concrete
terse or more detailed
active or directed
likes mathematical examples
likes Unix system examples
likes simple text-based example

TABLE 1. Example of learner information for customising C text

The level-3 representation of the user will associate values with these. Some come from

an initial interview phase where the user completes a simple form on their background and preferences. Some is inferred, using the level-1 model's representation of the structure of the domain. For example, if a learner is a competent Pascal programmer this means that they know about the concepts, call-by-value and call-by-reference. Since this is due to the domain structure, it is an inference based on the level-1 part of the model. Indeed, this type of inference enables the system to conclude that the competent Pascal programmer has a conceptual knowledge of the major common elements of Pascal and C.

The level-2 parts of the model enable additional but weaker inferences. For example, a learner who is skilled in Unix might be presumed to like 'abstract' and 'terse' descriptions since this is common among those who cope well with Unix (and its documentation). Although this is plausible, it is merely a reasonable starting assumption that is true of many but not all users in this class.

The tools for determining this and for storing the model are part of a toolkit for user modelling (Kay 1990, Cook 1994, Kay 1994) . Essentially, they use a range of sources to collect information. They also enable the learner to inspect the user model and to contribute information to it.

The information in the user model is used to customise the hypertext that is presented to the learner. So, for example, a learner who knows Pascal well, likes terse, abstract explanations that use jargon as appropriate will navigate a hypertext in this form.

3. Architecture

In our system, customised hypertext pages are constructed by an intermediary program that is invoked when a learner selects a link. This program consults the user model and, guided by what it finds, constructs a customised text of the target hypertext page to suit the learner.

In our first implementation we have used World Wide Web tools (server and browser) and the C preprocessor. Our customisation program uses a user modelling toolkit to take preferences from the user model and add them as "#define" statements to the raw hypertext. The raw HTML text contains "#if" statements to indicate sections to be included depending on the preferences. The C preprocessor is then used to deliver the

final form of the hypertext page.

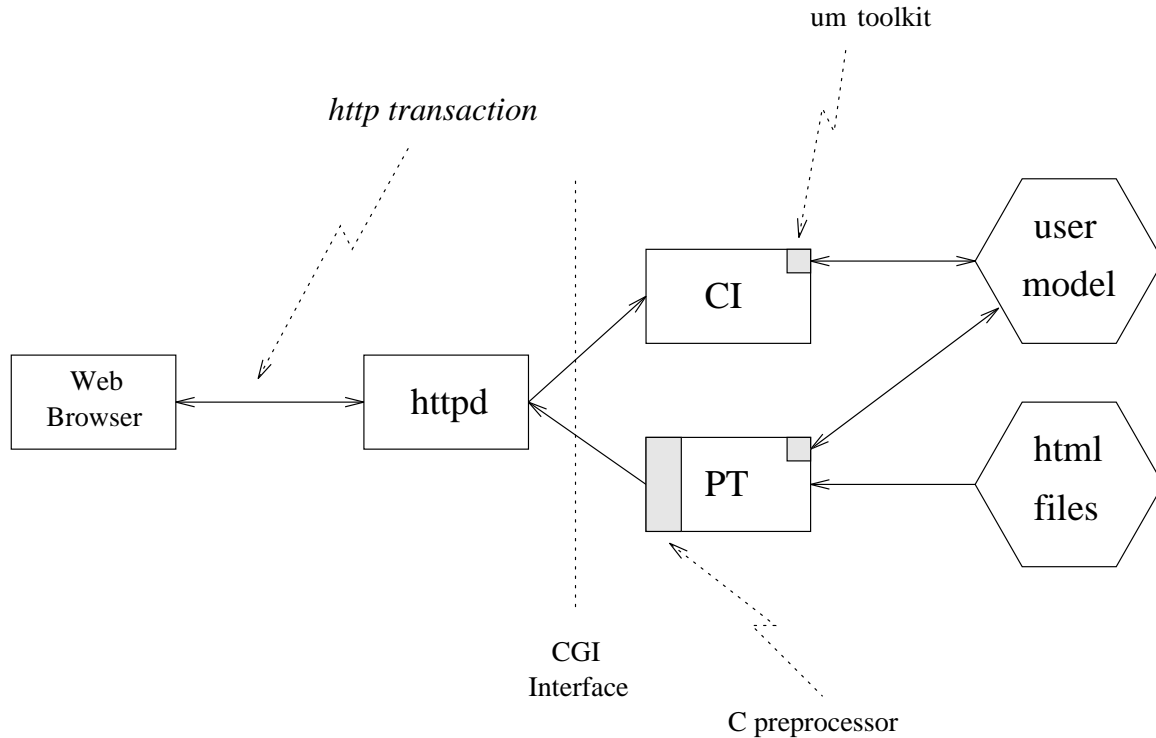


Figure 1. Architecture of system

4. Discussion

Here we discuss the ways that our approach differs from the possibilities offered by standard hypertext systems. We also discuss the problems that arise in applying our approach. From the learner's point of view, there is the difficulty of maintaining the sense of control that is associated with normal hypertext and the potential problems of document instability. From the system's point of view, the problems are in constructing the user model and then in managing the increased complexity of a meta-hypertext.

4.1 Customisation without repetition

The primary benefit of user modeling for adapting hypertext is that it permits far greater customisation without additional complexity in the hyperspace the learner sees. Our system operates on a large *meta-hyperspace* of the full range of customisability. Yet each learner should see a quite manageable hyperspace.

Figure 2 shows the way a current hypertext system can allow for learner's preferences among three classes of examples. At the left (A) is the hypertext space that allows the learner the choice of three classes of examples: mathematical, text oriented and systems oriented. The dark nodes across the top introduce the concept being explained. All are substantially the same, with small differences to account for the different examples that each will use. The next row of nodes is for the actual example used. These nodes are shown as lighter because they are quite different from each other. The next row of nodes

is for the discussion of the examples. Here, too, there is considerable commonality in all three since. In each case, the examples are being used to illustrate the same concept. Of course, there will also be considerable differences too because the each text discusses the concept in terms of different nodes. The rest of Figure 2A shows how each of the three possible paths continue to pass through nodes: darker nodes at the same level indicating texts that do the task described at the left with much in common and the lighter nodes showing text that is very different.

The major problem with this approach is that the author of the hypertext needs to manage several texts that are very similar. Any corrections may need to be made over each parallel text. From the learner's point of view, this approach partitions the hypertext, with the learner deciding at the top of the Figure which partition they will follow.

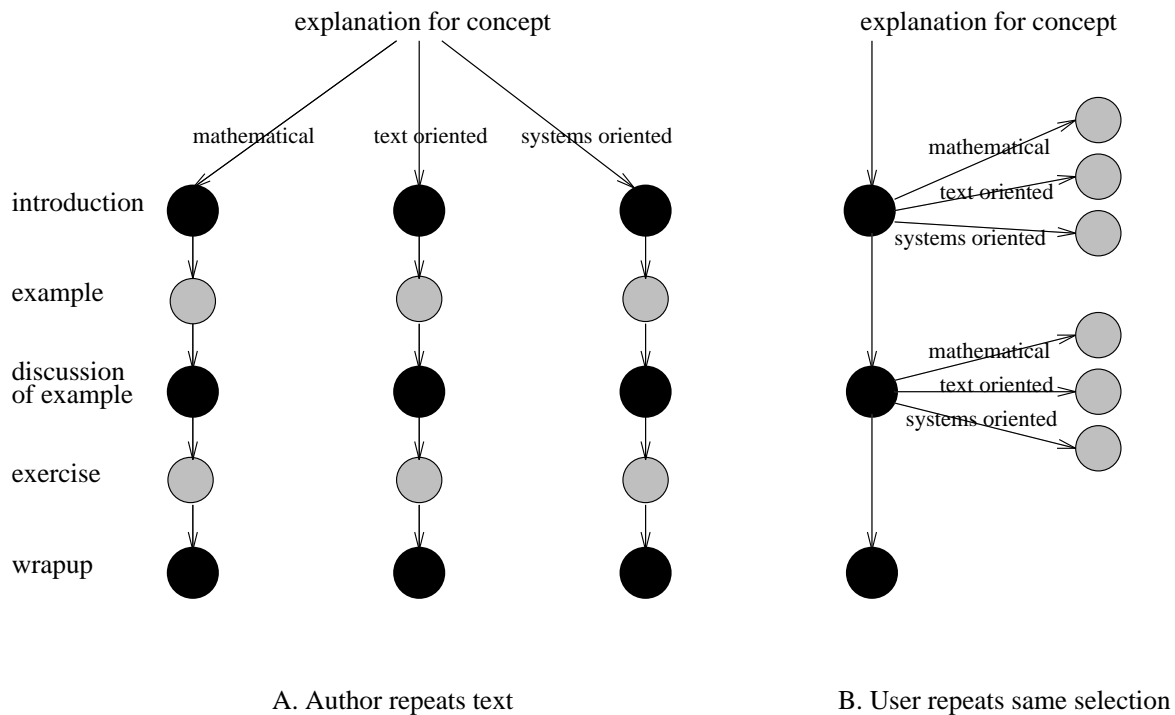


Figure 2. Standard hypertext allows for individual preferences

The right hand part of the figure (B) shows an alternate approach that avoids the almost identical repetitive nodes in the hypertext. Here the common aspects of the explanation are separated from those that are specific to the example. Now all learners see the same *introduction* node and then they must select the node for the type of example they prefer. Similarly for the *discussion* and *wrapup*. Now the repetition is with the learner. If they prefer the mathematical examples, they must select the *mathematical example* node every time they want an example of the concept. This approach also forces identical text for the introduction, discussion and wrapup nodes: any of these that are specific to the examples must be in their nodes, potentially causing near-repetition.

Figure 3 shows the way that our system deals with alternate explanations.

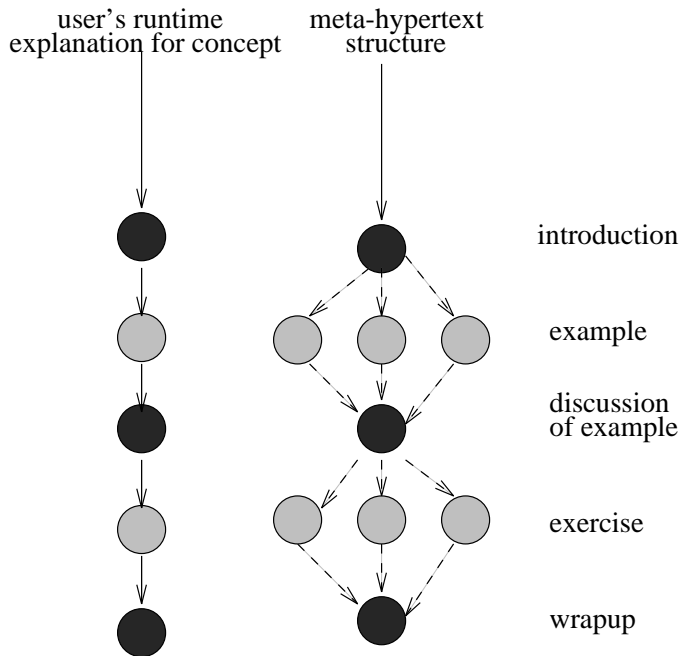


Figure 3. Metahypertext structure and the learner's hypertext

The path at the left depicts the user's actual hyperspace. At the right is the meta-hyperspace from which it came. The dark nodes are the elements that are common to all the hyperspaces generated for different users. The lighter nodes hold the alternate examples.

There are two important differences between the hyperspace at the left of Figure 3 and those in Figure 2. Most obvious is that in Figure 3 the user no longer needs to decide which example type they will select because this has been done by the system.

Less obvious is that we no longer have to limit customisation to the hypertext node level. Since we generate the hypertext, we can allow customisation at arbitrarily detailed levels. For example, we can now add a few words relating to the particular example in the 'wrapup'.

We can also extend the customisation. The example of Figures 2 and 3 illustrates customising exclusively in terms of three classes of examples. We can also provide customisation based on aspects like the learner's background and preferred style of explanation.

4.2 Document stability

One of the potential problems with our approach is that the learner could become disoriented and irritated if the revisited text were very different each time they come to it.

As present we have a number of strategies for dealing with this. None are enforced by

the system: they require judgement and discipline on the part of the hypertext author. We need several strategies because the different classes of customisation can bring different problems.

Consider first the effects of different learner's backgrounds. Our current approach is that as learners progress through their learning of C, the material presented converges to the form of the printed text (Kay 1989). This is intended for reference use by a reader who understands the underlying concepts but needs to check a detail or be reminded of some aspect of the language. Tutorial level material is still be accessible, but deeper in the hyperspace.

Note that in our context, there are many details that readers would need to check on starting a new class of task. For example, this is particularly common for the many library functions: a learner may never use certain groups of these for long periods and then need to do a burst of work with them. They may forget details like the order of the arguments in a standard function. We consider that the learner then needs descriptions of these, with examples and notes about potential pitfalls and common errors. This type of information tends to have less customisation of its presentation level and very quickly converges to the reference form.

For customisations of example types, the learner would not expect instability. This means that we cannot apply a convergence approach. Effectively, this type of customisation has the effect of defining sets of parallel books. We deal with this by applying a somewhat different convergence principle: the printed version, hence the goal of the convergence, tends to have small highly focussed examples compared with the teaching materials. The principle is that the learner actually understands the principles and is using this for reference so it is less critical to match their preferred example style.

Teaching style also has an interesting effect in the long term. For example, the active learner is provided with information in small steps, with the learner encouraged to guess or work out as much as possible. Once the learner has passed through such a sequence to complete the path to a concept, they are presented with direct information thereafter. It would be irritating to go through the other process repeatedly.

4.3 Learner Control

In all this the learner can control the customisation by altering their user model. If, for example, the model indicates the learner really knows a concept, like *call by value*, it will not make that a glossary button and explanations will be cast on this assumption. The learner can alter the user model and get alternate text.

The viewable (and adjustable) user model permits straightforward control at that level. If the learner wishes to explore the impact of their user model on hyperspace, they can do so by tinkering with the model and exploring the resultant hyperspace.

We are currently exploring the use of labels on the buttons to indicate the user model values that have caused the generation of links.

4.4 Building the model

At present we build the user model very simply: initially, we ask the learner a set of simple questions about themselves and then we use this to infer additional information.

We plan to provide several other ways to collect useful modelling information. Essentially, these come from three sources: the learner's actions as they move through their hyperspace; analysis of the learner's performance on exercises; and by allowing the learner to volunteer additional information to improve the customisation.

One important class of indirect user modelling information can come from the learner's comparative assessment of different forms of customisation. This is particularly useful for establishing the language style that the learner prefers. For example, it is typically easy for a learner to read two versions of an explanation and say which they prefer. If we try to determine this in other ways, the learner may need to be familiar with what we mean by terms such as "abstract" and "concrete".

Another important way to set values in the user model come from the learner's actions in traversing their hyperspace. So, for example, the learner can be presented with a screen that offers the choice indicated at the right of Figure 2: the learner may decide to select the mathematical example. If there are several points where the learner selects mathematical examples over other types, this can be used to infer that this learner may prefer mathematical examples. This approach permits us to avoid asking the user too many questions at the very start of their interaction.

We have explored several forms of customisation. We have done this in the context of the introductory 'chapter'. This is where the different backgrounds that learners bring are most significant. It is also a critical part of the hyperbook because its goal is that the learner acquire a useful subset of the language that serves as foundation: by the end of this section, the learner should be able to write useful programs and to start dipping into other parts of the language as needed.

One class of paths supports learning by active programming. The learner is presented a series of programming tasks. Each has links to potentially helpful information that the user can choose to view. There are example programs that do similar tasks and act as models. There are also pieces of expository text about relevant aspects of the language.

Another class of paths operates in terms of whether or not the learner is familiar with Pascal. If they are, they see notes about similarities and differences between Pascal and C, with particular warnings about common pitfalls for Pascal programmers. In addition, those aspects that are very similar in both languages are described very briefly in terms of the Pascal construct.

We have also explored more interactive approaches to the translation of book into hypertext. In one class of paths through the introductory material, we construct the text as a series of small inference tasks for the user. For example, rather than simply tell the user about the language's block structure, with braces, declarations and statements, we

present a small program and ask them to indicate features within it. So, for example, they are asked to click on the declarations. As each feature of the program has been made a link to an appropriate next node, clicking on the correct place moves to the next part of the hypertext. Clicking on the wrong place moves to a text explaining which the declarations are and the role of the feature that the learner selected.

Combining these with different classes of examples and degrees of abstractness in the descriptions makes for a large meta-hyperspace.

4.5 Complexity of the metahypertext

One of the problems with our current system is that it is even more complex for the author to manage than ordinary hypertext. This is a serious problem. We currently deal with it by systematic naming of the pages. In the longer term, we would like interface and management tools that aid the author in constructing and previewing the various forms of the text.

5. Conclusions

As the above description indicates, there is a merging of intelligent teaching systems and hypertext teaching materials. The user model that generates a customised hypertext is critical in building this bridge for customised communication.

Essentially, we create the pages and links for the individual learner dynamically as the learner interacts with the hyperspace. We call the whole of our system a *metahyperspace* since we actually represent many potential hyperspaces, each defined by the particular details of the learner's model.

Our approach remains quite close to current hypertext systems in that the metahypertext author must create each of the possible pieces of text that are presented to the learner. A more *intelligent* hypertext for C would be based on a knowledge base about C and the generation of an individualised hypertext.

The approach we have taken is predicated by the particular class of task we are concerned with. Learning programming is a complex task and as authors of a book, we feel that a considerable part of our contribution is in the exposition we provide in describing and explaining the various concepts, skills and other knowledge that we aim to communicate.

As authors, we find that our meta-hypertext is both a burden and a blessing compared with writing a conventional text. Clearly, the burden derives from the additional work of providing several versions of some parts of the text.

Less obvious is the blessing that comes with just this. When writing a conventional text or hypertext, we continually had to make compromises. For example, we had to decide when to add additional background for readers who may not have it. This is easier in a hypertext system as the background material can be moved to a separate page and made

available to the learner who selects it. However, the learner will only select it *if* they are aware they lack this background. In the tailored hypertext, a link leading to background is labelled to show the learner that it provides this background that their model indicates they lack.

The most satisfying elements for us were the many small additional notes that we wanted to offer to selected audiences. One important example of this is for learners who know Pascal. Since we have had considerable experience in teaching C to people who know Pascal, we know many of the pitfalls for them. These are clearly relevant and useful information to offer them. Equally, they are distracting clutter for people who do not know Pascal.

Hypertext offers considerable flexibility for the learner who wants to explore a text. The system we have described augments this by customising the contents of nodes and the links between them on the basis of the individual's user model. As authors, we have found this valuable for the introduction to a new programming language. In other domains where the learning task is challenging and the user's goals, background and preferences so varied, the customised hypertext can support an individualised learning environment.

Boyle 1994

C Boyle and A O Encarnacion, *MetaDoc: an adaptive hypertext reading system*, User modeling and user adapted interaction 1 1994

Cook 1994

R Cook and J Kay, *The justified user model: a viewable, explained user model*, UM94: User Modeling Conference 1994 also Sydney Univ. CS Dept. Technical Report 483 (ISBN 0 86758 922 1)

Cumming 1989

G Cumming and J Self, *Collaborative intelligent educational systems*, Artificial intelligence and education 1989

Encarnacion 1991

A Encarnacion and C Boyle, *A user model based hypertext documentation system*, Proc IJCAI Workshop W.4: Agent Modeling for Intelligent Interaction IJCAI-91 1991

Kay 1990

J Kay, *um: a user modelling toolkit*, Second Intl User Modelling Workshop 1990

Kay 1994

J Kay, *The um toolkit for cooperative user modelling*, submitted to User Modeling and User Adapted Interaction 1994 also Sydney Univ. CS Dept. SSRG Report 94/3/36.2

Kay 1989

J Kay and R J Kummerfeld, *C in the Unix Environment*, Addison-Wesley 1989

Lithgo 1991

R Lithgo, *A hypertext user modelling Unix manual system*, Honours Thesis, Basser Dept of Computer Science, University of Sydney 1991

London 1992

R V London, *Student modeling to support multiple instructional approaches*, User

modeling and user-adapted interaction 1-2 1992

Paris 1985

C L Paris, *Description strategies for naive and expert users*, Proc 23rd Annual Meeting of the Assoc for Comp Linguistics 1985

Paris 1988

C L Paris, *Tailoring object descriptions to a user's level of expertise*, Computational Linguistics Assoc for Computational Linguistics 3 1988

Paris 1989

C L Paris, *The use of explicit user models in a generation system for tailoring answers to the user's level of expertise*, User models in dialog systems Springer-Verlag 1989

Self 1992

J Self, *Bypassing the intractable problem of student modelling*, Proceedings of the 2nd International Conference on Intelligent Tutoring Systems Springer-Verlag 1992