An adaptation approach: query enrichment by user profile

Corinne Amel Zayani[1,2], André Péninou[2], Marie-Françoise Canut[2], and Florence Sedes[1,2]

[1] IRIT, 118 route de Narbonne, 31062 Toulouse cedex 4, France
zayani,sedes@irit.fr,
WWW home page: http://www.irit.fr

[2] LGC, 129 A, avenue de Rangueil B.P 67701, 31077 Toulouse cedex 4, France
peninou, canut@iut-blagnac.fr

Abstract. In semi-structured information systems, generally, the adaptation of documents is essential to give the user the feeling that the query result is adapted to his need. The user's needs can be defined in a user profile.
But, in the literature the adaptation systems are designed for a particular domain and are oriented towards either navigation adaptation or content adaptation. Adaptation takes place after the user's query has been evaluated. So, in this paper, we contribute to propose an adaptation algorithm which is domain independent. This algorithm consists in enriching the user query on the basis of user profile in order to adapt the results to the user.

1 Introduction

In semi-structured information systems, the adaptation of documents is defined for a particular domain, such as educational domain, e-commerce, tourism, etc. On the other hand, it depends on a user profile which aims to descibe some user characteristics [6]. The adaptation can be applied to the content or/and navigation, in order to answer respectively the problems of cognitive overload and disorientation [8]. The adaptation is very often supported by Event-Condition-Action rules [1] that are defined and implemented in adaptation engines. The engines trigger the rules over the attributes defined in the user model (e.g. number of visits, knowledge degree, etc.) and in the domain model. These rules can also update the user model. For a particular domain, it is relatively easy to design the rules to be implemented by an artificial intelligence expert.
In addition, the problem of adaptation within generic framework is recently tackled by [9]. In its thesis, [9] adapts the document presentation to a given user (blind user, normal user). The user defines explicitly his presentation preferences for each URL resource. We can then notice that adaptation systems are directly related to either a particular domain or to a particular user.
Within a generic framework and in order to use implicit user characteristics, we have yet proposed in [15] a domain independent architecture. This architecture

extends those defined in the literature such as AHAM [4] or Munich reference model [12]. The particularity of this proposed architecture is that the domain model has been replaced by a document model that was defined by [2] and the user model is built from the analysis of user's queries. Moreover, two adaptation processes have been defined in the architecture: upstream adaptation and downstream adaptation.
In this paper, we are interested in explaining the upstream adaptation process. We suppose that we have a document repository containing documents which can belong to any domain. To query a document or the document collection, the user builds his query textually or graphically [5]. On the other hand, we suppose that we have a user profile repository that clusters the needs of each user. The aim of this upstream adaptation process is to contribute to enrich the user query on the basis of his profile in order to adapt the result presented to the user.
The paper is structured as follows. Next section presents a brief summary of adaptation systems. Section 3 describes user's characteristics of the user model. Section 4 presents the adaptation process. In section 5 the adaptation algorithm is defined. In section 6 we present an application. An evaluation is made on this application in section 7. Conclusion and future work close the paper.

2 Related works

The AHAM (Adaptive Hypermedia Application Model) [4] model has been originally defined for educational domain. This model is based on the generic Dexter Model [10], but remains limited to applications in educational adaptive hypermedia systems. It splits up the storage layer of the Dexter Model into an adaptation model, a domain model, and a user model. The user model is an overlay of the domain model. The adaptation engine implements a set of adaptation rules for educational purpose. Like in AHAM, [12] have described the Munich reference model for adaptive hypermedia applications, where the adaptation engine can implement not only the educational-oriented rules but also other rules which can be defined for a particular domain by an expert. Like in this latter model, AHA! system [3] is defined for adaptive Web application. As explained earlier, AHAM, Munich reference model and AHA! perform rule-based adaptation. Consequently the adaptation engine only implements domain-based rules. Some works tried to reduce this restriction by defining non-persistent properties and post and pre concepts access rules execution. [13] have defined GAM that is a generic theoretical model for describing the user adaptive behaviour in a system in order to adapt interactive systems. This problem has been deal with [9] when blind users interact with information systems.
These different architectures for adaptive hypermedia system are oriented towards particular domains or interactive systms. So, we have yet proposed in [15] to change the domain model by a document model relative to any domain. In this proposal, the user profile can include different interests relative to any domain. It is built from the analysis of user's queries. Because of limitation of

content, we don't present the document model. The reader may refer to [15]. In this paper, we present the user profile and the adaptation process.

3 User profile

As adaptation depends on the user model, we suggest that the user model should describe structure of each user profile. We suppose that user profile is structured in XML format. The user profile contains characteristics that can be distinguished into two types [6] [11]:

- *permanent characteristics* which are constant over time. This type of characteristics introduces the user identity (last name, first name, etc.), his demographic data (age, kind and addresses), etc. It is useful in order to identify the user,
- *changing characteristics* evolve over time. This type of characteristics introduces the user interests, his preferences, his knowledge. So, the user profile is built from an analysis of his previous queries. It is useful in order to enrich user query and adapt the result. The structure of changing characteristics is the same as the structure of user query, which allows to make comparison easier between user profile, query and the document in order to enrich the query.

4 Adaptation process

The adaptation process is the key issue in this paper: it describes how the adaptive hypermedia system should perform its adaptation. In the literature, adaptation process is defined as a set of adaptation rules that make connection between domain model and user model. But each set of adaptation rules is designed for a particular domain because they take into account the relationships defined between the different concepts of the domain. The rules are triggered when the user chooses a link in a document. These rules adapt either the content or the links (navigation) of the chosen document that must belong to the targeted domain (domain model).
But such rules don't perform any adaptation when the user queries a document collection. To tackle this problem, we have distinguished two adaptation processes respectively named : downstream and upstream.

- *The downstream adaptation process* is useful when the user browses in hypermedia infrastructure. The process is based on adaptation rules as described above. Generally, these rules are defined by an artificial intelligence expert on the domain.
- *The upstream adaptation process* is used when the user queries the document repository. This process aims to enrich the user queries on the basis of his profile in order to enhance the result.

In this paper, we are interested in the upstream adaptation process. So, we present an algorithm which aim is to enrich the user query according to the user profile.

5 Proposal of an enrichment algorithm for upstream adaptation

In order to make easier the matching between user profile, query and document, we suppose that the query is an XML document that separats three intersting parts of the query : condition of the query, the structure of the results and the ordering of the results. We are interested in the condition part of query. We also suppose that user profile, and above all, changing characteristics are represented in XML documents. The profile is built from an analysis of user's previous queries. So, supposing that both query and profile are sets of conditions on elements, we note $Profile = \{condition_p + frequency\}$ and we note $query = \{condition_q\}$.
The main idea of the algorithm is to split enrichment in two parts :

- The first part aims to update user profile and to manage profile elements frequency. It verifies if the names of query elements are similar to the names of profile elements (see equation 2 below). If it is the case, the element frequency in the profile is incremented. Else, the element will be added to the profile.
- The second part aims to enrich the query. It verifies the similarity between the query elements proprieties[3] and each profile element proprieties. For query element, it is necessary to find their properties from the XML document. If the similarity is higher than 0, the profile element can be added to the query.

5.1 Algorithm

In order to simplify the reading of the algorithm we note user profile by UP, a profile element by PE, a query by Q, a query element by QE and a XML document collection by XML DC. In order to simplify the presentation, the first part of algorithm that consists in updating profile is not entirely presented. The algorithm for enrichment is:

```
program Enrichment
   var
    stageNumber := 0;
    frequency :=false;
   begin
    Traverse Q
```

[3] The properties of the element consist in name(e), attribute(e), parent(e), children(e), brother(e) and e refer to element.

```
Traverse UP
      Calculate the name similarity between PE and QE
       If  name similarity > 0  then   //equation (2), see below
PE.frequency++ and frequence = true
       EndIf
    EndTraverse UP
    If frequency =false then
       Traverse XML DC
       Until find the element propriety running of the query
       Save the element propriety found in VarElementPropriety
       Traverse UP
        calculate children similarity and parent similarity between
        PE and QE from VarElementPropriety
          If children similarity > 0   //equation (3), see below
             or parent similarity > 0 then  //equation (4), see below
          add PE in the TemporaryQuery
            stageNumber ++
      EndTraverse UP
    EndIf
EndTraverse Q
Traverse TemporaryQuery
    For i=0 until stageNumber
        Change predicate by the negation
    EndFor
EndTraverse TemporaryQuery
end.
```

The enrichment of the query is made stage by stage. Each stage leads to a new "partial" query, that we can consider as a part of the adapted query to be evaluated. At each stage, two parts are inserted in the query being built: a static part and an evolutionary part. The first part is the conditions of the initial query. The second part comprises the elements extracted from the user profile and that satisfies similarity degrees(see section 5.2). The evolution of this part depends on the change of predicates (e.g. $=$, $\leq$, $\geq$, etc) by their negation.
In the first stage, we keep the initial predicates which are extracted from the profile. Afterwards, in each stage, a predicate will be replaced by its negation. The negation of the predicates is made in the increasing order of frequency existing in the user profile.
In the last stage, for the second part (mentioned earlier), all predicates are the negation in comparison with the first stage.
Finally, in order to adapt the results to the user, the system should evaluate all the generated queries, in the order they have been generated (the different queries generated during the stages presented before).
The number of query enrichment stages is equal to the number of elements that will be extracted from the user profile (and added to the query) plus one. The number of stage $= N + 1$, where N is the number of elements extracted from

the user profile. For the moment, the user profile is updated by increasing the element frequency by one.

5.2 Similarity measurement

The similarity measurement between the user profile and the query is determined on the basis of their elements properties. From a general point of view, the similarity measurement is take from [14] and is given by the following equation (1) based on the set-theory functions of intersection (∩) and difference (/).

$$s(a,b) = \frac{\mid A \cap B \mid}{\mid A \cap B \mid + \alpha(a,b) \mid A/B \mid + (1-\alpha(a,b)) \mid B/A \mid} \text{for } 0 < \alpha < 1 \quad (1)$$

where a and b refer to elements being compared; A and B refer respectively to the set of properties of a and b; A ∩ B refers to the common proporties of A and B; A/B refers the proporties that belong to A but not B; ‖ is the cardinality of a set; and α is a weight that defines the relative importance of the non-common proporties.
The similarity measurement for the names of elements a and b is determined by the name matching of the two elements, i.e.[14]

$$s_{name}(a,b) = \frac{N_{n(a)\cap n(b)}}{N_{n(a)\cap n(b)} + \alpha(a,b)N_{n(a)/n(b)} + (1-\alpha(a,b))N_{n(b)/n(a)}} \quad (2)$$

where $N_{n(a)\cap n(b)} = \mid name(a) \cap name(b) \mid$ return 1 if the two names have a common string, else 0. $N_{n(a)/n(b)} = \mid name(a)/name(b) \mid$return 1 if a has a difference between b names, else 0.
For example, the two names "depart" and "department" have a common string "depart" and the set difference of two names is the string "ment". Therefore, $N_{department \cap depart} = 1, N_{department/depart} = 1, N_{depart/department} = 0$. Supposed that weight α =0 (the relative importance of $N_{n(a)/n(b)}$ and $N_{n(b)/n(a)}$ is the same), the similarity of two names is equal to $s_{name}(department, depart) = 0.5$. This equation is used at the first part of the enrichment query when the name of element queries is similar at the name of element profile.
At the second part of algorithm, the similarity of children and the similarity of parent must be be calculated. The similarity of the children of tow elements is calculated by the following equation:

$$s_{children}(a,b) = \frac{N_{c(a)\cap c(b)}}{N_{c(a)\cap c(b)} + \alpha(a,b)N_{c(a)/c(b)} + (1-\alpha(a,b))N_{c(b)/c(a)}} \quad (3)$$

Where $N_{c(a)\cap c(b)} = \mid children(a) \cap children(b) \mid$ returns the number of common child elements (with the same name property) of a and b. $N_{c(a)/c(b)} = \mid children(a)/children(b) \mid$returns the number of child elements of element a but not b. $N_{c(b)/c(a)} = \mid children(b)/children(a) \mid$ returns the number of child elements of element b and not a.

The formulas for similarity measurement for attributes and brothers of two elements a and b are similar to that of children.
The similarity of parents for a and b is defined as follows :

$$s_{parent}(a,b) = \begin{cases} 1 \text{ if } s_{name}(parent(a), parent(b)) > 0 \\ 0 \text{ if } s_{name}(parent(a), parent(b)) = 0 \end{cases} \quad (4)$$

6 Application

In order to experiment our algorithm, we use a collection of XML documents of cottage rentings. Each document describes a rented house with structured data (number of beds, person number, etc.) and also raw text (rooms description, leisure activities,etc.). This collection exists in the PRETI platform[4] that allows the user to retrieve information according to several querying mechanisms, such as by preferences, by flexible operator, etc.
Our application aims to improve this platform by adding a repository for user profiles and adding the enrichement algorithm in order to adapt the query results to the user. The figure 1 presents the cottage rentings document structure by a tree.

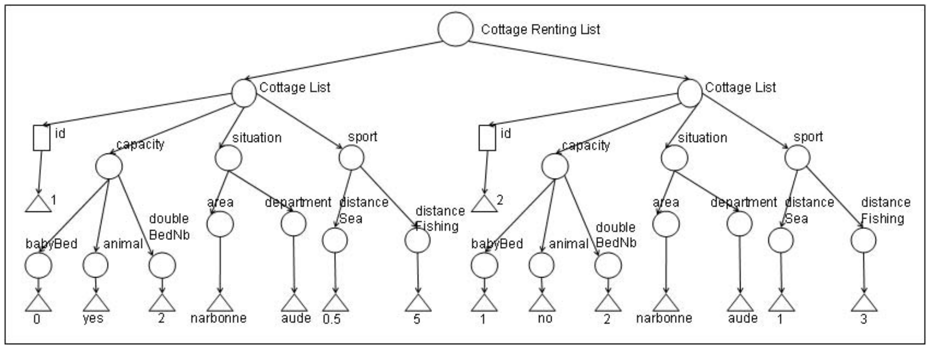
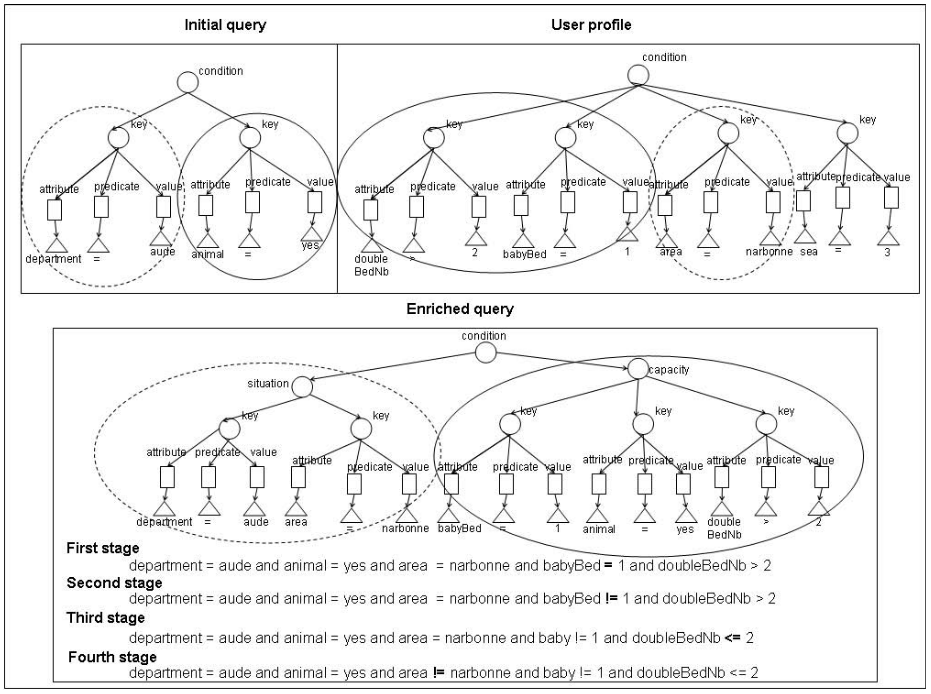


Fig. 1. Example of XML document of cottage renting

We suppose a given user is presented on the right of the figure 2 and that the current query is presented on the left of the figure 2. The query is: find rented cottage in "aude" department accepting animals (department = "aude" and animal = "‘yes").
We have applied the algorithm proposed in the previous section (sect. 5.1). The resulting enriched query is presented in the lowest part of the figure 2, int the form of four queries (one query by stage of enrichment).

[4] http://www.irit.fr/PRETI

We have simplified the problem of matching between user profile and his current query by only taking into account the conjunction operator between conditions. The figure 2 shows the elements which have similar proprieties by plain lines and broken lines groups.



department = aude and animal = yes and area = narbonne and babyBed **=** 1 and doubleBedNb > 2

department = aude and animal = yes and area = narbonne and babyBed **!=** 1 and doubleBedNb > 2

department = aude and animal = yes and area = narbonne and baby != 1 and doubleBedNb **<=** 2

department = aude and animal = yes and area **!=** narbonne and baby != 1 and doubleBedNb <= 2

Fig. 2. Example of query enrichment from a user profile

7 Evaluation

Generally the evaluation of the efficiency of query result can be done by both recall and precision measurements [7]. The recall is the ratio of the number of relevant documents retrieved by the query to the total number of relevant documents. This measurement is information retrieval oriented that is not our working context. In fact, in our work, the number of results is similar whether we take into account the user profile or not. Indeed, the different stages of a query Q, when combined with a profile P leeds to evaluate formulas of the form (Q and P) and then (Q and ¬ P). Finally, $\mid Q \mid = \mid$ Q and P $\mid + \mid Q\ and\ \neg P \mid$, assuming that |X| calculates the cardinality of the result set of the expression X. The precision is the ratio of the number of relevant documents retrieved to

the total number of documents retrieved (irrelevant and relevant).
In order to simplify the calculation of precision measurement, we have evaluated 700 documents from the collection of PRETI. For visualisation reasons, we present the result of an evaluation process that took place for 26 documents, knowing that this sample is a representative sample. When submitting the initial query shown in the figure 2 (highest left), there are 21 returned documents (see table 1).

Table 1. Result of initial query without user profile

rank of document	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
document id	1	2	3	6	7	8	9	10	12	13	14	15	16	17	19	21	22	23	24	25	26

When considering the results for initial query and if we study the user profile, there is only the document id 19 that corresponds to the enriched query at the stage 1, i.e. the result suits the user profile 100%, but this document take the 15th rank. Conversely, the document id 7 takes the 5th rank although it corresponds to the stage 4 and then, doesn't suit the user profile.
On the other hand, when we enriched the query by user profile, the document id 19 for example took the first place, i.e. the result is adapted at the user (see table 2).

Table 2. Result of enriched query

rank of document	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
document id	19	3	17	1	2	6	8	9	10	12	13	14	15	16	23	24	25	26	7	21	22

In our case, we have defined the precision by the following equation:

$$Precision_{i,j} = \frac{Number\ of\ document\ adapted\ to\ profile\ in\ stage\ j}{first\ i\ document\ returned} \qquad (5)$$

This equation (5) has been used to evaluate precision. We show the charts associated to the initial query and enriched query that take into account the stage 1, stage 2, stage 3 and stage 4 (see figure 2) respectively in figures 3, figure 4, figure 5, and figure 6. For the initial query,it is represented by break line curves, that show the precision measurement for i= 5, 10, 15, 20 or 25, where i is the number of first document returned. We evaluate the precision of this initial query by considering, for each retrieved document, its ranking when the query is enriched, more precisely, the stage at which the document is retreived by the enriched query. That leads to obtain the four series of results shown in figure 3 to 6. For example, figure 4 shows the results of the initial query by evaluating in the set of the 10 first documents, thoses that corresponds to the stage 2 of the enriched query. The enriched query is represented by contain line curves, that show the precision measurement for i= 5, 10, 15, 20 or 25, where i is the number of first document returned by the enriched query.

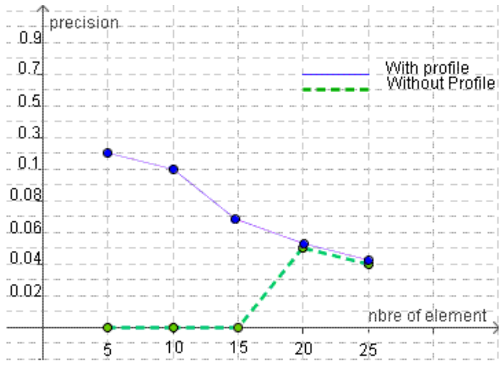


Fig. 3. Precision for the stage 1

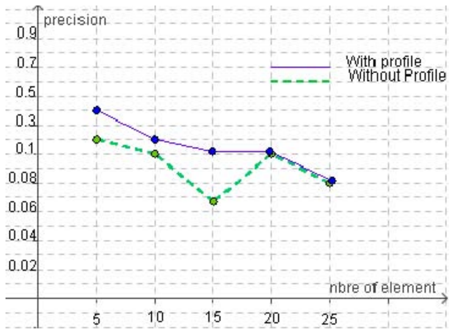


Fig. 4. Precision for the stage 2

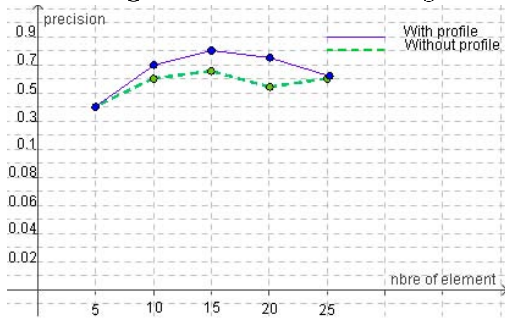


Fig. 5. Precision for the stage 3

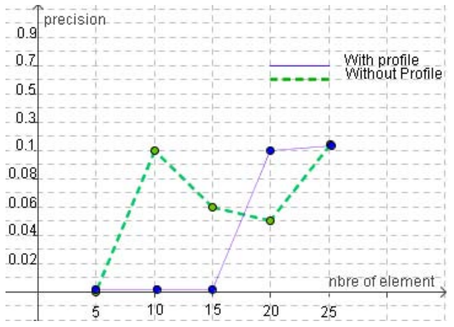


Fig. 6. Precision for the stage 4

We can notice that the precision of the returned result is better when the query is enriched by user profile. In figure 3 the precision of result returned by the initial query is null in the beginning, but when the query is enriched by user profile, the precision is very high. So, in this case, the result is better adapted at the user.
Normally the result of the initial query includes some document that doesn't correspond exactly to user profile and that may be shown in the first ranks. In order to put these documents at the end of the result, we have proposed to insert them in the final stage of query enrichment (last evaluated query of the complete enriched query). So, we can see on the figure 6 that the precision of the result of enriched query for the last stage is very low at the beginning, which is an expected outcome.

8 Conclusion

In this paper we have presented the upstream adaptation process that is based on the algorithm that consists in enriching the user query on the basis of the user profile in order to adapt the result. We have proposed an evaluation of query enrichment mechanism in the corpus of the PRETI documentary collection. This evaluation compares the precision measurement between the initial query and the enriched query. For the future work, we will improve the algorithm in order to better update the user profile and we will carry out large scale experiments. Moreover, we intend to take into account in the enrichment algorithm not only the conjunction operator ”and”, but also the disjunction operator ”or” and any combination. On the other hand, we will study the downstream adaptation process, and we intend to define active rules for XML documents which depends on DOM Event mode, in order to adapt the integrality of XML document that consist in documentary units.

References

1. A. Aiken, J. M. Hellerstein, and J. Widom. Static analysis techniques for predicting the behavior of active database rules. *ACM Trans. Database Syst.*, 20(1):3–41, 1995.
2. I. Amous. *Méthodologies de conception d'applications hypermédia - Extension pour la réingénierie des sites Web*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, décembre 2002.
3. P. D. Bra, A. Aerts, B. Berden, B. de Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash. Aha! the adaptive hypermedia architecture. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 81–84, New York, NY, USA, 2003. ACM Press.
4. P. D. Bra, G.-J. Houben, and H. Wu. Aham: a dexter-based reference model for adaptive hypermedia. In *HYPERTEXT'99: Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots*, pages 147–156, New York, NY, USA, 1999. ACM Press.
5. D. Braga and A. Campi. Xqbe: A graphical environment to query xml data. *World Wide Web*, 8(3):287–316, 2005.
6. P. Brusilovsky. Adaptive hypermedia. *User Model. User-Adapt. Interact.*, 11(1-2):87–110, 2001.
7. M. J. Cleverdon C. W. and K. M. Factors determining the performance of indexing systems. *ASLIB Cranfield Research Project*, 1966.
8. J. Conklin and M. L. Begeman. gibis: A hypertext tool for team design deliberation. In *Hypertext*, pages 247–251, 1987.
9. B. Encelle and N. Jessel. Adapting presentation and interaction with XML documents to user preferences . In *ICCHP'04: International Conference on Computers Helping People with Special Needs , Paris (FRANCE), 07/07/04-09/07/04*, pages 143–150, Heidelberg Germany, juillet 2004. Springer-Verlag.
10. F. Halasz and M. Schwartz. The dexter hypertext reference model. *Commun. ACM*, 37(2):30–39, 1994.
11. A. Kobsa, J. Koenemann, and W. Pohl. Personalised hypermedia presentation techniques for improving online customer relationships. *Knowl. Eng. Rev.*, 16(2):111–155, 2001.

12. N. Koch and A. Kraus. The expressive power of uml-based web engineering, 2002.
13. T. van der Weide and P. v. Bommel. GAM: A Generic Model for Adaptive Personalisation. Technical Report ICIS–R06022, Radboud University Nijmegen, Nijmegen, The Netherlands, EU, June 2006.
14. S. Yi, B. Huang, and W. T. Chan. Xml application schema matching using similarity measure and relaxation labeling. *Inf. Sci.*, 169(1-2):27–46, 2005.
15. C. Zayani, F. Sedes, A. Peninou, M.-F. Canut, and I. Amous. Interrogation de documents semi-structurés : extensions basées sur l'adaptation à l'utilisateur. In *INFORSID, Hammamet - Tunisie, 31/05/06-03/06/06*, pages 97–112, http://www.hermes-science.com/, juin 2006. Hermès Science Publications.