User models for customised hypertext

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 user’s different needs. The customisation of the text can take account of a variety of factors, including the user’s background, interests and preferred learning style.

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

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

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 user 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 user, 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 user’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 user audience and then to write the text so that it is a reasonable match to the needs of the expected audience, with some allowance for differences accommodated by allowing the user to select the links to hypertext pages that interest them.

Much of the appeal of hypertext is in the fact that it allows the user flexibility in what they choose to explore and read in the hyperspace. However, it is inadequate in a hypertext that allows complex forms of customisation. For example, in a hypertext that describes the programming language C, a user who likes mathematical examples could
be offered a set of links to mathematical examples of each concept. This poses problems in that either the user 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 user 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 user to select at each point.

Our approach is to extend hypertext, customising it on the basis of a user model. We currently construct the model largely from a dialogue with the user before the beginning of the learning task.

There is a growing move towards helping the user find what they want in a hyperspace. For example, Encarnacion and Boyle (4, 1) and Lithgo(8) customised Unix hypertext documentation on the basis of a user model, with the user’s knowledge defining what is presented in each manual entry.

At the same time, there is the recognition that different users need different classes of text. For example, Paris(10, 11, 12) found the need to provide quite different explanations of concepts for users 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(3) . There is also a growing number of researchers exploring the creation of systems that allow a range of teaching strategies, for example(9) .

2. User models to customise hypertext

A user model is a collection of information about a user. In a hypertext for teaching, the user model should contain the user’s preknowledge and preferences.

As an example, the authors have constructed a hypertext course for the C programming language. The user model in this system has information like that shown in Table 1.
Knowledge

competence in Pascal
understands run time structures
know about functions
know about function arguments
understand call by value
understand call by reference
understand call by name

Preferences

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

**TABLE 1.** Example of user information for customising C text

These preferences and other information about the user may come from the initial user model or may be inferred. For example, if a user says they are a competent Pascal programmer then it may be inferred that they understand concepts that are common to both Pascal and C.

The tools for determining this and for storing the model are part of a toolkit for user modelling(5, 2, 6). Essentially, they use a range of sources to collect information. They also enable the user 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 user. So, for example, a user who knows Pascal well, likes terse, abstract explanations that use jargon as appropriate will navigate a hypertext that is in this form.

**3. Architecture**

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

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
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 user’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 user sees. Our system operates on a large meta-hyperspace of the full range of customisability. Yet each user should see a quite manageable hyperspace.

Figure 2 shows the way a current hypertext system can allow for user’s preferences among three classes of examples. At the left (A) is the hypertext space that allows the user 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 are for the actual example used. These are shown
as lighted 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 this figure 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 will need to be made over each
parallel text. From the user’s point of view, this approach partitions the hypertext, with
the user deciding which partition they will follow.

A. Author repeats text

\begin{figure}
\centering
\includegraphics[width=\textwidth]{hypertext_diagram.png}
\caption{Standard hypertext allowances for individual preferences}
\end{figure}

B. User repeats same selection

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 users 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 user. if they
prefer the mathematical examples, they must select the mathematical example node
every time they want an example of the concept.
Figure 3 shows the way that our system deals with alternate explanations.

![Diagram of metahypertext structure and user's hypertext]

**Figure 3.** Metahypertext structure and the user’s hypertext

### 4.2 Document stability

One of the potential problems with our approach is that the user could become disoriented and irritated if revisited text were to be 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 different problems.

Consider first the effects of different user’s backgrounds. Our current approach is that as users progress through their learning of C, the material presented converges to the form of the printed text(7). This is intended for reference use by a reader who understands the underlying concepts but needs to check a detail. Tutorial level material is still be accessible deeper in the hyperspace.

Note that in our context, there are many details that readers would need to check whenever they begin doing a new class of task. This is particularly true of the many library functions: a user may never use certain groups of these for long periods and then need to do a burst of work with them. We consider that the user 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 user 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 far smaller and more focussed examples than the teaching materials. The principle is that the user actually understands the principles and is using this for reference so it is less critical to match their preferred example style. However, where an example is, for example, very mathematical, we usually include additional examples to supplement it.

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 user encouraged to guess or work out as much as possible. Once the user has passed through such a sequence to complete the path to a concept, they are presented with direct information thereafter. We consider that it would be irritating to go through the other process repeatedly.

4.3 User Control

In all this the user can control the customisation by altering their user model. If, for example, the models indicates the user 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 user can alter the user model and get alternate text.

The viewable (and adjustable) user model permits straightforward control at that level. If the user 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 show the user model values that have caused the generation of links.

4.4 Building the model

We currently build the user model very simply: initially, we ask the user 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 user’s actions as they move through their hyperspace; analysis of the user’s performance on exercises; and by allowing the user to volunteer additional information to improve the customisation.

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

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 user dynamically as the user 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 user’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 user. 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 user who selects it. However, the user 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 user that it provides this background that their model indicates they lack.
Another advantage for the author is being able to broaden the audience of readers. For example, there are C problems that are particular to Pascal programmers. Where we know that the reader is a Pascal programmer, we can generate the text with these. This reduces the number of compromises an author must make.