

# Personalization for the Semantic Web

Matteo Baldoni<sup>1</sup>, Cristina Baroglio<sup>1</sup>, and Nicola Henze<sup>2</sup>

<sup>1</sup> Dipartimento di Informatica, Università degli Studi di Torino  
c.so Svizzera 185, I-10149, Torino, Italy

Email: {baldoni,baroglio}@di.unito.it

<sup>2</sup> ISI - Semantic Web Group, University of Hannover,  
Appelstr. 4, D-30167 Hannover, Germany

henze@kbs.uni-hannover.de

**Abstract.** Searching for the meaning of the word “personalization” on a popular search engine, one finds twenty-three different answers, including “*the process of matching categorized content with different end users based on business rules ... upon page request to a Webserver*”, “*using continually adjusted user profiles to match content or services to individuals*”, and also “*real-time tailoring of displays, particularly Web pages, to a specific customer’s known preferences, such a previous purchases*”. A little more generally, personalization is a process by which it is possible to give the user optimal support in accessing, retrieving, and storing information, where solutions are built so as to fit the preferences, the characteristics and the taste of the individual. This result can be achieved only by exploiting machine-interpretable semantic information, e.g. about the possible resources, about the user him/herself, about the context, about the goal of the interaction. Personalization is realized by an inferencing process applied to the semantic information, which can be carried out in many different ways depending on the specific task. The objective of this paper is to provide a coherent introduction into issues and methods for realizing personalization in the Semantic Web.

## 1 Introduction

Personalized information systems aim at giving the individual user optimal support in accessing, retrieving, and storing information. The individual requirements of the user are to be taken into account in such different dimensions like the current task, the goal of the user, the context in which the user is requesting the information, the previous information requests or interactions of the user, the working process s/he is involved, the expertise level (an expert will be satisfied by information which is not suitable for a layman), the device s/he is using to display the information, the bandwidth and availability of the communication channel, the abilities (disabilities or handicaps) of the user, his/her time constraints (whether s/he is under time pressure, or is just browsing some information), and many, many more. Many different research disciplines have contributed to explore personalization techniques and to evaluate their usefulness

within various application areas: E.g. hypertext research has studied personalization in the area of so-called adaptive hypertext systems, collaborative filtering research has investigated recommender systems, artificial intelligence techniques have been widely used to cluster Web data, usage data, and user data, reasoning and uncertainty management has been adopted to draw conclusions on appropriate system behavior, etc.

Many attempts have been done to apply personalization techniques to the Web as a natural extension of work on hypertext and hypermedia, however, the Web is an information space thought for human to human communication, while personalization requires software systems (broadly speaking “machines”) to take part to the interaction and help. Such systems require knowledge to be expressed in a machine-interpretable format, which in the Web is not available. The development of languages for expressing information in a machine-processable form is characteristic of the *Semantic Web* initiative, as Tim Berners-Lee pointed out since 1998. This is the reason why the Semantic Web is the appropriate platform on which realizing personalization. Moreover, the aim of the Semantic Web is a content-aware navigation, because by exploiting knowledge it is possible to increase the precision of the answers, producing an overall behavior that is closer to the user’s intuition and desire. Thus, the Semantic Web not only is a good platform for developing personalization functionalities but it is also deeply connected to the idea of personalization in its very nature.

In the following we will see how the notion of personalization applies to the Semantic Web, overview the expectations, the mechanisms, the languages and tools, and set the state of the art. The paper is organized as follows. Section 2 introduces personalization as the key feature of the Semantic Web. Section 3 reports the state of the art in personalized Web systems, mainly based on the concept of “user model”. Section 4 explains how WWW adaptive systems can evolve in the Semantic Web. Conclusions follow.

## 2 Personalization and the Semantic Web

The objective of the Semantic Web is a content-aware navigation and fruition of the resources. This means being able, by means of proper mechanisms, to identify those resources that better satisfy the requests not only on the basis of descriptive keywords but also on the basis of *knowledge*. There is, in fact, a general agreement that the use of knowledge increases the precision of the answers. Such a knowledge, as we will see, represents different things, information about the user, the user’s intentions, the context. One of the key features that characterize the Semantic Web is that its answers are always personalized or adapted so to meet specific requirements. It will not be the case that the answer to a query about “book” will contain links to bookshops and links to travel agencies. This Web of knowledge is currently being built on top of the more traditional World Wide Web and requires the definition of proper languages and mechanisms. Let us now introduce a few basic concepts.

The first key concept is that of *user model*, that is a machine-interpretable representation of knowledge about the user. The user model, however, may contain different kinds of information; depending on what the user model contains, different reasoning technique might be necessary. Often the user model contains general information, e.g. age, education, etc. In this case, in the tradition of works on personalization, the adaptation occurs at the level of information selection and, especially, presentation. Different users better understand different ways of explaining things. Choosing the best possible communication pattern is fundamental in application systems that supply a kind of information which, because of its nature, might be difficult to understand but that it is important for the user to comprehend. Think, for example, to health-care systems, where medical information is supplied to persons of different age and education. In order for this kind of task to be executed, it is necessary to enrich the data sources and the data itself with semantic information. To this aim, one of the greatest difficulties is to define adequate ontologies.

More and more frequently, however, the Semantic Web is not seen as an information provider but as a *service provider*. This is actually in the line with the latest view of the World Wide Web as a platform for sharing resources and services. We can divide services in two families: “world services” and “web services”. A world service is, for instance, a shop, a museum, a restaurant, whose address, type and description is accessible over the Web. A Web service, instead, is a resource, typically a software device, that can be automatically retrieved and invoked over the Web, possibly by another service.

To begin with, let us consider services of the former kind, world services. The scenarios in which these services are considered adopt user models, in which a different kind of information is considered: the location of the user, which is supposed to vary along time. Typically this information is *not* supplied by the user but it is obtained by the system in a way that is transparent to him/her. In the simplest case, the user (a tourist or a person who is abroad for work) describes in a qualitative way a service of interest, as done with the regular Web browsers. The answer, however, contains only information about world services that are located nearby. The scenario can be made more complex if one adds the time dimension. In this case the user is not necessarily interested in a service that is available now, the system is requested to store the user’s desire and alert the user whenever a matching event occurs, that refers to a service that is nearby. As an example, consider a user who loves classical ballet. He is traveling, and has just arrived at Moscow. After a couple of days he receives an SMS informing him that in the weekend Romeo and Juliet is going to be held at the Boljsoi Theater and that tickets are available. Notice that besides a different kind of information contained in the user model, also the mechanism by which personalization is obtained is very different from the previous case: here the answer changes according to the *context*, in this case given by the position of the user in space and time, and the answer is not always immediately subsequent the query. As we have seen, in fact, a *triggering mechanism* is envisioned that alerts the user whenever an event that satisfies the description occurs. The word “triggering

mechanism” makes one think of a sort of reactive system, nevertheless, many alternatives might be explored and, in particular, inference mechanisms. Moreover, this approach is suitable also to a very different application domain, such as *ambient intelligence*, where appliances are the world services to be handled.

Strongly related to these topics, the recent success of decentralized applications has caused a growing attention towards decentralized approaches to user modeling. In this framework, the target applications include personal guides for navigation or ambient devices, integrated Web-sites (e.g. newspapers), portals (e.g. Yahoo), e-commerce Web-sites (e.g. Amazon), or recommender sites (e.g. MovieLens). In ubiquitous environments distributed sensors follow a user’s moves and, based on the tasks typically performed by him/her, on preferences induced from history and on the specific characteristics of the given context, perform adaptation steps to the ambient-dependent features of the supported functionalities.

As a last observation, when the answer is time-delayed, as described, the descriptions of the services (or more in general, of the events) of interest are sometimes considered as part of the user model. In this case the user model does not contain general information about the user but a more specific kind of information. Alternatively, this can be seen as a configuration problem: I configure a personalized assistant that will warn me when necessary. It is interesting to observe that no-one considers these as queries. An example application is a personalized agenda: the idea is to use an automatic configuration system for filling the agenda of a tourist, taking into account his/her preferences and the advertisements of cultural events in the visited area as they are published. Indeed, filling the agenda could be considered as the topmost level of a system that also retrieves services triggered by events and biased by the user’s location. Observe that this kind of systems should perform also personalization w.r.t. the device by which the user interacts with the system (mobile, laptop).

Many scenarios that refer to world services could naturally be extended so as to include *Web services*. In this case, the meaning of localization should be revised, if at all applicable, while the idea of combining services, as proposed in the case of the tourist agenda, should be explored with greater attention; Web service automatic composition is, actually, quite a hot topic as research in the field proves [20, 5]. Both, Web-service-based and ubiquitous computing applications can be considered as conglomerates of independent, autonomous services developed by independent parties. Such components are not integrated at design time, they are integrated dynamically at run-time, according to the current needs. A frequently used metaphor is a free-market of services where the user buys a complex service, that is composed dynamically on the basis of the available (smaller) services. For example, an e-learning course can be assembled dynamically by composing learning objects stored in independent repositories. The composition is performed so as to satisfy the specific characteristics of the student. For instance, a vision-impaired student will be returned audio materials.

Another, orthogonal, case is the one in which the user model contains (or is accompanied by) the description of what the user would like to *achieve*. There

are situations in which this description cannot be directly related to specific resources or services, but it is possible to identify (or compose) a set of resources so as to satisfy the user's *desires*. In this case a planning process is to be enacted. Sometimes besides the planning process other reasoning techniques are envisioned in order to supply a more complete support to the user. An application domain in which the goal-driven approach seems particularly promising is, once again, *e-learning*. In this case the goal is the learning goal of the user, that is to say a high-level description of the knowledge that s/he would like to acquire, and the plan contains the learning resources that the user should use for acquiring the desired expertise. The whole interaction with the user is supposed to be carried on through a browser. It is important to remark that students are not the only kind of users of this sort of systems. Also teachers should access them but with a different aim. For instance, a teacher might look for learning resources for a new course that s/he will teach. A new notion is, then, introduced, that of *role*. Not only user models contain general or specific information about the users' interests but they also contain the role that the user plays. Depending on the role, different views might be supplied by the system (personalization at the level of presentation) and different actions might be allowed. Rather than being just one of the many features from a user model, the role could, actually, be considered as orthogonal to it (the role is independent from the specific user). Beyond e-learning, the concept of role is useful in many application domains. In health care, there are patients and there are doctors and nurses. In tourism, there are tourists and there are travel agencies.

Another basic concept is that of *domain knowledge*. For understanding the meaning of this word, let us consider the intuitive application case of e-learning. Here the system needs to be supplied with a body of knowledge that not only contains the semantic description of the single learning resources, but it also contains definitions of *more abstract* concepts, not directly related to the courses and defined on the basis of other concepts. This knowledge is used to bias the construction of solutions that make sense from a *pedagogical* point of view. The use of a knowledge of this kind might be exported also to other application domains, whenever similar reasoning techniques are adopted.

Summarizing, the goal of personalization in the Semantic Web is to ease the access to the right resources. This task entails two orthogonal processes: *retrieval* and *presentation*. Retrieval consists in finding or constructing the right resources when they are needed, either on demand or (as by the use of automatic updates) when the information arises in the network. Once the resources have been defined they are presented in the most suitable way to the user, taking into account his/her own characteristics and preferences. To these aims it is necessary to have a model of the user, that is, a representation of those characteristics according to which personalization will occur. It is also necessary to apply inferencing techniques which, depending on the task, might range from the basic ontology reasoning mechanisms supplied by Description Logics (like subsumption and classification) to the most various reasoning techniques developed in Artificial Intelligence.

### 3 Personalization in the World Wide Web

*Personalization in the World Wide Web* can be compared to creating individual views on Web data according to the special interests, needs, requirements, goals, access-context, etc. of the current beholder. The ideas and solutions for creating these individual views are manifold and require interdisciplinary engagement: human computer interaction specialists, e.g. for creating meaningful user interfaces with good usability rankings; artificial intelligence experts, e.g. for mining Web data, or for creating dynamic and accurate models of users; and software engineers for creating generic infrastructure for maintaining personalized views on Web data, and for sufficient user interaction support.

In this article, we focus on those aspects of personalization which aim at improving the selection, access and retrieval of Web resources. The creation of appropriate user interfaces and user awareness is out of scope of this article.

**Definition 1 (Personalizing the access to Web data)** *Personalizing the access to Web data defines the process of supporting the individual user in finding, selecting, accessing, and retrieving Web resources (or meaningful sub-sets of this process).*

With this definition, we can more precisely say that the process of personalization is a process of *filtering the access* to Web content *according to the individual needs and requirements of each particular user*. We can distinguish two different classes of filters: those filter which have been created for a certain *hypermedia system*, and those, which have been created for a *network of Web resources*. The difference between these filters is in the way how they treat the underlying document space: if they have precise information on the structure and relations between the documents (this means the hypertext system), or whether they use dynamics and networking effects in the Web in order to provide individual views on Web data.

The first class of filters has been investigated since the beginnings of the nineties of the last century under the topic of *Adaptive Hypermedia Systems*. The second belongs to *Web Mining* techniques, both Web usage and Web content mining. The personalized systems based on Web mining are often called *recommender systems*, which are in focus of research since the mid-nineties of the last century.

In the following, we describe techniques and methods for personalization in the field of adaptive hypermedia (see Section 3.1), and Web mining (see Section 3.2). Afterwards, we will summarize approaches to user modeling.

#### 3.1 Adaptive Hypermedia Systems

An *adaptive hypermedia system* enlarges the functionality of a hypermedia system. It *personalizes* a hypermedia systems for the individual users: Each user has her or his individual view and individual navigational possibilities for working with the hypermedia system. A general definition of hypertext / hypermedia is given in [58]:

**Definition 2 (Hypertext)** *A set of nodes of text which are connected by links. Each node contains some amount of information (text) and a number of links to other nodes.*

**Definition 3 (Hypermedia)** *Extension of hypertext which makes use of multiple forms of media, such as text, video, audio, graphics, etc.*

Discussions on the definitions of hypertext can be found for example in [24, 47]. The terms hypertext and hypermedia are often synonymous [47]. Throughout this text, we use the term *hypermedia*. For a general, functionality-oriented definition of adaptive hypermedia systems, we follow the proposal of P. Brusilovsky [17].

**Definition 4 (Adaptive hypermedia system)** *“By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user.”*

The support of adaptive methods in hypermedia systems is advantageous if there is *one* common system which serves *many* users with *different* goals, knowledge, and experience, *and* if the underlying hyperspace is relatively *large* [17]. Adaptation of hypermedia systems is also an attempt to overcome the “lost in hyperspace problem” (for a discussion, see for example [47]). The user’s goals and knowledge can be used for limiting the number of available links in a hypermedia system.

**Techniques in Adaptive Hypermedia.** As we have explained, a hypermedia system consists of documents which are connected by links. Thus, there are generally two aspects which can be adapted to the users: the *content* and the links. Let us begin with *content level adaptation*.

By adapting the content to a user, the document is tailored to the needs of the user, for example by hiding too specialized information or by inserting some additional explanations. According to [17], we can identify the following methods for content level adaptation:

- *Additional explanations:* Only those parts of a document are displayed to a user which fit to his goals, interest, tasks, knowledge, etc.
- *Prerequisite explanations:* Here the user model checks the prerequisites necessary to understand the content of the page. If the user lacks to know some prerequisites, the corresponding information is integrated in the page.
- *Comparative explanations:* The idea of comparative explanations is to explain new topics by stressing their relations to known topics.
- *Explanation variants:* By providing different explanations for some parts of a document, those explanations can be selected which are most suited for the user. This extends the method of prerequisite explanations.
- *Sorting:* The different parts of a document are sorted according to their relevance for the user.

The following techniques are used for implementing the above stated adaptation methods [17].

- *Conditional text*: Every kind of information about a knowledge concept is broken into text parts. For each of these text parts, the required knowledge for displaying it to the user is defined.
- *Stretch text*: Some keywords of a document can be replaced by longer descriptions if the user’s actual knowledge requires that.
- *Page or page fragment variants*: Here, different variants of whole pages or parts of them are stored.
- *Frame based technique*: This technique stores page and fragment variants into concept frames. Each frame has some slots which present the page or page fragments in a special order. Certain rules decide which slot is presented to the user.

Content level adaptation requires sophisticated techniques for improved presentation. The current systems using content level adaptation do so by enriching their documents with meta information about prerequisite or required knowledge, outcome, etc. The documents or fragments contained in these systems have to be written more than once in order to obtain the different explanations.

*Link Level Adaptation*. By using link level adaptation, the user’s possibilities to navigate through the hypermedia system are personalized. The following methods show examples for adaptive navigation support.

- *Direct guidance*: Guide the user sequentially through the hypermedia system. Two methods can be distinguished, “next best” and “page sequencing” (or “trails”). The former provides a next-button to navigate through the hypertext. The latter generates a reading sequence through the entire hypermedia or through some part of it.
- *Adaptive sorting*: Sort the links of a document due to their “relevance” to the user. The relevance of a link to the user is based on the system’s assumptions about him/her. Some systems sort links depending on their similarity to the present page. Or by ordering them according to the required prerequisite knowledge. These methods are known as “similarity sorting” and “prerequisite knowledge sorting”.
- *Adaptive hiding*: Limit the navigational possibilities by hiding links to irrelevant information. Hiding of links can be realized by making them unavailable or invisible.
- *Link annotation*: Annotate the links to give the user hints to the content of the pages they point to. The annotation might be text, coloring, an icon, or dimming. The most popular method for link annotation (in the educational area) is the so called “traffic light metaphor”. Here the educational state of a link is estimated by the system due to the user’s actual knowledge state. The link pointing to the page is then annotated by a colored ball. A *red* ball in front of a link indicates that the user lacks some knowledge for understanding the pages; thus the page is not recommended for reading. A

*yellow* ball indicates links to pages that are not recommended for reading; this recommendation is less strict than in case of a red ball. A *green* ball is in front of links which lead to recommended pages. *Grey* balls give the hint that the content of the corresponding page is already known to the user. Variants in the coloring exist. A mix of traffic light metaphor and adaptive hiding is also used in some systems. For an evaluation about adaptive hiding and adaptive navigation we refer to [67].

- *Map annotation*: Here, graphical overviews or maps are adapted with some of the above mentioned methods.

Techniques for link level adaptation depend on the specific system and are, for example, discussed in [17]. Here the assumptions that the system makes about the user play an important role to decide what and how to adapt. Link level adaptation restricts the number of links and thus the number of navigational possibilities. It is useful to prevent the user from “getting lost in hyperspace”. As in the case of content level adaptation, a description of the content of the documents is required for implementing the adaptation tasks.

**Case Study: Adaptive Educational Hypermedia Systems** Adaptive educational hypermedia systems (AEHS) have been developed and tested in various disciplines and have proven their usefulness for improved and goal-oriented learning and teaching. In this section, we propose a component-based logical description of AEHS, in contrast to the functionality-oriented definition 4. This component-based definition is motivated by Reiter’s theory of diagnosis [62] which settles on characterizing systems, observations, and diagnosis in first-order logic (FOL). We decompose adaptive educational hypermedia systems into basic components, according to their different roles: Each adaptive (educational) hypermedia system is obviously a hypermedia system, therefore it makes assumptions about documents and their relations in a *document space*. It uses a *user model* to model various characteristics of individual users or user groups. During runtime, it collects *observations* about the user’s interactions. Based on the organization of the underlying document space, the information from the user model and from the system’s observation, the *adaptive functionality* is provided.

In this section, we will give a logic-based definition for AEHS. We have chosen first order logic (FOL) as it allows us to provide an abstract, generalized formalization. The notation chosen in this paper refers to [64]. The aim of this logic-based definition is to accentuate the main characteristics and aspects of adaptive educational hypermedia.

**Definition 5 (Adaptive Educational Hypermedia System (AEHS))** *An Adaptive Educational Hypermedia System (AEHS) is a Quadruple*

$$(DOCS, UM, OBS, AC)$$

*with*

- DOCS: Document Space:** A finite set of first order logic (FOL) sentences with constants for describing documents (and knowledge topics), and predicates for defining relations between these constants.
- UM: User Model:** A finite set of FOL sentences with constants for describing individual users (user groups), and user characteristics, as well as predicates and rules for expressing whether a characteristic applies to a user.
- OBS: Observations:** A finite set of FOL sentences with constants for describing observations and predicates for relating users, documents / topics, and observations.
- AC: Adaptation Component:** A finite set of FOL sentences with rules for describing adaptive functionality.

The components “document space” and “observations” describe basic data (DOCS) and run-time data (OBS). The user model and adaptation components process this data, e.g. for estimating a user’s preferences (UM), or for deciding about beneficial adaptive functionalities for a user (AC). A collection of existing AEHS, described according to this logic-based formalism, is reported in [35, 34]. In these works a characterization is given of the systems belonging to the first generation of AEHS (e.g. Interbook [18]), to the second generation of adaptive educational hypermedia systems (e.g. NetCoach [71] and KBS Hyperbook [33]), as well as of a recent system, which is also an authoring framework for adaptive educational hypermedia (AHA!2.0 [15]).

To make an example, let us then describe by the above formalism an AEHS, called *Simple*, having the following functionality. *Simple* can annotate hypertext-links by using the traffic light metaphor with two colors: red for non recommended, green for recommended pages. Later, we will extend this system to demonstrate the use (and the usefulness) of a domain model in an AEHS. *Simple* can be modeled by a quadruple  $(DOCS_s, UM_s, OBS_s, AC_s)$ , whose elements are defined as follows:

- $DOCS_s$ : This component is made of a set of  $n$  constants and a finite set of predicates. Each of the constants represents a document in the document space (the documents are denoted by  $D_1, D_2, \dots, D_n$ ). The predicates define pre-requisite conditions, i.e. they state which documents need to be studied before a document can be learned, e.g.  $preq(D_i, D_j)$  for certain  $D_i \neq D_j$  means that  $D_j$  is a prerequisite for  $D_i$ . N.B.: This AEHS does not employ an additional knowledge model.
- $UM_s$ : it contains a set of  $m$  constants, one for each individual user  $U_1, U_2, \dots, U_m$ .
- $OBS_s$ : A special constant (*Visited*) is used within the special predicate *obs* to denote whether a document has been visited:  $obs(D_i, U_j, Visited)$  is the observation that a document  $D_i$  has been visited by the user  $U_j$ .
- $AC_s$ : This component contains constants and rules. One constant (*Recommended\_for\_reading*) is used for describing the values of the “learning\_state” of the adaptive functionality, two constants (*Green\_Icon* and *Red\_Icon*) for representing values of the adaptive functionality. The learning state of a document is described by a set of rules of kind:

$$\begin{aligned} \forall U_i \forall D_j (\forall D_k \text{preq}(D_j, D_k) \implies \\ \text{obs}(D_k, U_i, \text{Visited})) \implies \\ \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading}) \end{aligned}$$

This component contains also a set of rules for describing the adaptive link annotation with traffic lights. Such rules are of kind:

$$\begin{aligned} \forall U_i \forall D_j \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading}) \\ \implies \text{document\_annotation}(D_j, U_i, \text{Green\_icon}) \end{aligned}$$

or of kind:

$$\begin{aligned} \forall U_i \forall D_j \neg \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading}) \\ \implies \text{document\_annotation}(D_j, U_i, \text{Green\_icon}) \end{aligned}$$

We can extend this simple AEHS by using a *knowledge graph* instead of a domain graph. The system, called **Simple1**, is able to give a more differentiated traffic light annotation to hypertext links than **Simple**. It is able to recommend pages (green icon), to show which links lead to documents that will become understandable (dark orange icon), which might be understandable (yellow icon), or which are not recommended yet (red icon). As in the previous case, let us represent **Simple1** by a quadruple  $(DOCS_{s1}, UM_{s1}, OBS_{s1}, AC_{s1})$ ,

- $DOCS_{s1}$ : The document space contains all axioms of the document space of **Simple**,  $DOCS_s$ , but it does not contain any of the predicates. In addition, it contains a set of  $s$  constants which name the knowledge topics  $T_1, T_2, \dots, T_s$  in the knowledge space. It also contains a finite set of predicates, stating the learning dependencies between these topics:  $depends(T_j, T_k)$ , with  $T_j \neq T_k$ , means that topic  $T_k$  is required to understand  $T_j$ .

The documents are characterized by predicate *keyword* which assigns a non-empty set of topics to each of them, so  $\forall D_i \exists T_j \text{keyword}(D_i, T_j)$ , but keep in mind that more than one keyword might be assigned to a same document.

- $UM_{s1}$ : The user model is the same as in **Simple**, plus an additional rule which defines that a topic  $T_i$  is assumed to be learned whenever the corresponding document has been visited by the user. To this aim, **Simple 1** uses the constant *Learned*.

The rule for processing the observation that a topic has been learned by a user is as follows (*p\_obs* is the abbreviation for “processing an observation”):

$$\begin{aligned} \forall U_i \forall T_j (\exists D_k \text{keyword}(D_k, T_j) \wedge \text{obs}(D_k, U_i, \text{Visited})) \\ \implies p\_obs(T_j, U_i, \text{Learned}) \end{aligned}$$

- $OBS_{s1}$ : Are the same as in **Simple**.
- $AC_{s1}$ : The adaptation component of **Simple1** contains two further constants (w.r.t. **Simple**), representing new values for the learning state of a document. Such constants are: *Might\_be\_understandable* and *Will\_become\_understandable* (the meaning is intuitive). Two more constants are added for representing

new values for adaptive link annotation. They are: *Orange\_Icon* and *Yellow\_Icon*. Such constants appear in the rules that describe the *educational state* of a document, reported hereafter.

The first rule states that a document is recommended for learning if *all* the prerequisites to the keywords of this document have already been learnt:

$$\begin{aligned} \forall U_i \forall D_j (\forall T_k \text{keyword}(D_j, T_k) \implies \\ (\forall T_l \text{depends}(T_k, T_l) \implies p\_obs(T_l, U_i, \text{Learned}) \\ \implies \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading}))) \end{aligned}$$

The second rule states that a document might be understandable if at least some of the prerequisites have already been learnt by this user:

$$\begin{aligned} \forall U_i \forall D_j (\forall T_k \text{keyword}(D_j, T_k) \implies \\ (\exists T_l \text{depends}(T_k, T_l) \implies \\ p\_obs(T_l, U_i, \text{Learned}) \\ \wedge \neg \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading}) \\ \implies \text{learning\_state}(D_j, U_i, \text{Might\_be\_understandable}))) \end{aligned}$$

The third rule entails that a document will become understandable if the user has some prerequisite knowledge for at least one of the document's keywords:

$$\begin{aligned} \forall U_i \forall D_j (\exists T_k \text{keyword}(D_j, T_k) \implies \\ (\exists T_l \text{depends}(T_k, T_l) \implies \\ p\_obs(T_l, U_i, \text{Learned}) \\ \wedge \neg \text{learning\_state}(D_j, U_i, \text{Might\_be\_understandable}) \\ \implies \text{learning\_state}(D_j, U_i, \text{Will\_become\_understandable}))) \end{aligned}$$

Four rules describe the adaptive link annotation:

- 1)  $\forall U_i \forall D_j \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading}) \implies \text{document\_annotation}(D_j, U_i, \text{Green\_Icon})$
- 2)  $\forall U_i \forall D_j \text{learning\_state}(D_j, U_i, \text{Will\_become\_understandable}) \implies \text{document\_annotation}(D_j, U_i, \text{Orange\_Icon})$
- 3)  $\forall U_i \forall D_j \text{learning\_state}(D_j, U_i, \text{Might\_be\_understandable}) \implies \text{document\_annotation}(D_j, U_i, \text{Yellow\_Icon})$
- 4)  $\forall U_i \forall D_j \neg \text{learning\_state}(D_j, U_i, \text{Recommended\_for\_reading}) \implies \text{document\_annotation}(D_j, U_i, \text{Red\_Icon})$

**Discussion: Why a logical characterization of adaptive (educational) hypermedia is needed.**

With Brusilovsky's definition of adaptive hypermedia, we can describe the general functionality of an *adaptive* hypermedia system, and we can compare which kind of adaptive functionality is offered by such a system.

In the literature, we can find reference models for adaptive hypermedia, e.g. the AHAM Reference Model [16], or the Munich Reference Model [43]. Both the AHAM and Munich Reference models extend the Dexter Hypertext Model [31], and provide a framework for describing the different components of adaptive

hypermedia systems. In both cases, the focus is posed on process modeling and on the engineering of adaptive hypermedia applications, so we can say that these models are *process-oriented*.

However, a formal description of adaptive educational hypermedia, which allows for a system-independent characterization of the adaptive functionality, is still missing. Currently, we cannot answer a request like the following: “I want to apply the adaptive functionality X in my system. Tell me what information is required with the hypermedia-documents, which interactions at runtime need to be monitored, and what kind of user model information and user modeling is required”. At the moment, we can only describe the functionality with respect to a *specific* environment, which means we can describe the functionality only in terms of the system that implements it. We cannot compare different implementations nor can we benchmark adaptive systems. A benchmark of adaptive systems would require at least a comparable initial situation, observations about a user’s interactions with the system during some defined interaction period, before the *result* of the system is returned, the *adaptive functionality* as well as the changes in the user model.

The logical definition of adaptive educational hypermedia given here focuses on the components of these systems, and describes which kind of processing information is needed from the underlying hypermedia system (the document space), the runtime information which is required (observations), and the user model characteristics (user model). The adaptive functionality is then described by means of these three components, or more precisely: how the information from these three components, the static data from the document space, the runtime-data from the observations, and the processing-data from the user model, is used to provide the adaptive functionality. The aim of this logical definition of adaptive educational hypermedia is to provide a *language* for describing the adaptive functionality, to allow comparison of adaptive functionality in a well-grounded way, and to enable the re-use of an adaptive functionality in different contexts and systems.

There is, actually, a need for a formalism expressing adaptive functionalities in a system-independent and re-usable manner, which allows their application in various contexts. In the educational context, a typical scenario where re-usable adaptive functionality is required would be: Imagine a learner who wants to learn a specific subject. The learner registers to some learning repository, which stores learning objects. According to his/her current learning progress, some of the learning objects which teach the subject s/he is interested in are useful, some of them require additional knowledge that the learner does not have so far (in accordance to his/her user model), and some might teach the subject only on the surface and are too easy for this learner. This kind of situation has been studied in adaptive educational hypermedia in many applications, and with successful solutions. However, these solutions are specific to certain adaptive hypermedia applications, and are hardly generalizable for re-use in different applications. Another reason why the adaptive functionality is not re-usable today is related to the so-called *open corpus problem* in adaptive (educational) hypermedia [36,

19], which states that currently, adaptive applications work on a fixed set of documents which is defined at the design time of the system, and directly influences the way adaptation is implemented, e.g. that adaptive information like “required prerequisites” is coded on this fixed set of documents.

### 3.2 Web Mining

In contrast to the approach in adaptive hypermedia, personalization with aid of Web mining does not work on such well-defined corpora like a hypertext system. Instead, it uses effects and dynamics in the network structure in order to detect (virtual) relations between Web resources.

The World Wide Web is seen as the *Web graph*. In this graph, Web resources are the nodes, and links between the Web resources are the edges. NB: as it is practically impossible to create a complete snapshot of the World Wide Web at a certain time point, this Web graph is not a completely known structure. On the contrary, in the case of adaptive hypermedia systems, the underlying hypermedia graph models completely the hypertext.

The approaches in Web Mining-based personalization are centered around detecting relations between Web resources. These relations can be *existing relations*, this means hyperlinks between Web resources, or *virtual relations*, this means that two or more Web resources are related to each other but are not connected via some hyperlink. These existing or virtual relations between Web resources are *mined* on basis of the Web graph. We can distinguish two main approaches for detecting the relations: *Mining based on the content* of the Web resources, or *mining based on the usage* of the Web resources. The two approaches can of course be combined.

Normally, Web Mining-based personalization has no external models like domain or expert models, as those used in adaptive hypermedia, but instead create dynamic models which grow with the number of Web resources integrated into the model.

**Recommendation Techniques for Web Mining.** In the following, we summarize major recommendation techniques according to Burke [21]. We can distinguish between *content-based*, *collaborative*, *demographic*, *utility-based*, and *knowledge-based* recommendations. Let  $U$  and  $I$  be respectively a set of users and a set of items, and  $U$  denotes an individual user. Let us outline these techniques.

- *Content-based recommendation*:
  - each user is assumed to operate independently of other users;
  - recommendations can exploit information derived from document contents;
  - The system builds user models in the following way: initially, users apply *candidate profiles* against their own preferences. For example, a candidate user profile for the rating of today’s news article is presented, the user can accept or reject the ratings for the articles. The profile is maintained by exploiting keywords and content descriptors which contribute to the rating of each article.

- The quality of the learnt knowledge is measured against the classical measures of Information Retrieval, i.e. precision and recall (see e.g. [4]).
  - The typical background consists of features of items in  $I$ , the typical input to the mining process consists of the user's ratings of some items in  $I$ . A learning process is enacted that generates a classifier fitting the user's preferences, expressed by the ratings. The constructed classifier is applied to all the items in  $I$ , for finding out which might be of interest.
  - limitations:
    - \* as in all inductive approaches, items must be machine-parsable or with assigned attributes;
    - \* only recommendations based on what the user has already seen before (and indicated to like) can be taken into account but negative information is as well important;
    - \* stability vs. plasticity of recommendations;
    - \* no filtering based on quality, style, or point-of-view (only based on *content*);
- *Collaborative recommendations (social information filtering)*
- This technique is basically a process of “word-of-mouth”, in fact the items are recommended to a user based upon values assigned by other people with *similar taste*. The underlying hypothesis is that people's tastes are not randomly distributed: there are general trends and patterns within the taste of a person as well as between groups of people. Also in this case a *user model* is to be built. To this aim the users are initially required to explicitly rank some sample objects.
  - The input used for computing the predictions is a set of “Ratings of *similar users*”, where the similarity is measured on the basis of the user profile values.
  - The mining process begins with the identification of those users in  $U$  that result similar to  $U$ , and extrapolates the possible interests and likings of the user at issue from the ratings that similar users gave to items in  $I$ .
  - Limitations:
    - \* a critical mass of users is required before the system can make recommendations;
    - \* how to get the first rating of a new object?
    - \* stability vs. plasticity of recommendations.

Demographic recommendation, utility-based recommendation and knowledge-based recommendation are variants which require additional data about the user beyond rating of items:

– *Demographic recommendations*

In this case, demographic information about all the users in  $U$  is exploited: similarly to the previous case, the users that are close to  $U$  are identified, but in this case similarity is computed on the demographic data. Then, the ratings of these users on items in  $I$  are used to produce recommendations to the user at issue.

– *Utility-based recommendations*

In this case the preferences of  $\mathcal{U}$  are coded by a *utility function*, which is applied to all the items in  $I$  for defining recommendations.

– *Knowledge-based recommendations*

The knowledge-based approach to recommendation works on a description of the user’s needs and on a body of knowledge that describes how items can meet various needs. An inferencing process is used to match the description of the user’s needs with the items that can help the user and, thus, are to be recommended.

**Case Study: Web Usage Mining in an online shop.** In this case study, we will see how we can improve selling strategies in an artificial online shop. Our online shop sells a small variety of products. Our goal is to find out which items are commonly purchased together in order to make for example some selected frequent-customers special bundle-offers which are likely to be in their interest.

To detect relations between data items, the concept of *association rules* can be used. Association rules aim at detecting *uncovered relations* between data items, this means relationships which are not inherent in the data like functional dependencies, and normally do not necessarily represent a sort of causality or correlation between the items. A database in which an association rule is to be found is viewed as a set of tuples: each tuple contains a set of items; the items represent the items purchased, and the tuples denote the list of items purchased together. For a definition of association rules, we follow [26]:

**Definition 6 (Association Rule)** *Given a set of items  $I = \{I_1, I_2, \dots, I_m\}$  and a database of transactions  $D = \{t_1, t_2, \dots, t_n\}$  where  $t_i = \{I_{i1}, I_{i2}, \dots, I_{ik}\}$  and  $I_{ij} \in I$ , an **association rule** is an implication of the form  $X \implies Y$ , where  $X, Y \subset I$  are sets of items classed itemsets and  $X \cap Y = \emptyset$ .*

To identify the “important” association rules, the two measures *support* and *confidence* are used (see [26]):

**Definition 7 (Support)** *The **support (s)** for an association rule  $X \implies Y$  is the percentage of transactions in the database that contain  $X \cup Y$ .*

$$\mathbf{support}(X \implies Y) = \frac{|\{t_i \in D : X \cup Y \subset t_i\}|}{|D|}$$

**Definition 8 (Confidence / Strength)** *The **confidence** or **strength ( $\alpha$ )** for an association rule  $X \implies Y$  is the ratio of the number of transactions that contain  $X \cup Y$  to the number of transactions that contain  $X$ .*

$$\mathbf{confidence}(X \implies Y) = \frac{|\{t_i \in D : X \cup Y \subset t_i\}|}{|\{t_i \in D : X \subset t_i\}|}$$

The support measures how often the rule occurs in the database, while the confidence measures the strength of a rule. Typically, large confidence values and

smaller support values are used, and association rules are mined which satisfy at least a *minimum support* and a *minimum confidence*.

The hard part in the association rule mining process is to detect the high-support (or *frequent*) item-sets. Computationally less costly is then the checking of the confidence. Algorithms for uncovering frequent item-sets exist in the literature [26], most prominent is the Apriori-algorithm [1], which uses the property of frequent itemsets that all subset of a frequent itemset must be frequent, too.

*Example: An online Book Shop* This (artificial) online book shop sells five different books: Semantic Web, Winnie the Pooh, Data Mining, Faust, and Modern Statistics.

Transaction	Items
t1	Semantic Web, Winnie the Pooh, Data Mining
t2	Semantic Web, Data Mining
t3	Semantic Web, Faust, Data Mining
t4	Modern Statistics, Semantic Web
t5	Modern Statistics, Faust

Customer  $X$  is a very good customer, and to tighten the relationship to customer  $X$ , we want to make a personal and attractive offer. We see him ordering a book on “Semantic Web”. Which bundle offer might be interesting for him? Which book shall we offer to a reduced price: Winnie the Pooh, Data Mining, Faust, or Modern Statistics? We are looking for association rules which have a minimum-support of 30% and a confidence of 50%. The association rules we are interested in are thus :

- Semantic Web**  $\implies$  **Data Mining** support: 60%, confidence: 75 %
- Semantic Web**  $\implies$  **Faust** support: 20%, confidence: 25%
- Semantic Web**  $\implies$  **Winnie the Pooh** support: 20%, confidence: 25 %
- Semantic Web**  $\implies$  **Modern Statistics** support: 20%, confidence: 25 %

An often seen pattern is that the books “Semantic Web” and “Data Mining” are bought together, and the association rule “Semantic Web  $\implies$  Data Mining” satisfies the minimum support of 30%. In 60% of the cases in which customers bought the book “Semantic Web”, they also bought the book “Data Mining” (confidence: 60%). Thus, we decide to offer our valuable customer the book “Data Mining” in a personal offer for an attractive price.

NB: The general “association rule problem” is to mine association rules which satisfy a given support and confidence; in the above example, we simplify the approach by asking whether a certain item is obtained in some association rule.

### 3.3 User Modeling

In a user model, a system’s estimations about the preferences, often performed tasks, interests, and so forth of a specific end user (or group of users) are specified (in the following, we will only refer to “the user” wherever a single user a sufficient homogeneous group of users can be meant). We can distinguish between

the *user profile* and the *user model*. A *User profile* provides access to certain characteristics of a user. These characteristics are modeled as attributes of the user. Thus, a user profile of user  $\mathcal{U}$  gives the instantiations of attributes for  $\mathcal{U}$  at a certain timepoint  $t$ . Instead, the task of the *user model* is to ascertain the values in the user profile of a user  $\mathcal{U}$ . Thus, the user model must provide updating and modification policies of the user profile, as well as instructions to detect and evaluate incidents which can lead to update or modification processes. Methods for drawing appropriate conclusions about the incidents must be given, as well as mechanisms for detecting discrepancies in the modeling process. Advanced user modeling approaches also provide mechanisms for dealing with uncertainty in the observations about a user, appropriate error detection mechanisms, and can prioritize the the conclusion on observed incidents.

A very simple user profile identifies all the pages that a user  $\mathcal{U}$  has visited, therefore, it is a set of couples:

$$(\mathcal{P}, \textit{visited})$$

A simple user model which can create this-like user profiles contains the following rule for interpreting incidents:

*“if  $\mathcal{U}$  visits page  $\mathcal{P}$  then insert  $(\mathcal{P}, \textit{visited})$  into the user profile of  $\mathcal{U}$  ”*

An extension of this simple user model is to recognize the observation that a user  $\mathcal{U}$  has bookmarked some page  $\mathcal{P}$  and note this in the user profile:

*“if  $\mathcal{U}$  bookmarks page  $\mathcal{P}$  then insert  $(\mathcal{P}, \textit{important})$  into the user profile of  $\mathcal{U}$  ”*

We will not go into detail on user modeling in this article (for more in-depth information refer to [41]). But even from this simple user models above, we can see that interpretation about the user interactions is not at all an easy task. E.g. if we observe a user  $\mathcal{U}$  bookmarking a page  $\mathcal{P}$ : How can we distinguish that  $\mathcal{U}$  has stored this page for future reference based on the content of the page from the fact that  $\mathcal{U}$  stored this page only because he liked the design of the page? Can we really be sure that bookmarking expresses favour for a page in contrast to denial? Appropriate mechanisms for dealing with uncertainty in the observations about the user, and for continuous affirmation of derived conclusions are essential for good user models (a good reference for studying numerical uncertainty management in user modeling is e.g. given in [38]).

User modeling approaches for *Adaptive Hypermedia* can take advantage of the underlying hypermedia structure or the domain models associated with the hypermedia system. Task models, expert models, or other, external models are used to model the user with respect to this external model. This approach is called *overlay modeling* [30]. As an example, for educational hypermedia systems, the learner’s state of knowledge is described as a subset of the expert’s knowledge of the domain, hence the term “overlay”. Student’s lack of knowledge is derived by comparing it to the expert’s knowledge.

The critical part of overlay modeling is to find the initial knowledge estimation. The number of observations for estimating the knowledge sufficiently

well must be small. In addition, a student's misconceptions of some knowledge concepts can not be modeled. A great variety of approaches for user modeling is available, see e.g. [42, 69]

*User Modeling for Web Mining* For Web Mining, the absence of a structured corpus of documents leads to different approaches for user modeling. An interest and/or content-profile of a user is generated (with the aid of classification or clustering techniques from machine learning) based on observations about the user's navigation behaviour. A *stereotype user modeling* approach [63] classifies users into *stereotypes*: Users belonging to a certain class are assumed to have the same characteristics. When using stereotype user modeling, the following problem can occur: the stereotypes might be so specialized that they become obsolete (since they consist of at most one user), or a user cannot be classified at all.

**Discussion** The user modeling process is the core of each personalization process, because here the system's estimations about the user's needs are specified. If the system identifies the needs not correctly, the personalization algorithms - regardless how good they are - will fail to deliver the expected results for this erroneous modeled user.

### 3.4 Conclusion: Personalization in the World Wide Web

To develop systems which can filter information according to the requirements of the individual, which can learn the needs of users from observations about previous navigation and interaction behavior, and which can continuously adapt to the dynamic interests and changing requirements is still one of the challenges for building smart and successful Web applications. Although the necessity to "support the users in finding what they need at the time they want" is obvious, building and running personalized Web sites is still a cost-intensive venture which sometimes underachieves [40].

Looking at the techniques in adaptive hypermedia, we can see that re-usability of these techniques is still an unsolved problem. We require a formalism expressing adaptive functionality in a system-independent and re-usable manner, which allows us to apply this adaptive functionality in various contexts, as it has been done e.g. for the adaptive educational hypermedia systems (see Section 3.1). Another reason why adaptive functionality is not re-usable today is related to the so-called *open corpus problem* in adaptive hypermedia, which states that currently, adaptive applications work on a fixed set of documents which is defined at the design time of the system, and directly influences the way adaptation is implemented, e.g. that adaptive information like "required prerequisites" is coded on this fixed set of documents. The introduction of standards for describing such metadata is a step forwards - and is currently undertaken in the Semantic Web.

Looking at the personalization techniques based on Web mining, we can see that the filtering techniques (content-based, collaborative-based, demographic-based, utility-based, knowledge-based, or others) are limited as they require a

critical mass of data before the underlying machine learning algorithms produce results of sufficient quality. Explicit, machine-readable information about single Web resources as given in the Semantic Web could be used for improving the quality of the input data for the algorithms.

## 4 Personalization for the Semantic Web

Functionalities for performing personalization require a machine-processable knowledge layer that is not supplied by the WWW. In the previous section we have studied techniques for developing adaptive systems in the WWW with all the difficulties and limitations brought by working at this level. Let us now see how adaptive systems can evolve benefiting of the Semantic Web. In particular, since the capability of performing some kind of inferencing is fundamental for obtaining personalization, let us see how the introduction of machine-processable semantics makes the use of a wide variety of *reasoning techniques* possible, thus widening the range of the forms that personalization can assume.

### 4.1 An overview

The idea of exploiting reasoning techniques for obtaining adaptation derives from the observation that in many (Semantic Web) application domains the goal of the user and the interaction occurring with the user play a fundamental role. Once the goal to be achieved is made clear, the system strives for achieving it, respecting the constraints and the needs of the user and taking into account his/her characteristics. In this context, the ability of performing a semantic-based retrieval of the necessary resources, that of combining the resources in a way that satisfies the user's goals, and, if necessary, of remotely invoking and monitoring the execution of a resource, are fundamental. All these activities can be performed by adopting automated reasoning techniques. To make an example, suppose that, for some reason, a student must learn something about the Semantic Web for a University course. Suppose that the student has access to a repository of educational resources that does not contain any material under the topic "Semantic Web". Let us suppose, however, that the repository contains a lot of information about XML-based languages, knowledge representation, ontologies, and so forth: altogether this information gives knowledge about the Semantic Web, the problem is retrieving it. A classical search engine would not be able to do it, unless the word "Semantic Web" is explicitly contained in the documents. This result can be obtained only by a system that is able to draw as an inference the fact that all these topics are elements of the Semantic Web.

In the Semantic Web every new feature or functionality is built as a new layer that stands on top of the previous ones. Tim Berners-Lee has described this process and structure as the "Semantic Web Tower". In this representation reasoning belongs to the logic and proof layers that lay on the ontology layer. This vision allows the Semantic Web to be developed incrementally.

*Data* on the Web is basically considered as the set of the available Web resources, each identified by a URI (uniform resource identifier). Such resources are mainly represented by plain XML (eXtensible Markup Language) descriptions. XML stands at the bottom of the tower. It allows a Web document to be written in a structured way, exploiting a user-defined vocabulary. It is perfect as a data interchange format, however, it does not properly supply any semantic information. Sometimes, when the domain is very closed and controlled, the tags can be considered as being associated with a meaning but the solution is risky and the application as such cannot be safely extended.

*Semantic annotation of data* is done by means of RDF (Resource Description Framework). RDF [59] is the basic Semantic Web (XML-based) language for writing simple statements about Web resources. Each statement is a binary predicate that defines a relation between two resources. These predicates correspond to logical facts. Given semantically-annotated data it is possible to perform some kinds of reasoning. In particular, some query languages have been developed that allow the automatic transformation of RDF-annotated data. Two of the main query languages that are used to transform data encoded in RDF are TRIPLE and RDQL. They are both quite simple in the inferencing that they allow.

TRIPLE [66] is a rule language for the Semantic Web which is based on Horn logic and borrows many basic features from F-Logic but is especially designed for querying and transforming RDF models. In contrast to procedural programming languages such as C or Java, it is a declarative language which shares some similarities with SQL or Prolog. TRIPLE programs consist of facts and rules, from which it is possible to draw conclusions for answering queries. The language exploits reasoning mechanism about RDF-annotated information resources; translation tools from RDF to TRIPLE and vice versa are provided. An RDF statement, i.e. a “triple”, is written as `subject[predicate -> object]`. RDF *models* are explicitly available in TRIPLE: statements that are true in a specific model are written as “@model”. Connectives and quantifiers (e.g. AND, OR, NOT, FORALL, EXISTS) for building logical formulae from statements are allowed as usual.

RDQL [61] is a query language for RDF and is provided as part of the Jena Semantic Web Framework [39] from HP labs, which also includes: an RDF API, facilities for reading and writing RDF in RDF/XML, N3 and N-Triples, an OWL API, and in-memory and persistent storage. RDQL provides a *data-oriented* query model so that there is a more declarative approach to complement the fine-grained, procedural Jena API. It is “data-oriented” in that it only queries the information held in the models; there is no inference being done. Of course, the Jena model may be “smart” in that it provides the impression that certain triples exist by creating them on-demand. However, the RDQL system does not do anything other than take the description of what the application wants, in the form of a query, and returns that information, in the form of a set of bindings.

Going back to *our example*, by using RDF we could semantically annotate the resources that give explanations about XML-based languages, ontologies, knowledge representation, etc. However, the use that we want to do of such