Adapting and Personalizing Hyperlink-Based Interaction

James Ohene-Djan, Alvaro A.A. Fernandes
Goldsmiths College, University of London
New Cross, London SE14 6NW, UK

ABSTRACT

This short paper summarizes the main features of an abstract model for adaptive, personalizable hyperlink-based interaction currently under development at Goldsmiths College. The model proposes loosely-coupling a core hyperlink engine to content and presentation services. To such a core a personalization engine and an adaptivity engine are, in turn, loosely-coupled. Besides greater conceptual clarity regarding the scope for *personalization and adaptivity* (P&A) in *hyperlink-based systems* (HLBSs), the formalization of the model induces a comprehensive set of very expressive P&A actions. The level of abstraction adopted facilitates implementation. Any concrete instantiation of the model benefits from both the low level of component coupling and their high cohesion.

KEYWORDS: Personalizable, Adaptive Hypermedia

INTRODUCTION

By the late 1980s, a number of different HLBSs had been implemented in the wake of intense hands-on development activity. This lead to a surge in interest in generalizing these instances into an abstract model of HLBSs. The *Hypertext Abstract Machine* (HAM) [2] is the result of the first major attempt to define an abstract model for HLBSs. Then, in two workshops [7], a group of researchers collaborated to propose a formal model of HLBSs, known as the *Dexter model* [5], which aimed at fixing a terminology and providing a formal specification (in Z) that generalized over most implemented HLBSs then in existence. Other models for HLBSs [4, 6] emerged from the same workshops.

A concrete implementation instantiating any of the models above is utterly monolithic. It tightly couples the functionality required by content and presentation management with core hyperlink-based functionality. In this view, HLBSs emerge by multiple inheritance from both user-interface and database management systems but the proposals above do not precisely spell out what are the non-inherited (i.e, the specific, characteristic, distinctive, identity-defining) aspects of HLBSs. Since the models above generalize existing implementations, this situation has not severely hindered research and development. However, with the intense interest in P&A in HLBSs, the monolithic nature of HLBSs and lack of clarity as to what, conceptually, is in scope for P&A actions represent obstacles for a principled exploration of P&A issues. The model summarized in this paper represents a contribution towards a better foundation to underlie that exploration.

THE MODEL

The model proposes an architecture (depicted in Figure 1) for HLBSs in which a core of hyperlink-based functionality is loosely-coupled to standard user-interface and database management systems, which act as providers of content and presentation services.

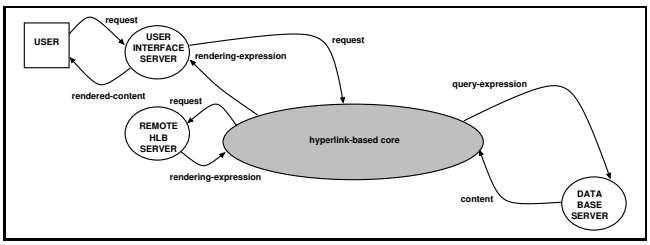

Figure 1: A Loosely-Coupled Architecture for HLBSs

This **core** is formalized as a *composer from specifications*, i.e., what a designer writes is not a document but rather a specification of how to build the document upon request. A formal language for writing such specifications has been defined along with a formal abstract machine to execute them thereby yielding renderable documents. This is depicted in Figure 2. For details, see [8].

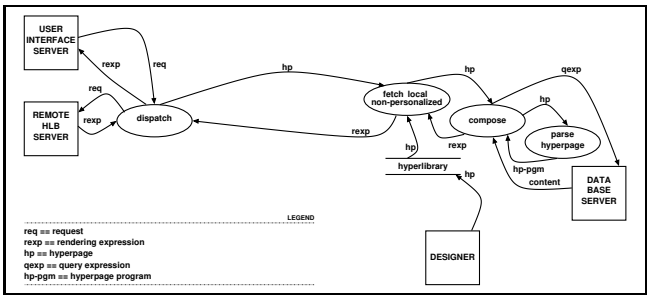

Figure 2: The Core Hyperlink-Based Functionality

At the most basic level, the core functionality provided is to process specifications of hyperdocuments into renderable

texts. In this approach, a designer writes specifications as to where the contents can be found, what rendering is the document to have and how to compose content and presentation features into a renderable text that a user sees. The basic dynamics of the core is the following. The user requests a page to be rendered, as usual. Such a page exists, and is fetched, as a specification of where to find its content (which may be, yes, hardwired into the specification) and how to render it. One then proceeds to fetch the contents by client-server querying designer-specified sources. The retrieved content is then composed, as specified by the designer, into the text to be rendered. Finally, the core responds to the request with the text thus composed.

Personalization is viewed as the process of handing over to the user the ability to *annotate specifications or to rewrite them or both*, thereby allowing the user to override, in principle, each and every designer specification. When personalization functionality is layered over the core the following capabilities are added (which assume some versioning scheme to be available in the environment). A designer can now annotate a page, e.g., on her estimate of things like theme, level of difficulty, etc., in preparation for differences in user's goals and background to be acknowledged. A user can personalize not only such annotations but the document specification as well, thereby overriding the designer's original decisions. If the user's preferences can be expressed at the specification level (e.g., a preference for one among many equivalent sources) then it can be enforced eagerly by versioning the hyperpage. Other preferences (e.g., to enforce British spelling) may need to be enforced lazily, because they need the content over which they range to be fetched from a server, so a kind of post-processing facility is supported. A formal language for tailoring the hyperdocument specified by the designer into a personalized version has been defined along with a formal abstract machine to generate them. This is depicted in Figure 3. For details, see [8].

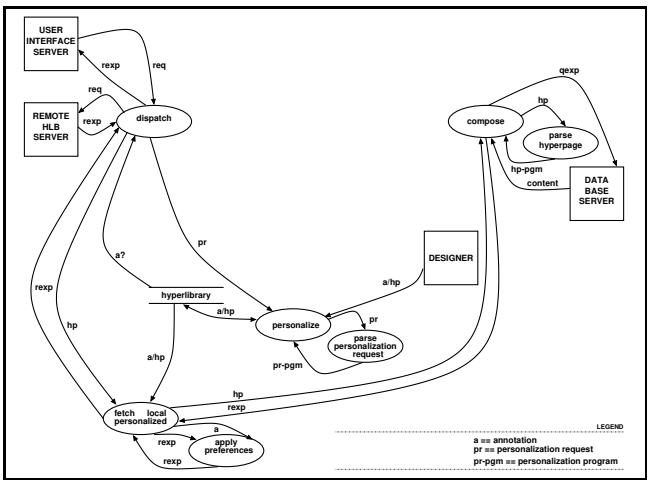

Figure 3: The Layer of Personalization Functionality

Adaptivity is viewed as the process of *allowing the system, by proxy, to take the initiative in personalization actions*, in the light of the system's inference of the information goals and the background of users. When adaptivity functionality is layered over the core and personalization layers, the following capabilities are added. Both users and designers can define strategies as to when the system should take the initiative and actively tailor the interaction to a user in the light of that user's information goals and history of use. A deductive engine will use an axiomatized theory expressed as active rules [3] to decide what personalization is warranted. This is depicted in Figure 4 and represents future work.

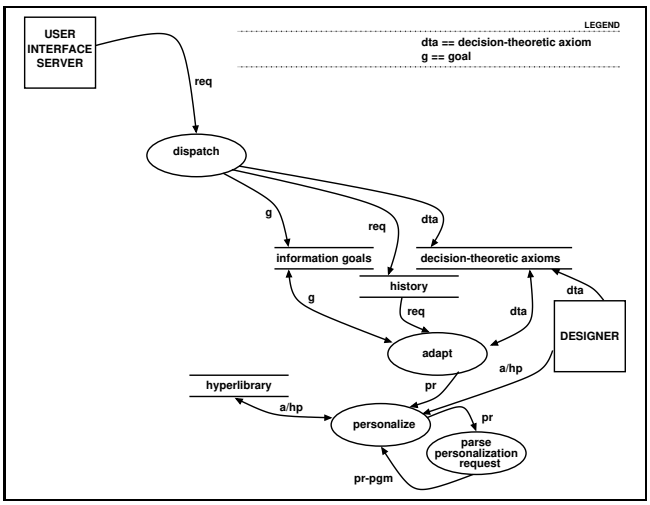

Figure 4: The Layer of Adaptivity Functionality

CONCLUSION

Besides greater conceptual clarity regarding the scope for P&A in HLBSs [8], the model summarized here induces a comprehensive set of very expressive P&A actions [8]. The level of abstraction adopted facilitates implementation. The model summarized in this paper thus represents a contribution towards a better foundation to underlie the exploration of P&A [1] issues in HLBSs.

REFERENCES

1. P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User Adapted Interaction*, 6(2–3):87–129, 1996.
2. B. Campbell and J.M. Goodman. HAM: A General Purpose Hypertext Abstract Machine. *Comm. of the ACM*, 31(7):856–870, July 1988.
3. A.A.A. Fernandes, M.H. Williams, and N.W. Paton. A Logic-Based Integration of Active and Deductive Databases. *New Generation Computing*, 15(2):205–244, 1997.
4. R. Furuta and P.D. Stotts. The Trellis Hypertext Reference Model. In Moline et al. [7], pages 83–94.
5. F. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. In Moline et al. [7], pages 95–133.
6. D.B. Lange. A Formal Model of Hypertext. In Moline et al. [7], pages 145–166.
7. J. Moline, D. Benigni, and J. Baronas, editors. *Proc. Hypertext Standardization Workshop*. US Department of Commerce, NIST SP 500-178, 1990.
8. J. Ohene-Djan and A.A.A. Fernandes. Adaptive, Personalizable HLBSs. Draft: `mailto:a.fernandes@gold.ac.uk`, 1998.