Taxonomy Based Notification Service for the ASSETS Digital Library Platform

Jitao Yang, Tsuyoshi Sugibuchi, and Nicolas Spyratos

Laboratoire de Recherche en Informatique, Université Paris-Sud 11
Rue Noetzlin, 91190 Gif-sur-Yvette, France
{jitao.yang,tsuyoshi.sugibuchi,nicolas.spyratos}@lri.fr

Abstract. In this paper, we report our taxonomy-based notification service for the *ASSETS* digital library platform, which is being developed in an EU co-funded project. Notification is a very fundamental functionality for every living digital library which is continuously updated and dynamically interacts with users. The ASSETS platform provides a common notification service and its extensions based on the publish/subscribe pattern as a message notification infrastructure. Our taxonomy based notification service is one of those extensions that enables users to define subscriptions for receiving notifications by using a hierarchically organized controlled vocabulary, namely a *taxonomy*. Through this service, users can easily subscribe to messages about specific domain of their interest with a small number of terms in a taxonomy. Then system can efficiently filter a stream of published messages to deliver notifications to proper subscribers by taking account of the taxonomy. This service works as an important piece for enabling various advanced features in the ASSETS platform such as personalized new item lists and a digital preservation service. In this paper, we show an outline of the ASSETS notification architecture, and give a description about a model for the taxonomy-based notification implemented in our service.

1 Introduction

A modern digital library system is formed as a stack of various principals including software components, external systems and human beings as users. Message exchange among those principals is sometimes asynchronous, and/or loosely coupled (associations between senders and receivers might be dynamically changed), and/or multicast (one message might go to more than one destinations). The *publish/subscribe pattern* is a well-known design for enabling such flexible massage exchange. The publish/subscribe pattern can be modelled as interactions among *publishers*, *subscribers*, and a *message broker* (or a *notification service*). A subscriber expresses his interest as *subscriptions* and registers them to a notification service. A publisher creates a new *message* with its *description* and submits it to the notification service. The notification service compares a description of each message with each subscription, then *notifies* a subscriber about new messages matching his interest.

M. Agosti et al. (Eds.): IRCDL 2011, CCIS 249, pp. 75–86, 2011.

We can find this pattern in many services in digital library systems. For instance:

1. **Personalized new item list**
   A user subscribes to messages about new items that match his interest.
2. **Digital preservation service**
   An expert subscribes to messages about operations affecting items that he is responsible for their consistency.
3. **User generated context service**
   A content creator subscribes to messages about reviewing processes of items he has submitted.

However required methods to compare descriptions and subscriptions are slightly different in each scenario. In the scenario 3, the notification service needs to find message descriptions that contain a reviewer's id, a content creator's id and an item id designated in a subscription. For this purpose, we need only exact matching over ids to implement this scenario. On the other hand, when a user subscribes to "European paintings" in the scenario 1, the notification service should notify him about not only new "European paintings" but also new "French paintings", "Italian paintings", "Spanish paintings", and so forth. Scenario 1 and 2 require a more sophisticated method which takes account of is-a relations (or *subsumption relations*) to compare terms in descriptions and subscriptions.

In the ASSETS (Advanced Service Search and Enhancing Technological Solutions for the European Digital Library), which is an EU co-funded project aiming to improve the usability of Europeana digital library platform by developing software services and user interfaces focused on search and browsing [1], we are trying to satisfy such heterogeneous requirements for notification by introducing a layered architecture to the design of the notification service. In the digital library platform developed by ASSETS, common functionalities shared by all notification services are implemented in the "common notification service". On the top of this common service, each technical partner in the ASSETS project can develop his own "extended" service which performs more specific application notification.

The taxonomy based notification service we would like to report here is one of those extended notification services in the ASSETS platform. This service enables to compare descriptions and subscriptions by taking account of is-a relationships (or subsumption relations) among terms represented in a taxonomy. In the rest of this paper, we would like to show an outline of the ASSETS notification facility in section 2. Then we would like to give a description about a model for the taxonomy based notification implemented in our notification service in section 3. In section 4, we review some related research works. The development of the project described in this paper is still ongoing. In section 5, we would like to conclude this paper with some remarks about what we are going to achieve in a short term, and what we will extend this work in more future work.

2 ASSETS Notification Facility

Through discussions with the developers from ASSETS partners, we have analyzed several usage scenarios of notification services including scenarios we explained in the last section. Based on the insight coming from this analysis, we have divided the ASSETS notification facility into a common part and a set of specific application parts. Fig 1 illustrates the architecture of ASSETS notification facility.

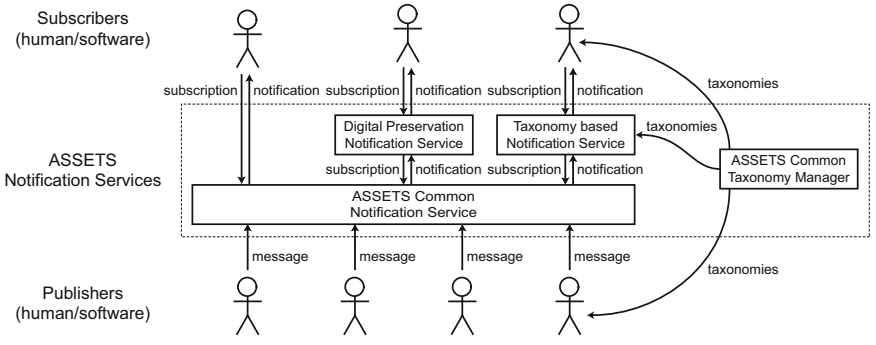

Fig. 1. Architecture of the ASSETS notification facility

The base of the notification facility is the ASSETS common notification service. On the top of the common notification service, several extended notification services are implemented for more specific application purposes. The common notification service is the single point to submit all messages created by publishers. On the other hand, every notification service accepts subscription requests from subscribers. The common notification service aggregates all published message and performs a light-weight exact matching to compare message descriptions and subscriptions. Each extended notification service works as a client of the common notification service and it subscribes to specific subsets of all published messages. On the extracted part of message streams, each extended service performs more advanced comparison for more specific application notification. This layered design for implementing the publish/subscribe pattern enables 1) decoupling of publishers and subscribers, 2) single-source and multiple-use of massages, and 3) easy development of extended notification service.

The taxonomy based notification service is one of those extended services. This service performs taxonomy based matching on condition that publishers, the notification service and subscribers use the same taxonomy. To share the same taxonomy with all principals in the notification workflow, the ASSETS notification facility has a taxonomy manager service which allows clients to register taxonomies and to retrieve registered taxonomies. By using this service, publishers and subscribers can get appropriate terms from a shared taxonomy for making

their messages or subscriptions. Our notification can also access the same taxonomy for taxonomy based matching through the taxonomy manager service.

3 Model of the Taxonomy Based Notification

The implementation of our taxonomy based notification is based on the approach proposed in [2]. The experimental results for evaluating the cost benefit obtained by this approach can be found in [3]. In this paper, we would like to introduce only the taxonomy based notification model used in this approach without details of algorithm and its cost evaluation.

Roughly speaking, the *notification* in our context is an activity to notify a subscriber about published *messages* whose *descriptions* match some *subscriptions* registered by the subscriber. The *taxonomy based notification* is a variation of notification that tests whether a message matches a subscription or not by taking account of a taxonomy. We would like to start our explanation from the formal definition of taxonomy in our model.

3.1 Terms, Subsumption Relations, and Taxonomies

Definition 1. Let T be a set of keywords, or *terms*. A *taxonomy* $\mathcal{T}$ defined over T is a tuple $(T, \preceq)$ where $\preceq$ is a reflective and transitive binary relation over T, called *subsumption relation*.

Given two terms s and t, if $s \preceq t$ then we say that s is *subsumed* by t, or that t *subsumes* s. In our work, we assumes that every taxonomy $(T, \preceq)$ is a tree in which the nodes are the terms of T and where there is an arrow $s \rightarrow t$ iff s subsumes t in $\preceq$.

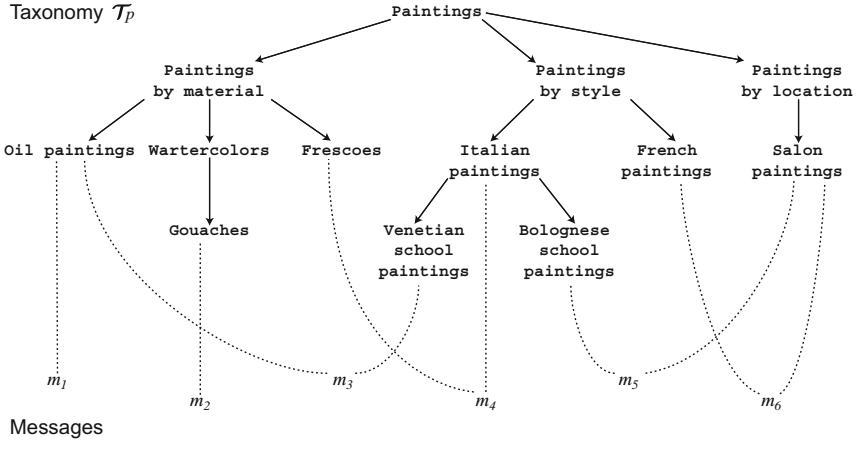

Fig. 2. A taxonomy and messages

Fig 2 shows the example taxonomy $\mathcal{T}_p$ we use in this section. In this example, the term `Watercolors` subsumes the term `Gouaches`, `Italian paintings` subsumes `Venetian school paintings` and `Bolognese school paintings`. Due to the transitivity of the subsumption relation, the term `Paintings` subsumes all terms in the tree including itself.

3.2 Descriptions

In order to deliver a message to appropriate subscribers, we need to provide a description of the message together with the identifier of the message. In informal definition, a description in our model is just a set of terms from a taxonomy. If the message is about an oil painting in a Venetian school style, the description of this message over $\mathcal{T}_p$ is {`Oil paintings, Venetian school paintings`}.

A description can be redundant if some of the terms it contains are subsumed by other terms. For instance, the description {`Watercolors, Gouaches, Italian paintings`} is redundant, as `Watercolors` subsumes `Gouaches`. Redundant descriptions are undesirable as they can lead to redundant computations during subscription evaluation. We shall limit our attention to non-redundant descriptions that have reduced sets of terms defined as follows:

Definition 2. Given taxonomy $(T, \preceq)$, a set of terms D from T is called *reduced* if for any terms s and t in D, $s \not\preceq t$ and $t \not\preceq s$.

Following the above definition, we shall make non-redundant descriptions by removing all but the minimal terms instead of removing all but the maximal terms. Because the former way produces more accurate descriptions than the other. This should be clear from our previous example, where the description {`Gouaches, Italian paintings`} is more accurate than {`Watercolors, Italian paintings`}. Finally, we formally define descriptions as follows:

Definition 3. Let m be a message. Given taxonomy $\mathcal{T}$, a *description* describing m over taxonomy $\mathcal{T}$ is a tuple $\langle D, m\rangle$ where D is a reduced set of terms in $\mathcal{T}$.

The dotted lines in Fig 2 indicate what descriptions actually represent. Conceptually, a description $\langle D, m\rangle$ can be thought as a set of pairs (t, m) for all $t \in D$. we can represent a set of those pairs as a binary relation R between terms and message identifiers. The relation R_p illustrated in table 1 is a binary relation that actually stores all descriptions for $m_1 \cdots m_6$ in Fig 2.

3.3 Subscription and Notification

In our model, a *query* is just a conjunction of terms from a taxonomy. In the following definition the symbol $tail(t)$ stands for the set of all terms in the taxonomy $\mathcal{T}$ strictly subsumed by t, that is $tail(t) = \{f | f \prec t\}$, R stands for the binary relation representing descriptions. Queries are defined over $\mathcal{T}$ and they are answered based on R.

Table 1. Relation R_p

Term t	Message Identifier m
Oil paintings	m_1
Oil paintings	m_3
Gouaches	m_2
Frescoes	m_4
Venetian school paintings	m_3
Italian paintings	m_4
Bolognese school paintings	m_5
French paintings	m_6
Salon paintings	m_5
Salon paintings	m_6

Definition 4. A query q over $\mathcal{T}$ is either a single term or a conjunction of terms from $\mathcal{T}$. Its answer, denoted by $ans(q)$, is a set of documents defined as follows:

Case 1: q is a single term t from $\mathcal{T}$, i.e., $q = t$
$ans(q) =$ **if** $tail(t) = \emptyset$ **then** $\{d|(t, m) \in R\}$ **else** $\bigcup\{ans(s)|s \in tail(q)\}$

Case 2: q is a conjunction of terms, i.e., $q = t_1 \wedge t_2 \cdots \wedge t_n$
$ans(q) = ans(t_1) \cap ans(t_2) \cdots \cap ans(t_n)$

For instance, let $q = \{$`Oil paintings`, `Italian paintings`$\}$ be a query over taxonomy $\mathcal{T}_p$ in Fig 2, its answer $ans(q)$ based on R_p in table 1 is $ans(q) = \{m_1, m_3\} \cap \{m_3, m_4, m_5\} = \{m_3\}$.

A *subscription* in our model is just a query describing (intentionally) the set of messages of interest to a user. In reality, a user can define his subscription by selecting terms from the taxonomy. The conjunction of the selected terms is the user's subscription. Actually, henceforth, we shall think of a subscription either as a set of terms (e.g., {`Oil paintings, Italian paintings`}) or as a query (e.g., `Oil paintings` $\wedge$ `Italian paintings`).

Now we define *notification* as an activity to inform about existence of answers for registered subscriptions (or queries).

Definition 5. Let q be a query registered as a subscription by subscriber s. For a set of published messages represented as binary relation R, $ans(q) \subseteq R$ answered based on R is *notified* to s.

3.4 Refinement Relations and Subscription Trees

A naïve implementation of the notification is to test whether each incoming message should be notified or not for *every* subscriptions. However, if the set of subscriptions is large, and/or the rate of events is high, the system might become quickly overwhelmed.

In our project, we use a more efficient approach based on the observation that this testing is basically a set membership test (i.e. testing whether a message belongs to a given set of messages). The idea is the following: if we have to perform test membership for every set in a collection of sets, we can save computations by starting with maximal sets first (maximality with respect to set inclusion). Indeed, if a message does not belong to a maximal set then we don't need to test membership for any of its subsets.

In order to implement this idea, we need to define first a notion of refinement between subscriptions. In fact, we need a definition that translates the following intuition: if subscription q_1 refines subscription q_2 then every event that matches q_1 also matches q_2.

Definition 6. Let q_1 and q_2 be two subscriptions. We say that q_1 is *finer* than q_2, denoted $q_1 \sqsubseteq q_2$, iff $\forall t_2 \in q_2, \exists t_1 \in q_1 | t_1 \preceq t_2$.

For instance, the subscription {`Gouaches, Venetian school paintings, Bolognese school paintings`} is finer than {`Watercolors, Italian paintings`}.

A set $\mathcal{S}$ of all subscriptions and refinement relation $\sqsubseteq$ over $\mathcal{S}$ becomes an upper semilattice. For details, see [4].

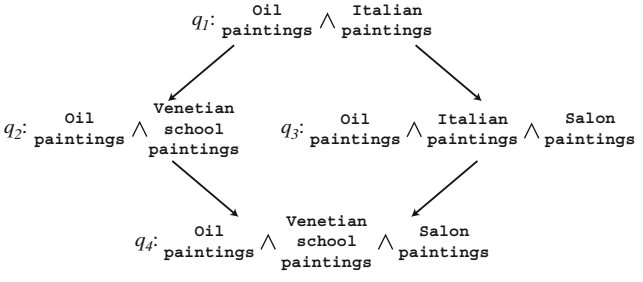

Fig. 3. A refinement relation

The graph G_s in Fig 3 represents the refinement relation over subscriptions $q_1, \cdots, q_4$. With this graph, we start testing of a message for notification from q_1 which is the root of this graph. However, there are two paths from q_1 to q_4 in G_s. As demonstrated in this example, the graph of a refinement relation can contains multiple paths leading to the same subscription. For a efficient traversal in a refinement relation graph, we need to extract a spanning tree from the graph. We call this tree *subscription tree*. An optimal subscription tree depends of probability of a message matching each subscription. An optimized algorithm for obtaining and incrementally updating optimal subscription trees is presented in [2].

3.5 Implementation Approach

A problem in implementing the "optimal" algorithm proposed in [2] is that we usually cannot obtain fixed filtering rates. A filtering rate of a subscription is a probability of a message matching the subscription. If we estimate filtering rates from statistics of past messages, filtering rates change every time when a notification system accepts a message. The optimal subscription tree also might change along with changes of filtering rates. Filtering rates and optimal trees dynamically change during a publish/notification process. A naïve approach to keep trees optimal is to construct a subscription tree from scratch when filtering rates are changed, in other words, when a message comes. Of course, it is not an acceptable solution because of its calculation cost.

Another problem is the cost to estimate filtering rates. To estimate a filtering rate of a given subscription, we need to test the subscription with past messages to count messages matching the subscription. It means that the notification system needs to perform many matching tests every time when a user registers a new subscription.

In this project, we adopt the following compromise approach for these problem:

- Subscription trees are not updated along with changes of filtering rates when a new incoming message comes.
- Subscription trees are periodically destroyed and re-constructed by using current instance of filtering rates.
- To estimate the filtering rate of a subscription, we use the *hit count* that indicates the number of messages that have matched the subscription during notification processes.

Filtering rates are firstly reflected to subscription trees by adding new subscription nodes to the subscription trees by taking account of current filtering rate based on the algorithm proposed in [2]. However, each change of filtering rates caused by a new incoming message is not reflected to the trees one-by-one. Those changes are reflected in bulk by periodically constructing new subscription trees from scratch. To construct a whole subscription tree is an expensive operation. Therefore, this operation is performed as a house-keeping task at a specific time interval.

Regarding filtering rates, we count numbers of messages that match each subscription in notification processes. We directly use this *hit count* to estimate filtering rate. Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of subscriptions, $h(s)$ be the hit count of subscription s. In this approach, we simply estimate filtering rate $\sigma(s_i)$ of subscription s_i as $\sigma(s_i) = h(s_i) / \sum_{j=1}^{n} h(s_j)$. Important difference from the naïve approach is that we do not compare a new subscription with any past messages. We test a subscription only with messages that come after the subscription has been registered. Therefore, a newer subscription usually has a less hit count than older one. This approach is based on an assumption that such bias will be flattened for the long-term period.

In future work, we would like to evaluate the quality of subscription trees constructed based on this compromise approach by comparing with ones constructed by the optimal approach.

4 Related Work

User notification has been studied for a long time and many systems support this function. The different methods of expressing the interest have led to several subscription schemas. Corresponding to different subscription schemas, the user notification systems can be classified as *subject-based* and *content-based*. In subject-based user notification systems, each publication belongs to a certain subject. The subscribers specify their interest in the particular subjects and they receive notifications whenever publications dispatched within the particular subjects. Subject-based notification systems limit the selectivity of subscriptions. The content-based system improves on subject-based system by allowing the subscription to contain complex query on publications' content.

As the amount of information of the information system increase very fast, efficient and timely dissemination of messages is a key to distributing information to assist end-users. The matching computing problem has been investigated a lot. For tree-based data, [5–8] give algorithms for filtering XML documents based on XPath. For graph-structured data, [9–12] provide some methods and algorithms for tackling the graph-structured data filtering.

We use subject based schema namely taxonomy based data model as presented in [2] in ASSESTS digital library platform for considering its following practical benefits:

- Give suggestions for publication description and interest subscription. With the taxonomy data, the publisher can find some suitable professional terms to describe the contents that going to be published; on the other side, the subscribers can get suggestions during the subscription process, such as the user is interested in the architecture of ancient Greece, however, she is not quite familiar with the arts of ancient Greece, and she does not know exactly the name of the ancient architectures that were constructed in that period, then by browsing and clicking on the node named "Architecture of ancient Greece" in the taxonomy tree, the node can pop out sub-nodes named "Temple of Hephaistos", "Erechtheion" and the other ancient Greece architecture nodes. Therefore, the taxonomy data model can help the user to find the interesting things and the user can subscribe some of them.
- Provide more exact notification messages for the subscribers. As described above, both the publication and subscription description use the terms from a same taxonomy, therefore the notification messages sent to the subscribers can exactly match the interests of the subscriber. Since no one likes spam messages, if the notification messages sent to the users are far from their interests, they may block the service. In addition, if we send a lot of notification messages to the users including the publications that have week matching

to the subscription, then the large volume of messages would make the subscribers overwhelmed and the user may neglect the service.
- Improve partially the "cold start" problem. "cold start" problem could be briefly described as at the beginning of the service goes live, there are not a lot of users would like to use the service. Taxonomy based data model can partially resolve the problem is because there are quite a part of people that do not know what to subscribe if just let them think about it by mind, therefore they do not use the service, however if we provide a taxonomy tree for them, possible the users just want to have a short time browsing of the taxonomy tree, while during the browsing, some of the terms may attract the interests of the users, then they use the service.
- Avoid "zero notification" problem. If we let the users express their interest with any words they would like to use, there is the possibility that no notification messages would be sent to the users, because the words typed by the users are possible have no connections with our platform or the words are miss typed that the system can not retrieve the relevant publications for the subscription.
- Simplify the subscription process. As mentioned above, if let the users express their interests just by thinking it in mind and write it down, there are a large part of the users would have no interest to express their interest, they just do not want to think too much. Therefore we provide a knowledge tree for the users, and by simple clicking operations they can find their interests occasionally, and then subscribe them. The taxonomy based data model can provide a simple and convenient service for the users.

From the above description, we can find that the taxonomy data is very important for the success of the whole service, therefore a period dynamically maintain and improve of the taxonomy data is necessary. We will improve the taxonomy data based on the following methods:

- Synchronize with the content augmentation of the digital library. Assume there will a large volume of "Middle Ages European paintings" be added to the digital library, while there is no such kinds of terms in the taxonomy tree, the taxonomy should be modified by adding a subtree with the terms describing the newly added contents.
- Pay attention to the publishers' feedback. Some of the publishers are very professional, possible they are not very satisfied with some of the terms in the taxonomy or there are no suitable professional terms as they expected. Then we encourage them to send feedback to us, based on the analysis of their feedback, we improve the taxonomy data.
- Take into account the subscribers' feedback. The subscribers' feedback is also very important for us, such as some of the users would like to book the notification information that does not exist in the taxonomy data tree, we will collect and analyze their requirements and consider if we can add the relevant contents to the digital library and modify the taxonomy. We also have a lot of professional users that their feedback is very important for our improvement.

– Analyze the search log. The search records usually give feedback about the interests of the users, on the one side we should analyze if the platform can response the contents that meet the users requirement; on the other side, we should check if the taxonomy data cover all or most of the interests of the users.

5 Conclusion

In this paper, we described an outline of our taxonomy based notification service for the ASSETS digital library problem. The major aim of this work is to make our service not only functional, but also reusable for implementing various features required by modern digital libraries. To achieve this goal, we carefully designed our notification model as an application-independent model. The current model is just a message exchange model and it does not include any concepts specific to digital libraries. Developers can implement actual notification applications on the top of (namely, as a client of) our notification service by associating concepts in each application domain with our model.

At present, the development task of our taxonomy based notification service is ongoing. As short-term goal we will finalize the implementation of our notification service, then we will develop a "new item list" service as an example application of our service. Through this short term task we would like to proof our concept and do some preliminary evaluations of performance. In future work we expect that some "spin out" technologies being developed in the ASSETS project are adopted as a part of the Europeana digital library portal. If we have a chance to apply our technology to the real Europeana, it is an exciting challenge to deal with real number of users visiting a big portal by using our technologies. To achieve enough performance under such a massive setting, we might need to introduce additional technique for subscription lookup like hashing or indexing.

Acknowledgement. This work was partially supported by the European project ASSETS: Advanced Search Services and Enhanced Technological Solutions for the European Digital Library (CIP-ICT PSP-2009-3, Grant Agreement no 250527).

References

1. ASSETS project, `http://www.assets4europeana.eu/`
2. Belhaj-Frej, H., Rigaux, P., Spyratos, N.: User notification in taxonomy based digital libraries. In: 24th Annual ACM International Conference on Design of Communication, Myrtle Beach, SC, USA, pp. 180–187 (2006)
3. Belhaj-Frej, H., Rigaux, P., Spyratos, N.: Fast user notification in large-scale digital libraries: experiments and results. In: 11th East European Conference on Advances in Databases and Information Systems, pp. 343–358. Varna, F.J. Curry House of Scientists, Bulgaria (2007)

4. Rigaux, P., Spyratos, N.: Metadata Inference for Document Retrieval in a Distributed Repository. In: Maher, M.J. (ed.) ASIAN 2004. LNCS, vol. 3321, pp. 418–436. Springer, Heidelberg (2004)
5. Diao, Y., Fischer, P., Franklin, M., To, R.: Yfilter: Efficient and scalable filtering of XML documents. In: 18th International Conference on Data Engineering, San Jose, USA, pp. 341–342 (2002)
6. Hou, S., Jacobsen, H.: Predicate-based filtering of xpath expressions. In: 22nd International Conference on Data Engineering, Atlanta, Georgia, USA (2006)
7. Chan, C., Felber, P., Garofalakis, M., Rastogi, R.: Efficient filtering of XML documents with XPath expressions. VLDB Journal 11, 354–379 (2002)
8. Gupta, A., Suciu, D.: Stream processing of xpath queries with predicates. In: ACM SIGMOD International Conference on Management of Data 2003, New York, USA, pp. 419–430 (2003)
9. Petrovic, M., Liu, H., Jacobsen, H.: G-ToPSS: Fast filtering of graph-based metadata. In: 14th International World Wide Web Conference, Chiba, Japan, pp. 539–547 (2005)
10. Petrovic, M., Burcea, I., Jacobsen, H.: S-ToPSS:Semantic Toronto publish/subscribe system. In: 29th VLDB Conference, Berlin, Germany, pp. 1101–1104 (2003)
11. Wang, J., Jin, B., Li, J.: An Ontology-Based Publish/Subscribe System. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 232–253. Springer, Heidelberg (2004)
12. Haarslev, V., Moller, R.: Incremental Query Answering for Implementing Document Retrieval Services. In: International Workshop on Description Logics 2003, Rome, Italy, pp. 85–94 (2003)