Personalization for the Semantic Web*

Matteo Baldoni[1], Cristina Baroglio[1], and Nicola Henze[2]

[1] Dipartimento di Informatica, Università degli Studi di Torino,
c.so Svizzera 185, I-10149, Torino, Italy
{baldoni, baroglio}@di.unito.it
[2] 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 as 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, the working process s/he is involved in, the level of expertise, the device s/he is using to

* This research has partially been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net). Matteo Baldoni and Cristina Baroglio have also been supported by MIUR Cofin 2003 "Logic-based development and verification of multi-agent systems (MASSiVE)" national project.

N. Eisinger and J. Maluszynski (Eds.): REWERSE 2005, LNCS 3564, pp. 173–212, 2005.

display the information, the bandwidth and availability of the communication channel, the abilities (disabilities or handicaps) of the user, his/her time constraints, and many, many more. 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, and so forth.

Many attempts have been done to apply personalization techniques to the *World Wide 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. Over this knowledge layer, the use of *inferencing mechanisms* is envisioned as a fundamental means for performing a content-aware navigation, producing an overall behavior that is closer to the user's intuition and desire. This is the reason why the Semantic Web is the most appropriate environment for realizing personalization. In other words, the Semantic Web is *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 take advantage of the Semantic Web. Conclusions follow.

2 Demands of Personalization in 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 and education. 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. 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 make easier 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 subsump-

tion 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 about 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 with respect 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: Doc*ument* **S***pace: 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: U*ser* **M***odel: 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: Obs*ervations: A finite set of FOL sentences with constants for describing observations and predicates for relating users, documents/topics, and observations.*

AC: A*daptation* **C***omponent: 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 [36, 35]. 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 [34]), 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, …, 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, …, 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:

$$\forall U_i \forall D_j (\forall D_k preq(D_j, D_k) \Longrightarrow obs(D_k, U_i, Visited)) \Longrightarrow learning\_state(D_j, U_i, Recommended\_for\_reading)$$

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

$$\forall U_i \forall D_j learning\_state(D_j, U_i, Recommended\_for\_reading) \Longrightarrow document\_annotation(D_j, U_i, Green\_icon)$$

or of kind:

$$\forall U_i \forall D_j \neg learning\_state(D_j, U_i, Recommended\_for\_reading) \Longrightarrow document\_annotation(D_j, U_i, Green\_icon)$$

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, ..., 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 $\forall D_i \exists T_j keyword(D_i, T_j)$, 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"):

$$\forall U_i \forall T_j (\exists D_k keyword(D_k, T_j) \wedge obs(D_k, U_i, Visited) \Longrightarrow p\_obs(T_j, U_i, Learned)$$

- 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:

$$\forall U_i \forall D_j (\forall T_k keyword(D_j, T_k) \Longrightarrow (\forall T_l depends(T_k, T_l) \Longrightarrow p\_obs(T_l, U_i, Learned) \Longrightarrow learning\_state(D_j, U_i, Recommended\_for\_reading)))$$

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

$$\forall U_i \forall D_j (\forall T_k keyword(D_j, T_k) \Longrightarrow (\exists T_l depends(T_k, T_l) \Longrightarrow p\_obs(T_l, U_i, Learned) \wedge \neg learning\_state(D_j, U_i, Recommended\_for\_reading) \Longrightarrow learning\_state(D_j, U_i, Might\_be\_understandable)))$$

  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:

$$\forall U_i \forall D_j (\exists T_k keyword(D_j, T_k) \Longrightarrow (\exists T_l depends(T_k, T_l) \Longrightarrow p\_obs(T_l, U_i, Learned) \wedge \neg learning\_state(D_j, U_i, Might\_be\_understandable) \Longrightarrow learning\_state(D_j, U_i, Will\_become\_understandable)))$$

  Four rules describe the adaptive link annotation:

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

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 [33, 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 $\mathcal{U}$ 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 $\sqcap$, 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 $\mathcal{U}$ 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* $I = \{I_1, I_2, \ldots, I_m\}$ *and a database of transactions* $D = \{t_1, t_2, \ldots, t_n\}$ *where* $t_i = \{I_{i1}, I_{i2}, \ldots I_{ik}\}$ *and* $I_{ij} \in I$*, an* **association rule** *is an implication of the form* $X \Longrightarrow Y$*, 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* $X \Longrightarrow Y$ *is the percentage of transactions in the database that contain* $X \cup Y$*.*

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

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

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

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 $\Longrightarrow$ **Data Mining** support: 60%, confidence: 75 %
Semantic Web $\Longrightarrow$ **Faust** support: 20%, confidence: 25%
Semantic Web $\Longrightarrow$ **Winnie the Pooh** support: 20%, confidence: 25 %
Semantic Web $\Longrightarrow$ **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 $\Longrightarrow$ 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 prioterize 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:

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

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* $(\mathcal{P}, visited)$ *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* $(\mathcal{P}, important)$ *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 favor 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 behavior. 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 resources requires that each of them is explicitly associated to every topic it might have correlations with. This should be done even though some of the topics are related with each other, for instance XML is related to RDF and XML is related to Semantic Web, but also RDF is related to Semantic Web, and ideally we could *exploit such relations* to infer properties of the available resources. What is still missing is the possibility of *expressing knowledge about the domain*.

RDF Schema [60] adds a new layer of functionalities by allowing the representation of *ontologies*. This is done by introducing the notion of "class" of similar resources, i.e. objects showing a set of same characteristics. Resources are then viewed as "individuals" of some class. Classes can be divided in "subclasses", the result is a *hierarchical structure*. From an extensional point of view, every instance of a class is also an instance of its super-class, as such it inherits the properties of that class. It is possible to exploit this mechanism to perform simple inferences about instances and classes w.r.t. the hierarchical structure. A more powerful ontology language is OWL [54] (Web Ontology Language). OWL, the W3C standard for ontology representation, builds on top of RDF and RDF-S and allows the representation of more complex relations, such as transitivity, symmetry, and cardinality constraints.

It is possible to reason about ontologies by means of techniques that are typical of Description Logics. Basically, such techniques are aimed at *classification*, that is, if a resource is an instance of a class, then it will also be an instance of its super-classes. Also, if a resource satisfies a set of properties that define a sufficient condition to belonging to a given class, then the resource is an instance of that class. By means of these techniques we can satisfy the goal of the user of our example: in fact, if we have an ontology in which Semantic Web has as subclasses XML-based languages, knowledge representation, and so on, and we have a set of resources that are individuals of such classes, it is possible to infer that they are also individuals of Semantic Web. The introduction of these inferencing mechanisms is a fundamental step towards personalization, although in order to have real personalization something more is to be done. Indeed, if two different users are both interested in the Semantic Web the system will return as an answer *the same set* of resources because it does not take into account any information about them.

So far, reasoning in the Semantic Web is mostly reasoning about knowledge expressed in some ontology and the ontology layer is the highest layer of the Semantic Web tower that can be considered as quite well assessed. The layers that lie on top of it, in particular the logic layer and the proof layer, are still at a primitive level. The lesson learnt from the analysis that we have done is that for making some personalization we need to represent and reason about *knowledge* and the Semantic Web offers this possibility. Let us, then, see what kinds of knowledge are necessary for performing personalization.

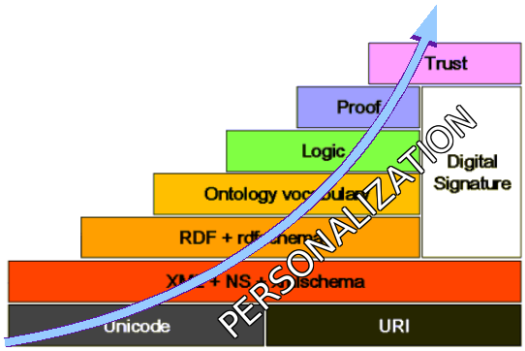

Fig. 1. The Semantic Web tower. Personalization occurs at the ontology layer but mostly at the logic and proof layers

4.2 Knowledge and Reasoning About Knowledge

A system that performs some kind of personalization needs to represent different kinds of *knowledge*: knowledge about the user, knowledge about the user's purpose (sometimes considered as included in the user's description), knowledge about the context, knowledge about the resources that can be queried, retrieved or composed, and domain knowledge that is used by the inferencing mechanism for obtaining personalization.

Knowledge about the user can roughly be viewed as partitioned in generic knowledge about the user's characteristics and preferences and in "state" knowledge. By the word "state knowledge" we hereby mean information that can change and that is relevant w.r.t. a specific application system, such as which exams have been passed in the case of e-learning.

A user's *goal* most of the times is considered as being coincident with a query but there are some distinctive features to take into account. First of all, a query presupposes an answer, and it implies a selection process, that can be performed by means of the most various techniques. The answer is supposed to be returned within a few seconds. In some applications, however, the goal corresponds to a general interest of the user. For example, the user might be a fan of a given music band and whenever the band performs in the user's town, s/he would like to be informed automatically. In this case, we can view the goals as conditions that can are embedded in rules: when some event satisfies a rule condition, the rule is triggered and, typically, the user is warned in a way that can be subject to further personalization (e.g. w.r.t. the physical device that is used –laptop, mobile, hand-held–). In this case, the answer, that depends on location and time, might be returned days or weeks after the rule has been set. Moreover, the same rule might be activated many times by many different events. A third kind of goal, that we have seen, is more abstract and not directly interpretable as a query. It is, for instance, the case of a learning goal: a learning goal is a description of the expertise that a user would like to acquire. The system uses this information

to build a solution that contains many Web resources, to be used as learning materials. None of them is (possibly) directly tied with the learning goal; the goal will be reached by the user if s/he will follow the proposed reading path. In other words, the composition of resources is a *means* for reaching the goal.

In performing resource selection, also knowledge about the *context* plays a very important part. In many applications, three kinds of contextual information can be identified: location in time, location in space, and role. *Location in time and space* is used for refining resource selection, that is, only those resources that fit the context description, are shown. The context description is not necessarily expressed by the user, since it might as well be obtained in other ways. In ubiquitous and in ambient computing it could be returned by a sensor network. *Roles* are predefined views, possibly with a limitation of the actions, that the role players can execute. They are used to personalize the selection of information sources, the selection of information and, of course, presentation.

For performing semantic-based processing on the Web it is necessary that the *Web resources* are semantically annotated. This is normally done by means of ontologies. Even though semantic annotation is not so much diffused, the languages for writing such annotations are pretty well assessed. One of the major difficulties is, actually, to retrieve –if any– an ontology that is suitable to the application at hand, avoiding to write a new one unless really necessary.

The last kind of knowledge that is often necessary in personalization tasks, that we called *domain knowledge*, is aimed at giving a structure to the knowledge. Domain knowledge relates the ontological terms in a way that can be exploited by other inferencing mechanisms, and not only to perform ontological reasoning. For instance, planning is a useful reasoning technique for obtaining personalization; there are proposals in the literature that suggest to bias the search of a plan by introducing solution schemas, that correspond to abstract descriptions of solutions that "make sense". For instance, in the e-learning applications when a course is constructed out of a set of available learning materials, the course must "make sense" also from a pedagogical point of view, see [7]. One can then imagine to have a high-level description of the structure of interest, not related to specific materials, which is personalized and filled with contents on demand, in a way that fits the specific user. Moreover, in many scenarios it is useful to express some event-driven behavior (e.g. in the already mentioned touristic application domain). It is especially at this level that rules can play a fundamental role in the construction of personalization systems in the Semantic Web.

Beyond Ontologies: Some Examples. The first scenario that we consider is set in one of the leading application areas for personalization: education. The most typical problem in this framework consists in determining an "optimal reading sequence" through a hyper-space of learning objects (a learning object is a resource with educational purposes). The word optimal does not mean that this is absolutely the best solution, it means that it specifically fits the characteristics and the needs of the given user. It is optimal for that user. So the aim is to support the user in the acquisition of some desired knowledge by identifying a reading path that best fits him/her. Considerable advancements have been yield

in this field, with the development of a great number of Web-based systems, like ELM-Art [70], the KBS hyperbook system [34], TANGOW [22], WLog [6] and many others, based on different, adaptive and intelligent technologies.

Different methods have been proposed on how to determine which reading path to select or to generate in order to support in the best possible way the learner's navigation through the hyper-space. All of them require to go *one step beyond* the ontology layer. In fact, pure ontological annotation and ontological reasoning techniques (though necessary) are not sufficient to produce, in an automatic way, the desired sequencing. If in our ontology the class "Semantic Web" is divided in the classes "XML-based languages", "knowledge representation", and "ontologies" we will be able to conclude that each of the individuals that belong to the sub-classes also belong to the super-class. What we cannot do is to impose that the student will be presented resources about *all* such topics, because only the conjunction of the three will let him/her satisfy his/her learning goal. Another thing that we cannot do is to impose that a given topic is presented before another one because only in this way the student will understand them.

If, on the one hand, it is necessary to annotate the actual learning objects, with the ontological terms that represent identifiable pieces of knowledge related to the learning objects themselves, on the other, it is also necessary to structure a domain knowledge in a way that it is possible to perform the personalization task. The desire is to develop an *adaptation component*, that uses such a knowledge, together with a representation of the user's *learning goal* and of knowledge about the user, for producing sequences that fit the user's requirements and characteristics, based on the available learning objects. Such an adaptation component exploits knowledge representations that are not ontologies (though they use ontologies) and it exploits reasoning mechanisms that are not ontological reasoning mechanisms. For instance, in the application domain that has been taken into account, *goal-directed reasoning* techniques seem particularly suitable.

To this purpose, one solution is to interpret the learning resources as atomic actions. In fact, each learning resource has a set of preconditions (competences that are necessary for using it) and a set of effects (the supplied competences). Competences can be connected by causal relationships. Rational agents could use such descriptions and the user's learning goal, expressed as well in terms of competences, for performing the sequencing task. This is, for instance, the solution adopted in the WLog system [6], which exploits techniques taken from the research area of "reasoning about actions and change" (*planning*, *temporal projection*, and *temporal explanation*) for building personalized solutions.

Another example concerns *Web services*. Generally speaking, a Web service can be seen as any device that can automatically be accessed over the Web. It may alternatively be a software system or a hardware device; a priori no distinction is made. The main difference between a Web service and other devices that are connected to a network stands in the kind of tasks that can be performed: a Web service can be automatically retrieved by searching for the desired functionality (in a way that is analogous to finding Web pages by means

of a search engine, given a set of keywords), it can be automatically invoked, composed with other Web services so to accomplish more complex tasks, it must be possible to monitor its execution, and so on. In order to allow the execution of these tasks, it is necessary to enrich the Web service with a machine-processable description, that contains all the necessary information, such as what the service does, which inputs it requires, which results are returned, and so forth. A lot of research is being carried on in this area and none of the problems that we have just enumerated has met its final solution yet. Nevertheless, there are some proposals, especially due to commercial coalitions, of languages that allow the description of the single services, and their interoperation. In this line, the most successful are WSDL [72] and BPEL4WS [14]. This initiative is mainly carried on by the commercial world, with the aim of standardizing registration, look-up mechanisms and interoperability.

Among the other proposals, OWL-S [55] (formerly DAML-S) is more concerned with providing greater expressiveness to service description in a way that can be *reasoned about* [20]. In particular, a service description has three conceptual levels: the *profile*, used for advertising and discovery, the *process model*, that describes how a service works, and the *grounding*, that describes how an agent can access the service. In particular, the process model describes a service as atomic, simple or composite in a way inspired by the language GOLOG and its extensions [45, 50]. In this perspective, a wide variety of agent technologies based upon the *action metaphor* can be used. In fact, we can view a service as an action (atomic or complex) with preconditions and effects, that modifies the state of the world and the state of agents that work in the world. The process model can, then, be viewed as the description of such an action; therefore, it is possible to design agents, which apply techniques for reasoning about actions and change to Web service process models for producing new, composite, and customized services.

Quoting McIlraith [51]: "[...] *Our vision is that agents will exploit user's constraints and preferences to help customize user's requests for automatic Web service discovery, execution, or composition and interoperation* [...]". In different words, personalization is seen as *reasoning* about the user's constraints and preferences and about the *effects*, on the user's knowledge and on the world, of the *action* "interact with a Web service". Techniques for reasoning about actions and change are applied to produce composite and customized services.

A better personalization can be achieved by allowing agents to reason also about the *conversation protocols* followed by Web services. Conversation protocols rule the interactions of a service with its interlocutors: the protocol defines all the possible "conversations" that the service can enact. Roughly speaking, we can consider it as a procedure built upon atomic speech acts. So far, however, no language for Web service specification, e.g. OWL-S, allows the explicit representation of the communicative behavior of Web services at an abstract level, i.e. in a way that can be reasoned about. Let us, however, explain with a simple example how this would be useful: an agent, which is a user's *personal assistant*, is requested to book a ticket at a cinema where they show a certain movie; as a

further constraint, the agent does not have to use the user's credit card number along the transaction. While the first is the *user's goal*, the additional request constrains the way in which the agent will *interact* with the service. In this case, in order to personalize the interaction according to the user's request, it is indeed necessary to reason about the service communications. Another possible task of the personal assistant is the organization of a journey: it is necessary to find and make work together (*compose*) services for finding a flight, renting a car, making a reservation at some hotel, maybe the user's personal calendar, etc. All services that have been developed independently and for simpler purposes.

Personalization may involve also other kinds of reasoning, that require knowledge to be represented in other ways. Among them *defeasible reasoning*, which allows taking into account degrees of preference represented as priorities between rules (e.g. DR-DEVICE [11]), *Answer Set Programming* [27], that can deal with incomplete information and default knowledge, *reactivity to events* [48] (the so called ECA rules –event, condition, action–), that allow the propagation of knowledge updates through the Web. All these approaches and techniques conceptually lie at the logic and proof layers of the Semantic Web tower and rely on some kind of rule language.

Rule languages and rule systems are, actually, in the mainstream of research in the Semantic Web area, especially for what regards *exchange of rule sets* between applications. Works in this direction include initiatives for the definition of *rule markup languages*. The aim of introducing rules is to support in a better and wider way the interaction of systems with users as well as of systems with other systems over the Web. Rule markup languages are designed so to allow the expression of rules as modular, stand-alone units in a declarative way, and to allow the publishing and interchange of rules among different systems. Different perspectives can be considered [68]. Rules can be seen as statements that define the terms of the domain, they can be seen as formal statements, which can be directly mapped to executable statements of a software platform, and they can also be considered as statements in a specific executable language.

Two examples of rule markup languages are RuleML [52] and SWRL [37]. The former is a deductive logic language based on XML and RDF. SWRL is a more recent proposal aimed at adding to the OWL language, for defining Web ontologies, the possibility of including Horn-like clauses. The idea is to add the possibility of making deductive inferences that cannot be accomplished by the ontology reasoning techniques. For instance, a consequence of this kind: if X has a brother Y and X has a son Z, then Y is an uncle of Z.

The most important aspect of the standards is its *adoption*, which implies a diffusion of the inference engines that implement them. The hope is that in the near future browsers will support RuleML engines, SWRL engines, and so forth, enabling the use of knowledge over the Web, in the same easy way in which they currently support languages like Java and JavaScript. On the other hand, besides the standards, the way is open for building, on top of the ontology layer, languages that support *heterogeneous* reasoning mechanisms, that fit the requirements of specific personalization problems. This is the reading key of the

following section, where a case study is presented together with reasoning techniques for tackling the personalization task. Further examples of personalization problems, reasoning techniques, and prototype systems can be found in [2].

4.3 Case Study: Personalization in an E-Learning Scenario

Let us focus on e-learning and see how reasoning can help personalization in this context. We will begin with the annotation of the learning resources, then, we will introduce some reasoning techniques, all of which exploit a new level of knowledge thus allowing a better personalization.

A learning object can profitably be used if the learner has a given set of prerequisite competences; by using it, the learner will acquire a new set of competences. Therefore, a learning object can be interpreted as an action: in fact, an action can be executed given that a set of conditions holds, and by executing it, a set of conditions will become true. So, the idea is to introduce at the level of the learning objects, some annotation that describes both their *pre-requisites* and their *effects*. Figure 2 shows an example of how this could be done. To make the example realistic, the annotation respects the standard for learning object metadata LOM. LOM allows the annotation of the learning objects by means of an ontology of interest (see for instance [56]), by using the attribute *classification*. A LOM classification consists of a set of ontology elements (*taxons*), with an associated role (the *purpose*). The taxons in the example are taken from the DAML version of the ACM computer classification system ontology [53]. The reference to the ontology is contained in the *source* element. Since the XML-based representation is quite long, for the sake of brevity only two taxons have been reported: the first (relational database) is necessary in order to understand the contents of the learning object, while the other (scientific databases) is a competence that is supplied by the learning object.

The proposed annotation expresses a set of *learning dependencies* between *ontological terms*. Such dependencies can be expressed in a declarative formalism, and can be used by a reasoning system. So, given a set of learning objects each annotated in this way, it is possible to use the standard planners, developed by the Artificial Intelligence community (for instance, the well-known Graphplan [13]), for building the reading sequences. Graphplan is a general-purpose planner that works in STRIPS-like domains; as all planners, the task that it executes is to build a sequence of atomic actions, that allows the transition from an initial state to a state of interest, or goal state. The algorithm is based on ideas used in graph algorithms: it builds a structure called *planning graph*, whose main property is that the information that is useful for constraining the plan search is quickly propagated through the graph as it is built.

General-purpose planners search a sequence of interest in the whole space of possible solutions and allow the construction of learning objects on the basis of any learning goal. This is not always adequate in an educational application framework, where the set of learning goals of interest is fairly limited and the experience of the teachers in structuring the courses and the learning materials is important. For instance, a teacher due to his/her own experience may believe

```
<lom xmlns="http://www.imsglobal.org/xsd/imsmd_v1p2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imsmd_v1p2 imsmd_v1p2p2.xsd">
  <general>
    <title>
      <langstring>module A</langstring>
    </title>
  </general>
  ...
  <classification>
    <purpose>
      ...
      <value><langstring>Prerequisite</langstring></value>
    </purpose>
    <taxonpath>
      <source>
        <langstring>http://daml.umbc.edu/ontologies/classification.daml
        </langstring>
      </source>
      <taxon>
        <entry>
          <langstring xml:lang="en">relational database</langstring>
        </entry>
      </taxon>
    </taxonpath>
  </classification>
  ...
  <classification>
    <purpose>
      ...
      <value><langstring>Educational Objective</langstring></value>
    </purpose>
    <taxonpath>
      <source>
        <langstring>http://daml.umbc.edu/ontologies/classification.daml
        </langstring>
      </source>
      <taxon>
        <entry>
          <langstring xml:lang="en">scientific databases</langstring>
        </entry>
      </taxon>
    </taxonpath>
  </classification>
</lom>
```

Fig. 2. Excerpt from the annotation for the learning object 'module A': "relational database" is an example of prerequisite while "scientific databases" is an example of educational objective

that topic A is to be presented before topic B, although no learning dependence emerges from the descriptions of A and B. This kind of constraint cannot be exploited by a general-purpose planner, being related to the teaching strategy adopted by the teacher.

On the other hand, it is not reasonable to express schemas of this kind in terms of specific learning objects. The ideal solution is to express the aforementioned schemas as *learning strategies*, i.e. a rule (or a set of rules) that specifies the overall structure of the learning object, expressed only in terms of *competences*. The construction of a learning object can, then, be obtained by refining a learning strategy according to specific requirements and, in particular, by choosing those components that best fit the user.

Reasoning About Actions. Reasoning about actions and change is a kind of temporal reasoning where, instead of reasoning about *time* itself, one reasons on *phenomena* that take place in time. Indeed, theories of reasoning about actions and change describe a *dynamic world* changing because of the execution of actions. Properties characterizing the dynamic world are usually specified by propositions which are called *fluents*. The word *fluent* stresses the fact that the truth value of these propositions depends on time and may vary depending on the changes which occur in the world.

The problem of reasoning about the effects of actions in a dynamically changing world is considered one of the central problems in knowledge representation theory. Different approaches in the literature took different assumptions on the temporal ontology and then they developed different abstraction tools to cope with dynamic worlds. However, most of the formal theories for reasoning about action and change (*action theories*) describe dynamic worlds according to the so-called *state-action model*. In the state-action model the world is described in terms of states and *actions* that cause the transition from a state to another. Typically it is assumed that the world persists in its state unless it is modified by an action's execution that causes the transition to a new state (*persistency assumption*).

The main target of action theories is to use a logical framework to describe the effects of actions on a world where *all* changes are caused by the execution of actions. To be precise, in general, a formal theory for representing and reasoning about actions allows us to specify:

1. *causal laws*, i.e. axioms that describe domain's actions in terms of their *precondition* and *effects* on the fluents;
2. action sequences that are executed from the initial state;
3. *observations* describing the value of fluents in the *initial state*;
4. *observations* describing the value of fluents in later states, i.e after some action's execution.

The term *domain description* is used to refer to a set of propositions that express causal laws, observations of the fluents values in a state and possibly other information for formalizing a specific problem. Given a domain description,

the principal reasoning tasks are *temporal projection* (or prediction), *temporal explanation* (or postdiction) and *planning*.

Intuitively, the aim of *temporal projection* is to predict an action's future effects based on even partial knowledge about the current state (reasoning from causes to effect). On the contrary, the target of *temporal explanation* is to infer something on the past states of the world by using knowledge about the current situation. The third reasoning task, planning, is aimed at finding an action sequence that, when executed starting from a given state of the world, produces a new state where certain desired properties hold.

Usually, by varying the reasoning task, a domain description may contain different elements that provide a basis for inferring the new facts. For instance, when the task is to formalize the temporal projection problem, a domain description might contain information on (1), (2) and (3), then the logical framework might provide the inference mechanisms for reconstructing information on (4). Otherwise, when the task is to deal with the planning problem, the domain description will contain the information on (1), (3), (4) and we will try to infer (2), i.e. which action sequence has to be executed on the state described in (3) for achieving a state with the properties described in (4).

An important issue in formalization is known as the *persistency problem*. It concerns the characterization of the invariants of an action, i.e. those aspects of the dynamic world that are not changed by an action. If a certain fluent f representing a fact of the world holds in a certain state and it is not involved by the next execution of an action a, then we would like to have an efficient inference mechanism to conclude that f still hold in the state resulting from a's execution.

Various approaches in the literature can be broadly classified in two categories: those choosing classical logics as the knowledge representation language [49, 44] and those addressing the problem by using non-classical logics [57, 23, 65, 29] or computational logics [28, 10, 46, 8]. Among the various logic-based approaches to reasoning about actions one of the most popular is still the situation calculus, introduced by Mc Carthy and Hayes in the sixties [49] to capture change in first order classical logic. The situation calculus represents the world and its change by a sequence of *situations*. Each situation represents a state of the world and it is obtained from a previous situation by executing an action. Later on, Kowalski and Sergot have developed a different calculus to describe change [44], called *event calculus*, in which *events* producing changes are temporally located and they initiate and terminate action effects. Like the situation calculus, the event calculus is a methodology for encoding actions in first-order predicate logic. However, it was originally developed for reasoning about events and time in a logic-programming setting.

Another approach to reasoning about actions is the one based on the use of modal logics. Modal logics adopts essentially the same ontology as the situation calculus by taking the state of the world as primary and by representing actions as state transitions. In particular, actions are represented in a very natural way by modalities whose semantics is a standard Kripke semantics given in terms of

accessibility relations between worlds, while states are represented as sequences of modalities.

Both situation calculus and modal logics influenced the design of logic-based languages for agent programming. Recently the research about situation calculus gained a renewed attention thanks to the cognitive robotic project at University of Toronto, that has lead to the development of a high-level agent programming language, called GOLOG, based on a theory of actions in situation calculus [45]. On the other hand, in DyLOG [9], a modal action theory has been used as a basis for specifying and executing agent behavior in a logic programming setting, while the language IMPACT is an example of use of deontic logic for specifying agents: the agent's behavior is specified by means of a set of rules (the agent program) which are suitable to specify, by means of deontic modalities, agent policies, that is which actions an agent is obliged to take in a given state, which actions it is permitted to take, and how it chooses which actions to perform.

Introducing Learning Strategies. Let us now show how the schemas of solution, or *learning strategies*, can be represented by means of rules. In particular, we will use the notation of the language DyLOG.

Learning strategies, as well as learning objects, should be defined on the basis of an ontology of interest. One common need is to express *conjunctions* or *sequences* of learning objects. So for instance, one can say that in his/her view, it is possible to acquire knowledge about *database management* only by getting knowledge about *all* of of a given set of topics, and, among these, *relational databases* must be known before *distributed databases* are introduced.

An example that is particularly meaningful is preparing the material for a basic computer science course: the course may have different contents depending on the kind of student to whom it will be offered (e.g. a Biology student, rather than a Communication Sciences student, rather than a Computer Science student). Hereafter, we consider the case of Biology students and propose a *DyLOG* procedure, named *'strategy('informatics_-for_biologists')'*. This procedure expresses, at an abstract level, a learning strategy for guiding a biology student in a learning path, which includes the basic concepts about how a computer works, together with a specific competence about databases. Notice that no reference to specific learning objects is done.

$$
\begin{array}{l}
strategy('informatics\_for\_biologists') \textbf{ is} \\
\quad achieve\_goal(has\_competence('computer\ system\ organization')) \wedge \\
\quad achieve\_goal(has\_competence('operating\ systems')) \wedge \\
\quad achieve\_goal(has\_competence('database\ management')). \\
\quad \ldots \\
achieve\_goal(has\_competence('database\ management')) \textbf{ is} \\
\quad achieve\_goal(has\_competence('relational\ databases')) \wedge \\
\quad achieve\_goal(has\_competence('query\ languages')) \wedge \\
\quad achieve\_goal(has\_competence('distributed\ databases')) \wedge \\
\quad achieve\_goal(has\_competence('scientific\ databases')).
\end{array}
$$

strategy is defined as a procedure clause, that expresses the view of the strategy creator on what it means to acquire competence about *computer system organization*, *operating systems*, and *database management*.

Suppose that *module A* is the name of a learning object. Interpreting it as an action, it will have preconditions and effects expressed as in Figure 2. We could represent *module A* and its learning dependencies in DyLOG in the following way:

$$access(learningObject('module\ A')) \textbf{ possible if}$$
$$has\_competence('distributed\ database') \wedge$$
$$has\_competence('relational\ database').$$
$$access(learningObject('module\ A')) \textbf{ causes}$$
$$has\_competence('scientific\ databases').$$

Having a learning strategy and a set of annotated learning objects, it is possible to apply *procedural planning* (supplied by the language) for assembling a reading path that is a sequence of learning resources that are annotated as required by the strategy. Opposite to general-purpose planners, procedural planning searches for a solution in the set of the possible executions of a learning strategy. Notice that, since the strategy is based on competences, rather than on specific resources, the system might need to select between different courses, annotated with the same desired competence, which could equally be selected in building the actual learning path. This choice can be done based on external information, such as a user model, or it may be derive from a further interaction with the user. Decoupling the strategies from the learning objects results in a greater flexibility of the overall system, and simplifies the reuse of the learning objects. As well as learning objects, also learning strategies could be made public and shared across different systems.

Other Approaches to Rule-Based Personalization in an e-Learning Scenario. The above example is just one possible way in which personalization can be realized in the Semantic Web in a practical context. Remaining in the e-learning application domain, many other forms of personalization can be thought of, which require other approaches to rule representation and reasoning. Hereafter, we report another example that is taken from a real system. The personalization rules that we will see realize some of the adaptation methods of adaptive educational hypermedia systems (see Section 3.1). The application scenario is a *Personal Reader*[3] [32, 12] for learning resources. This Personal Reader helps the learner to view the learning resources in a *context*: In this context, more *details* related to the topics of the learning resource, the *general topics* the learner is currently studying, *examples*, *summaries*, *quizzes*, etc. are generated and enriched with personal recommendations according to the learner's current learning state [32, 25]. Let us introduce and comment some of the rules that are used by the Personal Reader for learning resources to determine appropri-

[3] http://www.personal-reader.de

ate adaptation strategies. These personalization rules have been realized using TRIPLE.

Generating links to more detailed learning resources is an adaptive functionality in this example Personal Reader. The adaptation rule takes the *isA* hierarchy in the domain ontology, in this case the domain ontology for Java programming, into account to determine domain concepts which are details of the current concept or concepts that the learner is studying on the learning resource. In particular, more details for the currently used learning resource is determined by `detail_learningobject(LO, LO_DETAIL)` where `LO` and `LO_Detail` are learning resources, and where `LO_DETAIL` covers more specialized learning concepts which are determined with help of the domain ontology.

```
FORALL LO, LO_DETAIL detail_learningobject(LO, LO_DETAIL) <-
   EXISTS C, C_DETAIL(detail_concepts(C, C_DETAIL)
     AND concepts_of_LO(LO, C) AND concepts_of_LO(LO_DETAIL, C_DETAIL))
     AND learning_resource(LO_DETAIL) AND NOT unify(LO,LO_DETAIL).
```

Observe that the rule does neither require that `LO_DETAIL` covers all specialized learning concepts, nor that it exclusively covers specialized learning concepts. Further refinements of this adaptation rule are of course possible. The rules for embedding a learning resource into more general aspects with respect to the current learning progress are similar.

Another example of a *personalization rule* for generating embedding context is the recommendation of quiz pages. A learning resource `Q` is recommended as a quiz for a currently learned learning resource `LO` if it is a quiz (the rule for determining this is not displayed) and if it provides questions to at least some of the concepts learned on `LO`.

```
FORALL Q quiz(Q) <-
   Q['http://www.w3.org/1999/02/22-rdf-syntax-ns#':type ->
   'http://ltsc.ieee.org/2002/09/lom-educational#':'Quiz']

FORALL Q, C concepts_of_Quiz(Q,C) <-
   quiz(Q) AND concept(C) AND
   Q['http://purl.org/dc/elements/1.1/':subject -> C].

FORALL LO, Q quiz(LO, Q) <-
     EXISTS C (concepts_of_LO(LO,C) AND concepts_of_Quiz(Q,C)).
```

Recommendations are personalized according to the current learning progress of the user, e. g. with respect to the current set of course materials. The following rule determines that a learning resource `LO` is `recommended` if the learner studied at least one more general learning resource (`UpperLevelLO`):

```
FORALL LO1, LO2 upperlevel(LO1,LO2) <-
   LO1['http://purl.org/dc/terms#':isPartOf -> LO2].

FORALL LO, U learning_state(LO, U, recommended) <-
   EXISTS UpperLevelLO (upperlevel(LO, UpperLevelLO) AND
   p_obs(UpperLevelLO, U, Learned) ).
```

Additional rules deriving stronger recommendations (e. g., if the user has studied *all* general learning resources), less strong recommendations (e.g., if one or two of these haven't been studied so far), etc., are possible, too. Recommendations can also be calculated with respect to the current domain ontology. This is necessary if a user is regarding course materials from different courses at the same time.

```
FORALL C, C_DETAIL detail_concepts(C, C_DETAIL) <-
   C_DETAIL['http://www.w3.org/2000/01/rdf-schema#':subClassOf -> C]
   AND concept(C) AND concept(C_DETAIL).

FORALL LO, U learning_state(LO, U, recommended) <-
   EXISTS C, C_DETAIL (concepts_of_LO(LO, C_DETAIL)
   AND detail_concepts(C, C_DETAIL) AND p_obs(C, U, Learned) ).
```

However, the first recommendation rule, which reasons within one course will be more accurate because it has more fine–grained information about the course and therefore on the learning process of a learner taking part in this course. Thus, a strategy is to prioritize those adaptation rule which take most observations and data into account, and, if these rules cannot provide results, apply less strong rules. This can be realized by defeasible rules [3]: Priorities are used to resolve conflicts, e.g. by giving external priority relations (N.B.: these external priority relations must be acyclic). For example: Rule `r1` determines that the learning state of a learning object is recommended for a particular user if the user has learnt at least one of the general, introductory learning objects in the course, while `r2` says that a learning object is not recommended if the learner has not learnt at least one of the more general concepts. In the following code, `r1 > r2` defines a degree of preference: only when the first rule cannot be applied, the system tries to apply the second.

```
r1: EXISTS UpperLevelLO (upperlevel(LO, UpperLevelLO) AND
      p_obs(UpperLevelLO, U, Learned))
    => learning_state(LO, U, recommended)

r2: FORALL C, C_DETAIL (concepts_of_LO(LO, C_DETAIL)
      AND detail_concepts(C, C_DETAIL) AND NOT p_obs(C, U, Learned)
    => NOT learning_state(LO, U, recommended)

and r1 > r2.
```

5 Conclusions

Personalization, which has become one of the major endeavors of research over the Web, has been studied since the mid 90's in fields like Adaptive Hypermedia and Web Mining. In Adaptive Hypermedia each user has a personalized view of the hypermedia system as well as individual navigation alternatives. Personalization is carried out either selecting the proper level of contents, that the user

can read, or by modifying the set of links to other documents (for instance by hiding certain connections). Web Mining, on the other hand, is mostly concerned with the identification of relations between Web resources which are not directly connected through links. These new relations can be induced on the basis of resource contents or on the basis of regularities in the behavior of a set of independent users. All these approaches have been applied to the WWW, allowing the realization of adaptive systems even in absence of a universally agreed semantics and of standard languages and tools for representing and dealing with semantics. This heterogeneity entails some limitations. In fact, any technique used to deliberate whether a certain resource or link is to be shown to the user requires a lot of information, about the user, about the reasons for which the user should access that resource, and so on. Actually, most of the early personalization systems either managed "closed-world" resources, as it was the case of many systems for e-learning that handled given repositories of learning materials as well as of e-commerce tools, or they were based on user models refined during the direct interaction with the user.

The birth of the Semantic Web brought along standard models, languages, and tools for representing and dealing with machine-interpretable semantic descriptions of Web resources, giving a strong new impulse to research on personalization. Just as the current Web is inherently heterogeneous in data formats and data semantics, the Semantic Web will be heterogeneous in its reasoning forms and the same will hold for personalization systems developed in the Semantic Web. In this lecture we have analyzed some possible applications of techniques for reasoning about actions and change and of techniques for reasoning about preferences, the so called defeasible logic, but, indeed, the availability of a variety of reasoning techniques, all fully integrated with the Web, opens the way to the design and the development of forms of interaction and of personalization that were unimaginable still a short time ago. To this aim it is necessary to integrate results from many areas, such as Multi-Agent Systems, Security, Trust, Ubiquitous Computing, Ambient Intelligence, Human-Computer Interaction and, of course, Automated Reasoning.

This paper is just an introduction to personalization over the Semantic Web, that presents issues, approaches, and techniques incrementally. We have started from the World Wide Web and, then, moved to more abstract levels step by step towards semantics and reasoning, a pattern that follows the classical view of the Semantic Web as a tower of subsequent layers. More than being exhaustive w.r.t all the different techniques and methods that have been proposed in the literature, we have tried to give a complete overview, that includes historical roots, motivations, interconnections, questions, and examples. In our opinion, personalization plays a fundamental role in the Semantic Web, because what is the Semantic Web but a knowledge-aware Web, able to give each user the answers that s/he expects? Research in this field is at the beginning.

Acknowledgements

The authors are indebted with all the researchers who took part to the stimulating discussions during the meetings of REWERSE and in particular of working group A3 in Munich and Hannover. Special thanks to Viviana Patti and Laura Torasso, who actively contribute to the project.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, 1994.
2. G. Antoniou, M. Baldoni, C. Baroglio, R. Baungartner, F. Bry, T. Eiter, N. Henze, M. Herzog, W. May, V. Patti, S. Schaffert, R. Schidlauer, and H. Tompits. Reasoning methods for personalization on the semantic web. *Annals of Mathematics, Computing & Teleinformatics (AMCT)*, 2(1):1–24, 2004.
3. G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. MIT Press, 2004.
4. R. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.
5. M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. Reasoning about interaction protocols for web service composition. In M. Bravetti and G. Zavattaro, editors, *Proc. of 1st Int. Workshop on Web Services and Formal Methods, WS-FM 2004*, volume 105 of *Electronic Notes in Theoretical Computer Science*, pages 21–36. Elsevier Science Direct, 2004.
6. M. Baldoni, C. Baroglio, and V. Patti. Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. *Artificial Intelligence Review*, 22(1), September 2004.
7. M. Baldoni, C. Baroglio, V. Patti, and L. Torasso. Reasoning about learning object metadata for adapting scorm courseware. In L. Aroyo and C. Tasso, editors, *Proc. of Int. Workshop on Engineering the Adaptive Web, EAW'04: Methods and Technologies for personalization and Adaptation in the Semantic Web*, pages 4–13, Eindhoven, The Netherlands, August 2004.
8. M. Baldoni, L. Giordano, A. Martelli, and V. Patti. An Abductive Proof Procedure for Reasoning about Actions in Modal Logic Programming. In J. Dix et al., editor, *Proc. of NMELP'96*, volume 1216 of *LNAI*, pages 132–150. Springer-Verlag, 1997.
9. M. Baldoni, L. Giordano, A. Martelli, and V. Patti. Programming Rational Agents in a Modal Action Logic. *Annals of Mathematics and Artificial Intelligence, Special issue on Logic-Based Agent Implementation*, 41(2-4):207–257, 2004.
10. C. Baral and T. C. Son. Formalizing Sensing Actions - A transition function based approach. *Artificial Intelligence*, 125(1-2):19–91, January 2001.
11. N. Bassiliades, G. Antoniou, and I. Vlahavas. A defeasible logic system for the semantic web. In *In Proc. of Principles and Practice of Semantic Web Reasoning (PPSWR04)*, volume 3208 of *LNCS*. Springer, 2004.
12. Robert Baumgartner, Nicola Henze, and Marcus Herzog. The personal publication reader: Illustrating web data extraction, personalization and reasoning for the semantic web. In *Proceedings of 2nd European Semantic Web Conference*, Heraklion, Greece, May 2005.
13. A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

14. BPEL4WS. http://www-106.ibm.com/developerworks/library/ws-bpel. 2003.
15. P. De Bra, A. Aerts, D. Smits, and N. Stash. AHA! version 2.0: More adaptation flexibility for authors. In *Proceedings of the AACE ELearn'2002 conference*, October 2002.
16. P. De Bra, G.J. Houben, and H. Wu. AHAM: A dexter-based reference model for adaptive hypermedia. In *ACM Conference on Hypertext and Hypermedia*, pages 147–156, Darmstadt, Germany, 1999.
17. P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87–129, 1996.
18. P. Brusilovsky, J. Eklund, and E. Schwarz. Web-based Educations for All: A Tool for Development Adaptive Courseware. In *Proceedings of the Sevenths International World Wide Web Conference, WWW'98*, 1998.
19. Peter Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110, 2001.
20. J. Bryson, D. Martin, S. McIlraith, and L. A. Stein. Agent-based composite services in DAML-S: The behavior-oriented design of an intelligent semantic web. In J. Liu N. Zhong and Y. Yao, editors, *Web Intelligence*. Springer-Verlag, Berlin, 2002. Agent-Based Composite Services in DAML-S: The Behavior-Oriented Design of an Intelligent Semantic Web.
21. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2002.
22. R.M. Carro, E. Pulido, and P. Rodruez. Dynamic generation of adaptive internet-based courses. *Journal of Network and Computer Applications*, 22:249–257, 1999.
23. M. Castilho, O. Gasquet, and A. Herzig. Modal tableaux for reasoning about actions and plans. In S. Steel, editor, *Proc. ECP'97*, LNAI, pages 119–130, 1997.
24. P. de Bra. Hypermedia structures and systems: Online Course at Eindhoven University of Technology, 1997. http://wwwis.win.tue.nl/2L690/.
25. P. Dolog, N. Henze, W. Nejdl, and M. Sintek. The Personal Reader: Personalizing and Enriching Learning Resources using Semantic Web Technologies. In *Proc. of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004)*, Eindhoven, The Netherlands, 2004.
26. M. H. Dunham, editor. *Data Mining*. Prentice Hall, 2003.
27. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9, 1991.
28. M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–321, 1993.
29. L. Giordano, A. Martelli, and C. Schwind. Dealing with concurrent actions in modal action logic. In *Proc. ECAI-98*, pages 537–541, 1998.
30. I.P. Goldstein. The genetic graph: A represenation for the evolution of procedural knowledge. In D. Sleeman and J.S.Brown, editors, *Intelligent Tutoring Systems*. Academic Press, 1982.
31. F. Halasz and M. Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.
32. N. Henze and M. Kriesell. Personalization Functionality for the Semantic Web: Architectural Outline and First Sample Implementation. In *Proccedings of the 1st International Workshop on Engineering the Adaptive Web (EAW 2004), co-located with AH 2004*, Eindhoven, The Netherlands, 2004.
33. N. Henze and W. Nejdl. Extendible adaptive hypermedia courseware: Integrating different courses and web material. In *Proccedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000)*, Trento, Italy, 2000.

34. N. Henze and W. Nejdl. Adaptation in open corpus hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems*, 12, 2001.
35. N. Henze and W. Nejdl. Logically characterizing adaptive educational hypermedia systems. Technical report, University of Hannover, April 2003. http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2003/ TechReportHenzeNejdl.pdf.
36. N. Henze and W. Nejdl. A logical characterization of adaptive educational hypermedia. *New Review of Hypermedia*, 10(1), 2004.
37. I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, and B. Grosof. SWRL: a semantic web rule language combining OWL and RuleML, 2004. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521.
38. A. Jameson. Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User Adapted Interaction*, 5(3/4):193–251, 1996.
39. Jena - A Semantic Web Framework for Java, 2004. http://jena.sourceforge.net/.
40. Jupiter Research Report, October 14th, 2003. http://www.jupitermedia.com/corporate/releases/03.10.14-newjupresearch.html.
41. A. Kobsa. User modeling: Recent work, prospects and hazards. In M. Schneider-Hufschmidt, T. Kühme, and U. Malinowski, editors, *Adaptive User Interfaces: Principles and Practice*. Elvesier, 1993.
42. A. Kobsa. Generic user modeling systems. *User Modeling and User-Adapted Interaction*, 11:49–63, 2001.
43. N. Koch. *Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process*. PhD thesis, Ludwig-Maximilians-Universitt Mnchen, 2001.
44. R. Kowalski and M. Sergot. A Logic-based Calculus of Events. *New Generation of Computing*, 4:67–95, 1986.
45. H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming*, 31:59–83, 1997.
46. J. Lobo, G. Mendez, and S. R. Taylor. Adding Knowledge to the Action Description Language $\mathcal{A}$. In *Proc. of AAAI'97/IAAI'97*, pages 454–459, Menlo Park, 1997.
47. D. Lowe and W. Hall. *Hypermedia and the Web*. J. Wiley and Sons, 1999.
48. W. May, J.J. Alferes, and F. Bry. Towards generic query, update, and event languages for the semantic web. In *in Proc. of Principles and Practice of Semantic Web Reasoning (PPSWR04)*, volume 3208 of *LNCS*. Springer, 2004.
49. J. McCarthy and P. Hayes. Some, Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1963.
50. S. McIlraith and T. Son. Adapting Golog for Programming the Semantic Web. In *5th Int. Symp. on Logical Formalization of Commonsense Reasoning*, pages 195–202, 2001.
51. S. A. McIlraith, T. C. Son, and H. Zenf. Semantic Web Services. *IEEE Intelligent Systems*, pages 46–53, March/April 2001.
52. Rule ML. http://www.ruleml.org.
53. Association of Computing Machinery. The ACM computer classification system, 2003. http://www.acm.org/class/1998/.
54. OWL, Web Ontology Language, W3C Recommendation, February 2004. http://www.w3.org/TR/owl-ref/.
55. OWL-S: Web Ontology Language for Services, W3C Submission, November 2004. http://www.org/Submission/2004/07/.

56. W. Nejdl P. Dolog, R. Gavriloaie and J. Brase. Integrating adaptive hypermedia techniques and open rdf-based environments. In *Proc. of The 12th Int. World Wide Web Conference*, Budapest, Hungary, 2003.
57. H. Prendinger and G. Schurz. Reasoning about action and change. a dynamic logic approach. *Journal of Logic, Language, and Information*, 5(2):209–245, 1996.
58. R. Rada. *Interactive Media.* Springer, 1995.
59. RDF. http://www.w3c.org/tr/1999/rec-rdf-syntax-19990222/. 1999.
60. RDFS. http://www.w3.org/tr/rdf-schema/. 2004.
61. RDQL - query language for RDF, Jena, 2005. `http://jena.sourceforge.net/RDQL/`.
62. R. Reiter. A theory of diagnosis from first principles. *Artifical Intelligence*, 32, 1987.
63. E. Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1978.
64. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.
65. C. B. Schwind. A logic based framework for action theories. In J. Ginzburg et al., editor, *Language, Logic and Computation*, pages 275–291. CSLI, 1997.
66. M. Sintek and S. Decker. TRIPLE - an RDF Query, Inference, and Transformation Language. In I. Horrocks and J. Hendler, editors, *International Semantic Web Conference (ISWC)*, pages 364–378, Sardinia, Italy, 2002. LNCS 2342.
67. M. Specht. Empirical evaluation of adaptive annotation in hypermedia. In *ED-Media and ED-Telekom*, Freiburg, Germany, 1998.
68. G. Wagner. Ruleml, swrl and rewerse: Towards a general web rule language framework. *SIG SEMIS Semantic Web and Information Systems*, 2004. http://www.sigsemis.org/articles/copy_of_index_html.
69. Geoffrey I. Webb, Michael J. Pazzani, and Daniel Billsus. Machine learning for user modeling. *User Modeling and User-Adapted Interaction*, 11:19–29, 2001.
70. G. Weber and P. Brusilovsky. ELM-ART: An Adaptive Versatile System for Web-based Instruction. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems*, 12, 2001.
71. G. Weber, H.C. Kuhl, and S. Weibelzahl. Developing adaptive internet based courses with the authoring system NetCoach. In *Proc. of the Third Workshop on Adaptive Hypermedia, AH2001*, 2001.
72. WSDL. http://www.w3c.org/tr/2003/wd-wsdl12-20030303/. version 1.2, 2003.