

The University of Sydney

**DOCUMENT SELECTION IN SITS
TECHNICAL REPORT NUMBER 532**

OCTOBER 2001

Sam Holden and Judy Kay

Document Selection in SITS

Sam Holden and Judy Kay
Basser Department of Computer Science
University of Sydney, NSW 2006
Australia
{sholden,judy}@cs.usyd.edu.au

Abstract

This technical report describes SITS a personalised teaching system. It uses its knowledge of a user to select documents which should fulfill a learning need. It does this by modelling aspects of the user's knowledge and learning preferences and combining this with both a model of the relationships between the main learning concepts in the domain and metadata about the documents available for teaching about those concepts.

We describe the overall architecture of the system. We illustrate how a single query from different users is handled by SITS to select the best document available for each user.

Keywords metadata, ITS, user modelling

1 Introduction

Producing personalised information when a user wishes to learn something is very useful. The user's existing knowledge can be used to adapt the available resources to suit them. Resources that are appropriate to the user can be selected from a large pool of resources. Individuals have individual learning preferences, some people like seeing lots of examples, others like being presented with abstract explanations - these personal preferences can be used to aid the learning process by selecting appropriate resources.

Learning is now a life long task, as things change around a person they must learn new things. People want to be able to learn independently and flexibly. Different people with different backgrounds and learning needs and preferences want to be able to locate documents suited to them from which to learn.

The world wide web provides an extremely large choice of documents from which the user can learn a new concept. The large range of available resources provides more potential for selecting resources that match the user's current knowledge and learning preferences.

One approach to this is to use some form of adaptive hypermedia [2, 9] in which the content is encoded with extra information that can be used to include or exclude sections according to some

information about the user. The main cost of this method is that creating the content, is more difficult - the information needs to be presented in multiple ways in order that the best form can be used for each user.

Another approach is to use the information about the user to select the best material from that which is available. Rather than making each resource customisable, a large number of resources are used. Some of them may be customisable themselves, but that is not necessary.

If we personalise the selection of documents we would like to ensure the user feels in control of the process. Scrutability is the ability of the system to explain to the user why it behaved the way it did. For example, the user might scrutinise the system to determine why it presented them with one resource instead of some other resource. This is an extension of the scrutable learner model seen in [10, 4, 14, 3].

Scrutability is an important element whenever the user is presented with a resource from a selection of possible resources. It provides a checking mechanism: if the user is presented with a resource that is unsuitable, they can ask the system to explain why it selected that resource. This will help determine why the system is not performing correctly. It provides a mechanism whereby the user can examine the model the system has of them in an intuitive way. It is also important from an ethical point of view: choosing resources can be viewed as a form of censorship. Some resources are presented and others are not. Scrutability forces the system to be able to justify the selection mechanism to the user.

Scrutability also provides the user with better control of the learning experience [7, 8]. Confronting the user with the reasoning behind the teaching behaviour should also support metaognition [16, 11, 17].

Scrutable Intelligent Tutoring System (SITS) is a web-based software tutor. It keeps an evidence based model of the user's knowledge of concepts related to the course being taught, and also learning preferences. Courses are constructed through a web interface which allows concepts to be entered,

Table 1: Epistemological Metadata

Relation	Description
prereq	Student must know the concept
shows	Can be used to teach the concept
uses	Knowledge of the concept useful

resources to be added, and links made between those concepts and resources.

2 SITS Overview

To illustrate our approach, we will use the following example. Suppose the user is learning to use the Standard Template Library (STL) of C++. There are two main problems the teaching system has to solve:

- Selecting the concept to teach
- Selecting the documents to show the user for the chosen concept

The system has information about the user’s history: concepts the user knows and does not know, resources the user liked learning from, resources the user succeeded in learning from, resources the user did not succeed in learning from.

Once a concept is selected to teach, the system has access to a collection of resources about that concept. It must determine how to select the most appropriate resource (or resources) for this user.

Our approach is to define metadata that can drive a scrutable teaching strategy. A fundamental goal is to produce a scrutable teaching system that can be used with modest demands on time and effort from the teacher. Reusing available content is thus one the important requirements. This means the system must adapt to suit the content it has access to. This is possible by exploiting metadata associated with the content itself. Then the addition of a new resource only requires the addition of metadata describing that resource. Importantly there is no need to update the system infrastructure with data about the concepts being taught and how they relate to each other.

2.1 Selection of Documents

Returning to our example, suppose we have determined that the user wants to learn about the ‘sort’ algorithm in the STL.

We now describe the way we associate metadata with documents. This metadata captures two categories of information: epistemological and preference.

Epistemological information about the resource is indicated with three metadata types: *prereq*, *uses*, and *shows* (see table 1). *Prereq* indicates the

document requires the user to have some *prerequisite* knowledge about a concept. *Uses* indicates the document *uses* knowledge of a concept, but it is not essential to in understanding that document. The user should be able to either infer its meaning from the context or understand the document even if they don’t understand this concept. However, some teaching strategies may avoid choosing a document which *uses* unknown concepts because the user has specified they prefer to progress slowly and feel confident at each step. Equally the teaching strategy may give preference to documents that *use* concepts that where recently taught in order to reinforce the user’s knowledge of them.

Shows indicates the document can be used as a resource for teaching that concept to the user.

These three types were carefully chosen to minimise the number of types of metadata. This should make it easier to define the metadata for a document. At the same time, they capture the relevant information well for the system to successfully select appropriate resources for presentation to users.

Preference information about the resource is indicated by a fourth type of metadata: *type*. This is used to hold information about the author of the document and the style of the document. The current system has 5 style types: *short-example*, *example*, *short-reference*, *reference*, and *tutorial*.

Short-example documents are those which consist solely of a small example of a concept. *Example* documents have an example with some explanation. *Short-reference* documents consist of reference information which is incomplete. *Reference* documents are reasonably complete reference information about a concept. *Tutorial* documents are tutorial style information describing a concept.

The *type* metadata is also used to indicate the author of the document, and other non-concept related metadata. There is no predefined list of types (aside from the style types), instead they are arbitrary strings which are matched against the users preference information.

For the purposes of this example we suppose the system has access to five documents about sort. These are shown below, as Sort Documents 1-5. They are indicative of the range of documents that the might typically system deals with. The first two are short pieces of example code which demonstrate the use of the sort algorithm. There are two extracts from books, Sort Document 3 providing short reference material best suited for revision and Sort Document 4 providing a long tutorial style guide with large code examples. Lastly there is the reference documentation for the sort algorithm itself, which provides a complete description of the usage of the sort algorithm.

For each document we show, in addition to the actual document, its associated metadata. It is

clear that even though each deals with the same concept, the STL sort, they present quite different information in quite different forms. The main differences for SITS teaching purposes are captured in the metadata as we now discuss.

- Sort Document 1 (Example Code)

```
#include <iostream>
#include <algorithm>
using namespace std;
int array[] = {1, 50, -10, 11, 42 19};
int main() {
    int count=sizeof(array)/sizeof(*array);
    ostream_iterator<int> iter(cout," ");
    cout << "before: ";
    copy(array,array+count,iter);
    cout << "\nafter: ";
    sort(array,array+count,greater<int>());
    copy(array,array+count,iter);
    cout << endl;
}
```

Metadata : prereq:arrays, uses:ostream_iterator, uses:greater, shows:sort, type:short-example.

- Sort Document 2 (Example Code)

```
#include <iostream>
#include <algorithm>
int array[6] = {1, 50, -10, 11, 42, 19};
int main() {
    std::sort(array, array+6);
    for (int i=0 ; i<6 ; ++i)
        std::cout << array[i] << ' ';
    std::cout << std::endl;
}
```

Metadata : prereq:arrays, shows:sort, type:short-example.

- Sort Document 3 (Stroustrup [15])

“The sort() algorithms require random access iterators. That is, they work best for vectors and similar containers.

```
template <typename Ran>
void sort(Ran first, Ran last);
template <typename Ran, typename Cmp>
void sort(Ran first, Ran last, Cmp cmp);

template <typename Ran>
void stable_sort(Ran first, Ran last);
template <typename Ran, typename Cmp>
void stable_sort(Ran first,Ran last,Cmp cmp);
```

The standard list does not provide random access iterators, so lists should be sorted using the specific list operations.

<Continues for another paragraph>”

Metadata : prereq:iterator, prereq:vector, shows:sort, shows:stable_sort, type:short-reference, type:stroustrup

- Sort Document 4 (Eckel [5])

“One STL container (list) has its own built-in sort() function which is almost certainly going to be faster than the generic sort presented here (especially since the list sort just swaps pointers rather than copying entire objects around)...

< Explanation continues for a few pages with 2 pages of sample code>”

Metadata : prereq:iterator, prereq:vector, prereq:deque, prereq:ifstream, prereq:string, shows:sort, shows:partial_sort, shows:nth_element, type:tutorial, type:eckel

- Sort Document 5 (SGI Documentation [1])

“Sort sorts the elements in [first, last) into ascending order, meaning that if i and j are any two valid iterators in [first, last) such that i precedes j, then *j is not less than *i. Note: sort is not guaranteed to be stable. That is, suppose that *i and *j are equivalent: neither one is less than the other. It is not guaranteed that the relative order of these two elements will be preserved by sort.

The two versions of sort differ in how they define whether one element is less than another. The first version compares objects using operator< and the second compares objects using a function object comp.

< Continues for a couple of pages>”

Metadata : prereq:iterator, prereq:arrays, prereq:lessthancomparable, shows:sort, type:reference

Sort Document 1 is an example of usage of the STL sort algorithm to sort an array. The metadata indicates that this document is a short-example. It uses a number of other concepts, as indicated in the metadata. If the user lacks this knowledge the example would not make much sense. We capture such required knowledge in the metadata associated with this object with the uses category. The metadata associated with this resource indicates that knowledge of array is a prerequisite meaning that the user will not understand the example if they do not understand arrays.

This resource would be valuable for users who like to learn by seeing examples. It would also be useful for someone wanting to quickly see how to use sort without learning the details. And it would be useful for someone who already knows sort but wants a quick reminder on its usage. In this case, the user would need to know all the other concepts the resource shows so that the sort usage didn't get lost in the clutter of new material.

Sort Document 5, the SGI Documentation is a reasonably long and detailed document describing

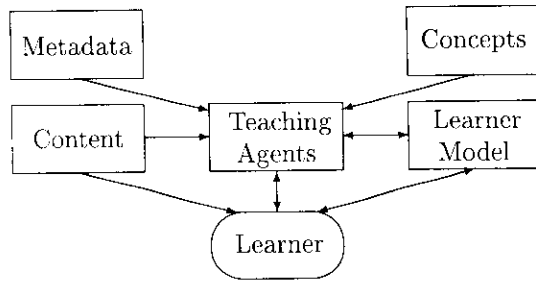


Figure 1: SITS Component Interaction

the sort algorithm and relating it to other concepts in the STL. The resource was written as documentation for programmers using the STL and thus is precise and complete. The metadata associated with the resource indicates that it requires knowledge of iterator, arrays, and less-than-comparable in order to be understood, and that it explains the sort concept and that it is reference material.

Users who have been introduced to sort briefly and want to cement the knowledge with precise documentation will find this resource useful. So will those users who want to learn sort and who prefer to learn from in-depth reference material.

2.2 Strategies for teaching

The information that will be shown to the student is determined by the strategy used for teaching. There are many different strategies that may be used, preferably one that suits the user's learning style [13, 12, 6] will be used.

One strategy might be to show tutorial information first, and then provide reference information. Another might be to provide examples and reference information together, and only use tutorial information if required.

A teaching strategy may favour resources that refer to back to things the user has recently been exposed to in order to reinforce the learning of that knowledge.

3 SITS Architecture

An expert human teacher can vary their teaching strategy to match the learner's existing knowledge and learning preferences. SITS aims to do this too. In addition it has been designed for scrutability so that the student can see the teaching strategy and how it determined the documents chosen for them.

Figure 1 shows the main component of SITS and the interactions between them.

The learner model uses an evidence based approach in which evidence that the user knows or does not know concepts is stored. This evidence is then interpreted by the Teaching Agents. The learner can scrutinise the learner model and add evidence if they believe the model is incorrect in some way.

- knowledge-iterator
- knowledge-arrays
- knowledge-less-than-comparable
- preference-eckel
- preference-reference
- preference-not-stroustrup

Figure 2: Student Model 1

- preference-stroustrup
- preference-short-description
- preference-example
- knowledge-iterators
- knowledge-vector
- knowledge-arrays

Figure 3: Student Model 2

The Concepts are the pieces of knowledge that SITS keeps track of in the learner model. They are simply identifiers and short and long descriptions which are used when presenting information to the user. For example the sort concept might be sort, "The Sort Algorithm", "A modifying sequence algorithm, which rearranges the elements of a range into sorted order".

The Content is a database of resources. The resources are not actually stored; just location information is held so they can be retrieved easily.

The Metadata store is used to keep track of the metadata related to the Content resources. In addition it provides extra information about the content (that the content was written by a particular author for example).

The Teaching Agent component drives the system. It interacts with the other components and the learner in order to determine what to teach next, what resources to use to teach a given resource, and to add evidence to the learner model.

SITS uses Teaching Agent components which allows different users to use different teaching strategies, and the same user to swap between different teaching strategies. It is the job of the Teaching Agent component to take the learner model, the documents and their metadata and determine which resources to present to the user.

3.1 Example of Operation

The selection of resources is a two step process in SITS. The first step is selecting the concept that the user is to be taught. The next step is select-

- preference-not-stroustrup
- preference-reference
- knowledge-iterators
- knowledge-vector
- knowledge-arrays

Figure 4: Student Model 3

ing the appropriate resources which address that concept.

Figures 2, 3 and 4 represent the learner model of three different users. The actual learner models consists of evidence, but that evidence is resolved to create a set of concepts the student knows. Preference information is included in this set. We will show how SITS would select a document for each of these users.

Selecting the concept to teach is performed by the Teaching Agent. SITS allows the user to select from a range of Teaching Agents to suit their preferences, thus providing some level of customisation. The Teaching Agent uses the student model and the resource metadata to select a concept that is suited to the student. The actual concepts themselves do not have any metadata - there is no representation of prerequisites for example. Instead SITS uses the metadata associated with the resources to derive this information.

Once the concept has been selected, SITS then selects the content to present to the user. This task is also handled by the Teaching Agent. It uses the student model to determine which of the resources should be used. The ability of the student to select from a number of Teaching Agents means that they can select one which matches their preferences. For example an agent that favours using examples, or an agent that favours reference information.

The actual method and implementation that a Teaching Agent uses to do these tasks is not restricted by SITS. However, the default Teaching Agent does it in the following way.

The mechanism for selecting which concept to teach will not be covered here, instead we will assume that the Teaching Agent has decided that the concept to be taught is sort.

It constructs a graph linking concepts via the resources using the metadata of the resources. The relevant section of the graph for the sort resources is shown in figure 5. A display of this graph is available to the user as part of the scrutability support.

The learner model data is then used with this graph to select the resources to use. Using the model shown in figure 2 the Teaching Agent would not use Sort Document 1, 3 and 4, because the student does not have the appropriate prerequisite

knowledge. Since the student has a preference for reference documentation Sort Document 5 would be selected as the most appropriate resource to use.

Using the model shown in figure 3 the Teaching Agent would not use Sort Document 1, 3 and 4 due to lack of prerequisite knowledge. Since the student has a preference for short reference information and for examples, both the remaining resources would be presented to the student.

Finally, for the model shown in figure 4 the Teaching Agent would not use Sort Document 1, 4 and 5 due to lack of prerequisite knowledge. However, the student does not like Stroustrup resources and likes reference information. This means none of the resources is suitable for the student. The Teaching Agent will try to first teach the student about lessthancomparable since that will allow the use of Sort Document 5. If no resources are suitable for teaching lessthancomparable, the Teaching Agent will be forced to use Sort Document 3 even though the user has a preference against the author.

4 Creating Courses for SITS

If a teacher wants to support a new set of resources the concepts that the system will use to model the learner's knowledge need to be enumerated and added to the concept database. It is difficult to add concepts when there is a large number of resources with metadata in the system, since existing resources which should reference that concept need to have their metadata updated. However, when creating a course deciding on the concepts to be taught is a necessary step even when not using SITS.

The resources that SITS will present to the learner then need to be added. This is done by giving SITS a URL for the resource and inputting the metadata. The metadata consists of relationships between the resource and the concepts as illustrated in our examples. Additional non-concept related metadata can also be added. This can be used to describe things like the resource being an example, or a reference, or using mathematics.

The fact that metadata is associated with each resource and not with the abstract concepts is an important factor. It means that the relationships between concepts do not need to be mapped out. Instead SITS will derive them from the resource metadata. This allows resources to be added independently of each other.

This also means the resources do not all have to have the same big picture outlook on the course. The resource's metadata lists the appropriate prerequisite knowledge and the resource is automatically integrated into the system. Individual resources can have different views of the relationships

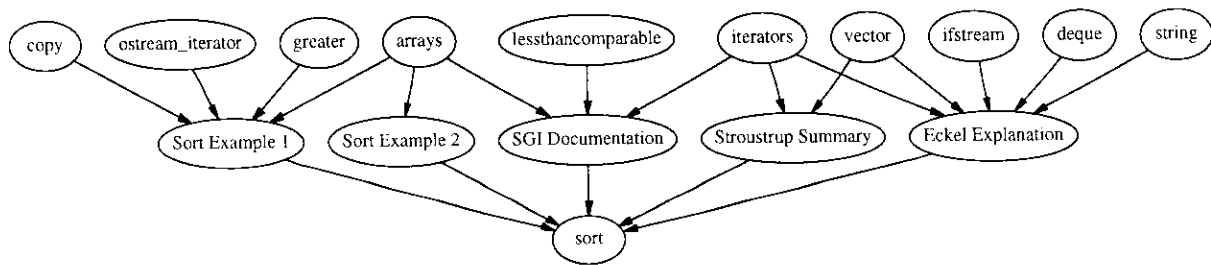


Figure 5: Sort Graph

between concepts. For example, one resource may treat sort as a concrete example of the more important abstract concept of an algorithm. While another may treat sort as a useful function provided by the STL independently of other functions.

5 Current Implementation

The administrative/teacher interface of SITS has been implemented as a web interface allows the creation concepts, addition of documents, and specification of metadata.

The current interface which presents documents to users is intended as a test bed of the teaching strategy and scrutability. It allows the scrutinising and modification of the learner model, the viewing of concepts to be taught, and the viewing of documents selected by SITS. The user can scrutinise SITS about why a document was chosen.

6 Discussion

SITS is a system designed in order to provide a scrutable tutoring system that uses a large number of resources in order to customise the learning experience to suit each user.

An important aspect of the design of SITS is that it should require a minimum in time and effort for the teacher to add each new resource. Nor should the basic set up for a new course be much more effort than documenting the learning outcomes and elements of the course.

Allowing additional of documents to be seamlessly added to the system without modification to the existing data is necessary in order for the system to be useful. This has also resulted in a system that uses elegant, simple but powerful techniques to select appropriate documents from a large set of possibilities, and ensuring the user can scrutinise the system's personalisation processes.

References

- [1] <http://www.sgi.com/tech/stl/>.
- [2] P. Brusilovsky. Methods and techniques of adaptive hypermedia. In *User Modeling and User-Adapted Interaction*, Volume 6 (2-3), pages 87-129. 1996.
- [3] S. Bull and H. Pain. Did I say what I think I said, and do you agree with me?: Inspecting and questioning the student model. In *Proceedings of the 7th World Conference on Artificial Intelligence in Education*, pages 501-508. AACE, Washington, DC, USA, 1995.
- [4] R. Cook and J. Kay. The justified user model. In *Proceedings of the Fourth International Conference on User Modelling, Hyannis, MA, USA*, pages 145-150, 1994.
- [5] Bruce Eckel. Thinking in c++ 2nd edition (volume 2). <http://www.mindview.net/Books/TICPP>.
- [6] M. Elsom-Cook, P. Byerley, P. Brooks, M Federici and C Scaroni. Using multiple teaching strategies in an its. In C. Frasson and G. Gauthier (editors), *Intelligent Tutoring Systems: At the crossroads of Artificial Intelligence and Education*. Ablex, Norwood, 1990.
- [7] J. Kay. Learner know thyself: Student models to give learner control and responsibility. In *Proc. ICCE97, International Conference on Computers in Education*, pages 18-26. Malaysia, Kuching, Sarawak, 1997.
- [8] J Kay. Learner control. *User Modeling and User-Adapted Interaction*, Volume 11, Number 1-2, pages 111-127, 2001.
- [9] J Kay and R J Kummerfeld. User models for customized hypertext. In J Mayfield and C Nicholas (editors), *Advances in hypertext for the World Wide Web*, pages 47-69. Springer Verlag, 1997.
- [10] Judy Kay. The um toolkit for cooperative user modelling. *User Modelling and User-Adapted Interaction*, Volume 4, Number 3, pages 149-196, 1995.
- [11] M J Lawson. Being executive about metacognition. In *Cognitive Strategies and Educational Performance*. Acedemic Press, Orlando, 1984.

- [12] Robert London. Student modeling to support multiple instructional approaches. *User Modeling and User-Adapted Interaction*, Volume 2, Number 1-2, pages 117–154, 1992.
- [13] N. Major and H. Reichgelt. Coca: A shell for intelligent tutoring systems. *ITS Lecture notes*, pages 523–530, 1992.
- [14] J. Self. Bypassing the intractable problem of student modelling. In *Proceedings of Intelligent Tutoring Systems '88*, pages 18–24. University of Montreal, 1988.
- [15] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Publishing Company, third edition, 1997.
- [16] S Volet and J Lawrence. Goals in the adaptive learning of university students. In H Mandl, E De Corte, N Bennett and H F Frierich (editors), *Learning and Instruction*, pages 497–516. Pergamon, 1990.
- [17] S Yussen. The role of metacognition in contemporary theories of cognitive development. In D Firrest-Pressley (editor), *Metacognition, Cognition and Human Performance*, Volume 1. Academic Press, Orlando, 1985.

ISBN1 86487 431 7