A Personalisation Framework for Advanced Learning Technology

James Ohene-Djan
Department of Mathematical & Computing Sciences
Goldsmiths College, University of London
email:j.ohene-djan@gold.ac.uk

Alvaro A.A. Fernandes
Department of Computer Science
University of Manchester
email: a.fernandes@cs.man.ac.uk

Abstract

This paper focuses on ownership transfer *via personalisation as a value-adding strategy for advanced learning technology. It contributes an approach to endowing hyperlink-based systems (e.g., for learning) with features that enable the personalisation of the interactions between users and the system that embodies the learning material.*

1. Introduction

Personalisable, adaptive hyperlink-based systems, more often referred to as *adaptive hypermedia*, (AH) [7] constitute an area of research that aims to enhance the functionality of hyperlink-based systems (HLBSs) by making the user interaction process personalisable. The approach taken is to endow HLBSs with personalisation features which may be initiated by the users or by the system itself. AH are assumed to be useful in areas, such as learning, where users have different information seeking goals, histories and preferences. AH aim to use knowledge provided by (or captured about) specific users to tailor the information and the links presented to each specific user.

The predominant application areas for AH have been tutoring systems [2, 8, 13, 14] and educational hypermedia in general [4, 5, 9]. Such systems generally encompass relatively small bodies of information resulting in relatively small hypernetworks. Users usually have a specific goal, viz., to learn a subset of the information contained therein. An important requirement of AH is, therefore, the ability to accommodate different levels of user knowledge of the subject matter. As a result, in the educational domain, AH may be used to suggest to the user appropriate routes through the hypernetwork and to indicate, or even restrict users to, specific subsets of information on an individualised basis [6].

In this paper we focus on *ownership transfer* via personalisation as a value-adding strategy for advanced learning technology. We propose a system-independent approach to the personalisation of learning materials deployed as HLBSs. The approach facilitates the personalisation of the interaction between users and the learning materials, thereby enabling individual users to come closer to satisfying their specific, dynamic information needs. A longer version [18] is available for the interested reader.

2. Motivation

Consider the question as to how students interact with a textbook over extended periods of learning. Furthermore, consider interactions which would be difficult or impossible if the book were a hyperdocument. Some of the personalisation actions that fall into this category are now defined.

Selective Reading, i.e., to open and read some page(s) only, possibly consulting the table of contents or index, and perhaps only partially. In current hyperdocuments, options for traversal are defined by the links the designer of the hyperdocument has embedded in it. Users are not provided with facilities to make links between pages and therefore define their own paths through the material. Also, when a user does want to traverse a link defined by the designer, there is no metadata about the target, thereby making it impossible not to commit to the traversal. Since traversing links can incur high computational costs, frustration could ensue.

Annotation, i.e., to add to the original printed text handwritten material that, for the student, enriches the meaning of the text. Examples include underlining or highlighting a passage, writing comments on the margin, cross-referring to some other page of the book or some other information resource (e.g., a website, another book, etc.). At present, to the authors' knowledge, no HLBS deployed on the WWW provides features to allow users to annotate parts of a hyperdocument in context, i.e., at particular points in the text.

Content Maintenance, i.e., to add content (e.g., copied from another book, or a website) at specific points and to delete (or cross out, not necessarily physically) parts of the text. Features to enable the users of a hyperdocument to tailor its content have also not been widely implemented. In particular, users are not given the ability to remove parts of a hyperdocument which they deem inappropriate for study at a point in time.

Bookmarking, i.e., to leave markers at particular pages (e.g., to indicate that they are more important in some respect than others). Note that marked pages are easily identifiable for selective reading. Although WWW browsers allow users of HLBSs to bookmark pages, such bookmarks are often limited to recording only the Internet address of a page. Users are not given the opportunity of indicating the importance of a bookmarked page over another or why a page is bookmarked at all.

Each of the above actions may be admissible on paper-based materials that the student owns but markedly difficult on hyperdocuments because the student has no means of claiming ownership of the material nor of personalising the latter (even if ownership could be transferred from the designer to the student, because the student may not have the designer knowledge required for personalisation to be carried out).

Now consider the question as to how students would interact with a textbook implemented as a HLBS. Furthermore, consider interactions which would be difficult or impossible if the book were *not* a hyperdocument. Some actions that fall into this category are now defined.

Context-Based, Nonlinear Navigation, i.e., to benefit from contextual links, at points deemed appropriate by the designer of the hyperdocument, and traverse the material nonlinearly.

Dynamic Information Management, i.e., to make use of computational resources to react and respond to requests dynamically (e.g., to benefit from the latest version of, possibly remote, independently generated and maintained hyperdocuments).

Each of the above actions may be admissible on electronic materials that a designer prepares, but markedly difficult on paper-based documents even if the student owns them.

In summary, although users of hyperdocuments benefit from the concept of linking and from the fact that electronic media typically exist in a computing environment, there are many useful actions that the owner of paper-based documents can take that are not open to users of hyperdocuments. Transfer of ownership via personalisation actions seems, therefore, to be central to deploying HLBSs in learning.

3. Framework

A general, open architecture for HLBSs, of the kind depicted in Figure 1 as a simplified data flow diagram, is assumed. Here we propose one view of the interior of the shaded oval in Figure 1. It is assumed that hyperlink functionality is provided as a client technology dependent on loose couplings to (at least) a user-interface server (UIS) and a database server (DBS). The classical example of a UIS is a WWW browser (e.g., Netscape Communicator). Among other functions, browsers broker requests and have rendering capabilities. Browsers can render formal texts (e.g., rendering expressions authored in HTML or XML). Examples of DBSs are database management systems (DBMSs) which support client server architectures (e.g., Oracle, or Informix).

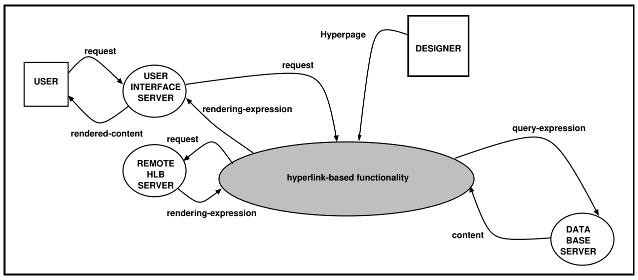

Figure 1. An Architecture for HLBSs

Broadly, the dynamics associated with Figure 1 are as follows. The UISs capture requests for desired hyperpages. The UISs channel requests for hyperpages into the HLBS proper. If a request is for a hyperpage which resides in a remote HLBS, then the core of hyperlink-based functionality interacts with it to obtain the requested hyperpage in the form of a rendering expression that it passes back for the UIS to render. If the request is for a local hyperpage (e.g., one which is known to the core) then the latter responds by returning a rendering expression to the UISs, possibly after querying one or more DBSs to fetch some or all of the content specified for the requested hyperpage.

Implicit in Figure 1 is the assumption that personalisation actions in HLBSs should not, and need not, be confounded with personalisation actions that might be provided by user-interface and database components in HLBS architectures. The shaded oval in Figure 1 is responsible for what users experience as hyperlink-based information retrieval. Notwithstanding the fact that users may well want to personalise database and user-interface features, clearly it can be argued that whatever is in scope for personalisation actions *in HLBSs* resides inside the shaded oval.

To model adaptive, personalisable hyperlink-based interaction, a framework is proposed in which the shaded oval in Figure 1 is partitioned into regions. Non-adaptive, non-personalisable HLBSs are modelled by a group of functions referred to as the *H-region*. Personalisable HLBSs require the addition to the H-region of the functions provided by the *P-region*. Adaptivity, seen as user-specific, system-initiated personalisation, comprises another region of functionality but is outside the scope of this paper (see [16, 17] for details).

The H-region models a core of hyperlink-based functionality. Conceptually, the H-region behaves as a *composer of hyperdocuments from specifications*, i.e., what the designer of a hyperpage designs is not a hyperpage, but rather a specification of how to build the hyperpage upon request. Hyperpages are modelled as formal specifications and a formal language has been defined for this purpose [16]. Within the H-region, users can only request for hyperpages to be rendered. The decisions that the designers of a hyperdocument have made with respect to content, navigation and rendering cannot therefore be overridden. Upon a request arriving from a UIS, a composition function parses the hyperpage specification into a series of actions that, when executed, convert the specification into renderable text that is sent to the UIS as the HLBS's response to the original request.

The semantics of a hyperpage specification (of which an example is shown in Figure 3) have been formalised as a program which, when interpreted, typically fetches content from a DBS, composes the content into a renderable text (making use of template variables as a binding mechanism) and finally responds to the original request with renderable text. The hyperpage specified in Figure 3 is renderable by a WWW browser.

The P-region comprises a group of functions that are non-disruptively added to the H-region in order to model personalisable hyperlink-based interaction. Personalisation is viewed as the process of handing over to the user the ability to *annotate specifications or rewrite them or both,* thereby allowing the user to override, in principle, each and every designer specification. Within the P-region, users can not only request a hyperpage, but also annotate or rewrite it, thereby creating their own version of it. The decisions that the designers of that hyperpage have made with regard to content, navigation and rendering of the hyperpage can therefore be overridden by users and this kind of event characterises ownership transfer.

When personalisation functionality is layered over the core, a designer can annotate a hyperpage in preparation for differences in users' goals and histories. Later, a user can personalise, not only such annotations (of which an example is shown in Figure 4), but the hyperpage specifications as well. Personalisation requests (of which examples are shown in Figure 5) allow users to specify which hyperpages are to be personalised and how.

Annotation pairs a hyperpage specification with notes of interest to the user and, by doing so, implies that versioning takes place. Such notes take one of the following forms. Firstly, a note can assign user-specific values to user-generic attributes of interest (e.g., that the level of difficulty of a given page or component part is high, or that 'email' is a keyword of relevance to a given page). Secondly, a note can specify a rewriting action over the renderable text after it has been composed by the H-region, i.e., after content has been fetched and made renderable (e.g., to map American into British spelling forms). This form of post-composition rewriting can also be conditional on the environment (e.g., replace images with captions if the display unit is text-only).

The existence of annotations on hyperpages allows for: personalisation of a specified hyperpage; the specification of *alternatives* to a specified hyperpage; the specification of *comparable* hyperpages to a specified one and the recording of information about a hyperpage (i.e., what are the current values of attributes set by previous annotations). Annotations are not operations on hyperpage specifications, i.e., they do not alter the latter. Rather, they give rise to a user-specific pairing with a hyperpage specification. Rewriting also causes versioning and can be characterised simply as the editing of hyperpage specifications. If a user edits the hyperpage specification as conceived by its designers, ownership is thereby transferred. If, subsequently, the same user edits that hyperpage specification again, adjustment takes place. A formal language for annotations has been defined as has a formal language for the personalisation requests used to maintain annotations and hyperpage specifications [16]. The available personalisation requests (denoting actions that users can take to improve their interaction process) are now defined.

Content Tailoring is the insertion or deletion of hyperpages and, recursively, their page parts. Selective Content is the hiding from users of parts of the content of hyperpages which they wish not to see. Content Rewriting is the rewriting of specified phrases in hyperpages. Prerequisite Content Selection is the fetching and displaying of prerequisite content for a particular hyperpage or page part. Linking is the insertion of hyperlinks between hyperpages and their page parts. Link Annotation is the annotation of links made between hyperpages and their page parts. Link Hiding and Blocking is the hiding or blocking of links.

4. Personalisation at Work

Consider Figure 2 where an excerpt from a textbook on the Internet is transcribed.

IMAP, the Internet Message Access Protocol, is a newer protocol that has advantages over POP. For instance, it defines how a client program asks a mail server to present available mail. IMAP can request that the server download only selected e-mail messages rather than all messages.

Figure 2. An Excerpt from a Textbook

To specify a hyperpage, the approach taken is to divide it into a sequence of segments called *chunks*. Each chunk comprises a specification of its content (i.e., figures, text – either hardwired or returned as the result of queries sent to databases –, etc.) and a specification of how this content should be rendered for presentation (e.g., via HTML tags). A chunk typically contains template variables. Conceptually, a template variable is bound to bits of text or else acts as a place holder for content that becomes available after evaluation of the query associated with the variable. The rendering part is specified in a language which the UIS can render, except that this renderable text may be interspersed with template variables. After content is retrieved, the template variables are bound and textual replacement takes place by substituting values (i.e., retrieved content) for references (i.e., occurrences of template variables). The overall result is renderable text (e.g., HTML).

In addition, a chunk may be associated with two sets of identifiers. The first is the set of entry points to the chunk; the second, the set of its exit points. An entry point enables the chunk (as an anchor in the page) to be referenced in a request. An exit point enables a chunk to establish navigable links to the anchor denoted by the exit point. In WWW parlance, an entry point can be thought of as a *URL* and an exit point as a *hyperlink*.

```
page{
 chunk{entry{ [<a name="Email Message Protocol"></a>]  }
  content{
    A:= 'IMAP', B:= ', the',
    C:= ' Internet Message Access Protocol',
    D:= ' is a newer protocol that has advan-
tages over ',
    E:= 'POP',  F:= '. For instance, it defines how a',
    G:= ' client program', H:= ' asks a ', I:= 'mail server',
    J:= ' to present available mail. IMAP can request
         that the server download only selected  e-mail
         messages rather than all messages.'  }
  rendering{
    [<b><a href="imap.html">           A </a></b>] B
    [<b><a href="imap.html">           C </a></b>] D
    [<b><a href="pop.html">            E </a></b>] F
    [   <a href="clientprogram.html"> G </a>    ] H
    [   <a href="mailserver.html">    I </a>    ] J  } }
 chunk{entry{ [<a name="IMAP">IMAP </a>]  }
  Content{ A := 'select def from gloss where term='IMAP'}
  rendering{['is defined as: '] A  } } }
```

Figure 3. Fig. 2 as a Hyperpage Specification

Figure 3 shows one way in which Figure 2 might be written as a hyperpage specification. It is assumed that database queries for retrieving content are written in SQL and that HTML is used in the construction of a renderable text. Due to lack of space, we omit details as to how client-server relationships are established. For more detail on how the evaluation of Figure 2 into renderable text takes place, see [16].

A hyperpage annotation pairs a hyperpage specification with notes of interest. An annotation is, roughly, a collection of contextualised attribute-value pairs. An example of annotation for the text specified in Figure 3 is given in Figure 4. The attributes used are *description*, to let the user provide a summary or abstract; *keyword*, to let the user tag content and thereby classify the information contained; *level*, to let the user attach a measure (e.g., of difficulty) to the text; and *see-as-well*, to let the user specify hyperpages that describe comparable information. Again, [16] provides more details. The annotation in Figure 4 will be associated with the hyperpage specification in Figure 3. Conceptually, this causes the versioning of the page specifically for the annotator.

Personalisation requests allow a user (e.g., a student) to generate annotations, to update annotations(e.g., those provided at source by the designer), and to update (by versioning) the hyperpage specifications themselves. A few examples of personalisation requests (in a language, and with a semantics, formally defined in [16]) that a student might issue (to the hyperdocument of which Figure 3 is a page) are depicted in Figure 5.

Example 1 in Figure 5 is a request to tailor the content found in the book. The request applies to all hyperpages (because the selection condi-

```
annotation{
 page: description :=
                   'defines and contrasts IMAP with
                   other protocols for e-mail services';
      level       := 1;
      see-as-well := [http://www.internetmail.com/];
      keyword     := 'e-mail', 'electronic mail';
 [1, chunk]: description :=
                  'Internet Message Access Protocol'
            keyword     :=
                  'email', 'IMAP', 'POP',
                  'Internet Message Access Protocol';
            see-as-well := '[page-60.html]'
 [2, chunk]: description :=
                  'Definition of IMAP
                   retrieved as dynamic content';'
 [2, chunk (R-spec)]: keyword :=
                  'Plain text'; }
```

Figure 4. Annotating Fig. 3

```
select-page-if true                     % Example 1
hp-then-do { insert [1, chunk (R-spec)]
     "These pages belong to James Fernandes."  }
                                        % ---------
select-page-if                          % Example 2
  [5, chunk] contains "electronic mail"
hp-then-do { insert [6, chunk]
     chunk {
       content  { X := 'Most recent email systems
                        support sound and video.'  }
       rendering { [<I>] X [</I>]} } }
                                        % ---------
select-page-if true                     % Example 3
ann-then-do { insert page :
       "www.ebook.com" -> "www.ecommercebook.com"; }
```

Figure 5. Personalising Fig. 3

tion is vacuously `true`). Its effect is to insert the string `"These pages belong to James Fernandes."` into the rendering specification of the first chunk of each page. Example 2 is a request to add content. The request is applied to all hyperpages that contain the string "electronic mail" in its 5th chunk. The effect on each selected hyperpage is the insertion of a new 6th chunk. Example 3 in Figure 5 is a request to rewrite a specified string. The request applies to all hyperpages and has the effect of inserting into the annotation of each hyperpage a request (as denoted by the arrow `->`') that all occurrences of the string `"www.ebook.com"` in the composed text be rewritten as `"www.ecommercebook.com"` when rendered.

The example in this section, in spite of space limitations, illustrates how hyperpage specifications that can be annotated and maintained via personalisation requests enables users of hyperdocuments to claim ownership by actions that version pages by means of annotations and rewriting requests.

5. Comparison

Little research has been conducted into formalising personalisation in HLBSs themselves. Most work has concentrated on providing personalisation features via data querying mechanisms (e.g., [1, 3, 11]). Although these contributions show how personalisation may be experienced, it can be argued that they merely exploit DBMS functionality rather than account for how hyperlink-based interaction itself can be tailored in a principled manner. We have formalised (in [16]) a complete space of possibilities for personalisation actions and the adoption of our framework enables a system to support all the personalisation actions described in [6] and implemented in HLBSs such as Adaptive HyperMan [15], ELM-ART [8] and Hypadapter [12]. Moreover, while in our framework every design decision can be overridden no proposal in the literature is as expressive as ours. In this way, we provide a more complete and coherent account of how personalisation enables and fully characterises ownership transfer. In its aims, our framework subsumes the work on the AH reference model AHAM [10]. However, the explicit aim of the AHAM is to represent techniques that have been implemented and hence it serves as a primary reference for comparative studies. In contrast, the work reported here aims to induce a set of personalisation actions from a formal definition of a core of hyperlink functionality. Rather than simply formalising implemented examples of personalisation actions, we have induced a complete space of possibilities for personalisation actions. In this way, we provide a methodological approach to the area that goes beyond the effort embodied in the AHAM.

6. Contributions and Conclusions

The challenge which the research [16, 17] underlying this paper aims to meet is how to model, at a suitable level of abstraction, the space of possibilities for personalisation actions that could be made available to users of hyperlink-based learning systems. To respond to this challenge, we have adopted the notion of ownership transfer as a means of providing a HLBS user with a learning experience that approximates that of printed materials while preserving, and building on, the benefits associated with HLBS technology. To enable ownership transfer, we have focussed on personalisation actions. To make it possible for personalisation actions to become part of HLBSs we have proposed a framework based on hyperpages as specifications of dynamic content and rendering, preferences as hyperpage versioning by annotation, and personalisation requests as the events that cause that versioning to take place. We have exemplified how our framework is a step towards bringing together the complementary benefits derived from paper materials owned by a student and from hyperdocuments that the student can only use but not personalise. The framework constitutes, therefore, a contribution to the goal of enabling HLBSs to address user information needs on an individualised basis, as discussed, e.g., in [4, 5]. Our framework can be concretely used as a basis for the implementation of a class of HLBSs (one instance of which is described in [16]) as well as for the study of implemented systems such as [2, 8, 13, 14].

References

[1] F. Afrati and C. D. Koutras. A Hypertext Model Supporting Query Mechanisms. In *Proc. ECHT'90 European Conf. on Hypertext*, pages 52–66, 1990.

[2] I. Beaumont. User Modeling in the Interactive Anatomy Aystem ANATOM-TUTOR. *User Models and User Adapted Interaction*, 4(1):21–45, 1994.

[3] C. Beeri and Y. Kornatzky. A Logical Query Language for Hypertext Systems. In *Proc. ECHT'90 European Conf. on Hypertext*, pages 67–80, 1990.

[4] D. Benyon, D. Stone, and M. Woodroffe. Experience with Developing Multimedia Courseware for the World Wide Web: The Need for Better Tools and Clear Pedagogy. *Int. J. Human-Computer Studies*, 47(1):197–218, 1997.

[5] P. Brusilovsky. Intelligent Tutoring Systems for the World-Wide Web. In R. Holzapfel, editor, *Proc. 3rd Int. WWW Conf.*, pages 42–45, 1995.

[6] P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996.

[7] P. Brusilovsky, A. Kobsa, and J. Vassileva, editors. *Adaptive Hypertext and Hypermedia*. Kluwer, 1998. ISBN 0-7923-4843-5.

[8] P. Brusilovsky, E. Schwarz, and G. Weber. ELM-ART: An Intelligent Tutoring System on World Wide Web. *LNCS*, 1086:261–269, 1996.

[9] P. Brusilovsky, E. Schwarz, and G. Weber. World-Wide Intelligent Textbooks. *Proc. World Conf. on Educational Telecommunications*, pages 302–307, 1996.

[10] P. de Bra, G.-J. Houben, and H. Wu. AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. In *Proc. 10th ACM Conf. on Hypertext and Hypermedia*, 1999.

[11] C. Hockemeyer, T. Held, and D. Albert. RATH — A Relational Adaptive Tutoring Hypertext WWW–Environment Based on Knowledge Space Theory. In C. Alvegård, editor, *Proc. CALISCE 98*, pages 417–423, 1998.

[12] H. Hohl, H.-D. Boecker, and R. Gunzenhaeuser. Hypadapter: An Adaptive Hypertext System for Exploratory Learning and Programming. *User Modelling and User-Adapted Interaction*, 6(2-3):131–156, July 1996.

[13] J. Kay and B. Kummerfeld. An Individualised Course for the C Programming Language. In *Proc. 2nd Int. WWW Conf.*, 1994.

[14] A. Kobsa, D. Müller, and A. Nill. KN-AHS: An Adaptive Hypertext Client of the User Modeling System BGP-MS. In *Pro. 4th Int. Conf. on User Modeling*, pages 99–106, 1994.

[15] N. Mathe and J. Chen. User-centered Indexing for Adaptive Information Access. *User Modeling and User-Adapted Interaction*, 6(2-3):225–261, July 1996.

[16] J. Ohene-Djan. A Formal Approach to Personalisable, Adaptive Hyperlink-Based Interaction. Technical report, Department of Mathematical and Computing Sciences, Goldsmiths College, University of London, 2000. Available from http://homepages.gold.ac.uk/ohene-djan/pa-hlbs.ps.zip.

[17] J. Ohene-Djan and A. A. A. Fernandes. Modelling Personalizable Hyperlink-Based Interaction. In *Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia (HYPERTEXT98)*, 1998. Available from http://www.contrib.andrew.cmu.edu/∼plb/HT98_workshop/Proceedings.html.

[18] J. Ohene-Djan and A. A. A. Fernandes. Modelling Personalizable Hyperlink-Based Interaction, 2000. Long Version at http://www.cs.man.ac.uk/∼alvaro/djanj_personalisation.ps.zip.