On Personalizing the Catalogs of Web Portals

Nicolas Spyratos
Laboratoire de Recherche en Informatique,
Universite de Paris-Sud, France
spyratos@lri.fr

Yannis Tzitzikas, Vassilis Christophides
Department of Computer Science,
University of Crete, Greece
Institute of Computer Science, ICS-FORTH
tzitzik@ics.forth.gr, christop@csi.forth.gr

Abstract

In this paper we propose a method for personalizing the catalogs of Web Portals. We propose $SCSL$, a declarative language for defining personal *semantic channels* over Web Portal catalogs. A semantic channel is actually a view of one or *more* Portal catalogs. $SCSL$ offers powerful primitives for filtering and restructuring available thematic topics and classified resources. A user can connect to a Portal *infomediary* in order to register, browse, or query his/her semantic channel. We describe the architecture and the functional modules of our infomediary acting as personalization server and we focus on a set of queries which can be considered as an API for browsing semantic channels.

Introduction

Portals are nowadays becoming increasingly popular by enabling the development and maintenance of specific communities of interest on corporate intranets or the Internet (Finkelstein & Aiken 1999; Carlson 2001). More precisely, Portals aggregate and classify, in a semantically meaningful way, various information resources for diverse target audiences (e.g., enterprise, professional, trading). Thus, the *Catalog* of a Portal consists of descriptive information about community resources. The complexity of the provided semantic descriptions (e.g., using taxonomies or ontologies) depends on the breadth of the community domain knowledge (targeting horizontal or vertical markets) as well as the nature of the available resources (e.g., sites, documents, data). In most *Web Portals* (e.g., Yahoo!, Netscape Dmoz or Chefmoz, MusicBrain, CNET, XMLTree[1]) Web resources are classified under large hierarchies of thematic topics or terms. To support personalization, Web Portals adopt a publish/subscribe approach (Manber, Patel, & John 2000; Ramakrishnan & Daya 1998), where users can choose from a predefined list, particular topics of information (called *channels*) they want to receive (e.g., travel information, business news).

In this paper, we propose a declarative language, called *Semantic Channel Specification Language* (or $SCSL$ for short), allowing users to specify their own *semantic channels* based on one or more Portal catalogs, created according to taxonomic schemas à la Yahoo!. The term *semantic channel* refers to a personal view of Portal catalogs constructed by extracting, reorganizing and enriching the available thematic topics and classified resources. A user can connect to a Portal *infomediary* (Maglio & Barrett 2000) in order to register, browse, or query the associated semantic channel. We propose an enhanced type of infomediary to act as personalization server, and describe its architecture and functional modules. Given the increasing popularity of RDF/S (Lassila & Swick 1999; Brickley & Guha 2000) for exporting Portal catalogs on the Web, we rely on robust RDF tools, like RDFSuite[2], for analyzing, storing and querying resource descriptions and schemas.

Personalization of Web sites and portals has received considerable attention both from a research and from a commercial perspective[3]. Our work aims at personalizing Web sites developed using Semantic Web technology(Berners-Lee, Hendler, & Lassila 2001), and in particular taxonomic schemas expressed in RDF/S. In this context, semantic channels are specified as appropriate views over Portal catalogs rather than as just flat lists of keywords. Motivation for our work comes from the fact that semantic and structural information of modern Web sites is more and more exploited to support user personalization (Lu, Eichstadt, & Ford 1998; Pretschner & Gauch 1999; Kurki, Jokela, & Sulonen 1999; Hearst 2000; Maedche *et al.* 2001). For example, (Lu, Eichstadt, & Ford 1998) propose some simple channel-specific predicates on the data of the channels selected by a user to specify particular companies (for stock prices), cities (for weather), or teams (for sports scores of interest). $SCSL$ goes one step further by also allowing to dynamically specify the schema parts of Portal catalogs that will be made available to the end users. Thus, $SCSL$ enables constructing user-specific hierarchies of terms as opposed to the notion of semantic bookmarks in (Maedche *et al.* 2001). These hierarchies are then used to navigate/query existing Portal catalogs. Another distinctive feature of our approach is that

[1] See www.yahoo.com, dmoz.org, chefmoz.org, musicbrain.org home.cnet.com, www.xmltree.com, respectively.

[2] 139.91.183.30:9090/RDF

[3] For a detailed list of related projects and products, readers are referred to www.sims.berkeley.edu/ sinha/personalization.html.

a semantic channel can combine terms and resources from *several* Web Portal catalogs.

The remaining of the paper is organized as follows: at first we recall briefly the structure of the catalogs of Web Portals like ODP. Then we introduce $SCSL$. Subsequently we discuss the architecture and implementation of the infomediary, and finally, we conclude the paper and identify issues for further research.

Web Portal Catalogs

In this section, we briefly recall the structure of Portal catalogs, like Netscape Dmoz (called Open Directory Project, ODP) or Yahoo!. To semantically classify Web resources, Internet Portals usually rely on large hierarchies of topics, or indexing *terms*. A Portal catalog is published on the Web as a set of interlinked html pages where each page contains Web resources classified under a specific term as well as various kinds of relationships with other terms.

A snapshot of the ODP Web catalog with tourist information about Crete is given in the top part of Figure 1. As we can see, each term is identified by a URL (e.g., `http://dmoz.org/Regional/Europe/Regions/Crete/`) from the root of the corresponding ODP thematic hierarchy (e.g., `Regional`) and it is displayed by a name (e.g., *Crete*) in an appropriate language (e.g., English, Greek). Furthermore, ODP terms are interconnected using three relations, namely *subtopic*, *symbolic* and *related*. The meaning of the *subtopic* relation is the well known subsumption (or isA) relation (e.g., `Travel&Tourism` specialize `Crete`), while the meaning of the remaining two (*symbolic* and *related*) is somehow overloaded. Sometimes they are used for referring to complex terms whose classified resources are spread over different hierarchies (e.g., `http://dmoz.org/World/Greek/Local/Europe/Greece/Regions/Crete`), or for linking synonym terms (e.g., `http://dmoz.org/Reference/Museum/ByRegion/Europe` and `http://dmoz.org/Reference/Museum/Art&Entertainment/ArtMuseum/European/`), or simply for defining shortcut navigation paths into the Portal catalog (e.g., `http://dmoz.org/Regional/Europe/Perfectures/Heraklion/`). The only difference between the ODP *symbolic* and *related* links is that symbolic links may have a different name from the topic they are referring to (e.g., *HeraklionCity* is the display name of the previous symbolic link within our example Web page).

More formally, we define the conceptual structure of a Web portal *catalog* (illustrated in the bottom part of Figure 1) as a pair $C = (H, I)$ where H is a *hierarchy* of terms and I is an *interpretation* of H.

A *hierarchy* is a pair $H = (T, R)$, where

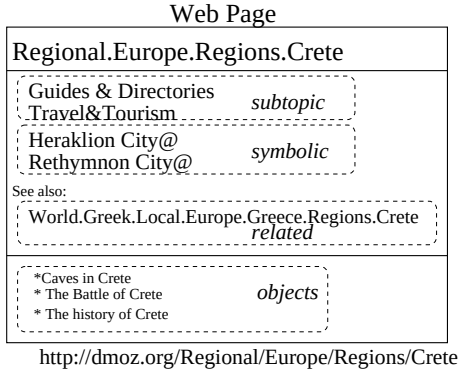
T is a set of term identifiers

$R : \subseteq T \times T$ is a binary relation over T

We use R to denote the union of *subtopic* (R_c), *symbolic* (R_s) and *related* (R_r) relations between terms, i.e., we can write $R = R_c \cup R_s \cup R_r$. If $(t', t) \in R$ then there is a link from t to t'.

The *interpretation* I associates the terms of the hierarchy with Web resources (i.e., object URLs) that are indexed by that catalog. If Obj denotes the set of all URLs of the Web, then I is a function $I : T \rightarrow 2^{Obj}$, where 2^{Obj} stands for the powerset of Obj. For example, I(`http://dmoz.org/Regional/Europe/Regions/Crete/`) denotes the set of resources that are indexed under this term, e.g. the resource `http://www.crete.gr/BattleofCrete` etc.

In the following we shall use $\bar{I}(t)$ to denote the set of objects which are indexed under the term t or a term narrower than t, i.e. $\bar{I}(t) = \bigcup\{I(s) \mid (s,t) \in R^*\}$, where R^* denotes the reflexive and transitive closure of R.



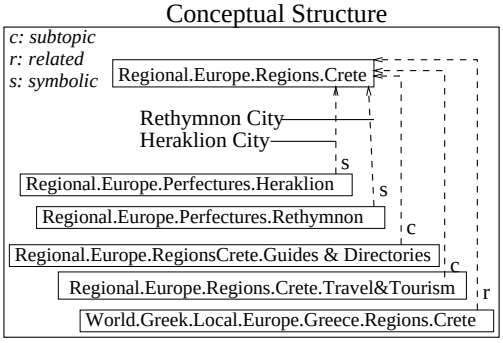


Figure 1: The interface and the structure of ODP

In addition, a number of properties may be assigned to each object, e.g. ODP uses the properties $LastModified$, $Title$ and $Description$.

Defining Semantic Channels with *SCSL*

Let $C_1 = (H_1, I_1), ..., C_k = (H_k, I_k)$ be a set of catalogs. Below we describe how we can define a semantic channel over these catalogs.

A semantic channel is *again* a catalog, i.e. a pair $C^u = (H^u, I^u)$ where $H^u = (T^u, R^u)$ is a hierarchy of terms and I^u is an interpretation of H^u. We do not partition the relation R^u to *subtopic*, *symbolic* and *related*, because we believe that this would rather confuse the user.

Roughly, a *semantic channel* over a catalog $C_i = (H_i, I_i)$ can be defined by *browsing*, and/or by *querying* the hierarchy of the catalog. In the first approach, also called the *check-box personalization*, a Web interface is used in order to "collect"

(and probably rename) the desired terms of T_i, as well as their relationships in R_i (e.g., see (Manber, Patel, & John 2000; Ramakrishnan & Daya 1998)). In the second approach, the user "describes" the desired terms of T_i (or relationships of R_i) by appropriate query expressions. Formulating these expressions directly in an RDF Query Language such as RQL (Alexaki *et al.* 2002) would involve a considerable human effort. For this purpose we designed the (high-level) language $SCSL$. Moreover, the $SCSL$ requires having the ability to create new terms and relationships, a feature which is not currently supported by the functionality of RQL.

Let us first describe how the hierarchy $H^u = (T^u, R^u)$ of a semantic channel is defined. Let $\mathcal{T}$ denote the set of all term identifiers, i.e., $\mathcal{T} = \bigcup_{i=1..k} T_i$. The set T^u may contain terms that belong to $\mathcal{T}$ and new terms which are introduced by the user (e.g. `MyFavoriteMusic`).

According to our approach, the hierarchy of a semantic channel is defined by a set of *SCSL declarations*. Each declaration d_i defines a pair (T_i^u, R_i^u) where T_i^u is a set of terms, and R_i^u is a subtopic relation over T_i^u. Now, a set of declarations $D = \{d_1, ..., d_k\}$ defines the hierarchy (T^u, R^u) where:

$$T^u = \bigcup_{i=1..k} T_i^u \quad \text{and} \quad R^u = \bigcup_{i=1..k} R_i^u$$

There are four kinds of $SCSL$ *declarations*, namely:

terms	$termExpr$
addSubTopic	$term \quad termExpr$
ReStruct	$termExpr$ **rels** $relTypeSet$
setObjectFilter	$termExpr \quad filter$

where $termExpr$ denotes a *term expression*, and $term$ denotes a single term. Let us first give some examples of term expressions that $SCSL$ supports.

- subtopics(`Crete`): specifies all immediate subtopics of the term `Crete`, i.e., the set $\{t \mid (t, \texttt{Crete}) \in R_c\}$
- subtopics(`Crete`)(2): specifies all subtopics of `Crete`, which can be reached by following at most two links of R_c, i.e., the set $\{t \mid (t, \texttt{Crete}) \in R_c^2\}$
- subtopics(`Crete`)(*): specifies *all* subtopics of `Crete`, i.e., the set $\{t \mid (t, \texttt{Crete}) \in R_c^*\}$, where R_c^* denotes the reflexive and transitive closure of R_c.
- related(`Crete`): specifies all topics which are related to `Crete`, i.e., the set $\{t \mid (t, \texttt{Crete}) \in R_r\}$
- symbolic(`Crete`): specifies all topics that can be reached by following a symbolic link from `Crete`, i.e. the set $\{t \mid (t, \texttt{Crete}) \in R_s\}$
- contains(`Crete`): specifies all terms whose name contains the string `Crete`, e.g. the term `Regional/Europe/Greece/Regions/Islands/Crete`. With this kind of term expressions the user can "collect" terms which are scattered in the hierarchy. Such expressions can be used for restructuring the whole catalog so that to result to a clear faceted structure such as the one proposed in (Tzitzikas *et al.* 2002).
- sub(`Crete`)(2): specifies all topics that can be reached by following two links of R, i.e., the set $\{t \mid (t, \texttt{Crete}) \in R^2\}$

Also note that term expressions can be combined using the set operations $\cup, \cap, \setminus$, to form more complex ones e.g.:

subtopics(`Arts`) **and** contains(`Crete`)

We shall use eval($termExpr$) to denote the set of terms specified by $termExpr$. *SCSL* also allows the user to specify which kind of evaluation he wants for each term expression. A term expression followed by the word *static* (which is the default) is evaluated immediately during registration, while if followed by the word *dynamic* is evaluated periodically, e.g., every month, or each time the user browses/queries his/her semantic channel. In this way a semantic channel can follow the evolution of the associated Portal catalogs. For example the term expression "subtopic(`Music`) dynamic" allows a user to include (in his/her channel) each new kind of Music that will appear in the future.

Let us now describe each kind of *SCSL* declaration.

A *terms*-declaration can be used for specifying *a set of existing terms*. For example,

terms contains(`jazz`)

adds to T^u all terms that concern jazz style music (e.g. Funk-Jazz, AcidJazz). Moreover, this kind of declaration can be used in order to introduce a new term (e.g. **terms** `MyMusic`) or to rename an existing one (e.g. **terms** $Jazz\#MyJazz$). However, if the declaration of a semantic channel contains only *terms*-declarations, then the defined channel is just a set of terms, i.e., $R^u = \emptyset$. In order to interconnect terms in a personal taxonomy we support the *addSubTopic* and *ReStruct*-declarations which specify pairs (T_i^u, R_i^u).

An *addSubTopic*-declaration has the syntax

addSubTopic $term \quad termExpr$

This declaration defines the pair (T^u, R^u) where

$$\begin{aligned} T^u &= \{term\} \cup \text{eval}(termExpr) \\ R^u &= \{(t, term) \mid t \in \text{ eval } (termExpr)\} \end{aligned}$$

For example the set R^u that is specified by the declaration:

addSubTopic `MyMusic`
sub(`Music`) **and** contains(`jazz`)

consists of all pairs (*t*, `MyMusic`) where *t* is a term specified by the term expression.

A *ReStruct*-declaration has the following syntax:

ReStruct $termExpr$ **rels** $relTypeSet$

where $relTypeSet$ is a subset of the set of keywords {SUBTOPIC, RELATED, SYMBOLIC} indicating the corresponding relations from the original Portal schema. A **ReStruct** declaration defines a pair (T^u, R^u), where T^u is the set of terms specified by $termExpr$, and R^u is the union of the restrictions, on the set T^u, of the relations that are specified in $relTypeSet$. For example, if $relTypeSet$= {SUBTOPIC, RELATED, SYMBOLIC}, then

$$\begin{aligned} T^u &= \text{ eval } (termExpr) \\ R^u &= R^*_{c|_{T^u}} \cup R_{r|_{T^u}} \cup R_{s|_{T^u}} \end{aligned}$$

It is important to note that we work on the transitive closure of subtopic relation (R_c). In this way the user can exclude

intermediate terms of the original hierarchy and still get a hierarchy of terms. For example consider the hierarchy shown in Figure 2(a) and suppose a ReStruct declaration where *termExpr* specifies all terms except those enclosed in the dashed square. Figure 2(b) shows the defined (T^u, R^u) in the case where *relTypeSet* = {SUBTOPIC, RELATED}, and Figure 2(c) in the case where *relTypeSet* = {SUBTOPIC}.

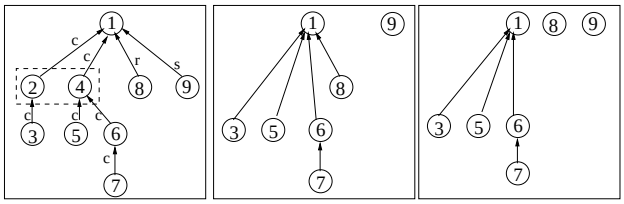


Figure 2: Example of the ReStuct declaration

For example, the declaration

ReStruct (subtopics(`Crete`)(*) **minus** contains(`Transportation`)) **rels** SUBTOPICS

specifies all subtopics of `Crete` which do not contain the "subterm" Transportation, and all subtopic relationships that hold (in R_c^*) between these terms.

A *setObjectFilter*-declaration associates a filtering condition to the objects (i.e., Web resources) classified under each term specified by the term expression. It is used to filter the objects that will appear under a term during browsing/querying. This condition is evaluated on the properties of the objects, e.g. in the case of ODP, on the properties $LastModified, Title$ and $Description$.

For example the declaration

setObjectFilter subtopics(`Music`)
LastModified > Jan2002

associates the condition LastModified > Jan2002 to each subtopic of the term `Music` in order to see only recent Web resources.

For defining the interpretation I^u of a semantic channel we proceed as follows. At first we define the interpretation I_0^u:

$$I_0^u(t) = \begin{cases} \bar{I}_i(t) & \text{if } t \in T_i \\ \emptyset & \text{otherwise} \end{cases}$$

This requires fetching and wrapping several Web pages (for more see (Tzitzikas, Spyratos, & Constantopoulos 2001)). The interpretation I^u is then derived by applying the associated object filters, i.e.:

$$I^u(t) = \{o \in I_0^u(t) \mid t.filter(o)\}$$

where $t.filter(o)$ means that the object o satisfies the filters which are associated with the term t.

We close this section with an example of a semantic channel over ODP which is defined by a set of *SCSL* declarations.

terms `Crete#MyCrete`

ReStruct
(sub(`Crete`)(4) **minus** contains (`Business`))
rels SUBTOPIC RELATED

terms `MyFavoriteMusic`

addSubTopic `MyFavoriteMusic`
(subtopics(`Music`) **and** contains(`Jazz`) **dynamic**)

setObjectFilter
`MyFavoriteMusic`
LastModified > Jan2002

The first declaration simply renames the ODP term `Crete`. The second one specifies the subtree under the term `Crete` which will be included in the channel by excluding the terms concerning Business and those reached through symbolic links. The third one introduces the new term `MyFavoriteMusic`, the fourth one hangs under this term all terms under the term `Music` that contain the word "Jazz", and finally, the fifth one assigns an object filter to this term.

Implementation

Figure 3 shows the architecture of the $SCSL$-based personalization server, which is based on an *infomediary* approach (Maglio & Barrett 2000). A user or an application can connect to the infomediary in order to define/register its semantic channel, and subsequently browse/query the thematic topics and Web resources of this channel.

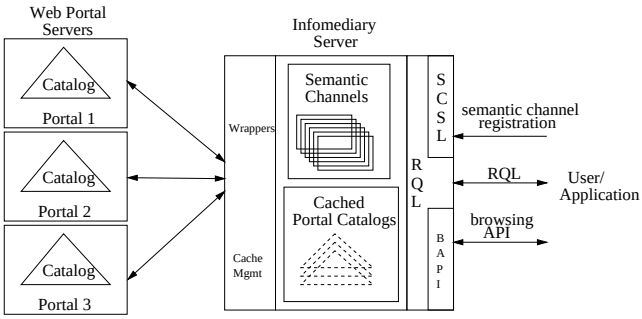


Figure 3: The infomediary server

In the sequel, we will use `P` to denote the Web server of a Portal and `M` to denote the Web server of the $SCSL$ infomediary. Let D be the declaration of a semantic channel as provided by the user. `M` stores D as a set of $SCSL$ declarations. Moreover, `M` evaluates D, i.e., `M` evaluates the term expressions contained in D, in order to compute and store the corresponding channel (H^u, I^u). Note that if D contains dynamically evaluated term expressions, then `M` periodically re-evaluates D and stores only the view updates. As we can see in Figure 3, `M` caches the term hierarchies of the catalogs, thus for evaluating an expression, `M` does not have to query the portal `P`. However, `M` has to update periodically its cache using appropriate wrappers for each Portal.

Since the term hierarchies and the Web resources of Portals are more and more exported in RDF/S (Lassila & Swick 1999; Brickley & Guha 2000) we employ RSSDB (Alexaki *et al.* 2001) for storing the cached copies of the catalogs as well as the user-defined semantic channels.

Browsing Semantic Channels

Clearly we can use RQL (Alexaki *et al.* 2002) in order to query the RDF repository of our infomediary. Here we focus on a particular set of queries allowing to implement browsing of the registered semantic channels. This set can be considered as a general API (implemented using SOAP[4]) for browsing personal taxonomies of terms and related Web resources, or for developing, on top, other value-added services and tools (e.g., recommendation, etc.). In the following we present the main queries of this browsing API where p stands for a channel identifier and t for a term coming from a catalog.

- $getChildren(p, t, n)$. Returns the descendants of t wrt. R^u at depth n, i.e., the set $\{t' \in T^u \mid (t', t) \in R^{u^n}\}$. If $n = 1$ it returns the direct children of t while if $n = *$ it returns transitively all its descendants. In case where t and n are set to 0 then the root terms (maximal elements) of H^u are returned. These queries are evaluated locally using only the cached schemas of the Portal catalogs which are updated periodically according to the infomediary policy. This evaluation strategy is justified by the cost of transitive closure computations requiring special purpose optimization DB support by the infomediary.
- $getObjects(p, t, n)$. Returns the objects which are associated with the term t. If $n = 0$ it returns the objects directly associated with t while if $n = *$ it returns the objects which are associated with all descendants of t. These queries can be evaluated either locally at the infomediary or remotely, by fetching and wrapping the corresponding Web page of a term t.

Summary and Further Research

In this paper we discussed the structure of portal catalogs, like Open Directory, and we described the architecture of a system for personalizing them. More precisely, we provided a high level language ($SCSL$) for specifying the desired part (of schema and resources) of several Web portal catalogs. An essential feature of $SCSL$ is its ability to construct personal views of taxonomic schemas by extracting, reorganizing and enriching the indexing terms of existing Portal catalogs. We believe that the expressive power of $SCSL$ is sufficient enough to restructure the entire catalog of a Portal, i.e., define views that overcome the semantic inconsistencies of the existing catalogs (e.g., see (Hearst 2000)). Of course this is a costly computation that goes beyond user personalization and can be performed offline by the Portal infomediary as an added-value service. However, such a restructuring would allow adopting the extended faceted classification scheme (introduced in (Tzitzikas *et al.* 2002)) which has many advantages comparing with the hierarchical structure of the existing catalogs. Moreover, a catalog having a clear faceted stucture would make the declaration of a semantic channel easier for the user.

Issues for further research include the storage optimization of the semantic channels (in order to avoid storing multiple copies of the same taxonomies), the re-evaluation of a semantic channel (in order to update it with new information from the Portals), and more generally the maintenance of the infomediary cache.

[4] www.w3.org/TR/SOAP/

Acknowledgements

Many thanks to Dimitris Plexousakis and Aimilia Magkanaraki for proof reading the paper.

References

Alexaki, S.; Christophides, V.; Karvounarakis, G.; Plexousakis, D.; and Tolle, K. 2001. "The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases". In *Procs of 2nd Int. Workshop on the Semantic Web (SemWeb'01)*.

Alexaki, S.; Karvounarakis, G.; Christophides, V.; and Plexousakis, D. 2002. "RQL: A Declarative Query Language for RDF. In *Eleventh Int. World Wide Web Conference (WWW)*.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The Semantic Web. *Scientific American*.

Brickley, D., and Guha, R. 2000. Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation.

Carlson, D. 2001. *Modeling XML Applications with UML: Practical e-Business Applications*. Addison Wesley.

Finkelstein, C., and Aiken, P. 1999. *Building Corporate Portals using XML*. McGraw-Hill.

Hearst, M. 2000. Next generation web search: Setting our sites. *IEEE Data Engineering Bulletin, Special issue on Next Generation, Luis Gravano (Ed.)* 23(3).

Kurki, T.; Jokela, S.; and Sulonen, R. 1999. Agents in delivering personalized content based on semantic metadata. In *Proc. 1999 AAAI Spring Symposium Workshop on Intelligent Agents in Cyberspace*.

Lassila, O., and Swick, R. 1999. Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation.

Lu, Q.; Eichstadt, M.; and Ford, D. 1998. Efficient profile matching for large scale webcasting. In *7th International World Wide Web Conference (WWW7)*.

Maedche, A.; Staab, S.; Stojanovic, N.; Studer, R.; and Sure, Y. 2001. *SEmantic portAL - The SEAL approach*. MIT Press. (to appear).

Maglio, P. P., and Barrett, R. 2000. Intermediaries personalize information streams. *Communications of the ACM* 43(8).

Manber, U.; Patel, A.; and John, R. 2000. Experience with personalization on yahoo! *Communications of the ACM* 43(8).

Pretschner, A., and Gauch, S. 1999. Ontology based personalized search. In *Proc.of the 11th IEEE International Conference on Tools with Artificial Intelligence(ICTAI'99)*.

Ramakrishnan, S., and Daya, V. 1998. The pointcast network. In *Proc. of ACM SIGMOD International Conference on Management of Data*.

Tzitzikas, Y.; Spyratos, N.; Constantopoulos, P.; and Analyti, A. 2002. "Extended Faceted Ontologies". In *Procs of the 14th Int. Conference on Advanced Information Systems Engineering, CAiSE-02*.

Tzitzikas, Y.; Spyratos, N.; and Constantopoulos, P. 2001. "Mediators over Ontology-based Information Sources". In *Second International Conference on Web Information Systems Engineering, WISE 2001*.