HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

Towards the Semantic Web and Web Services

Proceedings of the XML Finland 2002 Conference

Eero Hyvönen and Mika Klemettinen (editors)

October 21-22, 2002

HIIT Publications 2002–03

Towards the Semantic Web and Web Services

Proceedings of the XML Finland 2002 Conference

Eero Hyvönen and Mika Klemettinen (editors)

Helsinki Institute for Information Technology (HIIT)
Tammasaarenkatu 3, Helsinki
P.O. Box 9800
FIN-02015 HUT
FINLAND
http://www.hiit.fi

HIIT Publications 2002–03
ISBN 1458-951-22-6180-4 (printed version)
ISBN 1458-951-22-6181-2 (electronic version)
ISSN 1458-9451 (printed version)
ISSN 1458-946X (electronic version)

Foreword

This volume contains the research papers presented at the XML Finland 2002 conference in Helsinki, Finland, October 21-22, 2002 – the seventh annual conference organized by the XML Finland Association. The themes of the year 2002, the Semantic Web and Web Services, attracted some 200 participants.

The Semantic Web is a vision of the W3C consortium about the next generation Web, which is used not only by the humans but also by the machines. The vision is brought alive with XML-based "semantic" standards and interfaces that make web contents understandable to the machines. This enables the creation of more intelligent Web Services than before.

After the previous XML Finland conference in 2001, the feedback from the XML Finland Association members showed us the clear interest in these topics. At that time, Finnish universities, research centers, and companies were initiating research and development projects in this area. The XML Finland Association together with the patrons and sponsors of the conference are glad to have the opportunity to bring you the latest international and domestic news and research results concerning the Semantic Web and Web Services.

We were honored to have as the keynote speakers two leading experts: Prof. Dieter Fensel from the Leopold-Franzens Universität, Innsbruck, and the European OntoWeb network, and Research Fellow Ora Lassila from the Nokia Research Center, Boston. Their presentations enlightened the current state of the art in the Semantic Web and Web Services as well as envisioned the future developments. In addition to the keynote presentations, several invited talks, hands-on tutorials, and technical papers were presented. To complement the conference, major vendors presented their products and solutions in an exhibition.

One of the most important goals of the XML Finland conferences is to bring together XML researchers, professional users, and interested people with less experience of using the technologies. We hope that the XML Finland 2002 conference enabled its participants to make contacts and share experiences and ideas. This proceedings – together with a separate volume containing the presentation slides of the other talks – makes the ideas and results available to a larger audience.

We would like to take the opportunity to thank all the members of the XML Finland 2002 Organizing Committee and Program Committee, authors, presenters, exhibitors, and the participants of the conference. We would also like to thank our supporters OntoWeb, Sun Microsystems, Microsoft, TIEKE Finnish Information Society Development Centre, Helsinki Institute for Information Technology (HIIT), the University of Helsinki, Nokia, Done Information, eCraft Management Solutions, Genio, Tieturi, Citec, Dataclub, Index IT, Innovative Ideas, Ontopia, Republica, Profium, and TietoEnator.

Helsinki, October 15, 2002

Kaisa Kostiainen
Done Information
Chair of the XML Finland 2002 Organizing Committee

Mika Klemettinen
Nokia Research Center
Chair of the XML Finland 2002 Program Committee

Eero Hyvönen
University of Helsinki and Helsinki Institute for Information Technology
Co-Chair of the XML Finland 2002 Program Committee

Conference Organization

Program Committee

Chair

Mika Klemettinen, Nokia Research Center

Co-Chair

Eero Hyvönen, University of Helsinki and HIIT

Members

Leyla Akgez, Nokia
Tomi Kauppinen, Genio Innovations
Kaisa Kostiainen, Done Information
Mikko Lounela, Research Institute for the Languages of Finland
Mikko Paananen, Republica
Maija Pennanen, Geological Survey of Finland
Petri Pusa, SysOpen
Jörgen Westerling, eCraft Management Solutions

Organizing Committee

Chair

Kaisa Kostiainen, Done Information

Members

Leyla Akgez, Nokia
Eero Hyvönen, University of Helsinki and HIIT
Tomi Kauppinen, Genio Innovations
Mika Klemettinen, Nokia Research Center
Mikko Lounela, Research Institute for the Languages of Finland
Mikko Paananen, Republica
Maija Pennanen, Geological Survey of Finland
Petri Pusa, SysOpen
Jörgen Westerling, eCraft Management Solutions

Table of Contents

Key Note

WSMF - Towards the Semantic Web Services 1
Dieter Fensel
Leopold-Franzens Universität and OntoWeb

Semantic Web Research and Applications

Yellow Pages on the Semantic Web 3
Eero Hyvönen, Kim Viljanen, and Antti Hätinen
University of Helsinki and HIIT

Ontology-Based Image Retrieval 15
Eero Hyvönen, Avril Styrman, and Samppa Saarela
University of Helsinki and HIIT

Ontology-Based Semantic Metadata Validation 28
Vilho Raatikka and Eero Hyvönen
University of Helsinki and HIIT

Semantic Interoperability on the Web: Case Finnish Museums Online 41
Eero Hyvönen, Suvi Kettula, Vilho Raatikka, and Kim Viljanen
University of Helsinki and HIIT

Building and Maintaining Web Taxonomies 54
Wray Buntine, Sami Perttu, and Henry Tirri
University of Helsinki and HIIT

Web Services and XML in eBusiness

Semantic Web and Web Services from Operator's Perspective 66
Lasse Pajunen, Vesa Huotari, and Seppo Heikkinen
Elisa Communications

Combining WSDL and RDF Data Models 75
Janne Saarela
Profium

Comparing XML Based B2B Integration Frameworks 84
Paavo Kotinurmi
Helsinki University of Technology

XML Research

A Journalist's Tool Using XML for Writing and Retrieving News Stories 96
Martin Fluch, Greger Lindén, and Andrei Popescu
University of Helsinki

Perinnedatan automaattinen muotoilu XML-tekniikoin 109
Merja Ek, Heli Hakkarainen, Pekka Kilpeläinen, and Tommi Penttinen
University of Kuopio

Aiming Towards Best Practices in XML Techniques for Text Corpora Annotation:
City of Helsinki Public Works Department Press Releases - A Case Study 123
Mikko Lounela
Research Institute for the Languages of Finland

Extracting More Relevant Document Descriptors Using Hierarchical Information 136
Antoine Doucet
University of Helsinki

Semantic Web Enabled Web Services

Dieter Fensel

Institut für Informatik, Leopold-Franzens Universität Innsbruck
Technikerstrasse 25, 6020 Innsbruck, Austria
Tel. (mobil): +31-(0)6-51850619, Fax: +43-512-940685
email: dieter.fensel@uibk.ac.at
http://www.google.com/search?q=dieter or http://www.fensel.com

Currently, computers are changing from single, isolated devices into entry points to a worldwide network of information exchange and business transactions called the World Wide Web. However, the easy information access based on the success of the web has made it increasingly difficult to find, present, and maintain the information required by a wide variety of users. In response to this problem, many new research initiatives and commercial enterprises have been set up to enrich available information with machine-understandable semantics. This **Semantic Web** will provide intelligent access to heterogeneous, distributed information, enabling software products to mediate between user needs and the information sources available (cf. [Fensel, 2001], [Davies et al., 2002], and [Fensel et al, 2002(a)]).

Web Services tackle with an orthogonal limitation of the current web. It is mainly a collection of information but does not yet provide support in processing this information, i.e., in using the computer as a computational device. Recent efforts around UDDI[1], WSDL[2], and SOAP[3] try to lift the web to a new level of service. Software programs can be accessed and executed via the web. However, all these service descriptions are based on semi-formal natural language descriptions. Therefore, the human programmer need be kept in the loop and scalability as well as economy of web services are limited. Bringing them to their full potential requires their combination with semantic web technology. It will provide mechanization in service identification, configuration, comparison, and combination.

Semantic Web enabled Web Services have the potential to change our life in a much higher degree as the current web already did. We will discuss the potential impact of this technology in areas such as knowledge management, enterprise application integration, supply-chain management, and virtual enterprises. Our vision is to provide technology that is scalable and economic that it enables small and medium sized enterprises to joined the game of eWork and eCommerce (cf. [Fensel et al., 2002(b)]).

[1] http://www.uddi.org/
[2] http://www.wsdl.org/
[3] http://www.soap.org/

There are already initiatives such as the DAML-S[4] effort in the U.S.A. and the WSMF[5] in Europe that provide initial proposals for merging semantic web with web service technology. We are currently in the process of merging these initiatives taking a key lesson of the web into account: scalability and economy in price relay essential on successful and truly international standardization.

References

[Davis et al., 2002]
J. Davis, D. Fensel, and F. van Harmelen (eds.): *Towards the Semantic Web: Ontology-Driven Knowledge Management*, Wiley, to appear 2002.

[Fensel, 2001]
D. Fensel: *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*, Springer-Verlag, Berlin, 2001.

[Fensel et al., 2002(a)]
D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster (eds.): *Spinning the Semantic Web: Bringing the World Wide Web to its Full Potential*, MIT Press, Boston, September 2002.

[Fensel et al., 2002(b)]
D. Fensel, B. Omelayenko, Y. Ding, E. Schulten, G. Botquin, M. Brown, and A. Flett: *Intelligent Information Integration in B2B Electronic Commerce*, to appear 2002.

[4] http://www.daml.org/
[5] http://swws.semanticweb.org/

Yellow Pages on the Semantic Web

Eero Hyvönen[1,2], Kim Viljanen[2,1], and Antti J. Hätinen[1]

[1] University of Helsinki, Department of Computer Science
`firstname.lastname@cs.helsinki.fi`,
`http://www.cs.helsinki.fi/group/seco/`
[2] Helsinki Institute for Information Technology (HIIT)

Abstract. Yellow pages catalogs and corresponding directory services on the web are a widely used business concept for helping people to find companies providing services and selling products. When on the web, matching the customer's need with the relevant services offerred by companies is typically based on keyword search, table-based search, a list of service categories listed in some order, a hierarchical category system, or a combination of these. In spite of the versatility of possibilities, it can still be difficult to the end-user to map a need on the services offered. On the other hand, for the catalog advertiser, it may be difficult to index the service in such a way that the prospects would not miss the service. We propose that in order to enhance the recall and precision of yellow page services, the advertisements should be annotated using semantic web ontologies. Based on such conceptual definitions, the user can be provided with new content-based searching and browsing facilities, which makes the service more profitable to the advertisers and the directory service provider. As a first step towards this goal, an experimental semantic yellow page implementation is presented.

1 Introduction

Yellow pages is a widely used and commercially successful service model for matching the need of a customer with the corresponding products and services offered by companies. In this model, the yellow pages service provider maintains a list or a hierarchy of product and service categories, such as "Electronic equipment" or "Car Rental". The provider's customers are companies that buy advertisement space in the catalog within these categories. The business logic of yellow pages is based on the advertising companies' belief on the fact that the catalog is extensively used by their prospects in finding products and services. Yellow page catalogs are an especially important marketing channel for small companies that cannot offer large marketing campaigns but still need get in contact with a large potential customer base. Yellow pages are traditionally published in printed form once a year as a part of local telephone books. On the WWW, lots of similar directory services are provided online.

In this article, we discuss how Semantic Web [9] technologies can be employed in yellow page services. Our idea is to design ontologies [10] classifying the yellow page contents from the usage point of view. By annotating the advertisements with metadata based on such ontologies, content-based search functions and service recommendation can be provided to the end-user. We focus on the problem of how to match the customer need with what the companies offer in their advertisements. The Helsinki area yellow pages[3] are used as a test case.

In the following, the current state of the art of yellow page services is first outlined. Section 3 presents our vision of how the semantic web technologies can help to solve problems in current online yellow page services. In section 4 we describe our first prototype of a semantic yellow page service. Section 5 summarizes the work and discusses some open problems.

2 Yellow pages: state of the art

Printed yellow pages The Helsinki area yellow pages (HYP) (2002) is a typical example of a yellow page catalog. HYP is based on an alphabetical category list of 1085 products and services. From the user's viewpoint, a problem of using the catalog is that a user's need may be related to several advertisement categories. For example, if one needs help for a headache, the possibly relevant categories in HYP include ”pharmacy shops” (in Finnish: apteekkeja), ”medical consultancy” (lääkeneuvontaa), ”doctors” (lääkäreitä), ”medical centers” (lääkärikeskuksia), ”medical labs” (lääketieteellisiä laboratorioita), ”hospitals” (sairaaloita), perhaps even ”nursing services” (sairaanhoitoalan palveluja). On the other hand, a category may be related to several needs. For example, ”building equipment” (rakennustarvikkeita) is potentially relevant in many building needs and problems. Furthermore, several categories are partly overlapping, such as ”barber's shops” (parturit) and ”hair dressers” (kampaamot). In the bilingual HYP, there is also the additional problem that many, but not all, categories have both Finnish and Swedish categories. For example, printing services are advertised under the Finnish label ”painopalveluja” and the Swedish term ”tryckerier”.

From the advertiser's viewpoint, the advertisement should be found in as many reasonable ways as possible without bothering all possible customer needs, categorizations or linguistic terminology issues. In the current category system, this not possible but the company may have to buy advertisement space in several categories for better coverage. This is not always feasible to the company and leads to the situation where many useful categories in the list are practically empty. This is not what the user expects from a good service. For example, the category ”garden furniture” (puutarhakalusteita) in HYP contains only one advertisement although many, if not most furniture shops sell also garden furniture. From the user's and advertiser's viewpoint, the challenge is to create a search mechanism with high recall (relevant advertisements found)[4] and high precision (irrelevant advertisements not found)[5].

[3] http://www.keltaisetsivut.fi/

[4] Recall r is defined as the ratio $r = f/a$, where f is the number of retrieved relevant documents and a the number of all relevant documents. [4]

[5] Precision p is defined as the ratio $p = f/n$, where f is the number of retrieved relevant documents and n the number of all retrieved documents. [4]

Online yellow pages Yellow page services online can benefit the user in providing 1) new information content (e.g., hyperlinks to up-to-date business web sites), 2) better searchability of contents, and 3) interactive features (e.g., online enquiry forms and feedback mechanisms) [2]. A typical[6] online yellow page service provides the user with the following functions:

Keyword and table-based search Keyword search is used for finding the category names, the advertisements, and product names (e.g., in Brabys.com). The search may be multi-lingual (e.g., in web.wlwonline.de). Some services, such as SuperPages.com and Keltaisetsivut.fi, provides the user with table-based search along various data fields, such as the business name and location. The search result is usually presented as a hit list of matching advertisements that may be grouped according to their categories (e.g., in SuperPages.com).

Hierarchical category navigation Some services provide the user with a hierarchical navigation facility by which the user can focus from top categories to subcategories and finally to the advertisements (e.g., Brabys.com).

Location services Most of the online directory services provide location-based services that typically include maps and driving directions for finding to a given place.

Editorial content An example of editorial content is the Yell.com's shopping guide for buying a car. It provides links to related categories which may be in the interest of the user, such as car dealers, driving schools, and garage services. In this paper, we will call such semantic associations *recommended links*. In SuperPages.com, recommended links associate a selected category to an other category and in SMARTpages.com pop-up banner windows show advertisements related to the selected category.

Search engines vs. web catalogs An increasing number of companies have their own web sites. Search engines, such as Google[7] and AltaVista[8], are getting ever better in finding relevant information. This means a threat to the well-established business model of yellow pages. Using search engines is appealing for several reasons:

Matching the need with the offer It is easy for the prospect to try to locate companies by just typing in a few keywords in a search engine. The prospect does not have to figure out what categories in the catalog may be relevant from the point of view of his need. For example, if my video camera needs to be repaired I do not have to know in what yellow page categories there may be relevant advertisements for my problem. This mental matching problem is

[6] We investigated the following representative sites: http://www.smartpages.com/, http://www.superpages.com/, http://uk.yell.com/, http://www.brabys.com/, http://web.wlwonline.de/, http://www.keltaisetsivut.fi/, http://www.yritystele.fi/.

[7] http://www.google.com

[8] http://www.altavista.com

especially severe if the prospect is not familiar with the cataloging terminology and business conventions in use. For example, different products/services may be sold by different kind of companies in different countries.

Coverage of the catalog A yellow pages catalog lists only its customer's advertisements, but a general search engine may find also companies not listed in the catalog.

Quantity of information The data on the web sites is more abundant than on the catalog advertisements. For example, full product information with price lists may be readily available there.

Dynamicality Dynamicality of web pages is far beyond the static advertisements used in many catalogs. For example, order forms may be filled and returned electronically, videos clips may be used for showing the products, etc.

Timeliness The data is usually more up-to-date on the WWW than in the catalog that is updated only now and then. For example, there may be special offers for the week, updated price lists etc. on the web.

The keyword-based search methods of search engines are often useful, but have also their shortcomings [10, 6]:

1. A keyword in a document does not necessary mean that the document is relevant. For an extreme example, an advertisement may contain the phrase "We do not sell *stamps*, but ...", which means that a search using the keyword *stamp* will include the page in the hit list.
2. The engines cannot differentiate between *synonyms*. For example, a service selling "personal computers" is not found using the keyword "PC". This lowers the recall rate of information retrieval.
3. The engines do not understand *homonyms*[9]. For example, the keyword "Nokia" would find not only pages related to the Finnish telecommunication company, but also pages related to a city in Finland. This leads to low precision in information retrieval.
4. The engines do not understand general terms or phrases. For example, if your are interested in finding out "accommodation" you do not find advertisements with the words "hotels" or "inns" unless also the word "accommodation" is present.
5. Relevance. It is difficult to evaluate the relevance of a document with respect to a query. A list of 1000 hits is not very useful unless they can be ordered according to their relevance to the user.
6. Implicit information. A textual description is found only if it contains the explicit keyword. For example, one may be interested in companies dealing with "astronomy". A telescope advertisement is not found unless it happens to mention the word "astronomy", which may be too obvious to be mentioned.
7. Images and other non-textual binary documents cannot be matched.

[9] A homonym is a word with several meanings.

A benefit of the catalog is that the data there is structured, which makes the catalog easier to search than a collection of semi-structured WWW pages. Furthermore, yellow page services typically contains lots of small companies, such as barber shops, restaurants etc. that do not (yet) usually have their own web site. Finding such companies with a general purpose search engine like Google is not easy. Still another benefit of yellow page catalogs is that the service provider guarantees the correctness and reliability of the information in the database.

According to the semantic web vision [5], the search for specific information on the (semantic) web will become easier with the help of semantic metadata annotations. However, the need for trusted centralized (semantic) directories and (semantic) search engines stays. Yellow page services can potentially become one of the trusted information providers of the future semantic web.

3 A vision for semantic yellow pages

Service ontology to reflect the user's needs From the viewpoint of the user's need, indexing a company's advertisements according to a business category is not very useful unless the category unambiguously implies the services offered to the user. For example, camera repair service can potentially be offered by an importer company, appliances shop, camera shop, foto shop, optician etc. What kind of companies offer repair services depends, e.g., on the service business at hand, on the thing and brand being repaired, and on local practices. In order to enhance the search capabilities of yellow pages the advertising companies should more clearly state what services they actually offer.

A service cannot be characterized extensively by a simple category label, because services have internal semantical structure. For example, transportation services are characterized by properties such as the cargo (people, furniture, oil, etc.), the instrument used (car, train, ship, airplane, etc.), and the area of transportation (city, country, international, etc.). The internal properties of services can be described by using *ontologies* [10, 3, 6]. Services are often related to each other and make combined services. A classical example is the service needed for making a conference trip, which involves using related services such as buying flight tickets, reserving a hotel room, and registering for the conference. Service combination can be described in terms of ontological structures.

In our view, yellow page online services should be based on ontologies, that define the terminology and concepts to be used by the advertiser and by the end-user[10]:

Ontology-based service annotation From the advertiser's viewpoint, each advertisement in the system is a metadata description of some service that the company offers. A single company may offer many services. It is not enough for a camera shop to put one advertisement under category "camera shops" but, for example, one advertisement under camera "selling services",

[10] This idea has been applied before in other domains, e.g., in photo annotation and retrieval [18, 12]

another under ”repair services”, and a third one under ”rental services” depending on the company at hand. The user’s viewpoint is central, and the offerings should be indexed accordingly.

Ontology-based service retrieval To the end-user, the service ontology provides new ”content-based” possibilities to finding advertised services. The ontology can be used as a multidimensional semantic navigation directory for browsing the advertisements in the spirit of Topic Maps [17]. For example, services related to the concept ”Piano”, such as ”Piano shops”, ”Piano transports”, ”Piano tuners” etc. can be found through the product class ”Piano”, because this concept is used in different roles in these services.

The ontology also provides a kind of semantic glue that relates the service advertisements of different service classes with each other. If you want buy a flight ticket to Helsinki, you are likely to be interested in booking hotel accommodation there as well. Furthermore, if there are some special events, such as musical festivals taking place near by at the same time, you probably would not mind being informed about them as well. Such semantic relations between service classes can be defined in general terms within the ontology, and be used for generating recommended links for the user. For example, if the user selects the category ”flight services” with properties `destination=Helsinki, time=2002-10-18`, then the yellow page system could automatically show links to accommodation services in Helsinki and events taking place there at that time.

4 A prototype of semantic yellow pages

As a first step towards semantic web based yellow page services, we have implemented a prototype called ”Keltsi”. The goals in the work were 1) to get hands-on experience on the practical issues related to implementing a semantic web based yellow page service, 2) to test semantic web technologies in the yellow page context, and 3) to get experience on how to model the service ontologies and to annotate advertisements.

The Finnish Yellow Pages Ltd provided us with 150 MB of authentical advertisement data, 26452 advertisements in 2574 categories. The database contains advertisements with properties such as the category name (”Restaurants”, ”Camera shops” etc.), the advertisement text, contact information, and location information (including the municipality and coordinates).

4.1 The knowledge base

The semantic web is based on knowledge presented 1) in the form of *ontologies* and 2) instance level *metadata* [1] conforming to it. An ontology is typically defined as a set of classes interlinked with each other by the subclass relation. For example, the class ”HouseCats” could be a subclass of the class ”Pets”. Classes have *properties* that are called also slots or attributes. Properties may have *constraints* on their usage and other properties that are often called *facets*

[15]. For example, the class "Pets" may have the property "name" whose value must be a literal. Instances are related with corresponding ontological classes with an "instance-of" relation and inherit class properties via the class hierarchy. For example, any particular instance of "HouseCats" would inherit the property "name" from the class "Pets". The combination of ontologies and instances is called *knowledge base*.

To create a knowledge base corresponding to the Yellow Pages database, following major components are needed. 1) A service ontology covering the services provided by advertisements. 2) Ontologies for representing the service properties. 3) Annotation metadata describing the contents of the advertisements.

Given the large size of the database, it was desired to create ontologies and annotate advertisements for only a small fraction of the test data first. In our case study, a service, product, and location ontology was created. The service and product ontologies where based on the Universal Standard Products and Services Code[11] (UNSPSC) classification. The service ontology described the service classes that an advertisement could be connected with. For example, a restaurant advertisement would be connected with the class "Restaurants". The product ontology contained information about the products that an service provides. For example, a camera shop retails products that are of the types "Cameras" and "Photography films". The location ontology was based on the official list of Finnish municipalities[12].

RDF Schema [7] was chosen as the language for representing the ontologies and RDF [13] for representing instance data. This choice was motivated by the possibility to use widely available semantic web tools, such as the HP Lab's Jena toolkit[13] [14] for querying RDF(S) data, and Stanford University's Protégé 2000[14] [15] for creating the ontologies. The number of product and service classes in the ontologies was 432, and 34 different class properties were used. The location ontology contained 461 classes. For test purposes, we annotated 37 advertisements (service instances) in terms of the service, product, and location ontologies in the RDF database. Figure 1 illlustrates the connections between two instance advertisements and ontologies.

4.2 Search functionalities

The Keltsi system provides the end-user with two main functionalities: 1) ontology-based searching and 2) semantic instance browsing.

Ontology-based search The information search process of the user can be divided into the following steps:

1. Formulate the intention (e.g., "My camera is broken. How can I get it fixed?").

[11] http://www.un-spsc.net/, http://www.eccma.org/unspsc/
[12] http://www.intermin.fi/suom/laanit/kunnat.html
[13] http://www.hpl.hp.com/semweb/
[14] http://protege.stanford.edu/

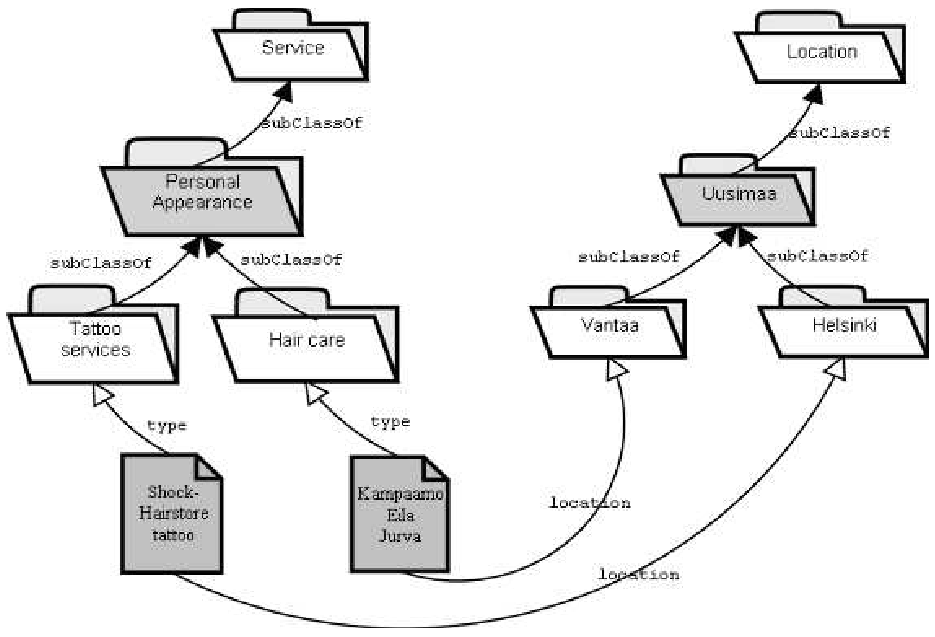


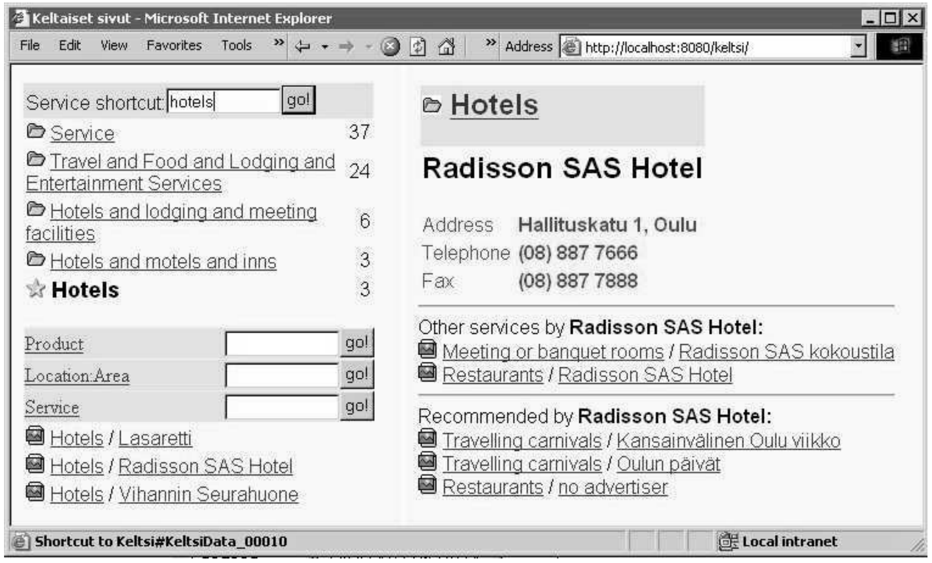
Fig. 1. Two advertisement instances connected to the service and location ontologies.

2. Choose a strategy for solving the problem (e.g., ”Look for a solution in the Keltsi system.”).
3. Present the intention using the models of the search interface (e.g., by using the user interface, create the following query: `product=Cameras`, `service=Repair Shops`, `location=Helsinki`)
4. Browse the results of the query (the advertisements), until the solution is found. If needed, the user can return to some of the previous steps. For example, one can reformulate the intention (”The repairing is too expensive compared to buying a new camera. Lets buy a new one.”), change the strategy (”No solution in Keltsi, let’s use Google.”), or redefine the query.

Ontology-based searching means that the user can query the advertisement collection by using the classes and properties of the the underlying ontologies.[15]. For example, when looking for a Chinese restaurant located in Helsinki, one could select the ”Restaurants” class and its properties `type-of-restaurant=Chinese` and `location=Helsinki`. The properties usually have a class value constraint, which means that the value must be selected from the corresponding ontology subtree, such as `location` and `type-of-restaurant`.

An interface supporting such functionality is shown in figure 2. The query is created by traveling the ontology three from general to more specific service

[15] This approach has been used, e.g., in [18].


Keltaiset sivut - Microsoft Internet Explorer

File Edit View Favorites Tools Address http://localhost:8080/keltsi/

Service shortcut: hotels go!

Service	37
Travel and Food and Lodging and Entertainment Services	24
Hotels and lodging and meeting facilities	6
Hotels and motels and inns	3
Hotels	3

Product go!
Location:Area go!
Service go!

Hotels / Lasaretti
Hotels / Radisson SAS Hotel
Hotels / Vihannin Seurahuone

Hotels

Radisson SAS Hotel

Address	**Hallituskatu 1, Oulu**
Telephone	**(08) 887 7666**
Fax	**(08) 887 7888**

Other services by **Radisson SAS Hotel:**
Meeting or banquet rooms / Radisson SAS kokoustila
Restaurants / Radisson SAS Hotel

Recommended by **Radisson SAS Hotel:**
Travelling carnivals / Kansainvälinen Oulu viikko
Travelling carnivals / Oulun päivät
Restaurants / no advertiser

Shortcut to Keltsi#KeltsiData_00010 Local intranet

Fig. 2. The user interface is divided into two parts: on the left hand side is the ontology-based searching tool and on the right is the semantic instance browser.

classes until the service class that best describes user's need is found. The class can be a very specific class, such as ”Restaurants”, or a more general class, such as ”Eating and drinking establishments”, which contains also bars, fast food services, etc. in addition to restaurants. The ontology class can alternatively be selected by writing the name of the class in the shortcut field, which makes the tree open up from the matching part. On the right of every ontology class, the number of corresponding service instances (advertisements) is shown. For example, the class ”Services” has 37 instances connected to it and the class ”Restaurants” has 10 instances.

The user can set property constraints that the searched service instances must fulfill. The properties and their possible values depend on the selected class. Therefore, if the selected class is ”Restaurants”, only properties related to ”Restaurants” and its superclasses are shown. The selection of the value for a property is either based on traversing through an ontology hierarchy, such as the ”location” of the service, by selecting the correct value from a list, or by writing the value in a text field.

The result of a search is a list of matching services where the type of the service and the name of the service is shown as a link. By clicking on the link, the information about the service and the advertisement is shown.

The semantic instance browsing The semantic instance browser shows the following information about the currently selected advertisement instance to the user: the literal properties of the service (such as the topic and contact information), the graphical advertisement (if available), the relations to the ontologies (such as to the type of service), and the recommended links.

The idea of the recommended links is to provide links to related services that are supposed to be of interest to the user. For example, if when looking at an advertisement of a hotel service, one could also be interested in activities near the hotel, such as recommended restaurants and entertainment services.

In this initial version of the Keltsi system the recommended links are provided explicitly by the advertisers. In future versions, the creation of recommendations will be automatic, as well, and based on the ontologies and the instance data. An application of this idea is presented in [12].

The technology The Keltsi system was build as a three tier web application. The top layer provides the user interface with the help of Java Server Page technologies [11], including JSP Tag libraries [19], and the Apache Tomcat servlet engine[16]. The middle layer handles the business logic of the service. The undermost layer provides the functionality for querying the RDF database. In the implementation HP Lab's Jena toolkit [14] was used. Keltsi is used by an ordinary web browser.

5 Discussion

This paper proposed that service ontologies provide a promising new way to organizing and annotating advertisements in yellow page directories on the web. The underlying ontology can be used as a navigational device by which the end-user can map her needs onto relevant offerings (advertisements) in the catalog. After finding a suitable service and provider, the ontological structure relating services and their instances, the advertisements, provide a means by which recommendations to related advertisements of interest can be generated. By annotating offerings semantically according to the ontologies, more accurate semantic-based search facilities can be provided to the end-user.

The work presented is related research and standardization work on service description and matching on the web, such as [16] where DAML-S [3] is proposed as an ontology for annotating service capabilities. However, the main focus there is on modeling service capabilities in terms of the input and output of a web service [8] for a software agent. In our work, the problem of matching human user needs with advertisements is considered.

A demonstrational web-based system based on RDF Schema ontologies and RDF instance advertisements was described. This system is, however, only a first small step towards the vision of semantic yellow pages presented in this paper. Several advances are needed before an actual semantic yellow page service could be deployed. Firstly, full blown service and other ontologies need to be designed and tested on real customer data. Secondly, the advertisements should

[16] http://www.apache.org/

be annotated accordingly. This is more demanding and time consuming than assigning advertisements to isolated categories or in relational tables. Annotation work is the price to be paid for better accuracy in information retrieval and for better servicing the end-user. We feel, that the advertising companies and the yellow page operators should be interested in providing such annotations since the matchmaking facility is the key point of the whole directory service business in the first place.

An important organizational question here is whether the task of annotating the services should be done by the yellow pages operator or by the advertiser. In our view, this could be done in a distributed fashion by the advertisers active on the web. The advertisements could be published on the customer's public web pages in RDF, and collected by a web crawled for a global repository. The advertisers should be entitled, on desire, to do the job because they have the best knowledge of their services. The advertiser also has the interest in maintaining the advertisements up-to-date, and possibly add new temporal advertisements, such as offers of the week, as time goes by. For this task, instance metadata editor is needed to guarantee that the metadata given really conforms to the ontologies in use.

Distributing metadata and annotation work in this way over the web changes not only the way in which yellow pages are maintained but has other consequences, too. For example, how to prevent the advertisers from corrupting the yellow pages service by generic service descriptions matching practically with any need? This problem was encountered, e.g., on the WWW when providing HTML with the META-tag facility for describing web page metadata. The information provided by the page publishers turned out to be quite unreliable and current search engines nowadays tend to neglect META-descriptions. Providing service and product metadata in this way also changes the ways in which advertisement space is sold to the companies. For example, how to set prices for different kind of service and product advertisements?

Acknowlegdements

Arno Aalto, Merja Jalava, Tuomo Kajava, and Petri Lindgren participated in the team that implemented the Keltsi system. Irmeli Rämö and Pekka Raitala of Finnish Yellow Paged Ltd provided us with the actual case database and helped our work with discussions concerning yellow pages services. Thanks to Vilho Raatikka for fruitful discussions.

References

1. Kal Ahmed, Danny Ayers, Mark Birbeck, Jay Cousins, David Dodds, Joshua Lubell, Miloslav Nic Daniel Rivers-Moore, Andrew Watt, Robert Worden, and Ann Wrightson. *Professional XML Meta Data.* Wrox Press Inc., 2001.
2. Blair Alexander and Karen V. Fernandez. Are publishers putting theory into practice? In *proceedings of Australian and New Zealand Marketing Academy Conference (ANZMAC)*, December 1-5 2001. ISBN 0-473-08206-3.

3. A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: web service description for the semantic web. In I. Horrocks and J. Hendler, editors, *The Semantic Web – ISWC 2002. First international semantic web conference*, pages 348–363. Springer–Verlag, Berlin, 2002.
4. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, New York, 1999.
5. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.
6. A. Bernstein and M. Klein. Towards high-precision service retrieval. In I. Horrocks and J. Hendler, editors, *The Semantic Web – ISWC 2002. First international semantic web conference*, pages 84–101. Springer–Verlag, Berlin, 2002.
7. D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
8. P. Cauldwell, R. Chawla, V. Chopra, G. Damschen, C. Dix, T. Hong, F. Norton, U.Ogbuji, G. Olander, A. Richman, K. Saunders, and Z. Zaev. *Professional XML web services*. WROX Press, Birminghan, UK, 2001.
9. D. Fensel (ed.). The semantic web and its languages. *IEEE Intelligence Systems*, Nov/Dec 2000.
10. D. Fensel. *Ontologies: Silver bullet for knowledge management and electronic commerce*. Springer-Verlag, 2001.
11. D. K. Fields, M. A. Kolb, and S. Bayern. *Java Server Pages*. Manning Publications Co., 2002.
12. E. Hyvönen, A. Styrman, and S. Saarela. Ontology-based image retrieval. Number 2002-03 in HIIT Publications, pages 43–45. Helsinki Institute for Information Technology (HIIT), Helsinki, Finland, 2002. http://www.hiit.fi.
13. O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.
14. B. McBride, A. Seaborne, and J. Carroll. Jena tutorial for release 1.4.0. Technical report, Hewlett-Packard Laboratories, Bristol, UK, April 2002. http://www.hpl.hp.com/semweb/doc/tutorial/index.html.
15. N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. Creating semantic web contents with Protege-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.
16. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web service capabilities. In I. Horrocks and J. Hendler, editors, *The Semantic Web – ISWC 2002. First international semantic web conference*, pages 333–347. Springer–Verlag, Berlin, 2002.
17. Steve Pepper. The TAO of Topic Maps. In *Proceedings of XML Europe 2000, Paris, France*, 2000. http://www.ontopia.net/topicmaps/materials/rdf.html.
18. A. T. Schreiber, B. Dubbeldam, J. Wielemaker, and B. J. Wielinga. Ontology-based photo annotation. *IEEE Intelligent Systems*, 16:66–74, May/June 2001.
19. G. Shachor, A. Chase, and Magnus Rydin. *JSP Tag Libraries*. Manning Publications Co., 2001.

Ontology-Based Image Retrieval

Eero Hyvönen[1,2], Avril Styrman[1], and Samppa Saarela[2,1]

[1] University of Helsinki, Department of Computer Science
firstname.lastname@cs.helsinki.fi,
http://www.cs.helsinki.fi/group/seco/
[2] Helsinki Institute for Information Technology (HIIT)

Abstract. The binary form of an image does not tell what the image is about. It is possible to retrieve images from a database using pattern matching techniques, but usually textual descriptions attached to the images are used. Semantic web ontology and metadata languages provide a new way to annotating and retrieving images. This paper considers the situation when a user is faced with an image repository whose content is complicated and semantically unknown to some extent. We show how ontologies can then be of help to the user in formulating the information need, the query, and the answers. As a proof of the concept, we have implemented a demonstrational photo exhibition using the promotion image database of the Helsinki University Museum based on semantic web technologies. In this system, images are annotated according to ontologies and the same conceptualization is offered to the user to facilitate focused image retrieval using the right terminology. When generating answers to the queries, the ontology combined with the image data also facilitates, e.g., recommendation of semantically related images to the user.

1 The problem of semantic image retrieval

Images are a major source of content on the WWW. The amount of image information is rapidly raising due to digital cameras and mobile telephones equipped with such devices. This papers concerns the problem when an end-user is faced with a repository of images whose content is complicated and partly unknown to the user. Such situations are recurrent, e.g., when using public image databases on the web.

We approach this general problem through a case study. The Helsinki University Museum[3] will open a permanent exhibition in the autumn 2003 at Helsinki city centre on the Senates Square. A central goal of the museum is to spread knowledge about university traditions to the large audience. One living tradition of the faculties of the University of Helsinki is the promotion ceremonies through which the students get their master's and doctoral degrees and the faculties grant honorary doctoral degrees to distinguished scientists and persons outside the university. The ceremonies consist of many occasions and last for

[3] http://www.helsinki.fi/museo/

several days. The museum database contains over 600 photographs about the ceremony events and documents, ranging from the 17th to 21th century, and more images are acquired after every promotion. The contents of this image repository would provide the audience with an interesting view into the life of the university .

There are two basic approaches to image retrieval: 1) content-based image retrieval (CBIR) and 2) metadata based image retrieval. In CBIR [4] the images are retrieved without using external metadata describing their content. At the lowest level, features such as color, texture, shape, and spatial location are used. At a higher conceptual level, images with an object of a given type or a given individual are searched (e.g., images with a person in the front or images of the Eiffel tower). At the highest level, retrieval of named events or activities or pictures with emotional or symbolic significance are retrieved (e.g., pictures about "tennis" or pictures depicting "atonement"). An example of the CBIR approach on the web is the PicSOM system [10]. In the metadata-based approach image retrieval is based on textual descriptions about pictures. In practice, this approach is usually employed in image retrieval due to the great challenges of the CBIR approach when dealing with conceptually higher levels of content[4].

A typical way to publish an image data repository online is to create a keyword-based query [1, 2] interface to an image database. Here the user may select filtering values or apply keywords to the different database fields, such as the "creator", "time", or to the content descriptions including classifications and free text documentation. More complex queries can be formulated, e.g., by using Boolean logic. Examples of museum systems on the web using this approach include the Kyoto National Museum search facility[5], Australian Museums Online[6], and Artefacts Canada[7].

Keyword-based search methods suffer from several general limitations [5, 8]: A keyword in a document does not necessarily mean that the document is relevant, and relevant documents may not contain the explicit word. Synonyms lower recall rate, homonyms lower precision rate, and semantic relations such as hyponymy, meronymy, antonymy [**?**] are not exploited.

Keyword-based search is useful especially to a user who knows what keywords are used to index the images and therefore can easily formulate queries. This approach is problematic, however, when the user does not have a clear goal in mind, does not know what there is in the database, and what kind of semantic concepts are involved in the domain. The university promotion ceremonies case discussed in the paper is an example of such a semantically complicated domain. Using the keyword-based approach would lead to the following problems:

[4] The term "content-based" in CBIR is unfortunate and confusing, since the textual metadata-based approach deals with explicit representations of content.

[5] http://www.kyohaku.go.jp

[6] http://amonline.net.au

[7] http://www.chin.gc.ca

Formulating the information need The user does not necessarily know what question to ask. One may only have general interest in the topic. How to help the user in focusing the interest within the database contents?

Formulating query The user cannot necessarily figure out what keywords to use in formulating the search corresponding to her information need. How to help the user in formulating queries?

Formulating the answer Generating image hit lists for keywords would probably miss a most interesting aspect of the repository: the images are related to each other in many interesting ways. In our case, the ceremonial occasions follow certain patterns in place and time and the people and surroundings depicted in the images reoccur in different events. These semantical structures should somehow be exposed from the data to the audience. The goal of an ordinary museum visitor is often something quite different from trying to find certain images. The user wants to learn about the past and experience it with the help the images.

We argue that semantic web technologies provide a promising new approach to these problems. In the following, semantic ontology-based annotation and retrieval of images is first discussed. After this the ontology used in our demonstrational system is presented, an annotation example is given, and the an ontology-based user interface to the image repository is illustrated. In conclusion, the contributions of this work are summarized.

2 Semantic image annotation and retrieval

The problem of creating metadata for images has been of vital importance to art and historical museums when cataloging collection items and storing them in a digital form. Following approaches are commonly used in annotating images:

Keywords Controlled vocabularies are used to describe the images in order to ease the retrieval. In Finland, for example, the Finnish web thesaurus YSA[8] is used for the task augmented with museum- and domain specific keyword lists.

Classifications There are large classification systems, such as the ICONCLASS[9] [20] and the Art and Architecture Thesaurus [14], that classify different aspects of life into hierarchical categories. An image is annotated by a set of categories that describe it. For example, if an image of a seal depicting a castle could be related to classes ”seals” and ”castles”. The classes form a hierarchy and are associated with corresponding keywords. The hierarchy enriches the annotations. For example, since castles are a subclass of ”buildings”, keyword ”building” is relevant when searching images with a castle.

Free text descriptions Free text descriptions of the objects in the images are used. The information retrieval system indexes the text for keyword-based search.

[8] http://www.vesa.lib.helsinki.fi
[9] http://www.iconclass.nl

Semantic web ontology techniques [5] and metadata languages [9] contribute to this tradition by providing means for defining class terminologies with well-defined semantics and a flexible data model for representing metadata descriptions. One possible step to take is to use RDF Schema [3] for defining hierarchical ontology classes and RDF [11] for annotating image metadata according to the ontology. The ontology together with the image metadata forms an RDF graph, a knowledge base, which can facilitate new semantic information retrieval services.

In our case study, we used this approach. Our idea is to first make ontological models of the concepts involved in the image repository. The ontologies form the core of the system and are used for three purposes:

Annotation terminology The ontological model provides the terminology and concepts by which metadata of the images is expressed.

View-based search The ontologies of the model, such as Events, Persons, and Places provide different views into the promotion concepts. They can hence be used by the user to focus the information need and to formulate the queries.

Semantic browsing After finding a focus of interest, an image, the semantic ontology model together with image instance data can be used in finding out relations between the selected image and other images in the repository. Such images are not necessarily included in the answer set of the query. For example, images where the same person occurs but in a different event of the same promotion may be of interest and be recommended to the user, even if such images do not match the query.

In the following, the construction of a comprehensive ontology for promotion concepts is discussed. It is then shown, how such an ontology facilitates semantic-based information retrieval of images as envisioned above.

3 The promotion ontology

The promotion ontology describes the promotional events of the University of Helsinki and it's predecessors, the Empirial Alexander's University and the Royal Academy of Turku. The top-level ontological categories are depicted in figure 1.[10] The classes of the ontology represent people of different roles (`Persons, roles, and groups`), events and happenings (`Happenings`) that take place in different locations (`Places`), physical objects (`Physical Objects`), speeches, dances, and other performances (`Performances, Performers, Creators, and Works`), and a list of all promotions from 17th century until 2001 (`Promotions`). The main goal of the ontologization process was to create an ontology suitable for the photograph exhibition and to offer the programmers the basis to implement the exhibition, either on the web or as an internal information kiosk application[11].

[10] In this paper we use English names for classes etc., but the actual implementation is in Finnish.

[11] The publication of the photographs representing, e.g., living people on the public WWW has legal consequences that have to be considered first.

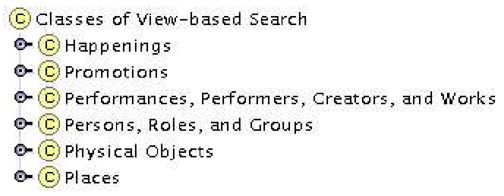

Classes of View-based Search
Happenings
Promotions
Performances, Performers, Creators, and Works
Persons, Roles, and Groups
Physical Objects
Places

Fig. 1. Top-level classes of Promotion-ontology.

The stable and unchanging things of the subject domain, i.e., continuants [19], are presented with classes in the ontology. The changing things, occurrents, are presented with instances. For example, in figure 2 the Cathedral of Helsinki has its own class. On the other hand, buildings that are not regularly used in promotions do not have a subclass of their own, but are instances of the general class `Buildings`.

The instances of the ontology have literal-valued properties, such as *name of the person*. These properties are typically used to provide a human-readable presentation of the instance to the user. Each instance, e.g., a particular person, is related to the set of promotions in which the instance occurs. In this way, for example, the persons performing in a particular promotion are easily found.

The top-level classes of the ontology are briefly discussed in the following.

Promotions Class `Promotions` has several properties to describe the features of every single promotion, such as the central person roles. All the instances of type `Promotions` hold at least 1) the date of the promotion ceremony, 2) the university under which the promotion was held, and 3) the faculty that arranged the promotion.

Persons, Roles, and Groups The same person may appear in many roles in several promotions. For example, a person, i.e., an instance of some (sub)class of `Persons`, can be a master-promovend in one promotion and a doctor-promovend in another one. To represent this, a person's unique person-instance is related to (possibly) many instances of the different `Roles`. The fundamental properties of instances of `Person` classes are the name, other name(s), and the promotion(s) in which the person participated to in different roles. The properties of the subclasses of class `Roles` include the person(s) in the role, and the promotion(s) in which the person participated in the role. The `Groups` classes represent collections of persons and roles in a group. Groups have a string-valued label and instance-valued property "person-roles of this group". For example, a group of master-promovends has a label "Master promovends of promotion x", and instances of the particular masters as the values of the property "person-roles of this group".

Places Class `Places` is divided into four subcategories (cf figure 2), `Squares and Parks`, `Buildings`,`Islands and Harbours`, and `Streets and Roads`. These classes define two literal-valued properties: name and address of the place. The name of the place describes the place and is used as the label of the instances.

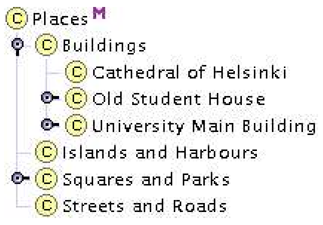


Fig. 2. Places and subclasses.

There is a regularly used set of `Squares and Parks` with direct relations to promotional ceremonies as well as a few `Buildings` that have become traditional sites for promotional happenings. These continuants have their own classes. `Islands and harbours` and `Streets and Roads` don't have further subclasses.

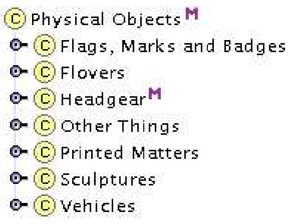


Fig. 3. Physical Objects and subclasses.

Physical Objects Class `Physical Objects` (cf. figure 3) has several subclasses, namely `Headgear`, `Sculptures`, `Vehicles`, `Flovers`, `Flags, Marks, and Badges`, `Printed Matters`, and `Other Things`. These classes have more subclasses that specify the physical objects involved in promotions in more detail. The properties of all physical objects are the name of the object, that is also used as the label of the instances, and the manufacturers of the object. In addition, the subclasses have their own additional properties, such as the literal-valued "language of a document" and the instance-valued "physical situatedness". `Printed Matters`, such as rune books may consists of several instances of `Pieces of Work`.

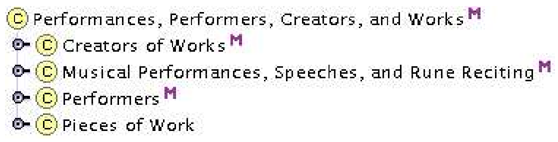

Performances, Performers, Creators, and Works
Creators of Works
Musical Performances, Speeches, and Rune Reciting
Performers
Pieces of Work

Fig. 4. Classes about audial performances.

Performances Class `Performances, Performers, Creators, and Works` (cf. figure 4) has subclasses `Musical Performances, Speeches, and Rune Reciting`, `Performers`, `Pieces of Work`, and `Creators of Works`.

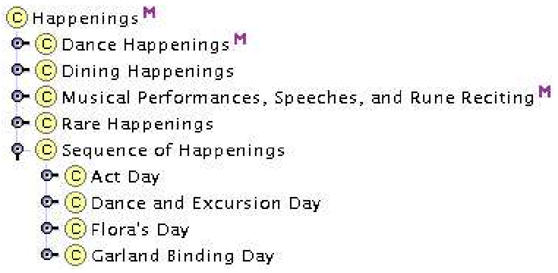


Fig. 5. Happenings and subclasses.

Happenings Class `Happenings` (cf. figure 5) ties all the other classes semantically together. Every happening has properties that describe

- the place of the happening (an instance of `Places`).
- the people who participated in the happening in different roles (instances of `Persons and Roles`),
- the performances of the happening (instances of `Performances, Performers, Creators, and Works`),
- the physical objects used in the happening (instances of `Physical Objects`),
- the `name of happening` (a literal value), and
- the `date of happening` (a literal value).

The class `Happenings` is divided into subclasses that collect happenings of similar nature together. The class `Sequence of Happenings` represents all the happenings in a hierarchy based on the chronological ordering of the happenings. The sequence of happenings is specified with a class-valued property `previous`. The sequence of two successive happenings, say, that `A` is followed by `B`, is specified by giving class `B` the property `previous` with the value *A*.

Ontology construction There are several partly conflicting goals to keep in mind when designing the ontology. The ontology not only should be semantically motivated, but also easy to construct and maintain to the ontologist. At the same time, the annotation work based on it should be simple to the annotator. Furthermore, the ontology and the annotated instance data should be in a form that is easy to use by the application programmer and efficient to run by the exhibition software.

In our work, two major difficulties were encountered during the annotation and implementation process:

1. Annotation process posed new demands to the ontology, which lead to changes in the ontology after many annotations were already done. How to manage such changes so that the annotator wouldn't have to redo the annotation work?
2. Application programmers pose new demands to the ontology and to the annotations in order to satisfy the demands of the end-user interface. As a result, changes in both the ontology and the annotations were needed.

4 Annotation of the images

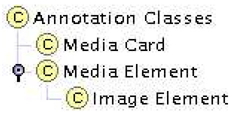


Fig. 6. Classes used in annotating the photographs.

We used the following annotation scheme: every image is associated with a set of instances of the promotion ontology. They occur in the image and hence characterize its content. This is basically how annotations are done when using, e.g., the ICONCLASS system [20]. The linkage between the image and its content is based on the classes of figure 6. Class `Image Element` defines information about the photographs: the name of the photographer, textual description about the subject of the photograph, a reference to the actual image file, and a reference to an instance of the class `Media Card` that has as one of its properties, the list of instances of the promotion ontology.

The ontology was created with the Protégé-2000 ontology editor[12][7], using RDF Schema as the output language. Protégé was also used for annotating the photograph.

For example, the image of 7 is annotated in the following way:

[12] http://protege.stanford.edu

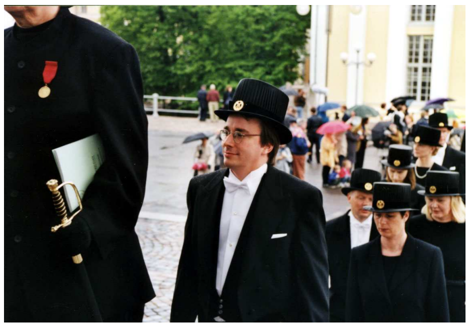
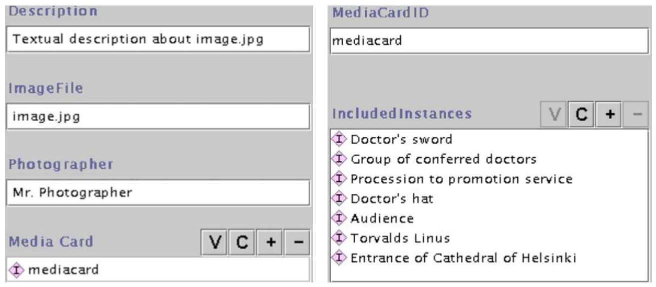


Fig. 7. Honorary doctor Linus Torvalds on a procession to divine service at the entrance of the Cathedral of Helsinki June 2, 2000.



Description	Textual description about image.jpg
ImageFile	image.jpg
Photographer	Mr. Photographer
Media Card	mediacard

MediaCardID	mediacard
IncludedInstances	Doctor's sword
	Group of conferred doctors
	Procession to promotion service
	Doctor's hat
	Audience
	Torvalds Linus
	Entrance of Cathedral of Helsinki

Fig. 8. Instances of annotation classes of figure 6. The left side depicts filled fields of an instance of `Image Element`, and the right side depicts an annotated instance of `Media Card`, that was used to annotate figure 7.

Step 1: The annotator takes the photograph and creates an empty instance of the class `Image Element`.

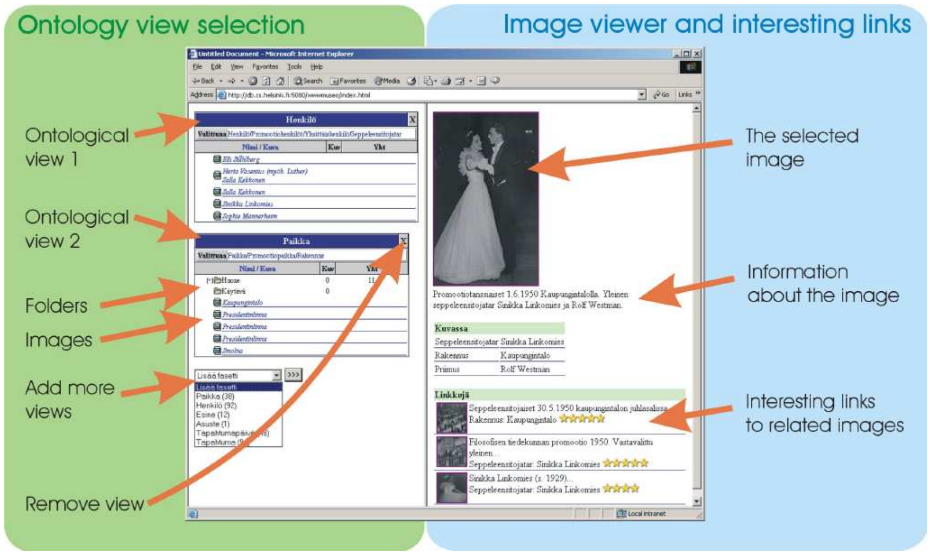
Step 2: The annotator fills in the empty fields of the `Image Element` instance in figure 8, and creates an instance of class `Media Card`.

Step 3: To include the needed metadata within the new `Media Card` instance, the annotator browses the ontology, starting from the top-level classes of figure 1. If a new instance is needed, e.g., if this is the first photo to be annotated where the `Person` Linus Torvalds is present, the annotator has to create one. Once the instance is created, the annotator can use it again to annotate other images about Linus Torvalds. The image of figure 7 is well-annotated with several instances as seen in figure 8. However, one could still add a few instances to it, such as the decoration in the upper left corner, and the diploma of the person carrying the decoration. The choice depends on how detailed semantics are needed and on the annotators choices.

The photos come from the image database of the Helsinki University Museum. This database was transformed into a repository of images and RDF-instance data and annotated further according to the ontology. At this phase the instance RDF metadata was checked and edited; the original information of the database was not sufficient alone and much of the data content was written in the free text description fields.

The ontologization annotation process lasted several months and included examining a wide range of historical materials. Several people took part of the process at different stages, including the domain experts and clients of the museum and programmers of the exhibition software. The full version of the ontology has 575 classes and 279 properties (slots). Currently, the knowledge base contains annotations of 619 photographs and has 3565 instances of ontology classes. The first demonstrational version of the exhibition discussed in this paper uses a much simpler ontology and annotation data of about 100 photos.

5 Semantic image retrieval



Promootiotanssiaiset 1.6.1950 Kaupungintalolla. Yleinen seppeleensitojatar Sinikka Linkomies ja Rolf Westman.

Kuvassa	
Seppeleensitojatar	Sinikka Linkomies
Rakennus	Kaupungintalo
Primus	Rolf Westman

Linkkejä

Seppeleensitojaiset 30.5.1950 kaupungintalon juhlasalissa
Rakennus: Kaupungintalo ☆☆☆☆☆

Filosofisen tiedekunnan promootio 1950. Vastavalittu yleinen...
Seppeleensitojatar: Sinikka Linkomies ☆☆☆☆☆

Sinikka Linkomies (s. 1929)...
Seppeleensitojatar: Sinikka Linkomies ☆☆☆☆

Fig. 9. User interface to the image server.

Based on the ontology, a web server was was implemented to support semantic image retrieval. Figure 9 illustrates its appearance on an ordinary web browser. The system provides the user with the following semantics-based facilities.

View-based filtering On the left, the user can open ontologies for filtering photographs of interest. In the figure, the ontologies for Persons (Henkilö) and Places (Paikka) have been opened. Additional views could be opened with the tool below. The ontologies are the same that were used when annotating the images. They tell the user the relevant concepts related to the promotions and underlying images. In this way the ontologies help the user in formulating and focusing the information need.

Queries can be formulated by opening ontological views and by selecting their classes. A query is the conjunction of the selections made in the open ontologies. In the figure, the selection was `Person=GarlandBinder` and `Place=Building`. The metaphor of opening directory folders with images is used here. This view-based idea to information filtering along different indexing dimensions ("facets") is an adaptation of the HiBrowse system developed for a bibliographical information retrieval system [15].

Image recommendations The answers to filtering queries are hit lists as customary in search engines, such as Google[13] and AltaVista[14] on the web. However, in contrast to such systems each hit is semantically linked with other images based on the ontological definitions and the annotations. In figure 9 the thumbnail photos beneath the dancing image are links to recommended images. They do not necessarily match the filtering query but that are likely to be of interest. They point, for example, to other images where the same garland binder occurs during the same promotion but not in a `Building`, or photos taken within the same `Place` but depicting only persons in other roles.

By clicking on a recommended thumbnail photo, the large image in view is switched and a new set of recommended images is dynamically generated beneath it. This idea is vaguely related to topic-based navigation used in Topic Maps [13] and the book recommendation facility in use at Amazon.com.

Our image server keeps a log of the session in order not to recommend same images over and over again. Furthermore, a persistent counter for visited images in maintained in a log file. In this way, more popular images can be ranked higher in the recommendations than less popular ones.

The system was implemented in Java with the help of Java Server Page technologies [6], JSP Tag libraries [18], the Apache Tomcat servlet engine[15], and HP Lab's Jena toolkit (version 1.4.0)[16] [12].

6 Discussion

This paper showed that ontologies can be used not only for annotation and precise information retrieval [16, 17], but also for helping the user in formulating

[13] http://www.google.com
[14] http://www.altavista.com
[15] http://www.apache.org/
[16] http://www.hpl.hp.com/semweb/

the information need and the corresponding query. This is important in applications such as the promotion exhibition, where the domain semantics are complicated and not necessarily known to the user. Furthermore, the ontology-enriched knowledge base of image metadata can can be applied to constructing more meaningful answers to queries than just hit-lists. For example, in our demonstration implementation, the underlying knowledge base provided the user with a semantic browsing facility between related recommended images.

The major difficulty in the ontology based approach is the extra work needed in creating the ontology and the detailed annotations. We believe, however, that in many applications – such as in our case problem – this price is justified due to the better accuracy obtained in information retrieval and to the new semantic browsing facilities offered to the end-user. The trade-off between annotation work and quality of information retrieval can be balanced by using less detailed ontologies and annotations, if needed.

Acknowledgements

Kati Heinämies and Jaana Tegelberg of the Helsinki University Museum provided us with the actual case database and helped in annotating the images. Tero Halonen and Tiina Metso provided expert knowhow about promotions and university traditions. Robert Holmberg, Kim Josefsson, Pasi Lehtimäki, Eetu Mäkelä, Matti Nykänen, Taneli Rantala ja Kim Viljanen participated in the team that implemented the demonstrational system. Our work was partly funded by the National Technology Agency Tekes, Nokia, TietoEnator, the Espoo City Museum, and the Foundation of the Helsinki University Museum, and was supported by the National Board of Antiquities.

References

1. M. Agosti and A. Smeaton, editors. *Information retrieval and hypertext*. Kluwer, New York, 1996.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, New York, 1999.
3. D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
4. J. P. Eakins. Automatic image content retrieval — are we getting anywhere? pages 123–135. De Montfort University, May 1996.
5. D. Fensel (ed.). The semantic web and its languages. *IEEE Intelligence Systems*, Nov/Dec 2000.
6. D. K. Fields, M. A. Kolb, and S. Bayern. *Java Server Pages*. Manning Publications Co., 2002.
7. W. Grosso, H. Eriksson, R. Ferguson, J. Gennari, S. Tu, and M. Musen. Knowledge modelling at the millenium (the design and evolution of protege-2000. In *Proceedings of 12th Workshop of on Knowledge Acquisition, Modeling and Management (KAW-1999), Banff, Alberta, Canada*, 1999.

8. E. Hyvönen, K. Viljanen, and A. Hätinen. Yellow pages on the semantic web. Number 2002-03 in HIIT Publications, pages 3–15. Helsinki Institute for Information Technology (HIIT), Helsinki, Finland, 2002. http://www.hiit.fi.
9. Eero Hyvönen, Petteri Harjula, and Kim Viljanen. Representing metadata about web resources. In E. Hyvönen, editor, *Semantic Web Kick-Off in Finland*, number 2002-01 in HIIT Publications. Helsinki Institute for Information Technology (HIIT), May 2002. http://www.cs.helsinki.fi/u/eahyvone/stes/semanticweb/.
10. M. Koskela, J. Laaksonen, S. Laakso, and E. Oja. The PicSOM retrieval system: description and evaluations. In *The challenge of image retrieval, Brighton, UK*, May 2000. http://www.cis.hut.fi/picsom/publications.html.
11. O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.
12. B. McBride, A. Seaborne, and J. Carroll. Jena tutorial for release 1.4.0. Technical report, Hewlett-Packard Laboratories, Bristol, UK, April 2002. http://www.hpl.hp.com/semweb/doc/tutorial/index.html.
13. Steve Pepper. The TAO of Topic Maps. In *Proceedings of XML Europe 2000, Paris, France*, 2000. http://www.ontopia.net/topicmaps/materials/rdf.html.
14. T. Peterson. Introduction to the Art and Architechure thesaurus, 1994. http://shiva.pub.getty.edu.
15. A. S. Pollitt. The key role of classification and indexing in view-based searching. Technical report, University of Huddersfield, UK, 1998. http://www.ifla.org/IV/ifla63/63polst.pdf.
16. A. T. Schreiber, B. Dubbeldam, J. Wielemaker, and B. J. Wielinga. Ontology-based photo annotation. *IEEE Intelligent Systems*, 16:66–74, May/June 2001.
17. G. Schreiber, I. Blok, D. Carlier, W. van Gent, J. Hokstam, and U. Roos. A mini-experiment in semantic annotation. In I. Horrocks and J. Hendler, editors, *The Semantic Web – ISWC 2002. First international semantic web conference*, number LNCS 2342, pages 404–408. Springer–Verlag, Berlin, 2002.
18. G. Shachor, A. Chase, and Magnus Rydin. *JSP Tag Libraries.* Manning Publications Co., 2001.
19. J. Sowa. *Knowledge Representation. Logical, Philosophical, and Computational Foundations.* Brooks/Cole, 2000.
20. J. van den Berg. Subject retrieval in pictorial information systems. In *Proceedings of the 18th international congress of historical sciences, Montreal, Canada*, pages 21–29, 1995. http://www.iconclass.nl/texts/history05.html.

Ontology-based Semantic Metadata Validation

Vilho Raatikka[2,1] and Eero Hyvönen[1,2]

[1] University of Helsinki, Department of Computer Science
`firstname.lastname@cs.helsinki.fi`,
`http://www.cs.helsinki.fi/group/seco/`
[2] Helsinki Institute for Information Technology (HIIT)

Abstract. Much of the Semantic Web content is generated from databases, especially the instance data based on the ontology classes used in applications. A recurring problem is that the instance data does not always semantically conform to the ontology used. It may be ambiguous, incomplete, or partly erroneous. Validating the data is necessary when it is transformed to a more semantic format. This may be a difficult and laborious task given the large and complex ontologies and databases in use. This paper discusses the problem of validating existing database instance data according to an ontology. We show how Semantic Web standards can augment the syntactic data validation schemes of relational databases and XML towards semantic validation. A metadata editor for semantic validation is being implemented. Its idea is to provide an XML-based view to a relational database and transform the data into RDF instances conforming to an RDF Schema, the ontology. The tool will be applied in tranforming museum collection data into a semantically interoperable form in the Finnish Museums on the Semantic Web project.

1 Introduction

The goal of our work is to combine data from heterogeneous museum databases into a single open WWW service called *Finnish Museums on the Semantic Web* (FMS) [7]. With this system, people interested in museum collections are offered a single access point to a large national level collection of exhibits. To access the service, only a common PC with a WWW browser and an Internet connection is be needed. The idea of FMS is to combine collection data at the semantic level by using semantic web technologies. In this way the user can be provided with new semantic-based facilities for information retrieval, such as ontology-based information retrieval and semantic browsing [9].

In order to reach this goal, the collection data stored in heterogenous databases must first be harmonized. Combining data of heterogeneous data sources is a challenge, which has been widely studied [1, 14]. The problem can be addressed at syntactical and semantical levels.

On the syntactic level, naming conventions for the tables and fields of the databases may be different. Also storing formats may be different in expressing equal concepts and data. For instance, the age of a person can be either of a `numeric`, an `integer`, or a `number` type. A more difficult question is how to combine different database schemas. Syntactic and semantic harmonization entails that one is be able to tell how to find the corresponding data values in different databases, e.g., the name or the material of a collection item. This can be tricky, if the databases store semantically different data, e.g., salaries with or without taxes. In the museum case, databases record more or less the same data of similar collection objects. This is partially because museums catalog collections use standards, such as Spectrum[3] and the CIDOC Guidelines[4]. Roughly speaking, only the schema for storing the data is different, although some museums catalog more metadata than others and syntactical differences may still be remarkable. There is a great variety of different *database management systems* (DBMS) and different database *schemas* in use, and the systems run on different platforms.

Harmonization on the semantic level includes schema intergation and terminological interoperability. For example, different synonyms for the same object may be used in different museums and by different catalogers, data about artists and organisations may be cataloged by different naming conventions. There are some classification standards and controlled vocabularies addressing the terminology problem, such as OCM[5], SHIC[6], and ICONCLASS[7]. However, in practise the terminology used in the data records is herogeneous.

In this paper a solution for harmonizing museum collection data on the syntactic and semantic level is developed. We show how semantic web ontology technigues, especially RDF [10] and RDF Schema [3] (RDFS), can be used in making the database semantics explicit and for representing the database contents is a harmonized uniform way. However, during the transformation of the existing databases into a harmonized RDF repository, both syntactic and especially semantic validitation is needed. Valid semantic metadata is a prequisite for the semantic information retrieval services provided by the envisioned FMS system.

The paper is organized as follows. First representation and extraction of semantics in databases is discussed. After this the steps needed in transforming heterogeneous databases into a semantically harmonized RDF repository are discussed. In conclusion, the implementation of a tool for helping the user in this task is discussed.

2 Semantics in Databases

The design of a database starts typically by modeling a *conceptual schema* [17, Ch.7.2]. The conceptual schema is based on the entities of the data and relations between them. The relations are typically directed and labeled.

[3] http://www.mda.org.uk/spectrum.htm
[4] http://www.willpowerinfo.myby.co.uk/cidoc/
[5] http://www.silverplatter.com/newFieldGuides/hraf/Outline_of_Cultural_Materia.htm
[6] http://www.holm.demon.co.uk/shic.htm
[7] http://www.iconclass.nl/

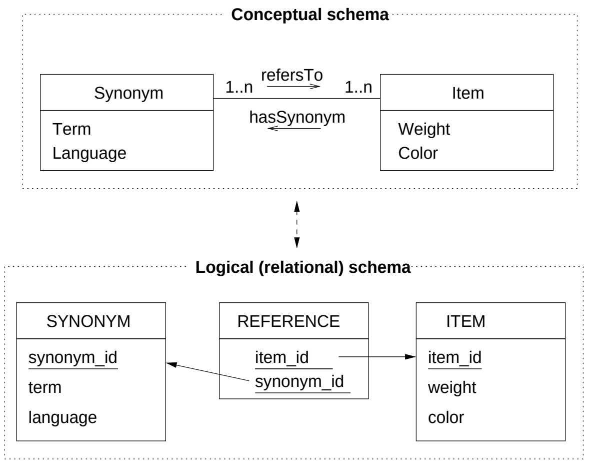


Fig. 1. The mapping between the conceptual schema and the logical schema. "Each synonym refers to at least one item and vice versa."

The modeling notation may be, for example, UML [17, Ch.7.3] and based on an *entity-relationship*-diagram (ER) [16, Ch.2]. The conceptual schema is mapped to a logical model, which usually is a *relational model* [16, Ch.3]. The entities are divided into tables as in Figure 1. Finally, the logical schema is transformed into a physical schema in which all necessary platform specific requirements are taken into account. The actual database is based on the physical model.

The semantics can be expressed in a relational schema in many ways. A table and its attributes (denoted as TABLE.attribute) are usually mapped to a class and its attributes. For example, the table ITEM can be mapped to the class with equal name and attributes ITEM.color and ITEM.weight. If an attribute refers to another table as in Figure 1, the meaning of the relation can easily be seen from the conceptual model. Referential integrity [4, Ch.6.2.4] in the relational schema is achieved by *foreign keys*. Say that there is a foreign key *r* in the table *T1* which refers to the attribute *s* in the table *T2*. The referential integrity means that for a given value of T1.r, T2.s having the equal value must exist. In Figure 1 the REFERENCE.item_id, for instance, refers to ITEM.item_id. Thus, any given value of REFERENCE.item_id must exist in the ITEM. Foreign keys in Figure 1 express that there cannot be an item without at least one synonym and vice versa. References of a database schema can also be explicitly named. The reference from REFERENCE to ITEM could be named *isSynonym*, for instance. Naming the references of a database further increases the expressiveness of a particular database schema.

A mechanism called *trigger* is used in ensuring cardinality constraints presented in the conceptual model. The trigger is launched as a consequence of a predefined action, for instance, if a row is included in the SYNONYM (see

Figure 1). The trigger performs an action, which checks that for the given SYNONYM.synonym_id, there is at least one REFERENCE.synonym_id with an equal value. That is, on the logical level the trigger may be used to force the given cardinality constraints defined on the conceptual level.

Usually the database semantics must be extracted from the logical schema because the up-to-date conceptual schema usually does not exist. Even if the conceptual schema exists, it seldom is synchronized with the logical schema after the logical schema is updated. The logical schema guarantees relatively efficient operations and good update properties. However, the semantic readability of the conceptual schema is lost in the logical model. In the *Karlsruhe Ontology and Semantic Web tool suite* [13] (KAON) the logical schema of the database is mapped straight to the ontology.

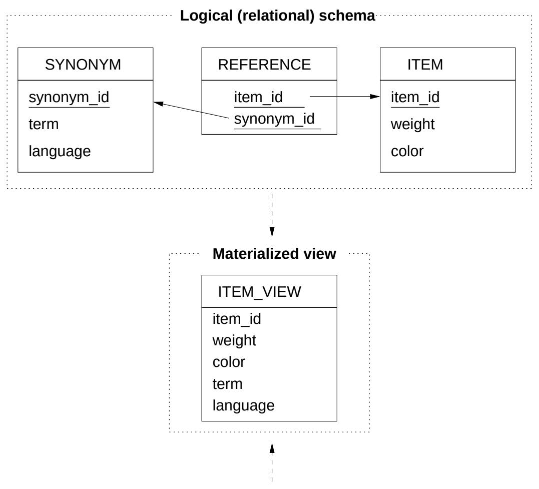
An alternative method is to re-create the conceptual schema from the logical schema. It might be useful for two reasons. Firstly, reading semantics is much easier from a conceptual schema than from a logical schema. Secondly, it is also easier to combine two conceptual models of different databases than two logical models.

It has been shown that any relational schema can be transformed into an ER-diagram if the relational schema is in an *entity-relationship normal form* (ERNF) [11, Ch.11.5]. By following certain rules in logical modeling, the database will always be in the ERNF. Transforming a relational schema to an ER-diagram does not result in a single ER-diagram but a restricted group of valid ER-diagrams. It is the user's task to choose the right one among the diagrams produced [11, Ch.11]. By using conceptual models of databases it may be possible to further automate the database mapping process.

All logical models cannot be mapped to a single concept schema. Instead, the mapping may result in a set of formally correct schemas. As an example, if the conceptual schema included a hierarchy of classes and was mapped to a relational schema, then the re-engineering of the conceptual schema would probably produce a set of models, all of which are not equal to the original schema. For this reason human interaction will be needed.

The semantics of a database application is typically embedded into both the business application and the database schema. The main goal of the database is to offer safe and efficient operations with the data. Parts of the system's semantics are stored into the database, such as entities, relations between entities and both cardinality and referential constraints. The semantics can be expressed in many ways in the database thus making the reliable extraction of the database semantics difficult.

The FMS system uses semantic web technologies in which the semantics are not embedded but explicitly represented by a set of ontologies. An *ontology* "describes a formal, shared conceptualization of a particular domain of interest" [6]. In FMS, ontologies are used for defining the semantics of stored museum data. An ontology can be described with an *RDF Schema* (RDFS) [3] specification. With RDFS one is able to define classes, class properties, basic property constraints, and inheritance between classes. For example, it is possible to state



```
<xsd:element name="item">
 <xsd:attribute name="itemId" type="xsd:positiveInteger"/>
 <xsd:complexType>
  <xsd:element name="weight">
   <xsd:complexType>
    <xsd:element="type" type="xsd:string"/>
    <xsd:element="quantity" type="xsd:positiveInteger"/>
   </xsd:complexType>
  </xsd:element>
  <xsd:element name="color" type="xsd:string"/>
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="synonym" maxOccurs="unbounded" minOccurs="1">
     <xsd:complexType>
      <xsd:element name="term" type="xsd:string" />
      <xsd:element name="language" type="xsd:string" />
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:complexType>
</xsd:element>
```

Fig. 2. Constructing a materialized view including the data required in the XML Schema.

that the cat crossing the street is an instance of the class Cat, which in turn is a subclass of Mammal. The *Resource Description Framework* (RDF) [10] can be used to represent real-world instance data in terms of the ontology class(es) and properties. The databases to be combined are transformed into an RDF database, a knowledge base, conforming the ontologies in use.

3 Steps of Data Validation

Museum collections typically reside in protected databases to which only privileged applications have the read access. Alternatively, the data can be republished outside the database in a form that anybody can read and understand. The latter approach was chosen in the FMS system. The collection data to be published in FMS will reside in the ordinary public HTML directories of the participating museums. No privileged access rights are needed. The FMS system collects the data every now and then, and creates a central repository for information retrieval. In this way the museums can select, edit, and separate the information to be published from the database. In this way, there is no fear of the original database being corrupted by unauthorized users. An additional benefit is that the FMS system can be developed independently from the rest of the museum information systems. The public repository can be used by other applications as well.

In *relational database management system* (RDBMS) data is divided into rows. Rows are stored into tables defined in the database schema. When data is read from the database and copied for the application, it must be presented in such a format that the application can use it. In an open system such as FMS, the data transfer format must be such that it is commonly known and accepted, and that anyone can translate to it. *XML*[8] [2] fulfills these requirements and is used in the FMS project as a data transfer format. XML is an open well-known standard family with plenty of public domain and proprietary software available.

In order to publish a museum database in FMS, the following syntactic and semantic validation steps must be taken. On the syntactic level, there may be errors in the data, inconsistent usage of formats, or data missing. On the semantic level, the terminology must be disambigiated and semantic validity constraints taken into account.

Syntactic Validation

1. **Database to XML transformation** The data to be published is retrieved from the data source and transformed to an XML format locally in the museum. For this purpose, every joining museum is given the *FMS XML Schema* defined by the FMS authorities.
2. **XML Schema[9] based validation** The XML data is syntactically validated against the FMS XML Schema. The schema consists of both obligatory and optional elements. Fulfilling the obligatory requirements is the museum's own responsibility.

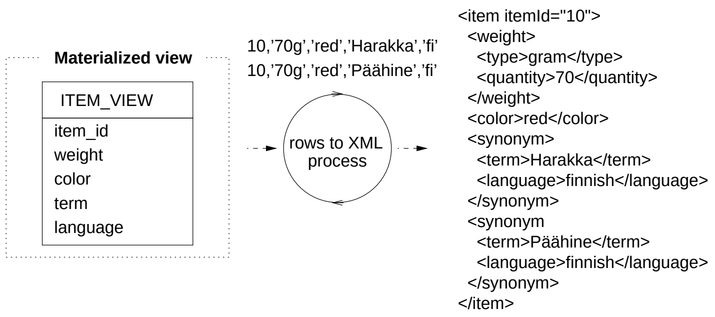
Semantic Validation

1. **XML to RDF transformation** After producing syntactically valid XML data, the records are transformed into RDF. This phase has two components: 1) The meaning of the XML elements is defined by mapping them to ontology concepts. 2) The terminology used in the XML element values is disambiguated by mapping them to RDF Schema ontology classes.
2. **RDF Schema based validation** The correctness of the RDF-statements is validated against the property constraints of the RDF Schema.

In the following these steps are discussed in more detail.

[8] http://www.w3c.org/XML/
[9] http://www.w3c.org/XML/Schema

3.1 Syntactic validation



```
10,'70g','red','Harakka','fi'
10,'70g','red','Päähine','fi'
```

```
<item itemId="10">
  <weight>
    <type>gram</type>
    <quantity>70</quantity>
  </weight>
  <color>red</color>
  <synonym>
    <term>Harakka</term>
    <language>finnish</language>
  </synonym>
  <synonym
    <term>Päähine</term>
    <language>finnish</language>
  </synonym>
</item>
```

Fig. 3. Transforming the data from the database view to the XML-document.

The data to be published in FMS must first be transformed to the XML format conforming to the FMS XML Schema. The data can be read from the database through a *view* (see Figure 2) [4, Ch.6.4], which helps in XML-transformation. A view is a virtual table, a query-able interface, which hides the physical and the logical details of the database from the user. The view constitutes of an SQL query, which may join multiple tables. The query is executed every time the view is queried. The remarkable difference to the database table is that updating a view is restricted. By rewriting the query, the database schema may be updated or the DBMS can be replaced with another without having to interfere with the application. The DBMS guarantees that the data of a view remains consistent if the original data is updated. In FMS the view should include at least all obligatory fields defined in the XML Schema.

If the query constituting the view is complicated or the view is frequently queried, the use of a *materialized views* is recommended. The materialized view is a physical structure rather than a virtual table. Using materialized views results in faster query execution than in logical views.

The data is queried through the view so that the rows are grouped by items. For each item there is a set of rows, which are combined to a single *XML card* (see Figure 3). After the translation, the document is validated against the XML Schema. If the card is valid, the process continues to the next phase, semantic validation.

3.2 Semantic Validation

The semantic validation phase has three components. Firstly, the meaning of XML *elements* is defined by mapping them to ontological structures. Secondly, the XML element *values* are mapped to the semantically corresponding ontology classes. This mapping is semi-automatic; human intervention may be needed to resolve semantic ambiguities. Thirdly, the RDF descriptions corresponging to an XML card is created based on the mappings and it is validated against the constraints expressed in the RDFS ontology.

Mapping XML Elements to Ontology Concepts The elements defined in the XML Schema must be mapped to the classes and properties of the RDFS ontology. Based on such a mapping, an XML card can be transformed into a set of RDF triplets. The mapping defines the meaning of the XML elements by a set of *mapping rules*. A mapping rule is a template of RDF triplets where *XPath*[10] expressions are used to identify the actual element values. As an example of XPath, assume the XML Schema below telling that the element `item` has a subelement `name` with a string value. The XPath referring to the element `name` is `/item/name`.

```
<xsd:element name="item">
  <xsd:element name="name" type="xsd:string"/>
  ...
</xsd:element>
```

When applying a template rule to an XML card, the XPath expressions are instantiated with matching element values. If some XPath in a rule cannot be matched, the rule cannot be applied to the card. If the rule matches, then the RDF template evaluates to a set of RDF triples where XPath expressions are substituted by the corresponding element values. For example, three template rules within the `fms` namespace are given below:

```
R1:   </item/id,        rdf:type,          /item/name>

R2:   </item/id,        fms:madeOf,        /item/material>

R3:   </item/id,        fms:hasLength,     /item/length/id>
      </item/length/id, rdf:type,          fms:Length>
      </item/length/id, fms:unitOfMeasure, /item/length/unit>
      </item/length/id, fms:lengthValue,   /item/length/value>
```

By applying them to the XML card

```
<item>
  <id>id74</id>
  <name>chip</name>
  <material>silicon</material>
  <lenght>
     <id>len7</id>
     <unit>mm</unit>
     <value>12</value>
  </length>
</item>
```

[10] http://www.w3.org/TR/xpath

the following result is obtained:

```
<id74, rdf:type,          'chip'>

<id74, fms:madeOf,        'silicon'>

<id74, fms:hasLength,     len7>
<len7, rdf:type,          fms:Length>
<len7, fms:unitOfMeasure, 'mm'>
<len7, fms:lengthValue,   '12'>
```

The rules R1, R2, and R3 describe the meaning of the elements `name`, `material`, and `length`, respectively.

Mapping Element Values to Classes The next step is *term mapping*. Here the literal subject and object values of the RDF-triples are mapped to the corresponding ontology classes. However, based on the special character of the `rdf:type` property, the literals in its domain position, id74 and len7 in the example, are unique identifiers of instances

In FMS, term mapping is based on synomyn sets, *synsets* that are attached to the ontology classes (concepts). They tell the possible classes that the literal names may refer to. For example, the class `SemiconductorChip` may have the synset {'chip'}. The mapping from names to concepts is not unique due to homonymy[11]. A syntactically valid literal may have several semantic interpretations. For example, the literal `chip` in our example may refer to a squirrel species, a semiconductor component, a kind of coin, and a piece of wood. As a result, the synset of a class `WoodChip` in addition to `SemiconductorChip` may contain the literal `chip` .

The result of applying term mapping to an RDF template is a set of RDF triples where literal data values (other than identifiers and numbers) are replaced by the URI references of the corresponding RDFS classes of the ontology. In our example, the result is two alternative sets of triplets:

```
<id74, rdf:type,            fms:SemiconductorChip>
<id74, fms:madeOf,          fms:Silicon>
<id74, fms:hasLength,       len7>
<len7, rdf:type,            fms:Length>
<len7, fms:unitOfMeasure,   fms:Mm>
<len7, fms:lengthValue,     12>

<id74, rdf:type,            fms:WoodChip>
<id74, fms:madeOf,          fms:Silicon>
<id74, fms:hasLength,       len7>
<len7, rdf:type,            fms:Length>
```

[11] A homonymous word has several meanings.

```
<len7, fms:unitOfMeasure,    fms:Mm>
<len7, fms:lengthValue,      12>
```

Validation against RDF Schema constraints The final step is to validate the corresponding RDF triple sets with the RDF Schema constraints and, if multiple interpretations still remain, select the right one by asking the human user of the validator.

There are two property constraints defined in RDFS: The *domain* constraint restricts the set of classes whose instances may have a particular property attached to them. The *range* constraint defines the set of classes whose instances can be values of a particular property. For example, the following description can be used to tell that a `SemiconductorChip` is `madeOf Silicon`:

```
<rdf:Property rdf:ID="madeOf">
  <rdfs:domain rdf:resource="#SemiconductorChip"/>
  <rdfs:range rdf:resource="#Silicon"/>
</rdf:Property>
```

In our example, using this constraint would mean that the latter RDF triplet interpretation in which the `WoodChip` is made of `Silicon` can be ruled out and the semantic meaning of the XML card is disambiguated. However, using this constraint means that it would not be possible to make another valid statement, where a `WoodChip` is `madeOf Wood`. This is of course not the intention. A disjuntion of range-domain-constraints allowing either wooden `WoodChips` or `SiliconChips` made of silicon is needed. However, such a construct is not supported by RDFS. In more advanced ontology languages, such as DAML+OIL[12], mechanisms for defining more sophisticated constraints like this are available. Another possibility is to design a special RDF(S) representation for disjuctive constraints and use it for validation.

In addition to disambiguating term mapping, semantic validation based on RDFS constraints can be used to detect semantic errors in the XML data. For example, an XML card claiming that the chip is `madeOf Water` can be detected semantically invalid. Also typos in names can be found because then the corresponding classes cannot be found (unless the typo results in a another valid class name).

RDFS constraints allow only simple validation of binary constraints. It is not possible to validate the data in a larger context by n-ary constraints, where $n < 2$. For example, assume that the length of the chip is given in liters in our example, which would be a semantic mistake. However, this cannot be detected by considering the binary constraints on hasLength and unitOfMeasure. A tertiary constraint on the length instance, its class type, and the unit of measure is needed. Liter is a valid unit but not for lengths.

The result of term mapping and constraint validation is a unique set of RDF triplets, the *RDF card*. It represents the original XML card on the semantic

[12] http://www.daml.org/2001/03/daml+oil-index

level. The union of such RDF cards, generated from the XML cards of the heterogeneous databases, constitues a knowledge base. This knowledge base is the harmonized semantic representation of the underlying heterogeneous databases.

4 Meedio: A Metadata Editor for Semantic Validation

To evaluate the ideas presented in this paper, an ontology about museum textiles is being designed in the FMS project. The ontology describes the common concepts about the textile domain so that museums can map their data sources to it. This ontology will be used as the first test case for combining collection data in the FMS system. To start with, XML data from the Espoo City Museum[13] and the National Museum of Finland[14] museum will be used. For this purpose, an initial FMS XML Schema has been specified and a database to XML transformator for the database of the National Museum of Finland has been implemented. The next step is generate RDF data conforming to the ontology.

For this work, a semantic metadata validator is needed. A first version of such a system, called *Meedio* has been implemented in a student project in the summer 2002. The goal of this project was to implement a tool by which a museum cataloger without background knowledge of XML or RDF could edit and transform XML cards into semantically valid RDF form. Meedio allows the user to manually edit the metadata, complete missing data, and add new information. The user can browse the ontologies and attach RDF statements to items or remove statements. This allows a convenient way to find and pick correct instance values for a particular property. When the user saves the edited results, the semantics of the RDF descriptions is validated against the range and domain constraints of the ontology. If conflicts occur, the user is notified about the problem. The user is responsible for any more refined semantic validation of the data. For example, to notice that the length of a `chip` cannot be 12m. The resulting valid RDF is stored to a location publicly available for the FMS system.

Since the Meedio is an instance editor, ontologies cannot be updated. This guarantees that users cannot corrupt the ontology by mistake. The management of ontologies is centralized. If one realizes that the ontology is incomplete or erroneous, one must request an update from the administrative staff of the FMS.

At the moment, Meedio is still in its infancy. For example, the term mapping function described in this paper is not finished yet. The first version of the system was implemented in Java with the help of Java Server Page technologies [5], JSP Tag libraries [15], the Apache Tomcat servlet engine[15], and HP Lab's Jena toolkit[16] [12]. Meedio is used by an ordinary web browser.

[13] http://www.espoo.fi/museo
[14] http://www.nda.fi
[15] http://www.apache.org/
[16] http://www.hpl.hp.com/semweb/

Acknowledgements

Thanks to Samppa Saarela and Kim Viljanen for fruitful comments concerning the paper. Mikko Apiola, Ari Inkovaara, Miikka Junnila, Justus Karekallas, and Pekko Parikka participated in the team that implemented the demonstrational system. Our work was partly funded by the National Technology Agency Tekes, Nokia, TietoEnator, the Espoo City Museum, and the Foundation of the Helsinki University Museum, and was supported by the National Board of Antiquities.

References

1. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *Computing Surveys*, 18(4):323–364, 1986.
2. N. Bradley. *The XML Companion.* Addison-Wesley, 2002.
3. D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
4. T. Connolly and C. Begg. *Database Systems - A Practical Approach to Design, Implementation, and Management, Third Edition.* Addison-Wesley, New York, 2002.
5. D. K. Fields, M. A. Kolb, and S. Bayern. *Java Server Pages.* Manning Publications Co., 2002.
6. T. R. Gruber. A translation approach to portable ontology spesifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
7. E. Hyvönen, S. Kettula, V. Raatikka, S. Saarela, and Kim Viljanen. Semantic interoperability on the web. Case Finnish Museums Online. In Hyvönen and Klemettinen [8], pages 16–29. http://www.hiit.fi.
8. E. Hyvönen and M. Klemettinen, editors. *Towards the semantic web and web services. Proceedings of the XML Finland 2002 conference. Helsinki, Finland*, number 2002-03 in HIIT Publications. Helsinki Institute for Information Technology (HIIT), Helsinki, Finland, 2002. http://www.hiit.fi.
9. E. Hyvönen, A. Styrman, and S. Saarela. Ontology-based image retrieval. In Hyvönen and Klemettinen [8], pages 43–45. http://www.hiit.fi.
10. O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.
11. H. Mannila and J.-P. Räihä. *Design of Relational Databases.* Addison-Wesley, New York, 1992.
12. B. McBride, A. Seaborne, and J. Carroll. Jena tutorial for release 1.4.0. Technical report, Hewlett-Packard Laboratories, Bristol, UK, April 2002. http://www.hpl.hp.com/semweb/doc/tutorial/index.html.
13. B. Motik, A. Maedche, and R. Volz. A conceptual modeling approach for semantics-driven enterprise applications. Technical report, University of Karlshure, 2002.
14. E. Rahm and P. A. Bernstein. On matching schemas automatically. Technical Report MSR-TR-2001-17, Microsoft Research, Microsoft Corporation, Redmond, WA, USA, 2001.
15. G. Shachor, A. Chase, and Magnus Rydin. *JSP Tag Libraries.* Manning Publications Co., 2001.

16. A. Silbershcatz, H. F. Korth, and S. Sudarshan. *Database System Concepts, third edition*. McGraw-Hill, New York.
17. J. Sowa. *Knowledge Representation. Logical, Philosophical, and Computational Foundations*. Brooks/Cole, 2000.

Semantic Interoperability on the Web: Case Finnish Museums Online

Eero Hyvönen[1,2], Suvi Kettula[2,3], Vilho Raatikka[2,1],
Samppa Saarela[2,1], and Kim Viljanen[2,1]

[1] University of Helsinki, Department of Computer Science
`firstname.lastname@cs.helsinki.fi`,
`http://www.cs.helsinki.fi/group/seco/`
[2] Helsinki Institute for Information Technology (HIIT)
[3] Espoo City Museum

Abstract. This paper discusses the problem of making the contents of heterogeneous databases semantically interoperable on the web. The problem is addressed through a real-world case study by considering the new possibilities and challenges that the WWW gives to museums when publishing their collections online. We argue that interoperability at syntactic, semantic, and pragmatic levels is needed in order to combine the collections logically and to provide the museum visitor with useful search and navigation services into the semantically rich cultural contents. A technical solution to the problem is proposed based on semantic web technologies and a demonstrational implementation that combines two Finnish museum databases is discussed.

1 Museums on the web: a new challenge

The computer screen cannot replace the experience of visiting a museum itself, but can offer several benefits when studying the collections:

Access to larger collections Large collections that cannot be put on display physically can be shown to the public using the computer.

Usability of information Information technology enables the retrieving, combination, and display of information in ways that are technically or economically impossible in traditional exhibition spaces.

Interactivity and audiovisuals Information technology provides interactivity and can be used to extend and enrich the collection material with images, sound, and video material.

The World Wide Web (WWW) [2] can offer still another complement to the traditional museum visit. In order to study the collections, the visitor does not have to visit the museum during certain opening hours at a specific location.

There are several technical challenges in developing the WWW channel for transmitting collection contents to the public. Cataloging, annotating, and maintaining the data contents for WWW publication already poses a challenge to any single museum. This problem is enhanced further when using the WWW for combining collection data from the databases of several museums. To the user such a global system would, however, be very helpful for many reasons:

- The distribution of collection objects in museums at different locations creates an obstacle to information retrieval, for both the public and for researchers.
- One does not have to bother how the different museums store their collection data and what particular vocabularies (e.g., MASA[4] [13] or museum specific word lists) and classification systems (e.g., Outline of Cultural Materials[5] [18]) are used.
- The collections of different museums can be studied through a single interface and access point, which simplifies usage.

From the information retrieval viewpoint, it should be possible to use the collections as if they were in a single database.

The problem of making heterogenous databases homogeneous is an old one. A difficulty there is how to make the different database schemas mutually interoperable. In schema integration [1, 17], the idea is to construct a global homogeneous view of independently developed database schemas. The global repository is created before querying it. Another possibility is to keep the schemas as they are and integrate the database systems by a central application that uses a general information retrieval protocol. The data integration is done in a lazy fashion during the query evaluation time. For example, the Z39.50[6] protocol is used by museums and libraries for searching and retrieving information from remote databases.

Using these approaches in providing the user with a homogeneous view to collection data means that the databases are integrated either in a tight fashion physically or through the application using the network. In this paper, a more loosely coupled approach will be proposed instead. The collection contents are published in the local public directories of the museums in the same way as ordinary web pages. A web crawler is used to fetch and index the contents and a search facility is provided to the users as a server-side application. For example, search engines such as Google[7] and AltaVista[8] are based on this idea. A major benefit of such a loosely coupled system is that that the content providers can publish and maintain their contents independently from the service provider. The idea of publishing one's data on the web independently in this fashion is one of the key points underlying the whole WWW.[9]

[4] http://www.nba.fi/DEVELOP/asiasana.htm
[5] http://www.yale.edu/hraf/ocm_list.html
[6] http://lcweb.loc.gov/agency/
[7] http://www.google.com
[8] http://www.altavista.com
[9] A recent development in this direction on the web is Web Services [5], the idea of publishing functional services on the web for the machines to use.

In addition to making the database schemas interoperable, semantic interoperability with respect to content is needed. For example, the different vocabularies used in the collection data must be made mutually coherent. This issue is not addressed by the traditional schema-based approaches above in which explicit semantic descriptions of the content do not exist.

In this paper we address the problem of making existing distributed collection databases mutually interoperable on the semantic level. We argue that semantic web technologies offer a promising approach to facilitating homogeneous, semantic information retrieval based on heterogenous databases on the web.

In what follows, the problem of logical combination of heterogeneous collections at syntactic, semantic, and pragmatic levels is first discussed. Application of the web languages XML [3], RDF [12], and RDF Schema [4] in data and knowledge representation is discussed. After this, a system architecture for addressing the semantic interoperability problem is proposed, and the case implementation of our prototype, "Finnish Museums on the Semantic Web" (FMS), is discussed. To begin with, this system will integrate collection data from the heterogeneous collection databases of the Espoo City Museum[10] and the National Museum of Finland[11].

2 Logical combination of collections

In order to combine the collection data of different museums into one logical virtual entity, the data must be combined according to both syntax and semantics.

2.1 Syntactic interoperability

The museum databases are distributed (exist in physically different places) and heterogeneous, i.e., they 1) use different database systems by different manufacturers, and 2) their logical structure (schema, tables, fields, names, etc.) may vary. On the level of structure, combining collection data means that the collection record data fields meaning the same thing but under different labels in different databases, such as "name of object" and "object name", are identified as the same, common labels are given to the fields, and a common way of presenting collection data is agreed upon. What is needed is a shared, mutually agreed presentation language for collection data.

In web applications, XML (eXtensible Markup Language) [3] has quickly been adopted as a formal way to agree on data representation languages. XML is a meta-language, a language to define languages. With the help of XML, different communities can easily and accurately define their own domain specific presentation languages. XML is penetrating the museum information systems,

[10] http://www.espoo.fi/museo
[11] http://www.nba.fi

as well. For example, the SPECTRUM cataloging system of the mda[12] has an XML specification that is being tested during 2002 by the Consortium for the Interchange of Museum Information CIMI[13] (CIMI).

Using XML gives us the following advantages:

Open standards XML data is simple text. The format is not restricted to the products of certain manufacturers, although major manufactures and communities are committed to using and supporting it. There are and will be several tools available for XML data including free public domain software. The use of such open standards makes it easier to change from using one commercial museum system to another and to combine different systems with each other.

Transporting collection data Collection data is to be preserved and collected for a long time. This poses a conflict to the hectic world of information technology, where enterprises and products come and go at a fast rate. Thus, museums have to change and transfer their databases to new systems time and time again. With open standards, this can usually be accomplished both flexibly and economically.

Combining the collections Open standards enable the combination of data from different collections. Depending on the situation, several collections can be physically joined into one database, or the use of several collections can be integrated over the web with the help of common data representation languages and data transmission practices.

Multi-channel publishing With XML it is easy to produce different manifestations of the collection data, e.g. both for the WWW and in print (XSLT transformations).

With the help of XML, museums can agree on a mutual standard for recording collection data, regardless of the collection system in use. When the museums have agreed on the common language, the transmission, combination, and WWW publishing of the collections becomes significantly easier.

In the FMS system, the combination of museum data is based on a common XML language at the structural level. This language is used to express the collection data to be published on the WWW. As a first step towards such an XML language, an FMS XML Schema specification has been designed [16].

2.2 Semantic interoperability

XML is a modern solution for combining data on the level of syntax. For a deeper combination and integration of data, the terminology used in the collection data has to be united in meaning, i.e., with respect to semantics. For example, the same kind of piece of furniture might be classified as a "footstool" in one museum, while it is called a "bench" in another. Thus, a search for "bench" would leave out the items classified as "footstools".

[12] http://www.mda.org.uk, previously known as the Museum Documentation Association.

[13] http://www.cimi.org/wg/xml_spectrum

Mapping terminologies with ontological concepts One of the main semantic relations that is used in controlled vocabularies and thesauri [9], such as the WordNet[14] [6] is the relation between super- and subordinate meanings (hyponymy), e.g. the fact that "footstools" are a kind of "furniture". If this semantic relationship is known, a search for "furniture" should find "tables" as well as "chairs," "cabinets" and other types of furniture. Meronymy, for its part, is the semantic relationship between parts and a whole, e.g., in the way that a "table top" is part of a "table," or "Helsinki" is part of "Finland." In addition, the collection data contains many straightforward semantic relationships. For example, the "creator" relationship may connect a collection object to a certain "manufacturer" or "artisan", and the relationship for the "place of production" to a certain "location". Such relationships can be defined formally by ontology languages that are being developed by the World Wide Web Consortium W3C[15] and others.

In order to combine collection data semantically, an exact definition, the ontology, of physical object classes and other concepts, as well as the relationships between them, is needed. For example, the International Council of Museums (ICOM)[16] develops a general ontology called CIDOC object-oriented Conceptual Reference Model (CRM)[17] for the documentation of cultural heritage. In addition to such an ontology, one needs general thematic ontologies of place, time, style etc. and application domain specific ontologies focusing on particular areas of collections, such as textiles or furniture.

When defining ontologies in WWW applications, a natural "standard" choice is the RDF recommendation [12] and RDF Schema specification [4] of the W3C. These languages are syntactically based on XML[18] but their semantic data model is a labeled directed graph, a semantic network of relationships between web resources and literal data items. RDF is a way to present metadata as named properties of the data. With RDF Schema we can agree on the terminology and constraints to be used in the RDF descriptions. The first version of the FMS system is based on RDF(S) semantics when presenting ontologies and metadata.

In practice, the semantic combination of data in the FMS means that the linguistic expressions that stand as data field values on the structural level, i.e., as the values of the XML elements, are mapped onto the classes of the general ontology. There are two basic problems involved: 1) Mapping synonymous terms on the same concepts. 2) Disambiguating polysemous terms. Both tasks are handled in the FMS by a metadata editor tool called Meedio [16] before the data is published. Meedio gets as input collection records conforming to the FMS XML Schema, and transforms them into RDF format (to be read in by the web crawler of the FMS system). During the interactive Meedio transformation, the values in the XML data elements are mapped on the corresponding semantic concepts

[14] http://www.cogsci.princeton.edu/~wn/
[15] http://www.w3.org
[16] http://www.icom.org/
[17] http://cidoc.ics.forth.gr/
[18] Other syntaxes could be specified as well.

defined by an RDFS ontology. The transformation is based on associating ontological classes with linguistic synonymous words and expressions.

In synonym mapping, terms are simply mapped on concepts. For example, the class `Benches`[19] may have the property `synonyms` and the set {`footstool, bench`} as its value. Assume then that the FMS-system finds a collection data record of a "bench" of the form

```
<collectionObject>
   <objectName>bench</objectName>
   ...
</collectionObject>
```

from the web site of a museum and a "footstool" record

```
<collectionObject>
   <objectName>footstool</objectName>
   ...
</collectionObject>
```

from the web site of another museum. With the help of the ontology, these two objects can be recognized as instances of the same ontology class `Benches` resulting into two RDF instances of the same `type`:

```
<rdf:Description rdf:about='URI of the collection object'>
   <rdf:type rdf:resource='http://www.fms.fi/furniture#Benches'/>
   ...
</rdf:Description >
```

In this way the different terminologies used in different museums by different catalogers can be integrated (assuming that the terms have the same meaning in different organizations).

In a polysemous situation the same term refers to several different concepts. For example, the word "chip" refers to a squirrel species, a semiconductor component, a kind of coin, and a piece of wood. Meedio handles semantic disambiguation by asking the user to make the choice whenever it finds a disambiguous term-class mapping according to the ontology.

Enriching the semantic content When a data value in an XML element, e.g. term "footstool" above, is attached with a class in the ontology, the collection record is merged into the semantic RDF graph defined by the ontology and other collection instance data. An important side effect of this process is semantic enrichment where new meaning is automatically added to collection items in two ways. Firstly, the generic ontological relations defined once by the ontologist are automatically inherited from the ontological definitions to instance

[19] We use English names and symbols in our examples, but the actual names in the FMS system are in Finnish.

data. For example, if the class "Benches" is given a category according to the OCM classification [18] in the ontology, then all footstools and benches will have this classification as well. As a result, the museum database or the museum cataloger does not have to provide this piece of information for each item in the class. Secondly, semantic associations emerge automatically with other related collection item instances. For example, a particular bench from a museum may have the same manufacturer as a footstool in another museum, which may be an important piece of information to the user.

The enriched semantic RDF graph forms the underlying database on which services for the end-user can be created, an approach similar to some earlier semantic web systems, such as [7, 8]. In the WWW there are two basic ways of retrieving information: by browsing pages through links and by generating hit lists by search engines. In the FMS system, a semantic WWW browser for such data is being developed as well as an ontological search engine for the Finnish language.

2.3 Pragmatics of collection objects

During cataloging, information on the collection objects such as object type and description, history of usage, provenience etc. is added into the database. With the help of controlled vocabulary keywords and classifications, contextual metadata is added to the item in order to enhance information retrieval facilities later on. On the structural level, data fields like object type, donator etc. can be identified and connected to the item. On the semantic level, names that stand as the values of the data fields can be connected with the semantic concepts defined by the ontologies. In linguistics, the next level after semantics is pragmatics describing the usage and purposes of language meanings. Museum collection data have their pragmatic dimension, too.

In our view, pragmatics of collection objects tell how the exhibits are used and in what roles they participate in different processes, events, and skills of life. For example, let us consider the process of "fishing". This process is associated with different instruments, agents, places, and time. A "fish net" and a "boat", for example, are connected with "fishing" as instruments. The "fishing" event or skill thus offers the possibility to relate different objects, agents, locations, and time with each other. From the point of view of the museum visitor, this kind of pragmatic level is especially interesting, because such context data "brings the objects to life".

Pragmatic processes are intangible abstract events. Until now they have not been considered as collection objects of their own to be stored in databases. A fishing event can be recorded on video, or a documentary can be written on the production of linen but they are stored as pieces of documentation, not as abstract processes. With semantic web technologies it is possible to design ontologies that describe the processes, and data can be represented and stored according to those ontologies. By storing processes in addition to objects in museum databases, it would become possible to create a new approach to retrieving and perceiving exhibit data. The processes where a physical collection object is used in different roles could be found automatically through the process ontology. This gives the user the possibility to find semantic links to other concepts and items related by the processes. For example, from "fishing net" we can move forward to "fishing processes" and from there to "fishing boats", "fishermen", and other fishing topics. In the user interface, these associations can be visualized, for example, as normal hypertext links when browsing the collection on the WWW in the same way as is Topic Maps [14].

3 Finnish museums on the semantic web

In order to realize the goal presented in the previous section, a research and development project[20] is being carried out at the Helsinki University Computer Science Department and the Helsinki Institute for Information Technology (HIIT). As a case study, the collection databases of the Espoo City Museum and the National Museum of Finland are used. (Other museums may join the system later.) These databases use different DBMSs (SQL Server and Ingres), have different relational schemas, and are located physically in different cities. The systems are representatives of two major camps of Finnish museum information systems: the Antikvaria group (the Espoo City Museum) and Musketti (the National Museum). For the sake of simplicity, the implementation of the system is first restricted to only one part of exhibit collections, the textiles. The technical solution is, however, applicable to any object domain and the exhibition can be complemented later with other classes of museum objects.

3.1 Viewing collections as a virtual WWW space

The traditional approach in publishing collections on the web is to use keyword-based search for the collection database. The user types in keywords and the system generates hit lists matching with the given words. This approach is used, e.g., in the Australian Museums Online system[21]. In our view, such functionality is useful especially to an expert user looking for some specific objects. The goal the ordinary museum visitor is, however, usually something quite different from trying to find certain objects. One would rather want to learn about the past and experience it with the help of the collections. In physical exhibitions the cognitive museum experience is often based on the interesting semantic and thematic combination of exhibits and their contextual information. The same principle applies to WWW exhibitions, as well.

To take a step towards this ambitious goal, the FMS system transforms collection databases into a virtual semantic web space (see figure 1). Its pages are linked with each other with semantic links that are useful for finding information based on its content. The idea is to offer to the user a semantic browsing and searching facility in the combined collection knowledge base [11]. This facility

[20] http://www.cs.helsinki.fi/group/seco/

[21] http://www.amonline.net.au/

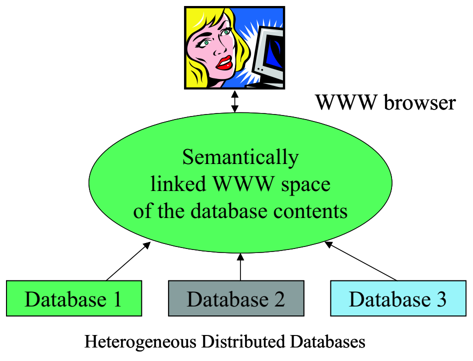

WWW browser
Semantically linked WWW space of the database contents
Database 1
Database 2
Database 3
Heterogeneous Distributed Databases

Fig. 1. The idea of the Finnish Museums on the Semantic Web system from a user's point of view.

is implemented by a piece of server-side software, called Ontogator. When the user views the exhibition entry page (URL) with a web browser, Ontogator dynamically generates WWW pages with links to other pages of interest. The FMS home page is the single entry point through which the user enters the virtual museum collections' WWW space.

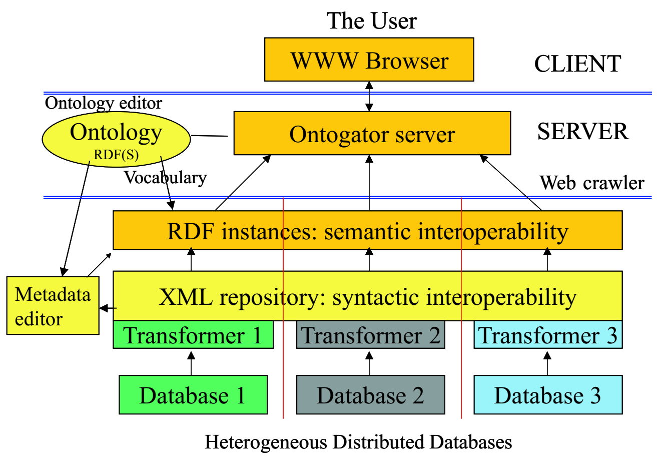


Fig. 2. The architecture of the Finnish Museums on the Semantic Web system.

The general architecture of the FMS system is presented in figure 2. Museums join the system by producing exhibit cards according to the FMS XML Schema, and by checking the semantic validity of the results with the metadata editor

Meedio [16]. The metadata in RDF form produced by Meedio is placed in a public directory on the museum's WWW server. In this way, the museum can easily and completely control of the information it wants to and can publish. The museum does not need to allow the FMS system access their internal database system through the web. Data security, firewall, and related problems do not arise.

The collection data of different museums is read in by the web crawler of the FMS system and combined into a global RDF database, a semantic graph that consists of the ontology and the collection data. Based on this graph, Ontogator dynamically generates the collection WWW space (figure 1) for the user's web browser.

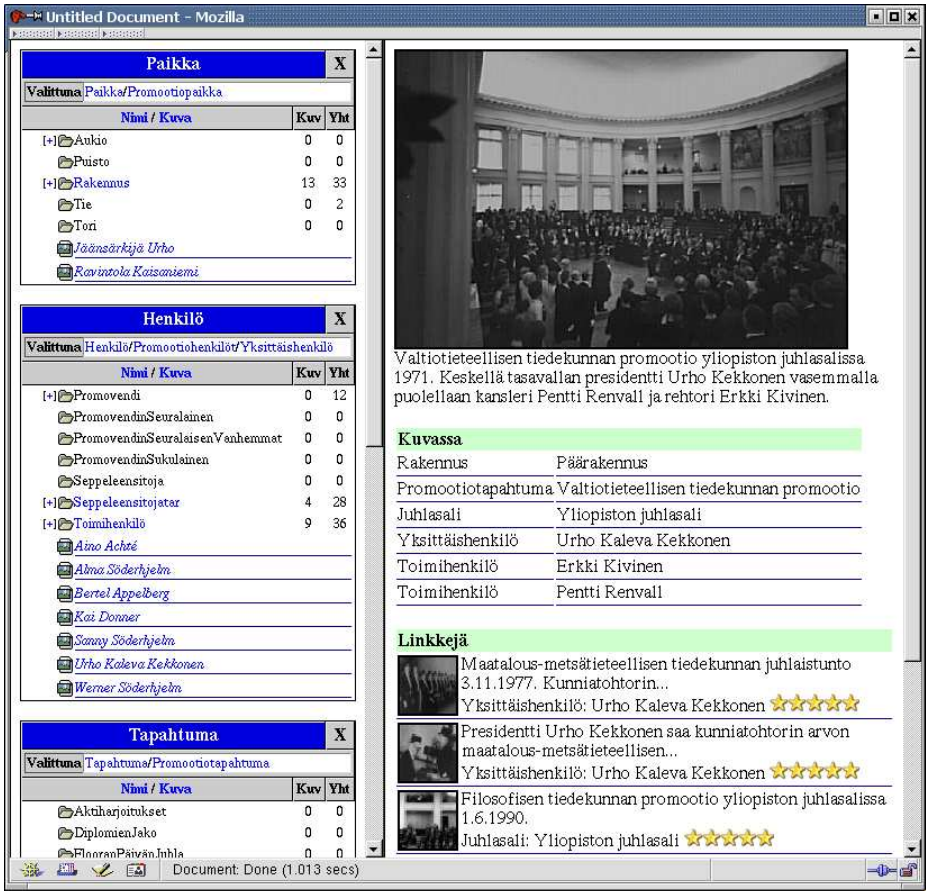
Ontogator will provide the user with the following semantics-based facilities.

View-based filtering Ontogator shows the multiple ontologies used in annotating collection data, such as ObjectType, Material, etc. By selecting ontological classes from these hierarchies, the user can express the search profile easily in the right terminology. For example, by selecting `ObjectType=carpet` and `Material=silk`, silk carpets are found. This view-based idea to information filtering is adapted from the HiBrowse system developed for a bibliographical information retrieval system [15]. Using the system is based on the metaphor of opening directory folders – the idea used in Windows Explorer and in many other systems.

Topic-based navigation Ontogator supports topic-based navigation according to the underlying idea of Topic Maps [14]. The creation of semantic links between topics of interest is based on 1) the collection domain ontologies (classes and their relations) and 2) on actual collection data (instance data). The links give the user contextual and pragmatic information about the objects in the collection.

Ontological search engine for Finnish A search engine is being developed for generating hit lists in the same fashion as search engines on the WWW. However, our engine will understand and make use of the semantic relationships between keywords. The conjugation of Finnish words is also taken into account of.

The actual implementation of the system is underway. To illustrate the use of the FMS system, figure 3 shows the user interface of our first experimental implementation of Ontogator. This system [11] was originally created for an image database of the Helsinki University Museum, but illustrates our current idea to be used in FMS as well. The system implements view-based filtering and facilitates topic-based navigation in a restricted sense by recommending semantically related images. On the left in the window, the user may choose different ontological views of the images related to the promotional ceremonies of the University of Helsinki. Here the ontologies of "People", "Places", and "Events" are open. The ontologies show the user the concepts that will be needed when searching for images. This is necessary, because the end-user might not be familiar with the promotion-related concepts. By opening the ontology hierarchies, the user



Paikka X

Valittuna Paikka/Promootiopaikka

Nimi / Kuva	Kuv	Yht
[+] Aukio	0	0
Puisto	0	0
[+] Rakennus	13	33
Tie	0	2
Tori	0	0
Jäänsärkijä Urho		
Ravintola Kaisaniemi		

Henkilö X

Valittuna Henkilö/Promootiohenkilöt/Yksittäishenkilö

Nimi / Kuva	Kuv	Yht
[+] Promovendi	0	12
PromovendinSeuralainen	0	0
PromovendinSeuralaisenVanhemmat	0	0
PromovendinSukulainen	0	0
Seppeleensitoja	0	0
[+] Seppeleensitojatar	4	28
[+] Toimihenkilö	9	36
Aino Achté		
Alma Söderhjelm		
Bertel Appelberg		
Kai Donner		
Sanny Söderhjelm		
Urho Kaleva Kekkonen		
Werner Söderhjelm		

Tapahtuma X

Valittuna Tapahtuma/Promootiotapahtuma

Nimi / Kuva	Kuv	Yht
Aktiharjoitukset	0	0
DiplomienJako	0	0
FlooranPäivänJuhla	0	0

Valtiotieteellisen tiedekunnan promootio yliopiston juhlasalissa 1971. Keskellä tasavallan presidentti Urho Kekkonen vasemmalla puolellaan kansleri Pentti Renvall ja rehtori Erkki Kivinen.

Kuvassa

Rakennus	Päärakennus
Promootiotapahtuma	Valtiotieteellisen tiedekunnan promootio
Juhlasali	Yliopiston juhlasali
Yksittäishenkilö	Urho Kaleva Kekkonen
Toimihenkilö	Erkki Kivinen
Toimihenkilö	Pentti Renvall

Linkkejä

Maatalous-metsätieteellisen tiedekunnan juhlaistunto 3.11.1977. Kunniatohtorin...
Yksittäishenkilö: Urho Kaleva Kekkonen ★★★★★

Presidentti Urho Kekkonen saa kunniatohtorin arvon maatalous-metsätieteellisen...
Yksittäishenkilö: Urho Kaleva Kekkonen ★★★★★

Filosofisen tiedekunnan promootio yliopiston juhlasalissa 1.6.1990.
Juhlasali: Yliopiston juhlasali ★★★★★

Document: Done (1.013 secs)

Fig. 3. A semantic browser for the image database of the Helsinki University Museum.

chooses the ones that are of interest, and the browser shows all the images that fit all the chosen concepts. On the right, an image found by this method is shown. Underneath it, the system displays a number of related recommended images that are connected semantically with the main image in some way. The system finds such images automatically based on the underlying RDF database. For example, in the recommended images, the same persons may appear as in the main image, but in another context. By browsing the collections according to the associations, the user may make a journey into the world of promotional events and images.

4 Conclusions

This paper discussed the problem of making heterogeneous databases semantically interoperable on the web. It was argued that in addition to syntactic interoperability semantic interoperability is needed for combining the data logically. Ontology techniques of semantic web research were proposed to solving the interoperability problem on the semantic level. The Finnish Museums on the Semantic Web system was used as a case application.

Using ontologies has several benefits.

Terminological coherence The different terminological conventions of different museums and individual catalogers can be made mutually interoperable.

Enriching collection data Ontological class definitions enrich collection data semantically by, for example, property inheritance.

Pragmatic contexts Object ontologies can be complemented with pragmatic level immaterial contextual patterns, such as descriptions of processes and skills. They can provide the user with further insight on how the objects were used, manufactured etc.

The resulting enriched database can be used as a basis for implementing WWW exhibitions richer in content. Two ways of doing this were proposed. Firstly, ontological classes can be exposed to the user in order to facilitate view-based information filtering. Secondly, the ontological relations between collection data elements can be used as an associative structure that facilitates semantic browsing between related concepts.

Our practical goal is to test and demonstrate feasibility of these ideas in practice by implementing the FMS systems as a prototype. The architecture of the system was presented and first implementational experiments of some parts of the whole system were discussed.

Acknowledgements

Our work is funded mainly by the National Technology Agency Tekes, Nokia, TietoEnator, the Espoo City Museum, and the Foundation of the Helsinki University Museum, and was supported by the National Board of Antiquities. Fruitful co-operation and discussions with Sten Björkman, Kati Heinämies, Anneli Rantanen, Petteri Saarinen, and Sirkka Valanto is acknowledged.

References

1. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *Computing Surveys*, 18(4):323–364, 1986.
2. T. Berners-Lee, M. Fischetti, and M. Dertouzos. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web.* Harper Business, 2000.
3. N. Bradley. *The XML Companion.* Addison-Wesley, 2002.
4. D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
5. P. Cauldwell, R. Chawla, V. Chopra, G. Damschen, C. Dix, T. Hong, F. Norton, U.Ogbuji, G. Olander, A. Richman, K. Saunders, and Z. Zaev. *Professional XML web services.* WROX Press, Birminghan, UK, 2001.
6. C. Fellbaum, editor. *WordNet. An electronic lexical database.* The MIT Press, Cambridge, Massachusetts, 2001.
7. D. Fensel, J. Angele, S. Decker, M. Erdman, H. Schnurr, S. Staab, R. Studer, and A. Witt. On2broker: semantic-based access to information sources at the WWW. In *World bonference on the WWW and Internet (WebNet 99)*, 1999.
8. D. Fensel, F. van Harmelen, M. Klein, H. Akkermans, J. Broekstra, C. Fluit, J. van der Meer, H. Schnurr, R. Studer, J. Hughes, U. Krohn, J. Davies, R. Engels, B. Bremdal, F. Ygge, T. Lau, B. Novotny, U. Reimer, and I. Horrocks. On-to-knowledge: ontology-based tools for knowledge management. In *eBusiness and eWork, Madrid, Spain*, 2000.
9. D. J. Foskett. Thesaurus. In *Encyclopaedia of Library and Information Science, Volume 30*, pages 416–462. Marcel Dekker, New York, 1980.
10. E. Hyvönen and M. Klemettinen, editors. *Towards the semantic web and web services. Proceedings of the XML Finland 2002 conference. Helsinki, Finland*, number 2002-03 in HIIT Publications. Helsinki Institute for Information Technology (HIIT), Helsinki, Finland, 2002. http://www.hiit.fi.
11. E. Hyvönen, A. Styrman, and S. Saarela. Ontology-based image retrieval. In Hyvönen and Klemettinen [10], pages 43–45. http://www.hiit.fi.
12. O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.
13. R. L. Leskinen, editor. *Museoalan asiasanasto.* Museovirasto, Helsinki, Finland, 1997.
14. Steve Pepper. The TAO of Topic Maps. In *Proceedings of XML Europe 2000, Paris, France*, 2000. http://www.ontopia.net/topicmaps/materials/rdf.html.
15. A. S. Pollitt. The key role of classification and indexing in view-based searching. Technical report, University of Huddersfield, UK, 1998. http://www.ifla.org/IV/ifla63/63polst.pdf.
16. V. Raatikka and E. Hyvönen. Ontology-based semantic metadata validation. In Hyvönen and Klemettinen [10], pages 30–42. http://www.hiit.fi.
17. E. Rahm and P. A. Bernstein. On matching schemas automatically. Technical Report MSR-TR-2001-17, Microsoft Research, Microsoft Corporation, Redmond, WA, USA, 2001.
18. P. Sihvo, editor. *Kulttuuriaineiston luokitus. Outline of cultural materials.* Museovirasto, Helsinki, Finland, 1996.

Building and Maintaining Web Taxonomies

Wray Buntine, Sami Perttu and Henry Tirri

Helsinki Inst. of Information Technology
HIIT, P.O. Box 9800
FIN-02015 HUT, Finland
{wray.buntine,sami.perttu,henry.tirri}@hiit.fi

Abstract. A recognized problem for internet commerce is the task of building a product taxonomy from web pages, without access to corporate databases, and then populating a database with link information about service, spare parts, reviews, product specifications, product family, etc. A key precursor for this task is the ability to build classification hierarchies in an unsupervised manner of web pages potentially useful. However, a nasty aspect of the real world is that most web pages have multiple facets. A web page might contain information about cameras and computers, as well as having both specification and sale data. We are interested in methods for unsupervised learning of multiple facet models which automatically allow these multiple facets to be extracted or predicted from examples. We will present some examples using the Reuters 2000 XML Corpus containing 800,000 Reuters newswires from the year beginning 20th August 1996. Our system, called Ydin, inputs XML/XHTML pages and displays results using a custom web interface. It extracts different facets and provides a navigation tool for investigating the results.

1 Introduction

A recognized problem for internet commerce is the task of building *a consumer guide*, building a product taxonomy from web pages, without access to corporate databases, and then populating a database with link information about service, repair, spare parts, reviews, product specifications, product family and company home pages, and purchase information from retailers. While within individual corporations this task is arguably well-handled by using the emerging standards and methodology of the semantic web, for web services such as consumer guides, where collections of disparate information sources would be used, top-down standards enforcement is not a realistic option, and nor will be labor intensive knowledge engineering in general. Many believe more automatic methods are required to build the ontologies and other knowledge structures needed [1], such as semantic-based search and the automated construction of taxonomies from web data.

While we do not believe full automation of the task (building consumer guides) is generally feasible, significant support for the task is provided by the ability to build classification hierarchies, that is, taxonomies, in a supervised or unsupervised manner. However, a nasty aspect of the real world is that most webpages have multiple facets. A web page might contain information about cameras and computers, as well as having both specification and sale data. Whereas another page might mix a product index with partial specification and sales data. We are interested in methods for supervised and unsupervised learning of multi-faceted models, where facets cover both the content and the style (e.g., index, sales, specs, etc.).

Clustering or unsupervised learning is a standard method for analysing discrete data such as documents, and is now being used in industry to create taxonomies from web pages. A rich variety of methods exist borrowing theory and algorithms from a broad spectrum of computer science: spectral (eigenvector) methods, kd-trees [3], using existing high-performance graph partitioning algorithms from CAD [4], hierarchical algorithms [5] and data merging algorithms [6], etc. These methods are used both to structure a corporation's document database, and to organise results from a web search.

These methods have one significant drawback for typical application in areas such as document analysis: each document is to be classified exclusively to one class. Their models make no allowance for instance, for a product page to have 40% digital camera content and 30% laptop computer content, and 10% of strictly sales jargon. Rather these methods require pages to be 100% one way or another, and any uncertainty is only about whether to place the 100% into one or the other class. In practice documents invariable mix a few topics, readily seen by inspection of the human-classified Reuters newswire, so the automated construction of topic hierarchies needs to reflect this. One alternative is to make clustering *multi-faceted* whereby a document can be assigned proportionally (i.e., using a convex combination) across a number of clusters rather than uniquely to one cluster.

A different task, supervised multi-faceted learning, covers the same kind of problem: a document can belong to several classes simultaneously, though perhaps with varying degrees. However, in this case, the taxonomy (or classification hierarchy) is pre-supplied. Given a set of pre-assigned classes for each document, one is to build a predictive classifier that will assign a set of classes to new documents. This supervised multi-faceted task has been addressed quite well in the literature, oftentimes by modifying an existing classification algorithm, for instance the multi-class multi-label perceptron of Crammer and Singer [7]. In our internet commerce example, the way to use these techniques is to have an expert build a taxonomy, assign say 20 pages to each node, train a classification system on these examples, and then classify the remaining pages under consideration. However, because this requires a pre-existing taxonomy, it does not address the general discovery task we proposed.

Several authors have recently developed methods that address the task of multi-faceted clustering. These are discrete analogues to principle components analysis (PCA) intended to handle discrete or positive only count data of the kind used in the bag-of-words representation of web pages. By their discrete relatively efficient nature, they offer significant computational advantages over the more general non-linear methods of Karhunen and others [8]. These new methods include probabilistic latent semantic analysis [9] latent Dirichlet allocation [10], multinomial PCA [11]. A good discussion of the motivation for these techniques can be found in [9].

In this paper we present an application of multinomial PCA to the new Reuters Corpus[1], containing over $806,791$ news items from 1996 and 1997. This gave us the opportunity to evaluate the effectiveness and scaling potential of the algorithm for the task of automatically constructing taxonomies from large collections of web pages.

2 Contrasting Clustering and Multi-Faceted Clustering

For concreteness, consider the problem in terms of the usual "bag of words" representation for a document [12], popular for analysing web pages. Here the items making up the sample are documents and the features are the counts of words in the document, and counts of any stylistic features considered relevant (though ignored here). A document is represented as a sparse vector of words and their occurrence counts. All positional information is lost. With J different words/features, the dimensionality for words/features, each document becomes a vector $\boldsymbol{x} \in \mathcal{Z}^J$, where the total $\sum_j x_j$ might be known. Traditional clustering becomes the problem of forming a mapping into a single integer (i.e., a class assignment or discretization, $\mathcal{Z}^J \mapsto \{1, \ldots, K\}$, where K is the number of clusters). Whereas techniques such as PCA form a mapping into a real-valued vector of lower dimension (i.e., $\mathcal{Z}^J \mapsto \mathcal{R}^K$ where K is considerably less than J).

The problem we consider, however, is to represent the document as a convex combination, thus to form a mapping into a lower dimension probability vector (i.e., $\mathcal{Z}^I \mapsto \mathcal{C}^K$ where $\mathcal{C}^K$ denotes the subspace of $\mathcal{R}^K$ where every entry is non-negative and the entries sum to 1, $\boldsymbol{m} \in \mathcal{C}^K$ implies $0 \leq m_k \leq 1$ and $\sum_k m_k = 1$).

For instance, suppose we are performing a coarse clustering of newswires into topics: the topics found might be "sports", "business", "travel", "international", "politics", "domestic", and "cultural". Consider a document about a major sports-star and the overlap of his honeymoon with a big game. Then traditional clustering might output the following: "the document is about sports". A more refined clustering system that represents uncertainties as well might output: "with 90% probability the entire document is about sports, with 7% probability it is about cultural, and 3% probability about something else". General multinomial PCA considered in this paper might output: "50% of the document is about sports, 35% of the document is about cultural, 7% about business, 5% about international". The supposed business content is really a discussion of the hotel for the honeymoon and the supposed international content comes from the location of the honeymoon. Note here general multinomial PCA plays the role of dimensionality reduction, and places similar kinds of words into the same bucket for compression purposes rather than any real topic identification.

[1] Volume 1: English Language, 1996-08-20 to 1997-08-19.

The problem we consider is also to perform multi-faceted clustering, which serves the purpose of extracting multiple mutually occurring topics from a document. Suppose $\boldsymbol{m}$ is the probability vector ($\boldsymbol{m} \in \mathcal{C}^K$) corresponding to a particular document. For multi-faceted clustering, $\boldsymbol{m}$ should have most entries zero, and only a few entries significantly depart from zero. A measure we shall use for this is entropy, $H(\boldsymbol{m}) = \sum_j m_j \log(1/m_j)$. Thus multi-faceted clustering prefers low entropy vectors from $\mathcal{C}^K$, i.e., those where the entropy is much less than $\log K$. In the limit, when the average entropy of the probability vectors is 0, the mapping becomes equivalent to standard clustering. With the simple honeymoon example above, the output could be reduced to: "70% of the document is about sports, 30% of the document is about cultural". This makes the document have $2^{H(\boldsymbol{m})} = 1.85$ effective topics, as opposed to the original PCA example above with more proportions (0.5,0.35,0.07,0.05) which had $2^{H(\boldsymbol{m})} = 3.17$ topics.

3 Overview of the Multinomial PCA Method

We give a brief review of the multinomial PCA method and algorithm here as a basis for the experimental work presented later.

3.1 The Basic Model

To begin, consider Tipping *et al.*'s representation of standard PCA [13]. The hidden variable $\boldsymbol{m}$ is sampled from K-dimensional Gaussian noise. Each entry represents the strength of the corresponding component and can be positive or negative. This is folded with the $J \times K$ matrix of component means $\boldsymbol{\Omega}$ and then used as a mean for J-dimensional Gaussian noise.

$$\begin{aligned} \boldsymbol{m} &\sim Gaussian(0, \mathbf{I_K}) \\ \boldsymbol{x} &\sim Gaussian(\boldsymbol{\Omega m} + \boldsymbol{\mu}, \mathbf{I_J}\sigma) \end{aligned}$$

This relies on the data being real-valued and somewhat Gaussian, which fails badly for some image and document data. Techniques such as ICA address this problem using a more general non-linear framework [8].

A discrete analogue to the above formulation is first to draw a probability vector $\boldsymbol{m}$ that represents the proportional weighting of components, and then to mix it with a matrix $\boldsymbol{\Omega}$ whose columns represent a word probability vector for a component. This yields a distribution over the word counts $\boldsymbol{x}$:

$$\begin{aligned} \boldsymbol{m} &\sim Dirichlet(\boldsymbol{\alpha}) \\ \boldsymbol{x} &\sim Multinomial(\boldsymbol{\Omega m}, L) \end{aligned}$$

Here L is the total number of words in the document, and $\boldsymbol{\alpha}$ is a vector of K-dimensional parameters to the Dirichlet. Thus, the mean of each entry x_j is a convex combination of a row of $\boldsymbol{\Omega}$.

3.2 The Basic Algorithm

Basic algorithms for this problem as casted use the hidden variable framework of the Expectation-Maximization (EM) algorithm and its variants widely used in clustering (e.g., [14]). This yields an algorithm that estimates the model parameters $\boldsymbol{\Omega}$, the distribution of words per component, from the document mixing proportions $\boldsymbol{m}$, then in turn estimates the mixing proportions $\boldsymbol{m}$ from the model parameters $\boldsymbol{\Omega}$. This iterative approach is termed a re-estimation algorithm. The version we use applies a so-called variational extension of the EM algorithm [11], and was first applied to this problem by Blei, Ng, and Jordan [10]. However, they ran their algorithm on the old Reuters Corpus maintained by David Lewis that contains about 20,000 documents, using a network of computers in parallel, and only built 20 component models. It is difficult to assess the properties of their results from the simple experiments they reported.

3.3 Scaling Up the Algorithm

If 1000 component models are to be built, that would mean $806,791,000$ floats are needed to store the estimate of $\boldsymbol{m}$, denoted here as $\widehat{\boldsymbol{m}}$, which would require 3.2Gb if stored naively.

Thus, in order to run the system on the full Reuters data set, we needed to make some changes to our software. First, we store each entry of $\widehat{\boldsymbol{m}}$ in 16-bits as a factor of 2^{-16}, which reduces the storage requirement to 1.6Gb. Note that part way through the run, this becomes sparse, but initially at least the full $1.6Gb$ is used. Second, we process the $\widehat{\boldsymbol{m}}$ data using sparse vector processing. Part way through the run, $\widehat{\boldsymbol{m}}$ becomes sparse and a single cycle runs by up to three times faster. Third, we process the $\widehat{\boldsymbol{m}}$ data directly off the disk using memory mapping (`mmap()` in Linux) and do not keep the full matrix in memory. It turns out that the processing per document is sufficiently slow that this use of disk makes no difference in speed. The $\boldsymbol{\Omega}$ matrix, of size 260Mb (with 1000 components and $65,000$ words), needs to be stored in main memory since access to it is effectively random.

Space requirements for runtime are $O(K*(I+J)+S)$ where S is the size of the input data represented as sparse vectors, and each iteration takes $O(K*(I+J+S))$. Thus the computational requirements are comparable to an algorithm for extracting the top K eigenvectors usually used for PCA. Convergence is maybe 10-30 iterations, depending on the accuracy required, slower than its PCA counterpart. Our code is written in C using Open Source tools and libraries such as the GNU Scientific Library (GSL), and we intend to release an Open Source version once development is sufficiently advanced.

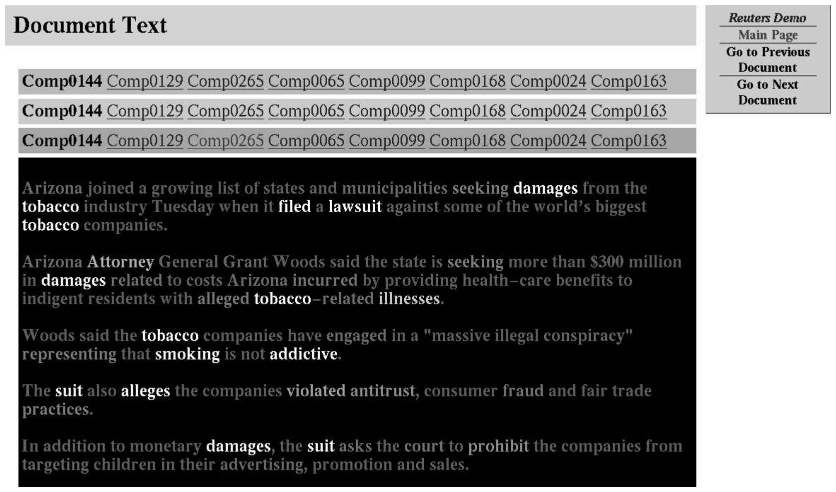
4 Reuters Experiments

We collected bag-of-words data from the new Reuters Corpus, containing $806,791$ news items from 1996 and 1997. On average, each news item is 225 words long, which translates to a total of 180 million word instances. About 390, 000 of these are distinct words; we kept the most frequent 65, 000 in the dictionary, accounting for 99.995% of the data. This lets us store the full set of documents in a bag-of-words format in 220 MB which can be kept for random access in a computer's main memory. The individual documents themselves, when individually compressed, take up about 3 GB on disk. We were able to process the full 806, 791 Reuters news documents on a single desktop Linux computer with 1 GB of memory and a 1.3 GHz processor in an overnight run. Parallel processing (e.g., [10]) is the next technique which we will need when running the algorithms on 20 GB web/HTML repositories. Runs were done training on 700, 000 documents and using the remainder as a hold-out set to ensure safe stopping and prevent over-fitting.

4.1 Inspecting Results

With data sizes of this magnitude, it is no longer feasible to analyze the results from a static printout. For instance, a cross-referenced report listing details of components, words and documents and their statistics can easily reach a few megabytes for even 10, 000 documents. Hence, we have written a custom web server in C++ that allows the user to examine different aspects of a model via a web browser. The model data takes many gigabytes of space with the larger models; only a small part of it can be kept in memory at once.

Documents, components and words each have their own pages that are dynamically generated and fully cross-referenced. Fig. 1 shows a part of a (origi-



Document Text

Comp0144 Comp0129 Comp0265 Comp0065 Comp0099 Comp0168 Comp0024 Comp0163

Comp0144 Comp0129 Comp0265 Comp0065 Comp0099 Comp0168 Comp0024 Comp0163

Comp0144 Comp0129 Comp0265 Comp0065 Comp0099 Comp0168 Comp0024 Comp0163

Arizona joined a growing list of states and municipalities seeking **damages** from the **tobacco** industry Tuesday when it **filed** a **lawsuit** against some of the world's biggest **tobacco** companies.

Arizona **Attorney** General Grant Woods said the state is seeking more than $300 million in **damages** related to costs Arizona incurred by providing health–care benefits to indigent residents with alleged **tobacco**–related **illnesses**.

Woods said the **tobacco** companies have engaged in a "massive illegal conspiracy" representing that **smoking** is not **addictive**.

The **suit** also **alleges** the companies **violated antitrust**, consumer fraud and fair trade practices.

In addition to monetary **damages**, the **suit** asks the court to prohibit the companies from targeting children in their advertising, promotion and sales.

Fig. 1. Color-Coded Document Display

nally) color-coded display of a document. In our interface, the three color components red, green and blue can be assigned individually to different topics; in the figure, they are all assigned to a single topic to get a gray-level display. Each word is colored according to its frequency in the component divided by its frequency in the data. The highlighted component seems to be about lawsuits against tobacco companies.

Naturally, components in the model reflect the Reuters Corpus content: more than a half of the components are related to subareas of finance and economy. In the 1000 component run, many components correspond to breaking out these reports into topics like "Australian stock market closing prices", "USA Research alerts," "European union shipping reports," "Corporate profits," "Nuclear power companies," etc.

Table 1 summarises the components for a $K = 20$ component run, ordered with the most numerous components at the top. Listed in the table, the effective

Code	Description	Prop.	Eff. words	Eff. docs. (1000s)
01	investigations/interviews/legal	0.08	310	347
10	corporate finance	0.07	307	279
07	stock exchange	0.07	275	226
11	international finance/currency	0.07	343	212
17	international/government relations	0.06	525	196
06	corporate research/intelligence	0.08	700	191
18	EU politics, economy	0.04	528	155
19	business/markets in greater europe	0.03	405	145
12	business in asia/sth africa	0.03	492	128
05	politics	0.03	940	123
13	corporate results/forecasts	0.09	199	122
00	production/capacity	0.03	552	120
15	bonds	0.06	587	119
16	commodities	0.06	584	113
09	terrorism	0.04	656	108
14	EU records, imports/exports	0.02	617	105
03	sports, conflicts	0.03	702	104
08	soccer and cricket	0.02	692	71
04	shipping reports, tennis/golf	0.01	725	57
02	indicators/ecomony	0.01	112	38

Table 1. Component Summary for K=20

number of words in a component is the log2 entropy of the component word vector (entry in $\boldsymbol{\Omega}$) raised to the power 2. The effective number of documents containing a component is the log2 entropy of documents given components (probability of being in a document given a word from the component) raised to the power 2.

For this run with components, each document might have between 2 and 8 components contributing to its word distribution, many with one dominant one. The top component (with code 01) reflects this because while it occurs in effectively 43% of documents, it only makes up 8% of the component contributions, thus when it occurs, it occurs on average as 20% of a document. Note the documents with a high proportion of the component with code 01 are all legal cases, whereas in general the component might occur in articles including an interview of a CEO or politician.

The following components are taken from a run for $K = 300$ components. Table 2 shows two components from the model. Typical words are ordered according to their frequency in the component. Unexpected words are scored by the $\log_2$-difference of their frequency from word prior. Component 37 is broadly religion-related, with a large expected number of words, 580. The typical documents contain a news item about the war on flies declared by the mayor of Manila and an item about the Pope blessing Christmas crib figurines.

Component 37		Component 79	
Typical Words	Unexpected Words	Typical Words	Unexpected Words
0.029 church	9.53 resplendent	0.053 internet	8.89 Vebacom
0.027 mayor	9.51 Tuckey	0.047 telecommunicat.	8.89 Viinanen
0.017 hundreds	9.51 Perafan	0.038 telecom	8.88 Dancall
0.017 pope	9.50 cobras	0.032 phone	8.88 xylitol
0.016 catholic	9.50 botulism	0.032 access	8.88 Sudirman
0.016 children	9.50 Balabagan	0.030 communications	8.88 Rhenus
0.015 people	9.50 Arreckx	0.027 telephone	8.88 Hotwired
0.014 who	9.49 Burman	0.025 service	8.88 Sunpage
0.013 roman	9.49 manifestations	0.022 mobile	8.87 Scaglia

Table 2. Top Words In Components 37 and 79

Component 153 in table 3 is interesting in that its typical words are per-

Component 153		Component 48	
Typical Words	Unexpected Words	Typical Words	Unexpected Words
0.193 we	6.62 Ospel	0.880 loss	10.47 Penril
0.043 very	6.62 Maucher	0.015 widens	10.47 Reddi
0.042 our	6.62 Jagmetti	0.012 write	10.47 Rayrock
0.027 do	6.62 Zinkernagel	0.011 narrows	10.47 Puretec
0.023 are	6.62 Kulczyk	0.009 charges	10.47 Yuoka
0.023 good	6.61 Peltola	0.005 restructuring	10.47 jpe
0.021 can	6.61 Integrion	0.003 writedown	10.47 Dorman
0.020 is	6.61 Bols	0.003 discontinued	10.46 Schmiedeknecht
0.020 going	6.60 Cees	0.003 disposal	10.46 Digene

Table 3. Top Words In Component 153 and 48

sonal pronouns and opinionated words. The typical documents are invariably interviews or press releases.

Component 48 in table 3 is an example of a low-entropy component with an expected number of words of only 2.36. It is dominated by the word "loss", which has a frequency of 0.88. The typical news items are financial news about losses. This component is basically triggered by the occurrence of the word "loss", making all the other high frequency words in the component more likely.

4.2 Explaining Component Dimensionality

To understand components and their relationship to the task of multi-faceted clustering, we plotted some diagnostics extracted from the result matrices $\boldsymbol{\Omega}$ and $\boldsymbol{m}$. The effective number of components in a document is the log2 entropy of its component proportion vector (entry in $\boldsymbol{m}$) for that document raised to the power 2. For instance, a document where the $\boldsymbol{m}$ vector is $(0.33, 0.33, 0.33, 0, 0, 0, \ldots)$ would have an effective number of components of 3, and a document where the $\boldsymbol{m}$ vector is $(1/K, 1/K, \ldots, 1/K, 0, 0, 0, \ldots)$ would have an effective number of components of K. Fig. 2 shows the relationship between document size and

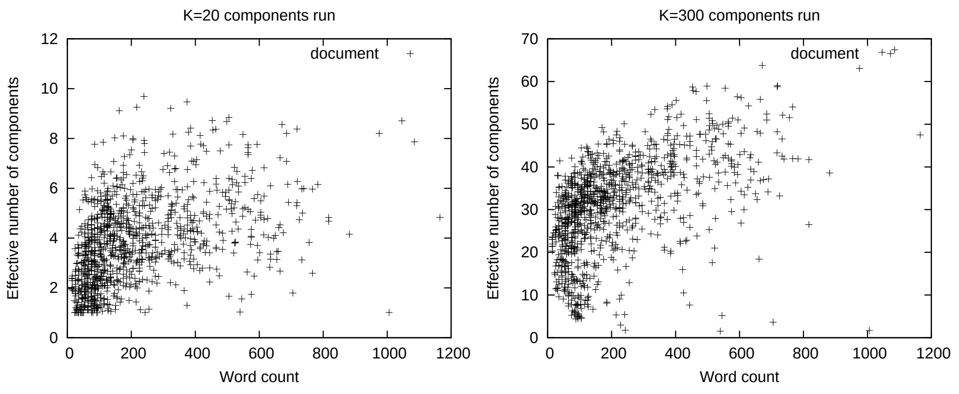


Fig. 2. Document Size versus Components. (a) 20 components, (b) 300 components

number of components for the runs using $K = 20$ and $K = 300$ components. Clearly, in this case, the number of components for each document indicates we are operating in the regime of dimensionality reduction. Most components are not synonym sets, they are topical sets of words. While for each component there will be a corresponding set of documents where this dominates, the majority of documents mix many different components.

Fig. 3 shows the effective number of words for different components runs. For instance, for $K = 20$ this plots data from the fourth column of Table 1. Components have been ordered in terms of their effective number of words, and then the effective number of words plotted (which is now increasing). For the $K = 300$ run for instance, some components are very specific, the extreme being component 48 above, and some components are far more topical with a large number of effective words. Note, however, that the components for the different runs are very different from each other and amenable to hierarchical combination (unlike the components obtained using PCA).

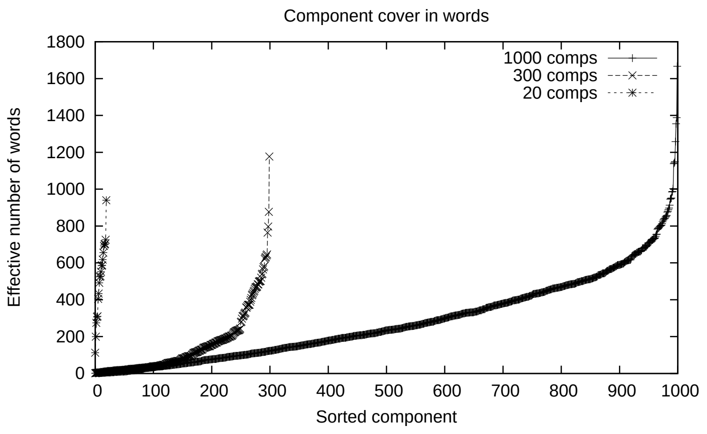


Fig. 3. Effective words per component

5 Overview of the Ydin Server

The six modules comprising the Ydin server, some of which are described above, are shown in Fig. 4:

XML/XHTML parser: Reads and parses XML/XHTML documents, creating a tree model of each document. XML elements are leaves in the tree.

Document analyzer: Converts document content to words and sentences. Words are initially stemmed, i.e. reduced to a root word, and placed in a dictionary and a word index.

Document database: Distributed storage for processed and compressed documents and information about them: dictionary, inverted index, bag-of-words data.

MPCA: Statistical component described above to generate component models ($\boldsymbol{\Omega}$) and document-component probability vectors ($\boldsymbol{m}$).

HTML interface: Interfaces with the MPCA and document database modules to provide the user with a high-level view of the documents. The interface has a number of optional functions:

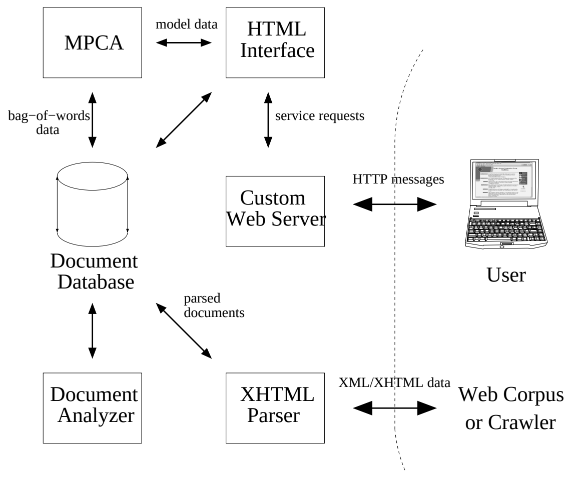


Fig. 4. Module diagram of the Ydin Server

- **search-by-topic:** enter topical words to do a topic search
- **find-similar:** find documents similar to the current document
- **component viewer:** view a component in terms of common words and typical documents
- **word viewer:** view a word in terms of its semantically related words

Custom web server: Our server responds to HTTP requests and dispatches them to the HTML interface, which then generates the desired pages dynamically. We use a custom server for maximum efficiency but a standard web server such as Apache could be used in its place.

We are extending Ydin to the task of extracting web taxonomies, which goes beyond component analysis.

For instance, to be able to provide links to reviews of a product the user is interested in, we need to be able to identify which documents contain product reviews. This entails matching the components generated by a statistical model with the semantics we are interested in. In other words, the statistical components need to be identified; we could use hand-made or supervised criteria for inferring such information. Other services such as comparisons of product features require similar functionality.

A move beyond simple bag-of-words analysis to richer context and sentential models may be mandatory to reach this level of sophistication.

6 Conclusion

We have demonstrated that recent extensions to PCA for multinomial data provide some support for multi-faceted clustering. We argued that while multinomial

PCA is really a dimensionality reduction algorithm, and not designed for multi-faceted clustering, if used right components can be extracted that are useful for the purpose of building a taxonomy. For this, we would need at least the ability to generate full topic hierarchies, and to perform automatic labelling/naming of topics in the hierarchy. We expect that by doing this for multiple companies at once, useful hierarchies could be established.

We have also presented our Ydin server which has been developed for the purpose of exploring the results of a multinomial PCA analysis on a large repository of HTML/XML data. We are currently extending the tool with view to building an Open Source system for information navigation and assisting the maintenance of semantic web.

Acknowledgements Thanks to Reuters for providing their Corpus as described in Footnote 1.

References

1. Cherry, S.: Weaving a web of ideas. IEEE Spectrum **39** (2002) 65–69
2. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000) 888–905
3. Moore, A.: Very fast EM-based mixture model clustering using multiresolution kd-tree. In: Neural Information Processing Systems, Denver (1998)
4. Han, E.H., Karypis, G., Kumar, V., Mobasher, B.: Clustering based on association rule hypergraphs. In: SIGMOD'97 Workshop on Research Issues on Data Mining and Knowledge. (1997)
5. Vaithyanathan, S., Dom, B.: Model-based hierarchical clustering. In: UAI-2000, Stanford (2000)
6. Bradley, P., Fayyad, U., Reina, C.: Scaling clustering algorithms to large databases. In: Proc. KDD'98. (1998)
7. Crammer, K., Singer, Y.: A new family of online algorithms for category ranking. In: 25th Annual Intl. ACM SIGIR Conference. (2002)
8. Karhunen, J., Pajunen, P., Oja, E.: The nonlinear PCA criterion in blind source separation: Relations with other approaches. Neurocomputing **22** (1998) 520
9. Hofmann, T.: Probabilistic latent semantic indexing. In: Research and Development in Information Retrieval. (1999) 50–57
10. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet allocation. In: NIPS*14. (2002) to appear.
11. Buntine, W.: Variational extensions to EM and multinomial PCA. In: ECML 2002. (2002)
12. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley (1999)
13. Tipping, M., Bishop, C.: Mixtures of probabilistic principal component analysers. Neural Computation **11** (1999) 443–482
14. Kontkanen, P., Myllymaki, P., Silander, T., Tirri, H.: BAYDA: Software for bayesian classification and feature selection. In: Knowledge Discovery and Data Mining. (1998) 254–258

Semantic Web and Web Services from Operator's Perspective

Lasse Pajunen, Vesa Huotari, and Seppo Heikkinen

Elisa Communications Corporation, P.O.Box 40, FIN-00061 ELISA
{Lasse.Pajunen, Vesa.Huotari, Seppo.Heikkinen}@elisa.fi

Abstract. Traditionally, operators have been operating single use networks like phone networks targeted to pure voice transmission. Later, some services like Short Messaging System have offered methods for sending data on top of these networks. In addition, there have been many extensions to these networks (faxes, logos, and ring tones) but the core network has been designed originally for single use.

Now, as computer networks (mainly Internet networks) are advancing to a level, where they are becoming capable of handling tasks that previously required specialized networks, operators are starting to provide solutions for this new platform. This means transferring current products to new environment and at the same time developing new ones.

What are the products and services that the customers are asking for and can expect from the operators? Current offerings include things like network infrastructure (access and routing), network services (DNS and firewalls), service platforms (WWW and email), and complete solutions (including all communication with integration services). In the future, new types of services on the application level (including security and services like UDDI and ebXML Registry) will emerge after these services become commercially feasible.

Semantic Web and Web Services define a new layer of interoperability on top of current Internet technologies. That layer implements a commonly understandable knowledge serialization method on top of basic binary streams. In the future services, Semantic Web and Web Services are playing an important role. The main idea of Internet and XML is interoperability: interoperability between many independent service providers, solutions providers, and customers. Interoperability makes it possible to connect separate services to build larger and more complex systems with relative ease.

However, current technologies do not solve all the problems. These problems are not new, but their importance is growing while more systems are integrated. To be able build large systems suitable for business operation, one has to take into account requirements like quality of service, security, and service management. The support for these properties has to be built into the systems. In this paper, we discuss the relevance of Web Services and Semantic Web for the operators.

1 Introduction

Computer networks, mainly Internet Protocol (IP) based networks, are important technologies for current product offerings from operators. Therefore, *operators* are companies that provide services related to information transmission networks. Nowadays, IP technologies provide the core network that is used for data transmission. Operator services can be divided in two categories based on what kinds of services are considered. *Network operators* provide services related to lower level IP networks. *Service operators* provide services on top of these networks.

Network operator products are typically connection or access offerings that link customers to the core backbone IP network. These access services and technologies include dialup-based services like Integrated Services Digital Network (ISDN), broadband services like Asynchronous Digital Subscriber Line (ADSL), and dedicated services like Asynchronous Transfer Mode (ATM).

Nowadays application level services are implemented on top of these IP networks. For example, the web technologies are built on them. The other similar services are mail and voice over IP services. In addition to these application services, service operators offer infrastructure services like Domain Name System (DNS) management, firewalls, and Public Key Infrastructure (PKI) management.

Network operator has switched from telecommunication networks to computer networks. Declining revenues from voice networks are replaced by revenues from IP based services. The number of households having broadband IP access has increased a lot and it is still increasing rapidly.

On the other hand, IP and web technologies are stabilizing. The initial problems have been solved and standardized. The other ones are currently being researched. This research is very important and finalizing technologies is critical for operators. After finalizing issues related to quality of service (QoS), it is, for example, possible for service operators to start providing technology as service instead of each company provides it by itself. This includes services like managing of networks and applications that companies can outsource to operators.

However, before technology is ready and operators can take full advantage of it, some requirements for technologies need to be answered. In this paper, we will analyze few of them. For example, in addition to the quality of service issues, the technology has to provide ways to integrate services in an efficient way.

The rest of this paper is organized as follows. Section 2 describes current operator services. Section 3 describes how operators see Web Services and Semantic Web from their point of view. Section 4 lists requirements for future development. Finally, Section 5 concludes.

2 Current Operator Services and Technologies

Traditionally operators have offered just bit transfer services, but gradually they have started moving from these bottom layers toward the top in order to increase their revenues. This has lead them to be Internet Service Providers (ISP) and Application Service Providers (ASP). The meaning of the term operator itself has become bit ambiguous because it no longer refers solely to telecom operators as new players have entered the growing markets.

Internet access has been “a must” for some time now and only the means of access have changed as the broadband connections like ADSL have really started flying. The term broadband might be bit misleading though as the customers have been more interested in flat rates and always on kind of capabilities. This, however, is just transfer and routing of IP traffic, although some mandatory services, like Domain Name System (DNS), have been needed to provide a “standard” way of accessing Internet. Some supplementary services like time service with Network Time Protocol (NTP) have also been offered, but they may not have been as relevant to all the customers. Operators have also offered traffic transfer in a more limited fashion in order to provide possibilities to form virtual private networks (VPN), for example to connect remote users working at home to company networks. VPNs have also been created with cryptographic techniques, but usually this has been done with the help of security companies and their products. Operators have also provided filtering services with the use of firewalls to block unwanted traffic for customers not wanting to manage their own firewalls. Some operators have also provided network-monitoring services to guarantee fast response times in case of device failures. Usually this kind of service has been relevant only to bigger customers.

On application level, the email and the World Wide Web (WWW) are the most widely used services, even though discussion forums like Usenet news and Internet Relay Chat (IRC) have been popular too to a certain extent. For many customers WWW has actually been a synonym for the Internet, but in simplistic case, it is transfer of HyperText Markup Language (HTML) encoded data with the use of HyperText Transfer Protocol (HTTP) riding on top of Internet protocols. WWW cannot actually be seen as operator provided service as the actual data is coming from various different places around the world. As this was the case, the operators and other players saw it fitting to try to collect the interesting information in one place for easy access and at the same time increase the traffic to their own sites. These places, portals, could contain interesting link lists, weather forecasts, news, discussion forums, and so on. Often these were done using some proprietary tools and techniques although scripting and Java Servlet technologies got some popularity. The choices done could have been influenced or mandated by the used platforms, like UNIX versus Windows kind of setting. Some of the services in the portals were bought from outside sources and then integrated to the overall service to give a consistent view. It was quite possible that each of these services was done in its own way and some extra effort was needed every time to get the single service to fit into the overall system. For example, when offering some news service, portal operator may have been forced to implement some proprietary form of communication with the actual news provider to fetch the news articles. As these portals grew in size, some application server frameworks were needed to manage the whole system and provide consistent ways to interconnect with other systems. Interconnection could be for example Remote Method Invocation (RMI) between different Java application and Open Database Connectivity (ODBC) call to a database to fetch some information. Several vendors offered their application servers to handle such tasks in one contained environment, often using Java and its Enterprise Edition (J2EE) concepts.

The variety of services has proliferated and especially mobile related services like logos and ring tones have been popular. Many other Short Message Service (SMS) based services have also increased service operator revenues although service provider may have been forced to make contracts and service implementations individually with all the different mobile operators in order to guarantee that the service is working for all the customers.

Operator have also offered some security aware services but with lesser extent. Often this has been just securing web pages with the use of Secure Socket Layer (SSL) so that the application level traffic has been encrypted and the server authenticated. This has usually left the client unauthenticated. On individual company level, adaptation of Public Key Infrastructure (PKI) ideas has been bit more widespread but no real widely accepted and used infrastructure beyond that has existed even though national government has put some effort on it.

3 Important Aspects of Web Services and Semantic Web

3.1 Web Services and Semantic Web through Operator Glasses

In this section, we will analyze Web Services and Semantic Web from operator's perspective. At first, we discuss Web Services and Semantic Web from casual view to be able to compare them to operators' view.

The common view describes *web* as an abstract (imaginary) space of information [1]. Information covers documents, sounds, videos etc. Information is collected as pages that are linked together by hyperlinks. Therefore, the web is large hyperlinked data storage. A browser agent retrieves one page at the time for display.

The World Wide Web is used more and more for application communication. The programmatic interfaces available are referred to as *Web Services* [3]. These APIs describes how the services or applications can be used. The interoperable way of describing interfaces enables then for example development of intelligent programming IDEs (Integrated Development Environment).

The third thing is *Semantic Web*. The Semantic Web is the abstract representation of data on the World Wide Web, based on the RDF standards and other standards that are being defined [5]. It is being developed by the W3C in collaboration with a large number of researchers and industrial partners. The Semantic Web therefore adds meta information to the web. This for example, helps browsers to find relevant pages, if metadata vocabulary matches. Then metadata makes it possible to do improved versions of Google and Altavista services. The goal is to make the web easier to use for intelligent agents. Semantic Web itself does not create concrete new services, but it enables them.

Operator's role as a network provider (of course, operators can operate many other services too) is to transfer information. Operators' main objective is not a storage view, but active information transmission view. Therefore, the operators see these concepts little bit differently.

The web can be seen as communication between customer (end user) and service provider (storage provider and content creator). The operators transfer bits and provide proxy kinds of functionalities to improve quality.

Web Services describe services and, in addition, they describe service invocation model. The important thing is that it describes concept of intermediaries. Therefore, in Web Services the network is the critical part. By offering intermediate services, more functionalities and features beyond the capabilities of normal proxies can be offered like quality of service, security, encryption, non-republication, and reliable transfers.

Semantic Web provides ways to add meta information for services and use it to provide additional services. This can be used to improve other services, billing, and provide semantic services, like trust, security, previous services, agents, and brokers. By providing information, for example, on how much each piece of information costs, the operator can implement billing for the content provider. On the other hand, operators can provide matchmaking services for application integrators as brokering services. Therefore, operators often build services on top of their networks. Semantic Web provides technologies to build these independently of data streams.

The biggest advantage of Web Services and Semantic Web is achieved, when they are combined. The Semantic Web Services are services that can take advantage of Web Services to describe functionality and information transfer and can take advantage of Semantic Web to provide meaningful intermediary services within the network.

3.2 Semantic Communications Protocol

In the network, there are a couple of support services. Providing these services requires understanding of semantics. RosettaNet, ebXML, OASIS etc are important standardization bodies for middleware. Standardized processes and vocabularies make it possible for building infrastructure services.

Web Services and Semantic Web technologies provide a new layer to the communication protocol stack. Web Services describes how information can be exchanged. Semantic Web helps to provide additional services on top of Web Services. Figure 1 shows a semantic communication protocol stack.

On the bottom of protocol stack, there are transmission technologies. The bottommost level is access level to network. This includes technologies like DSL. On top of access level, there is IP network. IP unifies all access and transmission technologies beneath it. On top of IP network, there are many IP application level protocols, like HTTP, SMTP, and Blocks Extensible Exchange Protocol (BEEP). These technologies provide a method to transport information from one user to another.

On top of IP network, there is a Web Service layer. Web Services layer is complex and means more than just SOAP and Web Services Description Language (WSDL). This perspective requires features described later. The layer includes messaging features found in RosettaNet Implementation Framework (RNIF), ebXML Messaging. Therefore, the Web Services layer provides a reliable messaging layer for upper level conversations.

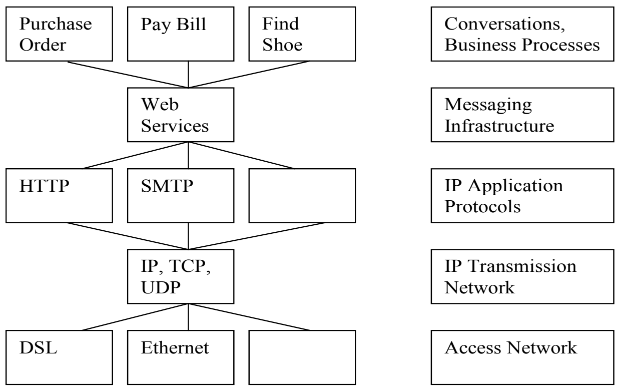


Fig. 1. Semantic Web Services Protocol Stack

On top of Web Services messaging layer, there is process description layer. Conversations are processes. Processes can be described as RosettaNet Partner Interface Processes (PIPs). Operator provides services to process designing consults and help in building these processes. These conversations connect different services semantically together.

To summarize, Web Services is a technology infrastructure for a standard way of implementing conversations on top of web. Web Services includes standards for the actual implementations (interfaces) and a usage model (process description frameworks).

Semantic Web means adding standardized metadata. There will be several different metadata families, like there are languages in our real life. The operators help to solve problems with these separate metadata spaces. Operators also provide some services for some functional spaces.

3.3 Role of Operator in Web Architecture

Web Services and Semantic Web are introducing meaning on top of bit protocols. This, on the other hand, makes it possible for the operators to provide services related to communication. Figure 2 shows a picture of services and functionalities in web architecture.

On most left and right, there are customers' internal systems. On top of them, there is company's internal integration. In the middle are the companies' public services and Internet with support services for providing functionality and integration to other companies.

At the operators view, on the edge, there are customer specific customizations and services. This means links to customers' private services. As long as there are legacy systems that do not support Web Services, connection systems can be implemented as separate integration projects.

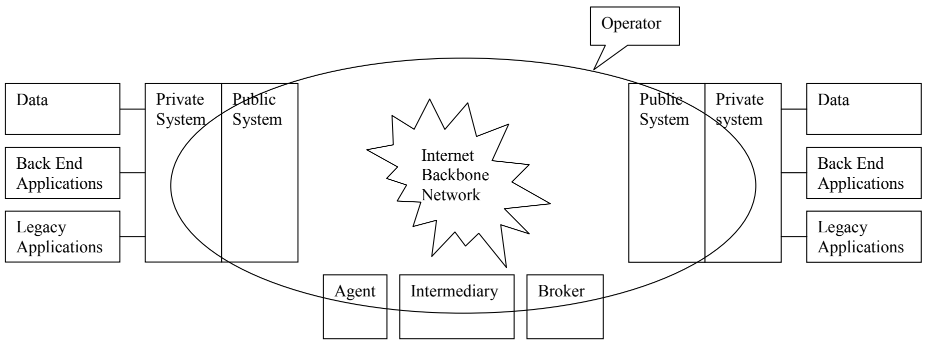


Fig. 2. Role of Operator in Web Architecture

Another option from operator's perspective comes, when computer systems have well-defined interfaces. This enables outsourcing service management. After systems have public interfaces for systems users and private interfaces with management support for system operators, the systems can be moved to hosted environments provided by the operators.

Intermediaries provide a concrete point to add services in the network for operators. Intermediaries are in the middle of message flow and therefore they provide a good opportunity to add features in communication. The previously mentioned services can be implemented within intermediaries.

4 Requirements for Future Technologies

When moving towards component-based architectures, the complexity increases. This brings vulnerability to different malfunctions. These malfunctions might relate to network errors or breakdowns, software malfunctions, or different hardware problems. When composite services are made by using different components from various parties, the demand for quality network in terms of low latency time and low error rate is imminent. The state and quality of service of its various components have to be monitored and adjusted if needed in order to guarantee proper workflow. If service or component becomes unavailable, otherwise fails or the quality decreases, there should be ways to re-configure the system. Preferably, re-configuration happens without user intervention. Although this does not remove the need for proper management interfaces; maintenance should be able to fluently change the system behavior and be able to view the state of the system with details. Eurescom [2] project P1108 (Workflow-based On-line Validation of Complex Component based Internet Services) has studied previously mentioned aspects.

Without previous experience, how can we rely on services made by others who might be totally unknown to us? The analogy with establishing partnerships within Web Services framework is similar like, for example, in real world where individuals make decisions based on recommendations, own decision or using other methods. Security functionalities may help building the trust but they offer only mechanisms of securing messages and authorization of different rights. Experience with security issues is extensively important when it comes to tightly coupled applications. Some of these solutions like SSL and other encryption methods may be adapted to loosely coupled Web Service applications. Support for PKI should also be considered as one element of building the trust. Still, simplicity raises new concerns that must be dealt with in order to build secure applications. Web Services security should offer at least the following functionalities [4].

- Confidentiality means that the information is available only to entities that are authorized to that particular information.
- Authorization grants rights to certain information and guarantees that sender is authorized to send messages.
- Data integrity ensures that message is not altered or destroyed.
- Proof of origin is evidence identifying the originator of a message.

Some services might be totally free of charge but in most cases service provider wants to make revenue. Payment models might vary but two obvious possibilities are consumption based charge and monthly charge. To be able to charge someone, there must be negotiated contracts between parties. These contracts may be very detailed including agreements about service level, cost, and sanctions just to mention few. Automated negotiations are not possible in large scale yet. Furthermore, there is need for interoperable, scalable, and reliable accounting mechanisms that can deal with large number of transactions coming from different sources.

Finally, different vendors may interpret W3C recommendations with their own interests or implement extensions that endanger interpretability. All these problems and requirements together create challenges to electronic commerce and more meaningful web.

Table 1 summarizes requirements for technologies. There are four main categories: security, quality of services, management, and business requirements. Each category has several requirements that were described above.

Table 1. Requirements for Technologies

Security	Quality of Service	Management	Business Requirements
Confidentiality	Latency	Composition	Trust
Authorization	Network errors	Monitoring	Payment models
Data integrity	Recovery	Validation	Accounting
Proof of origin			Interpretability
PKI-readiness			

5 Conclusions

IP and other network technologies are quite stable. Lot of technology is standardized, but some important improvements are still under standardization. The technology in general is well adopted and the need for the services is there.

Web Services and Semantic Web provide a framework for software developers to build software that can be operated by the operators. At first, the technology can be used for application integration and later it can provide methods to separate services from company internal IT departments to be operated by operation specialist companies.

Web Services and Semantic Web need to be standardized more. Before operators can take advantage of the new software, some aspects have to be improved. Especially, quality of service and management issues are important to the operators, as they have to be able to integrate the new solutions to their existing management procedures. In addition, security aspects have vital importance because traditionally operators have enjoyed certain level of trust and they want to retain this impression.

References

1. Tim Berners-Lee. Press FAQ web page. http://www.w3.org/People/Berners-Lee/FAQ.html.
2. Eurescom. Eurescom web site. http://www.eurescom.de.
3. Hugo Haas. Web Services Activity web page. http://www.w3.org/2002/ws/.
4. Heather Kreger. Web Services Conceptual Architecture (WSCA 1.0). IBM Software Group. 2001.
5. Eric Miller, Ralph Swick, Dan Brickley, Brian McBride, Jim Hendler, Guus Schreiber, Semantic Web web page. http://www.w3.org/2001/sw/.

Combining WSDL and RDF data models

Janne Saarela

Profium, Lars Sonckin kaari 12, FIN-02600 Espoo
janne.saarela@profium.com
http://www.profium.com

Abstract. Semantic Web technologies have been developed by the W3C in a bottom-up fashion where the basic building block is Resource Description Framework (RDF) technology. RDF provides a data model that can express machine-understandable semantics.
Web Services technologies have been developed and introduced to the marketplace by software vendors. W3C has followed up on these developments and is now standardising technologies in this space. Composed of three layers, Web Services has a nearly finalised protocol layer, Simple Object Access Protocol (SOAP), a service description layer Web Services Description Language (WSDL) currently worked out by the W3C working group, and service discovery mechanism Universal Description, Discovery and Integration (UDDI) currently considered by OASIS.
This paper explores different approaches of harmonising the RDF and WSDL technologies enabling Semantic Web and Web Services communities to benefit from each other's work.
The goal of this harmonization is to determine whether UDDI type of services can be provided by existing RDF repositories.

1 Introduction

Semantic Web has the ambitious goal of being able to encode machine-understandable semantics about networked objects. This goal partly overlaps with one of the goals of the Web Services movement, namely the service description. Service description is currently based on Web Services Description Language (WSDL) and resource description on the general RDF data model in the Semantic Web context.

The vocabulary used by WSDL has been specified using an XML schema. This vocabulary, however, could be augmented with properties outside the scope of Web Services. In fact, the DAML-S[2] specification addresses ontology level issues in service description and addresses this goal on a higher level.

Before we can determine whether the WSDL and RDF data models can be combined in practice, let us examine these data models briefly in the following subsections.

1.1 RDF data model

RDF data model and syntax specification [5] establishes a formal data model of a directed labeled graph. The nodes or resources in this data model are URI addressable objects. The properties associated to these resources are themselves URI addressable objects, Property objects. The RDF schema technology [3] provides tools for the creation of a simple type system based on object-oriented concepts such as classes and inheritance. The role of the RDF schema is to assist in validation of RDF data models and enforce the creation of valid descriptions when configured to authoring tools.

The following figure describes a simple RDF data model where a resources at URI has two properties associated with it: dc:coverage and my:participates. The values of these properties are a literal node and a resource respectively. We use here the prefixed notation for properties to remind that at the end the properties are URI addressable objects themselves.

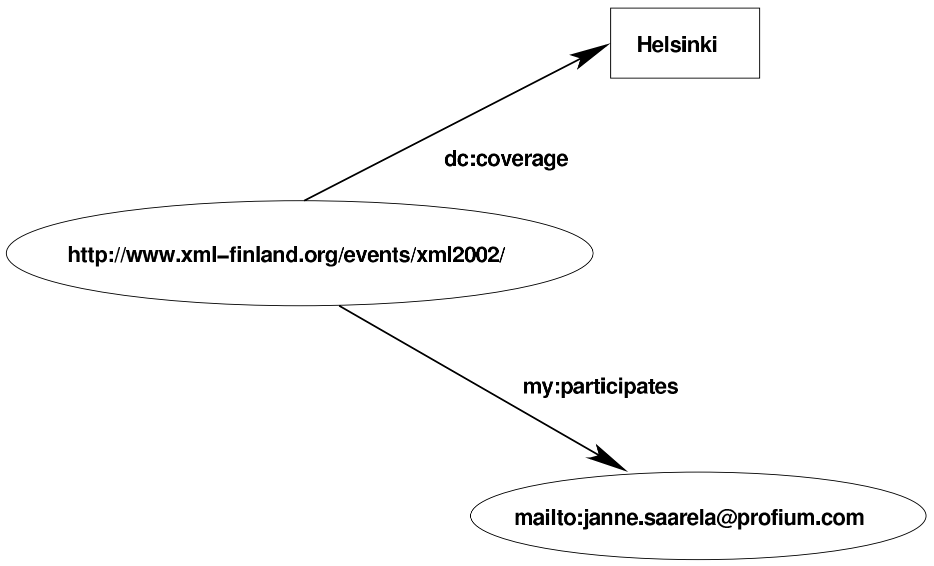

Helsinki
dc:coverage
http://www.xml–finland.org/events/xml2002/
my:participates
mailto:janne.saarela@profium.com

Fig. 1. Properties in the RDF data model can have resource and literal type of values

1.2 WSDL data model

WSDL was submitted to W3C in its version 1.1 to the W3C. The latest release of this technology is V1.2 [4] published by the Web Services Description Working Group at the W3C.

WSDL data model is constrained by a corresponding XML Schema definition. This definition effectively lays out a structure of five layers.

1. Types - an embedded XML Schema within a WSDL description
2. Messages - message structures that can use XML Schema basic data types or the ones defined in the preceding types section.

3. Ports - one port is a set of abstract operations and the message structures they use. The port definition effectively lays out the interaction model of the service. The interaction model can be one of four basic models:
   (a) one-way
   (b) request-response
   (c) solicit-response
   (d) notification
4. Bindings - A binding of the operations of a given port to actual formats and protocols. In fact, this is where WSDL binds messages to SOAP protocol but leaves the door open for other protocols, too.
5. Services - a grouping of ports together into one actual URI address where the messages can be invoked. One should note that a service URI alone does not uniquely identify an operation.

Whereas the WSDL description tries to be complete and unambiguous with respect to information content, binding of these service descriptions to actual protocols such as HTTP introduces a new problem: the mapping of HTTP traffic unambiguously to a WSDL port.

For example, given a service uri `http://www.profium.com/services` and an operation `register`, how would the HTTP GET or POST request be mapped internally within an HTTP server to a service implementation? This is a problem addressed by WSDL 1.2: Bindings[6] document. However, this version of the document does not consider the unique addressing of operations critical and therefore leaves the mapping a runtime issue. We will see in the later sections how the unique addressing would improve WSDL expression capabilities outside the scope of Web Services.

2 Approach 1: Linking to WSDL

This section lays out an attempt to use RDF technology to enrich WSDL service descriptions by using RDF in parallel to WSDL. The goal is to have RDF descriptions that uniquely identify information objects within WSDL and thus allow their description with the RDF data model.

In order to create a URI for information objects in WSDL, we are not looking for ways to address names within a WSDL document that might be stored on the Web. In this way we would only uniquely address information objects of a certain WSDL document. Rather, we wish to have a unique addressing mechanism of information objects within a Web Service described by the WSDL description.

In order to do this, we need to partly do the same task the WSDL 1.2: Bindings document does for HTTP. Their approach is to map HTTP GET and POST request URIs and message bodies to message structures. However, our task is simpler as we only need to unambiguously create a URI for a given operation at a given service URI. In addition, the reverse operation i.e. the determination of a service URI and an operation from a URI should be unambiguous.

Our simple solution is to use fragment identifier concatenated to the service URI to create a URI for operations. Following the earlier example, service URI `http://www.profium.com/services` and an operation `register` would result in `http://www.profium.com/services#register`. This simple approach is, however, not without implications. As Web Services typically use SOAP payload, the media type associated with the service URI is always either `application/xml` or `text/xml` in version 1.1 of SOAP. W3C Working Group is currently considering moving into media type `application/soap+xml` in version 1.2 of SOAP. This media type is defined in an IETF draft document[1] to have fragment identifier specified identically to the other XML media types.

The use of fragment identifiers in our approach thus potentially conflicts with users who address SOAP V1.1 or V1.2 compatible message structures with XPointer based addressing. XPointer[11] is the designated language of fragment identifiers of XML media types.

The following figure shows conceptually this approach how information objects, in this case operations, can become parts of the RDF data model and thus allow more elaborate service description.

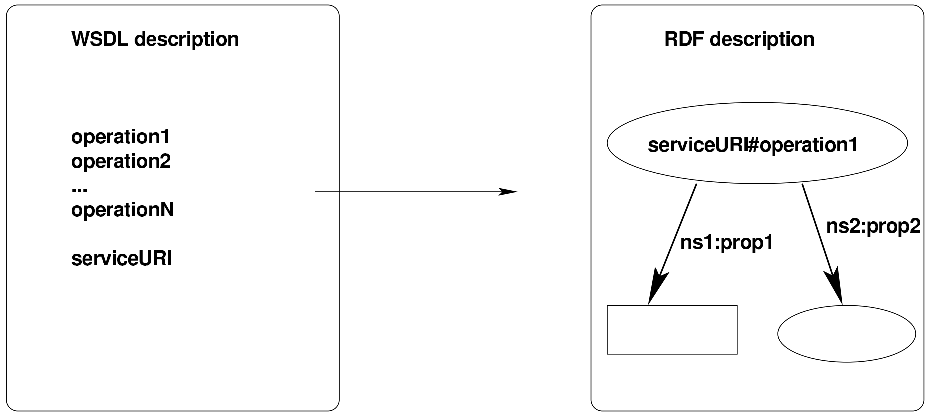


Fig. 2. Deriving RDF data model resources from WSDL operations

3 Approach 2: Interleaving WSDL and RDF

This section lays out an attempt to use RDF technology to enrich WSDL service descriptions by using RDF embedded inside WSDL. The goal is to have one single service description that satisfies Web Services interoperability and opens the door for RDF and Semantic Web interoperability.

We started out with a simple WSDL description from the WSDL specification itself that describes a trading data service with a request-response interaction model. As such, this XML based description is not RDF. The RDF context-free grammar does not allow some of the markup structures used by WSDL.

Our goal was to develop an XML markup structure that conforms to the RDF context-free grammar and at the same time preserves the original information objects and their relationships. This manual process could result in multiple XML encodings but the one we developed is described in Appendix 1 of this paper.

The underlying RDF data model of this WSDL description is presented in the following Figure.

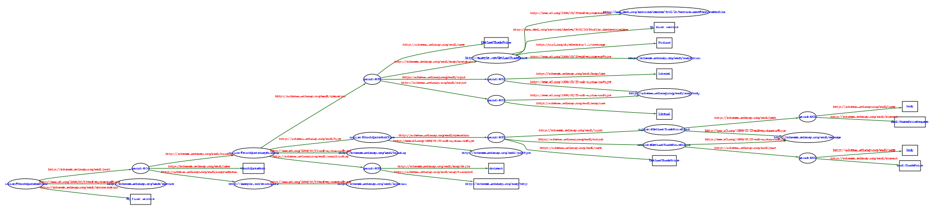


Fig. 3. Suggestion of WSDL in RDF

4 Validation of results

We wanted to validate the two approaches of this paper with practical tools to see if they were feasible to implement.

We have run the following tests with the commercial Profium Semantic Information Router (SIR) product. The product allows easy run-time configuration of new RDF schemas and was thus suitable for quick prototyping.

4.1 Approach 1

We first explored approach 1 by using DAML-S[2] RDF schema for Services and typing operation nodes as `Service` instances. This approach enables a query on the RDF data model to find services that match the properties of a queried service.

DAML-S describes services on a higher level of abstraction. DAML-S introduces the concept of a `ServiceProfile` which is used to find a service, not to invoke it. DAML-S also presents properties such as input, precondition and postcondition to let software processes or agents determine the type of data and types of other services a given service deals with.

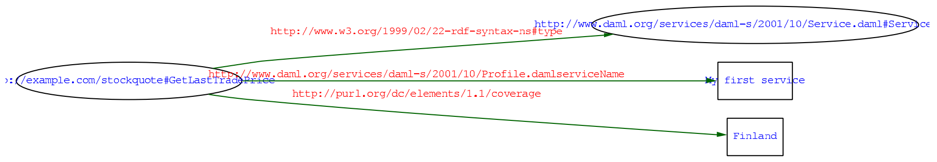
It is feasible to generate the following RDF markup from WSDL, either with a dedicated piece of software or an XSLT transformation.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
         xmlns:damls="http://www.daml.org/services/daml-s/2001/10/Service.daml"
         xmlns:damlp="http://www.daml.org/services/daml-s/2001/10/Profile.daml"
         xmlns:dc="http://purl.org/dc/elements/1.1/">

  <rdf:Description rdf:about="http://example.com/stockquote#GetLastTradePrice">
    <rdf:type
rdf:resource="http://www.daml.org/services/daml-s/2001/10/Service.daml#ServiceProfile" />
    <damlp:serviceName>My first service</damlp:serviceName>
    <dc:coverage>Finland</dc:coverage>
  </rdf:Description>
</rdf:RDF>
```

This approach effectively has the data model we have been looking for with approach 1. The following figure shows the RDF data model of the previous markup.



://example.com/stockquote#GetLastTradePrice
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.daml.org/services/daml-s/2001/10/Service.daml#Service
http://www.daml.org/services/daml-s/2001/10/Profile.damlserviceName
My first service
http://purl.org/dc/elements/1.1/coverage
Finland

Fig. 4. RDF data model with approach 1

This data model was received by Profium SIR and a query was run to find a service profile that resides in Finland. For clarity, this simple query can be expressed as a pseudo logic program query as

```
:- triple(X, dc:coverage, literal(Finland)),
   triple(X, rdf:type, resource(ServiceProfile)).
```

where triples are defined as `triple(subject, predicate, object)`.

4.2 Approach 2

The approach 2 was validated in a similar way i.e. the RDF schema was configured to SIR and the data model was received by the system. Once the complete RDF data model was in a persistent store, a similar query was developed and run succesfully.

5 Next steps

We are now ready to proceed to experiment UDDI type of services on the RDF descriptions stored using either one of these two approaches.

For example, the `find_service` API call of UDDI V3.0 servers shows very limited expressivity in terms of search terms. Using wildcarded strings to find a service with a certain name is a task human beings are willing to do. Fully automated search facilities ask for more precision.

To this end, UDDI V3.0 provides a type category system that can be used to find services. A categorisation is established with a logical name such as `uddi:ubr.uddi.org:categorization:geo3166-2` which can then have a human readable value such as `GEO:France` and machine-understandable value such as `FR`.

This (keyword,value) based approach has lower expressivity than RDF the data model which can have structured values due to the resource nature of objects of statements.

In addition to developing a more expressive search interface to services than what UDDI provides, the technical API for managing RDF type of service description needs more work. Seaborne[9] has suggested an API for manipulating RDF data models over the network and is a good starting point for further work.

6 Related work

Uche Obguji [7] has studied the harmonization of WSDL and RDF in an incremental fashion where the WSDL V1.1 compliant descriptions can be augmented with RDF specific markup. His work was the inspiration for the Approach 2 presented in this paper.

7 Summary

This paper has taken the goal of bringing the description language of the Web Services domain together with the description language of the Semantic Web domain. Web Services Description Language (WSDL) and Resource Description Framework (RDF) have been the two separately grown technologies which are now starting to converge.

This paper follows on the work initiated by Obguji[7] on the data model harmonization level. Higher service descriptions have been pursued in great detail by the DAML-S[2] initiative which has the ambitious target of nearly proving Web based services' correctness. Their pre- and postconditions for service access develop even further the formalised WSDL input and output message structure validation aspects currently supported by XML Schema technology.

The main contribution of this paper is the feasibility study of two different approaches to combining WSDL and RDF data models. The initial results look promising but more work is still needed to validate the ambitious goal that RDF repositories could provide UDDI type of service description and discovery facilities.

References

1. Baker, M., Nottingham, M.: The "application/soap+xml" media type. (2002) IETF draft.
`http://www.w3.org/2000/xp/Group/2/06/18/draft-baker-soap-media-reg-01.txt`
2. Ankolekar, A. et al.: DAML-S: Web Service Description for the Semantic Web. Proceedings of the 1st International Semantic Web Conference (2002).
3. Brickley, D., Guha, R.: Resource Description Framework (RDF) Schemas. W3C Working Draft. (2002)
`http://www.w3.org/TR/rdf-schema/`
4. Chinnici, R., et al.: Web Services Description Language (WSDL) V1.2. W3C Working Draft
`http://www.w3.org/TR/wsdl12/`
5. Lassila, O., Swick, R.: Resource Description Framework (RDF) data model and syntax (1999). W3C Recommendation.
6. Moreau, J-J.: Web Services Description Language (WSDL) V1.2: Bindings. W3C Working Draft. (2002)
`http://www.w3.org/TR/2002/WD-wsdl12-bindings-20020709/`
7. Obguji, U.: Supercharging WSDL with RDF (2000).
`http://www-106.ibm.com/developerworks/library/ws-rdf/?dwzone=ws`
8. Saarela, J.: The role of metadata in electronic publishing. Acta Polytechnica Scandinavica (1999).
9. Seaborne, A.: Accessing RDF remotely An RDF NetAPI using HTTP (2002).
`http://www-uk.hpl.hp.com/people/afs/Joseki/doc/RDF_NetAPI.html`
10. UDDI Version 3.0 (2002).
`http://uddi.org/pubs/uddi_v3.htm`
11. Grosso, P. et al.: XPointer Framework (2002). W3C Working Draft.
`http://www.w3.org/TR/xptr-framework/`

Appendix 1: WSDL as RDF

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns:w="http://schemas.xmlsoap.org/wsdl/"
         xmlns="http://schemas.xmlsoap.org/wsdl/"
         xmlns:damlp="http://www.daml.org/services/daml-s/2001/10/Profile.daml"
         xmlns:dc="http://purl.org/dc/elements/1.1/">

    <message rdf:ID="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message rdf:ID="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType rdf:ID="StockQuotePortType">
        <operation rdf:parseType="Resource">
```

```
           <name>GetLastTradePrice</name>
           <input rdf:resource="#GetLastTradePriceInput"/>
           <output rdf:resource="#GetLastTradePriceOutput"/>
        </operation>
    </portType>

    <binding rdf:ID="StockQuoteSoapBinding">
        <w:type rdf:resource="#StockQuotePortType" />
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation rdf:parseType="Resource">
           <name>GetLastTradePrice</name>
           <soap:operation>
             <soapAction rdf:about="http://example.com/GetLastTradePrice" />
           </soap:operation>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

    <service rdf:ID="StockQuoteService">
        <documentation>My first service</documentation>
        <port rdf:parseType="Resource">
           <name>StockQuotePort</name>
           <binding rdf:resource="#StockQuoteSoapBinding" />
           <soap:address>
             <location rdf:about="http://example.com/stockquote"/>
           </soap:address>
        </port>
    </service>

    <rdf:Description rdf:about="http://example.com/GetLastTradePrice">
      <rdf:type rdf:resource="http://www.daml.org/services/daml-s/2001/10/Service.daml#Servi
      <damlp:serviceName>My first service</damlp:serviceName>
      <dc:coverage>Finland</dc:coverage>
    </rdf:Description>

</rdf:RDF>
```

Comparing XML Based B2B Integration Frameworks

Paavo Kotinurmi

Helsinki University of Technology, Software Business and Engineering institute,
P.O.Box 9600, FIN-02015 HUT, Finland
Phone (+358-50-563 0276) Fax (+358 9 451 4958)
Paavo.Kotinurmi@hut.fi

The competition among supply chains have driven companies to cut costs on business communication by automating transactions between information systems. To accomplish this they need common understanding on issues communicated and there is need for B2B standard frameworks to help in system integration between companies. The amount of different e-business standard frameworks and the differing scope causes confusion on what are the problems these standards solve, how they solve them and how do they relate to other such standards. The fast technology development around XML technologies and number of new standards efforts further complicates the e-business standards landscape.

1 Introduction

B2B frameworks enable system-to-system integration between companies. Electronic data interchange (EDI) effort originated at 1970's started e-Business. Current developments around XML-based business-to-business (B2B) standard frameworks can be seen quite closely related to EDI. They are trying to extend B2B collaboration to new areas and enabling greater number of companies to participate in system-to-system e-business. EDI have not accomplished to bring e-business to small and medium sized enterprises (SME) in more than 20 years of its existence. Still EDI based commerce is according to value the most used standard. For instance, a recent IDC study showed EDI transactions to be six times the value of all non-EDI B2B e-commerce[1].

In recent years many standards for XML-based integration have been developed. XML and related technologies are suitable for representing and validating information and it is widely embraced by major software vendors. However, the amount of different e-business content standards makes the choosing a standard to support difficult. The goal of this paper is to classify different e-business standard frameworks and draw their commonalities to Web Service technologies and issues related to defining semantics in messages.

[1] http://www.idc.com/getdoc.jhtml?containerId=ebt20020110

B2B framework analyses can be found covering frameworks Open buying on Internet (OBI)[2], eCO[3], RosettaNet[4], Commerce XML (cXML[5]) and BizTalk [1]. A similar analysis covers BizTalk, Common Business Library (CBL), cXML, Internet Open Trading Protocol (IOTP[6]), Open Applications Group Integration Specification (OAGIS[7]), Open Catalog Format (OCF[8]), and Real Estate Transaction Mark-up Language (RETML[9]) [2]. Due to nature of this area of investigation, a lot have changed. For example BizTalk framework, that is Biztalk.org site and related messaging framework, is history. Furthermore, other frameworks like eCO and OBI have not gained significant usage. Also new frameworks like ebXML[10] have been developed. In addition, xCBL and UBL are worth mentioning. The mappings between business documents defined in different e-business standards are not easy XSLT translations due to semantic differences [3]. Therefore, supporting all these standards is very demanding. When analyzing current XML-based B2B standards today, it leaves us with initiatives like RosettaNet, ebXML, OAGIS, UBL, cXML and xCBL.

Here, we classify these standards according to what they standardise. We also specify related XML technologies defining messaging or semantics specific issues. This is to clear up the relationships between the standards and to see which standards are competing and which are extending one another. Because process-oriented e-business standards like ebXML and RosettaNet have most in common with Web Services and semantic issues related we analyse them more thoroughly.

The rest of the paper is organised as follows. Chapter 2 covers an overview on different B2B standard frameworks. Chapter 3 deals with comparison of different XML based standards. Chapter 4 discusses relationships between different standards. Chapter 5 concludes the paper.

2 B2B Integration Standards Overview

Standards can be divided to standards defining schemas for exchanged messages and standards defining both the processes and the message schemas associated. RosettaNet and ebXML specify ways for describing also processes. RosettaNet differs from the others by having vertical, industry specific issues in the standard. Of the standards mentioned cXML and xCBL can be classified as standards closely related to integration to electronic marketplaces due to companies defining the standard, but the business documents defined can be used in system-to-system integration.

[2] http://www.openbuy.org/
[3] http://eco.commerce.net/
[4] http://www.rosettanet.org/
[5] http://www.cxml.org/
[6] http://www.ietf.org/html.charters/trade-charter.html
[7] http://www.openapplications.org/
[8] http://www.martsoft.com/ocp
[9] http://www.rets-wg.org
[10] http://www.ebxml.org/

2.1 EDI

EDI standards development began in the 1970's. EDI has three syntax standards. In 1974, the U.K. EDI syntax standard TDI (Trade Data Interchange) was published as a draft. The first standards of U.S. version of EDI, ANSI X12, were published in 1983. The EDIFACT was originated in 1985 to address the problems caused by different standards on both sides of the Atlantic. The EDIFACT standard development is now UN lead. The X12 syntax is the most used EDI syntax in North America [4], while EDIFACT is the dominant standard in the rest of the world.

UN/EDIFACT syntax (ISO 9735) defines the structures that are to be used for interchange of data. The syntax defines the components of the language and how they relate to each other while the grammar that controls the message design is described by the Message Design Rules. The UN/EDIFACT syntax, in its earliest versions, had three basic assumptions on the data and how they are interchanged. They were character-based data, batch data transfer and predefined, structured messages. In the latest version of the syntax, version 4, capabilities for interactive data transfer and transmission of binary data have been added together with a set of comprehensive security mechanisms.

United Nations Trade Data Interchange Directory (UNTDID)[11] defines codes and identifiers by alphabetic, numeric and alphanumeric characters for the messages interchanged. The main work within UN/EDIFACT focuses on the design of such predefined message structures.

The EDI messages transportation happens normally in Value Added Networks (VAN). These special connections have been quite expensive and therefore the EDI syntax is very compact in size, which makes them hard to read. The recent advances in EDI messaging standards, such as EDIINT specifications from Internet Engineering Task Force (IETF), have enabled companies transacting EDI over Internet rather than the pricier VAN making it affordable for smaller companies.

2.2 RosettaNet

RosettaNet is an industry driven non-profit consortium working to create, implement and promote open e-business process standards. The participating companies represent companies in information technology, electronic components and semiconductor manufacturing industries. In addition, there are companies that provide software solutions. The most important components standardised in RosettaNet are Partner Interface Processes (PIP). RosettaNet PIP defines a common public inter-company process to which company specific internal, private processes interact. Each PIP contains specification document, Document type definitions (DTD) and message guidelines documents. Message guidelines, based on RosettaNet business dictionary, introduce additional semantics to XML elements defined in DTD's. Now there are over 70 PIPs published.

RosettaNet business dictionary (RNBD) defines semantics to DTD elements and technical dictionary (RNTD) organises product descriptions into reusable atomic

[11] http://www.unece.org/cefact/

properties and relationships. Actually RosettaNet technical dictionary (RNTD) is the only clearly industry specific part of the standard. RosettaNet recommends the use of certain unique identifiers to identify companies and trade items. Data Universal Numbering System (DUNS) codes identify trading partners and their locations by providing unique identifiers. Global Trade Item Number (GTIN) is a guaranteed unique product identifier for products throughout the supply chain. RosettaNet uses Universal Standard Product and Services Classification (UN/SPSC) classification of products according to technical dictionary.

RosettaNet Implementation Framework (RNIF) standardises packaging, routing, security and transferring of RosettaNet messages over Internet. It also specifies business signal messages used in the execution of RosettaNet PIPs. It defines issues like authentication, authorisation, non-repudiation and validation to messaging. RosettaNet messaging is event-based.

2.3 ebXML

UN/CEFACT[12] (United Nations Centre for Trade Facilitation and Electronic Business) and OASIS[13] (Organisation for the Advancement of Structured Information Standards) sponsored ebXML (Electronic Business using Extensible Mark-up Language) initiative started November 1999. At that stage, the opportunities that XML offers to small and medium sized companies and to developing countries were recognised as well as need for standardizing the use of XML itself. The mission of ebXML is to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties. Using ebXML standard framework, companies have a standard method to describing the exchange of business messages, conduct trading relationships, communicate data in common terms, define, and register business processes. While ebXML offers a complete framework, companies can implement parts of the framework, without having to take it all at once.

EbXML Business Process Specification Schema (BPSS) is an XML-based specification language that formally defines public business processes that allow business partners to collaborate [5]. EbXML Core components provide the business information encoded in business documents exchanged between business partners. Core Components was the only group that has failed to publish any approved specification to date. Core components guidelines define basic components and naming conventions for them. The effort of defining common semantics has proven to be so challenging that there still are no specification available. The ebXML Registry [6] provides a set of services that enable sharing of information between interested parties. This enables business process integration between collaboration parties based on the ebXML specifications. The shared information is maintained and managed by the ebXML Registry Services [7]. Submitted content to registries may be XML schema and documents, process descriptions, Core Components, context descriptions, UML models, information about parties and even software components. Collaboration

[12] http://www.uncefact.org
[13] http://www.oasis-open.org/

Protocol Profiles (CPP) and Agreement (CPA) are XML documents that encode a company's e-business capabilities and two companies' e-business agreements [8]. The CPP defines a Party's Message-exchange capabilities and the Business Collaborations that it supports. The CPA defines the way two Parties will interact in performing the chosen Business Collaboration. The ebXML messaging services (ebMS) provide a general-purpose messaging mechanism [9]. The ebMS defines the message enveloping and header document schema. EbXML messages can be transported over communication protocols such as HTTP, SMTP, etc. The ebXML Message Service is designed to allow reliability, persistence, security and extensibility to messaging.

The further development of ebXML continues in committees coordinated by UN/CEFACT and OASIS. OASIS technical process further develops EbXML Messaging Services, Registries and Repositories, Collaborative Protocol Profile and Implementation, Interoperability and Conformance work. UN/CEFACT further develops EbXML Core Components and BPSS specification.

2.4 OAGIS

The Open Applications Group was originally comprised of eight ERP vendors including American Software, CODA Financials, Dun&BradstreetSoftware, Marcam, Oracle, PeopleSoft, SAP, and Software 2000. Nowadays there are over 30 members including customer organizations and system integration providers. OAGIS was originally not XML based [2], but recently OAGIS have fully embraced XML.

OAGIS provides example scenarios to be used as a starting point for integration. OAGIS defines Business Object Documents (BOD) as a common message architecture. BODs are the business messages or documents that are exchanged between software applications between collaborating partners. BOD uses metadata to describe what kind of message to expect. It is able to communicate status and error conditions in messaging. The BOD has concept "Noun" for common business objects like "bill of material" and concept "Verb" to name actions like "get" or "confirm" to be applied to certain business objects. For instance, "GetBillOfMaterial" and "ListBillOfMaterial" are example BODs from OAGIS 8.0 specifications that can be used in collaborations.

The OAGIS does not define messaging itself. BODs can be transported via protocols like HTTP or SOAP, or it can be used with more complex messaging frameworks like ebXML messaging service or with RosettaNet Implementation Framework 2.0.

2.5 Other B2B Standards

There are also many active message-oriented standards like UBL, xCBL and cXML, which define business messages, but do not say anything about how or when the messaging happens.

OASIS host's activity named Universal Business Library (UBL[14]) defines similar things as the group developing Core Components in UN/CEFACT. The purpose of UBL is to develop a standard library of XML business documents (purchase orders, invoices, etc.) by modifying an already existing library of XML schemas.

Marketplace vendor Commerce One has been active in developing xCBL, an open XML specification for the cross-industry exchange of business documents such as product descriptions, purchase orders, invoices, and shipping schedules. XCBL is a set of XML building blocks and a document framework that defines documents for e-commerce. The CBL preceding xCBL has been part of now quite inactive eCO framework. The xCBL schemas are starting point for the work of UBL.

Still similar standard cXML defines similar issues than xCBL and UBL. Another electronic marketplace vendor Ariba actively developed this standard. The cXML standards further development seems now quite slow. This could be a sign of maturity or slow dying away.

In addition, OASIS hosts portal XML.org for information about hundreds of industry specific XML standards. Examples of such are Chemical Industry Data Exchange (CIDX[15]) and Petroleum Industry Data Exchange (PIDX[16]) defining business messages for certain vertical industries.

2.6 Web Services

According to W3C Web Services Architecture Working Group "A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML based messages exchanged via internet-based protocols."[17]

When speaking about web services the main technologies discussed are Simple Object Access Protocol (SOAP[18]), Web Services Description Language (WSDL[19]) and Universal Description, Discovery and Integration (UDDI[20]). SOAP is a protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. It is the messaging layer for Web services. WSDL defines services as collections of network endpoints or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. UDDI provides a mechanism for clients to find web services. Using a UDDI interface, businesses can dynamically lookup as well as discover services provided by external business partners.

[14] http://www.oasis-open.org/committees/ubl/
[15] http://www.cidx.org/
[16] http://www.api.org/faeb/pidx/
[17] http://www.w3.org/TR/2002/WD-wsa-reqs-20020819
[18] http://www.w3.org/TR/SOAP/
[19] http://www.w3.org/TR/wsdl
[20] http://www.uddi.org/

Web services offer some similar business functions of ebXML - messaging, service descriptions and registries - but in more component form. They both have their respective domains. Web Services are simple and supported by important vendors like Microsoft, IBM and Sun are all promoting and make these technologies are potentially important in the future cause decision makers don't have to deal with things like CORBA vs. DCOM vs. J2EE or UNIX vs. Windows vs. Linux. Another advantage is these standards are simple which makes them easily adapted.

3 Comparison of Different Standards

In general, the new XML based e-business standard frameworks can offer companies something EDI cannot, and that is flexibility. EDI transactions generally work specifying very fixed message formats. That works fine, as long as collaborating with the same partners with stable content. However, businesses today do not always have that kind of predictability and EDI transactions are difficult to map from one industry or one country to another. Furthermore, existing EDI messages are increasingly moved over the Internet in XML format like XML/EDI, while XML messages are easier to validate and translate into the formats needed by applications at each end of the exchange [10]. Therefore, traditional EDI is often examined to define the meaning of the transferred data (semantics), and XML is used, while being more readable by humans [11].

The different XML standards differ a lot in the scope, what they standardise. The following table 1 illustrates how different standard frameworks define issues related to B2B messaging. When there is standard messaging framework like RNIF in RosettaNet or ebXML messaging service (ebMS), they define how to package and transport the messages and validate the transport headers. Web Services define similar issues, but in less detail concerning authentication, authorisation and non-repudiation issues. In schema validation, the XML business document's validation options are presented if such are defined. The acronyms used are from file extensions. XSD stands for W3C XML Schema. XDR means XML Data-Reduced, which is widely used by older Microsoft product, when XML Schema was still under development. SOX Schema format is used in xCBL, while it was developed before XML Schema to define similar rules. Because ebXML has so far no approved schemas to use, it is blank. As OAGIS 8.0 standard gives also possibility to check the validity of the message by applying validation Extensible Stylesheet Language (XSL) stylesheet to BOD message, it is also listed. Additional semantics means how additional validation guidelines are given. For instance, RosettaNet business and technical dictionaries provide more exact rules to validation of DTD elements using message guidelines documents with each PIP. EbXML provides semantic support by defining Core components guidelines, but does not say how messages are validated. Registry is listed, if the standard framework defines a standard way to locate business partners. In Web Services standards, UDDI provides this service. EbXML has defined its own registry and repository. Trading partner agreements (TPA) define contractual issues on collaboration processes like what data is exchanged and when. Only ebXML defines standard to automate these, while for instance in RosettaNet that is needed to be agreed separately. Processes refer to standard processes between collaboration parties, that define the order and timing of individual messages. RosettaNet defines standard PIP processes. EbXML has a way of describing such processes, but it has not defined so far any processes to use. Process execution refers to standard language to describe the execution of the processes. Only Web Services standards define these issues by having multiple standards for this. Web Services standardisation has alternative for process executions. Business Process Execution Language for Web Services (BPEL4WS) is an XML-based flow language that defines how business processes interact. It is said to replace the existing IBM WSFL and Microsoft XLANG efforts by combining and extending the functions of these previous foundation technologies.

Table 1. List of features in XML based e-business standards

Standard/ Feature	**RosettaNet**	**EbXML**	**OAGIS**	**Web Services**	**xCBL**	**cXML**
Process execution	--	Guidelines for describing (BPSS)	--	WSFL, XLANG, BPEL4WS	--	--
Processes	PIP	Guidelines for describing (BPSS)	--	--	--	--
TPA	Manually with each PIP	CPP/CPA	Agreed separately	--	--	--
Registry	--	Registry/ Repository	--	UDDI	--	--
Additional semantic	Message guidelines (Dictionaries)	Core components	--	--	--	--
Schema validation	PIP DTD	--	BOD DTD, XML Schema, (XSL)	--	SOX, XDR, XSD, DTD	DTD
Messaging	RNIF 1.1, 2.0	ebMS 1.0, 2.0	RNIF 2.0, ebMS	SOAP, WSDL	--	--
Packaging	RNO (1.1), MIME (2.0)	SOAP	-	SOAP	--	--
Security	S/MIME	XMLDSIG	-	SOAP/sec	--	--

In general, it is possible to combine these standards for accomplishing larger functionality. OAGIS can utilise messaging frameworks from other standards. Similarly, a company could use for instance UDDI or ebXML registry to list the RosettaNet PIPs or OAGIS BODs it supports in communication. Also RosettaNet compliant software implementation can internally use certain standard language for execution of PIP processes, but the RosettaNet standard itself does not standardise this.

It is hard to say universal guidelines on which is the most suitable standard or set of standards to use. One can say that for message validation the use of XML schemas is more suitable for B2B collaboration. DTD do not validate element content. For instance, it is important that things like dates and currencies are specified exactly. Same date could be represented as 22.10.2002, 2002-10-22 or October the 22nd, 2002. While for humans they are understandable for computers to handle them automatically the format should be exactly as expected. For instance, XML Schemas "xs:date" only accepts format according to ISO 8601, which is 2002-10-22. Still some specifications like XML schema are quite recent W3C specification and thus is some standard framework have not yet defined their business messages as XML Schemas. However, some of them will most probably do so in the future.

Also messaging infrastructure is important for business critical solutions. In the messaging frameworks, exception handling capabilities and security mechanisms are readily standardised. In addition, they help to solve issues like non-repudiation and help to define how many times the message is send, if no message acknowledgement signal is given. Standard messaging is easy to set up multiple times if the alternative is agreeing separately on such issues each time.

With standardised processes, it is easier and faster to define inter-company processes and acceptable response times to messages send. Still OAGIS kind scenarios also help to define message sequences to the inter-company message exchange.

Registries are needed, when there are lots of collaboration partners and processes, which change dynamically. So far, inter-company integration is a big effort. This means that companies have for instance, only few RosettaNet PIPs in production use and thus there is no need for registries. It is hard enough to get one common process implemented. Even using the standard integration frameworks there are many issues that need to be agreed on in setting up collaboration.

One very important consideration is of course the standard documents schemas match to company's internal applications data models. Different standards define issues in very different levels. Industry specific standards classify industry specific items in more detail than horizontal, cross-industry standards. What standard company should use depends on the collaboration partners. For instance, communication between electronic device manufacturer and transportation provider would probably use horizontal standard, while two electronic device manufacturers would probably use industry specific standard messages.

4 Standard Convergence and Relationships to Other Standards

The different standard frameworks differ in maturity. For instance, CBL was originated already in 1997 and OAGIS 1995. Web Services and ebXML specifications are new. In addition, many standards are further developed. For instance, latest OAGIS standard 8.0 differs significantly from the first versions, which did not have anything to do with XML.

In developing ebXML, the RosettaNet standard was the standard that affected most the ebXML specifications. The certain parts of ebXML like CPP, CPA, BPSS and Registries are complementary to RosettaNet standards and maybe will be used in the future as part of the RosettaNet standard. The messaging services RNIF and ebMS are at the moment competing alternatives and only real standards for business messaging. In addition, core components and processes might provide competing alternatives after they are available.

The clearest competing definitions are schemas used to describe the documents exchanged. RosettaNet PIPs, OAGIS BODs, cXML and xCBL all provide differing schemas for the documents exchanged. UBL and ebXML Core Components try to unite these by introducing common alternative, but this is still work-in-progress. Only Web Services do not standardise any issues related to schemas used in documents exchanged.

The Web Services describe partly similar things to ebXML. For instance, Web Services WSDL provides information about a service name, parameters for that service and the endpoint to invoke it. EbXML CPP not only produces this information, but also other parameters such as the role of an organization, error-handling and failure scenarios. UDDI is used to publish Web Services definition like WSDL to a global UDDI Repository. In ebXML, Registry Service Interface is used to publish an organization's CPP. EbXML Registry Services provides information about for instance business processes, business documents and business profiles. In addition, Web Services BPEL4WS language competes with ebXML BPSS in some respects while similar issues can be described. However, ones again BPSS is meant for more complex business processes interoperability, while Web Services standards define application level interoperability issues. Web Services and ebXML are solving fundamentally different kind of problems although similar issues are standardised and therefore they do not directly compete. However, as the Web Services specifications do not mention ebXML at all, it may hinder the ebXML adoption to get the critical mass it needs.

RosettaNet and ebXML frameworks define basic messaging concepts like those that are standardised in Web Services. In RosettaNet and ebXML trading partner agreements define the security and non-repudiation issues between companies. Web Services define only implementation issues. When doing business electronically these issues have to be in place when there are significant economic issues involved.

5 Conclusions

EDI, RosettaNet, ebXML and Web services are a continuum rather than three distinct alternatives. EDI provides a fixed, predictable message format, which with high volumes and stable business processes has proven itself. RosettaNet and ebXML have the messaging features of EDI, plus a larger framework of functions combining business process models, trading partner agreements, and semantic interoperability. Web services can help future implementations by providing easier ways to combine applications. The first solutions with Web services in B2B collaboration are probably just internal integration of company systems. However, as the standards mature there might be tools also for B2B messaging that can be applied in standard frameworks.

Problems with different standards and classifications are discussed and use of semantic web techniques have been proposed. For example, the use of Semantic Web techniques like RDFT mapping meta-ontology has been developed for transformations between different standards [12]. So far, Semantic Web technologies have not affected the making of standard frameworks themselves. However, many problems in the frameworks are related to defining common meaning to information exchanged and universal-mapping rules between differing standards could help many.

The adoption of e-business standard frameworks is up to companies and what they are ready to implement in their operational information systems. These systems are ultimately those, which provide the content in e-business collaborations. There is lack of scientific research on applying standard frameworks in real life problems and presenting cases on supporting different standards. This would help giving guidelines on which standards or set of standards to select. Finally, cooperation between existing standards should be encouraged. The number of related standards is big and they poorly define their own relationships to other standards. This makes the selection of the standards difficult and that can slow down the adoption of B2B integration solutions overall.

References

1. Shim S.S.X., Pendyala V.S., Sundaram M. and Gao J.Z.; E-Commerce Frameworks, IEEE Computer 33,10: (2000) 40-47
2. Haifei Li; XML and Industrial Standards for Electronic Commerce. Knowledge and Information Systems, Volume 2, Issue 4, (2000) 487-497
3. Omelayenko B. and Fensel D.; An Analysis of Integration Problems of XML-Based Catalogs for B2B Electronic Commerce, In: Proceedings of the 9th IFIP 2.6 Working Conference on Database Semantics (DS-9), Hong-Kong, April 25-28, (2001) 232-246
4. Salminen, A.; EDIFACT for business computers: has it succeeded?, StandardView, ACM Perspectives on Standardization, 3 (1), (1995) 33-42
5. ebXML Business Process Specification Schema v1.01, (2001), http://www.ebxml.org/specs/ebBPSS.pdf
6. ebXML Registry Information Model v2.0, (2001), http://www.ebxml.org/specs/ebrim2.pdf
7. ebXML Registry Services Specification v2.0, (2001), http://www.ebxml.org/specs/ebrs2.pdf
8. ebXML Collaboration-Protocol Profile and Agreement Specification v1.0, (2001), http://www.ebxml.org/specs/ebCCP.pdf

9. Jones, I., et al; Message Service Specification v2.0, (2002), http://www.ebxml.org/specs/ebMS2.pdf
10. Glushko, R. J.; Tenenbaum, J. M.; Meltzer, B., An XML-Framework for Agent-based Ecommerce , Communications of the ACM, Vol.42, No.3, (1999) 106-114
11. Hasselbring, W. and Weigand, H., Languages for electronic business communication: state of the art, Industrial Management & Data Systems,101/5, (2001) 217-226
12. Omelayenko B., Fensel, D., Bussler, C.; Mapping Technology for business Integration, In Proceedings of the Fifteenth International FLAIRS Conference (FLAIRS-2002), Pensacola, FL, May 14-16, (2002) 419-424

A Journalist's Tool for Writing and Retrieving News Stories

Martin Fluch, Greger Lindén, and Andrei Popescu

Department of Computer Science,
P.O. Box 26 (Teollisuuskatu 23),
FIN-00014 University of Helsinki,
Finland,
{Martin.Fluch,Greger.Linden,Andrei.Popescu}@cs.helsinki.fi,
WWW home page: http://www.cs.helsinki.fi/research/doremi/

Abstract. We have developed a working environment for a journalist that consists of a news story editor (an XML editor) and a retrieval interface to a collection of news stories. The editor supports a subset of the News Industry Text Format standard specification (the NITF DTD) for news stories. When writing a story, the user is guided through the possible structures. Stories may contain metadata such as author, title, keywords, categorization, creation and release dates, relations to other stories, etc. A prototype of the editor has been built in Java.
The editor works in close co-operation with a retrieval interface to a news story collection. The interface allows for searching the collection based on search words, similarity to retrieved and selected stories, or similarity to the news story being written in the editor. The retrieval results may be refined or augmented with new results. The interface also shows the explicit grouping of related news stories in the interface. The retrieval interface uses a linguistic tool for stemming words (both in Finnish and English).
The editor and retrieval system have been enhanced with support for different news story types such as events, comments, fact sheets, and background information, as well as with support for news subject categorization suggested by the International Press Telecommunications Council (IPTC). The system also includes online retrieval based on the text already entered in a new story in the editor. When the user writes a story, the system automatically finds related news stories of a requested kind.

1 Introduction and system overview

Knowledge workers constantly run into work tasks were they need to collect, filter and refine data. A typical example of such a knowledge worker is a news reporter. News agencies and services receive stories, one-liners, and other data from heterogeneous sources. A first step in data refinement is therefore to unify the data into one format, to have it categorized and to add describing metadata. An important part of the work is to track ongoing events and also to pick out new interesting pieces of information. On the other hand, the same information may be received from several news feeds at the same time and replicated data must then be filtered out.

In the ideal working environment many of these subtasks are performed automatically. Received information may be automatically categorized, descriptive phrases and terms may be automatically found and marked as metadata, and repeated stories or part of stories may be filtered out automatically. In the ideal working environment, the data is also automatically transformed into the same format; or, even better, produced in the first case in such a format.

In the Knowledge Worker's Workstation project (TYTTI)[1], we have built a prototype for news story writing and management called *Itaxad* ("Interface to an XML article database"). Naturally, we have chosen XML as the story representation format. Our system consists of an XML editor, especially designed to support news stories, and a retrieval engine for intelligent retrieval of news stories from a database. The system is designed for an end user, a news reporter, who can easily find, check and even reuse previous news stories when writing a new story. The stories can also be categorized, both according to user-given keywords and grouping, and to standardized news categories, to ease later retrieval. Retrieval may be performed based on keywords or metadata. The system also provides automatic online retrieval, here called *live search*; old stories are automatically retrieved based on the text that the user is currently writing in the editor.

In the implementation we have used both the standard News Industry Text Format (NITF) DTD [NITF], an XML DTD for news stories and a categorization of news called the Subject Reference System (SUBJECTS) and provided by the International Press Telecommunications Council (IPTC) [IPTC]. Both specifications have become de facto standards in the media world [All01]. The format and the categories are explained further below.

As XML technology becomes more wide-spread and accepted, more and more XML supporting software is being developed. There are, of course, many general-purpose XML editors available, see, e.g., the software section at the Cover pages [Oasis]. Also NITF, is supported in some specialized tools, like e.g., in news feed integration [Sey99], content management [Net] and databases [CoW01]. Automatic online retrieval, also called just-in-time retrieval, has been used before, e.g. in an e-mail environment [RhM00].

We have opted for building our own editor and retrieval prototype for research reasons. In this case, we have better control of the software, integration is easier, and we are able to use the system as a platform to which we can add further enhancements, such as event detection and tracking (see Section 7).

In the following paper we explain how the Itaxad system is used and how it was implemented. We present the NITF DTD and the IPTC categories in Sec-

[1] The Knowledge Worker's Workstation (TYTTI, 2000-2003) project of the doremi research group (http://www.cs.helsinki.fi/research/doremi/), is a USIX project funded by Tekes, the National Technology Agency of Finland, and the following enterprises: Vaisala, Alma Media, Nokia, Sanoma-WSOY and Lingsoft.

tion 2. In Sections 3 to 5 we explain how to write, reuse, save, and retrieve a story. In Section 6, we briefly explain how the system was implemented. The system is still being developed and in Section 7 we present what future enhancements we have in mind.

2 Story structure and topic

The News Industry Text Format (NITF, [NITF]) is an XML vocabulary for defining the content and structure of news stories. It also includes a whole variety of additional information (metadata) that makes news stories more searchable. NITF was developed by the International Press Telecommunications Council (IPTC, [IPTC]) and is a public standard that is becoming more and more supported.

NITF is the recommended format for text in the NewsML standard [IPTC], an XML encoding for creation, transfer and delivery of news. When NITF concerns text news, NewsML is media independent and enables publishers to express relationships among different kinds of news components (multimedia).

A NITF document[2] consists of two parts, the document head, which holds metadata valid for the whole document, and the document body, which contains the news story itself. A NITF document serves as a source for creating transformed versions that are then published, e.g., as a HTML document, or in printed form in a news paper or a magazine. The NITF vocabulary is very large and allows the creator be very descriptive.

The document head includes the story title, publication data and data about the document itself (like the date of issue, copyright information, special keyword lists, document id, etc.), and categorization information. Most of the information is metadata and will not be shown in the published story. The metadata makes the story easier to find in large collections and can be used, e.g., for indexing in a database.

The document body contains the actual content of the news story. Most of the content will show up in the published story. The body is further divided into a head, a content and an end part. An example (found on the NITF home page [NITF]) of a very simple NITF document is shown in Figure 1.

The body head may contain, among others, the headline of the story, by- and datelines, and an abstract. The body content contains paragraphs, lists, tables, media and other containers. The body end holds a tagline and/or bibliography.

Organizations, persons, locations, etc. may be denoted by special markup. In the example in Figure 1, Microsoft has been marked as an organization and some additional information has been added such as a unique identifier for the organization. Other special markup allows hyperlinks to be included.

[2] In the following a *NITF document* is also referred to as an *article* or a *story*.

```
<nitf>
    <head> ... </head>
    <body>
        <body.head>
            <hedline><hl1>This is the main headline</hl1></hedline>
            <byline>By Joseph Q. Reporter</byline>
        </body.head>
        <body.content>
            <p>Today, <org value="MSFT" idsrc="NASD">Microsoft</org>
            announced the release of...</p>
            <p>This is the content of the second
            paragraph of the story.</p>
        </body.content>
    </body>
</nitf>
```

Fig. 1. A simple example of a NITF document [NITF].

3 Writing and reusing stories

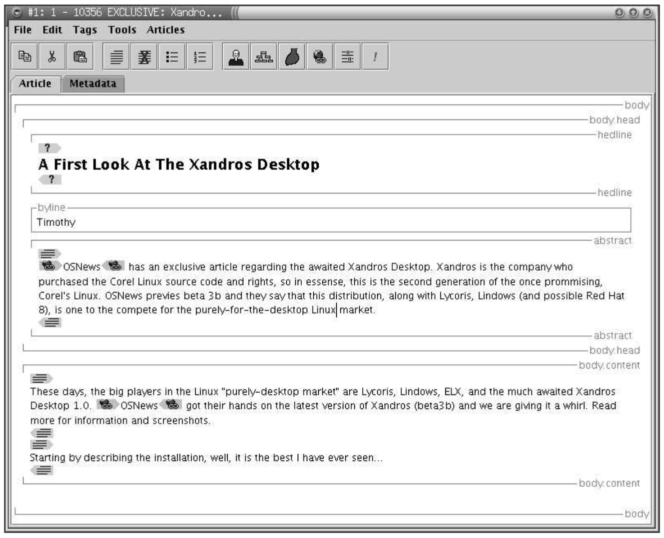
The journalist uses a structured editor for writing news stories. Using the editor is comparable to using a regular word processing program. The editor guides the user through the possible structures of the story according to the NITF DTD. The guidance also assures that the result is a valid NITF document, but to a certain extent, the editor also allows the use of other DTDs.

The XML document is displayed as naturally as possible. The story content with all the headlines, paragraphs, pictures and other elements are displayed as continuous text with appropriate marks to represent the XML markup. Metadata is (only) displayed separately in a different window.

When the user starts writing a new story, a minimal valid XML document is created by the system. The user modifies the document by inserting text, creating tags and marking up text passages. Copy, cut and past operations are supported in the usual manner, as long as the resulting document remains valid under the changes. An example of the editor window is shown in Figure 2.

A title, a byline and some paragraphs have been included. The text elements are all surrounded by XML markup, either in the form of small iconic marks (for inline markup) or in the form of window-wide lines where the element name is mentioned at the end (for block markup). Possible inline markup is chosen from the icon bar in the top of the editor window. Block markup can be selected from a scroll down menu, but when possible, new block markup is inserted automatically. For example, when the user hits return at the end of a paragraph, a new paragraph is inserted. Attributes, e.g. the address of the hyperlink, are inserted in a separate dialog box.

Metadata concerning the entire document is given in a separate window. The window is generated dynamically based on the DTD chosen for the document. Metadata allowed in the present version includes document title, date of issue,


#1: 1 - 10356 EXCLUSIVE: Xandro...
File Edit Tags Tools Articles
Article
Metadata
body
body.head
hedline
byline
abstract
body.content

A First Look At The Xandros Desktop

Timothy

OSNews has an exclusive article regarding the awaited Xandros Desktop. Xandros is the company who purchased the Corel Linux source code and rights, so in essense, this is the second generation of the once prommising, Corel's Linux. OSNews previes beta 3b and they say that this distribution, along with Lycoris, Lindows (and possible Red Hat 8), is one to the compete for the purely-for-the-desktop Linux market.

These days, the big players in the Linux "purely-desktop market" are Lycoris, Lindows, ELX, and the much awaited Xandros Desktop 1.0. OSNews got their hands on the latest version of Xandros (beta3b) and we are giving it a whirl. Read more for information and screenshots.

Starting by describing the installation, well, it is the best I have ever seen...

Fig. 2. An example of the news editor window.

release and expiration as well as keywords, and IPTC and story type categories. Available IPTC categories and story types can be chosen from a scrollable list and just dragged and dropped in the correct field. The metadata window is shown in Figure 3.

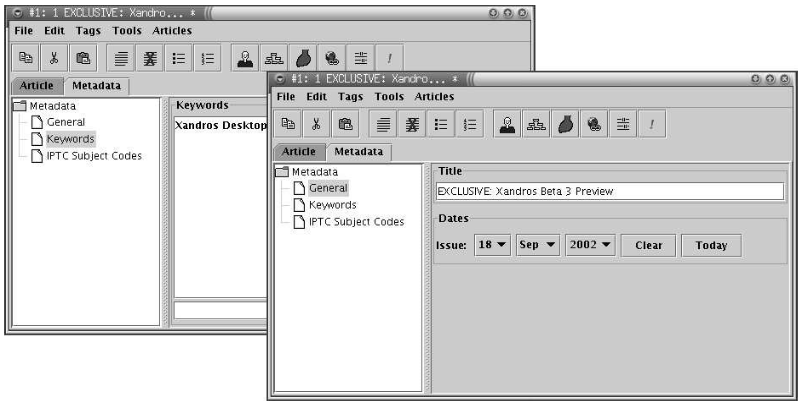
Stories retrieved from the database can be opened in the editor and reused, either as such or by copying parts to another editor window. When a story is opened, the system checks the document for validity against its DTD and then presents the contents and the metadata in separate parts of the editor as explained above.

4 Saving a story

The stories are saved in a database together with the information about the DTD that was used to create the document. Either a previous version of the story is replaced or a new entry in the database is created.

When the database receives the storage request, it checks the XML document and extracts some information for further use for the retrieval of documents. The database stores both a copy of the XML document, a processed version of the plaintext and the extracted metadata. The system extracts the title of the document, the author(s), and the dates of publishing, issue and expiration. All metadata is optional, but when present it is stored in a separate database table for the purpose of retrieval. Including, e.g., a title or author information is completely up to the writer.

The database has three different systems of grouping and categorizing the stored stories. Two of them are based on information stored within the XML



Keywords: Xandros Desktop

Title: EXCLUSIVE: Xandros Beta 3 Preview

Dates — Issue: 18 Sep 2002

Fig. 3. News story metadata is given in a separate window.

document (IPTC categories and story types), and the third one is done by explicit grouping of the stories (user-defined groups). All categorization is optional and performed explicitly by the user.

The IPTC grouping is a three level categorization system including predefined categories according to subject, matter and detail [IPTC]. The categorization system is very large and tries to cover all possible news story topics. Some examples of the seventeen available subjects are *Arts, culture and environment*, *Crime, law and justice*, and *Disaster and accident*. Subjects are divided into matters and matters into details. Examples of matters are *Corporate crime* with details like *Fraud* and *Embezzlement*, and *Transport accident* with details like *Road accident* and *Railway accident*. The DTD permits the document to belong to zero, one or several categories. A document in a certain category automatically belongs to all super-categories of the category.

Story types are, e.g., head stories, comments, background, fact sheets, etc. and are explicitly defined by the user (no predefined types). Depending on the DTD used, also this information may be part of the XML document. When the story is saved, the database is searched for the given story types. If no match is found, a new story type group, with the story as a single member, is created. A story may belong to zero, one or several story types depending on the DTD used.

The system also supports explicit linking by the user of stories in so called user-defined groups. A story may belong to any arbitrary number of groups. The information for this grouping is solely stored in the database, and not as in the previous two cases in the XML document.

When the story is saved in the database, its plaintext without markup is extracted and saved to allow search based on the textual content. For each language, we have built a stop-list of words that have extremely low discriminating value. These words can be either very common terms (such as prepositions, conjunctions, certain pronouns, etc.) or terms that occur very frequently throughout our collection. The lists are used to filter the stop-words out of the saving process.

The system is "bilingual" and knows how to process stories in both English and Finnish. If the language has not been explicitly marked in the story, the system is able to deduce which language the story has been written in by comparing it to stop lists in different languages. The plaintext is also stemmed, i.e., only base forms, not inflections, of words are stored. Stemming is of course language dependent and the appropriate stemmer is chosen according to the language (Finnish or English) that the system decides that the story is written in.

5 Finding stories

The Itaxad system allows the user to search through the news story collection. The search engine is designed in such a way that it takes advantage of the particularities of our repository. It benefits from the XML structure of the documents and from the various types of groupings in the collection by allowing the user to formulate complex queries and to visualize, browse and interact with the results in an effective manner.

The system is used in two major ways. The user can retrieve stories through the Itaxad Query Interface which provides an interface for building complex queries. The Live search module automatically finds news stories related to the story being written in the editor.

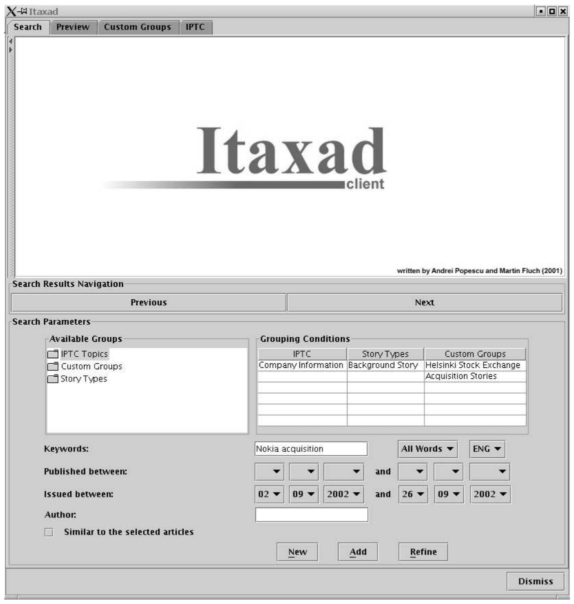
5.1 Itaxad Query Interface

The query interface (Figure 4) is a tool for building search queries and for visualizing the results. It also offers various other facilities: previewing retrieved stories, browsing the IPTC categories and managing user-defined groups.

The interface allows the user to build two types of queries. In *keyword queries* the system retrieves stories relevant to the user-defined query. In *similarity queries* the system retrieves stories similar to the ones indicated by the user. Both types of queries can be enhanced with a rich set of constraints based on the existing groupings of documents in the collection, as well as on the information provided by the XML metadata present in each story. For instance, one might want to restrict the results of a query by asking only for those stories that

– provide a *background story* for a certain event
– discuss a topic related to *company information*
– belong either to the user defined group *Helsinki Stock Exchange* or *Acquisition Stories*
– are issued between *02.09.2002* and *26.09.2002*

The first three conditions belong to the grouping constraints and can be easily specified by dragging the desired groups from the list of available ones and dropping them on the grouping conditions table. The last one is a metadata constraint. Figure 4 shows the interface corresponding to the above conditions.



Grouping Conditions

IPTC	Story Types	Custom Groups
Company Information	Background Story	Helsinki Stock Exchange
		Acquisition Stories

Keywords: Nokia acquisition All Words ENG

Published between: and

Issued between: 02 09 2002 and 26 09 2002

Author:

Similar to the selected articles

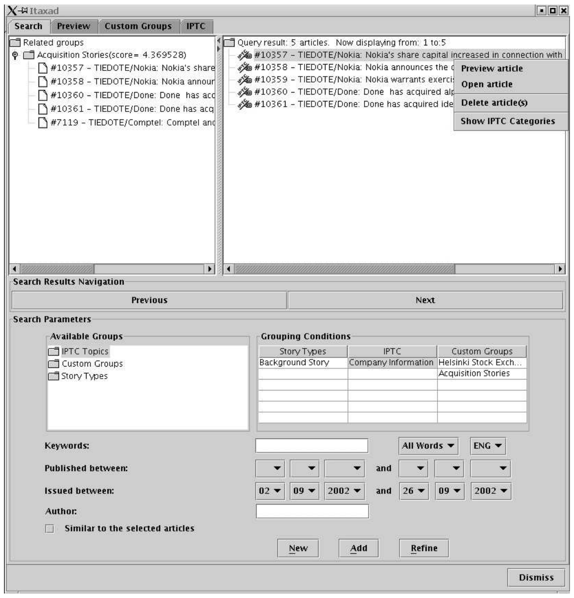
Fig. 4. Itaxad Query Interface.

For any query, the user has the option to determine in which way the system will deal with the results. She can display separately the results of the latest query sorted by their relevance scores, or she can combine the latest results with the ones from the previous queries. The similarity scores (and, consequently, the display order) will be modified such that they reflect the relevance to the sequence of queries run so far. Or, she may retrieve the relevant stories only from the list of results of the previous query.

With these options users can define searching strategies similar to *relevance feedback* [Sal89,SaB97]. They can start with a tentative keyword query, then they can continue by choosing certain interesting stories, finding others similar to them and then refining the results by filtering them through new keyword queries.

Keyword Queries. Keyword queries are formulated using natural language. The system returns the list of stories relevant to the vector of terms extracted from the user query. So far we have implemented two types of keyword queries, vector queries [Sal89] consisting of terms without Boolean operators and Boolean AND-queries [Sal89].

In order to improve retrieval, we are using a linguistic tool [Con02] for stemming words as part of the query preprocessing step. Considering the specific of our repository, we are giving the possibility for the user to formulate queries in English and Finnish. Depending on the chosen language, the query is stemmed and stop-words are removed.



Related groups
Acquisition Stories(score= 4.369528)
#10357 – TIEDOTE/Nokia: Nokia's share
#10358 – TIEDOTE/Nokia: Nokia announ
#10360 – TIEDOTE/Done: Done has acc
#10361 – TIEDOTE/Done: Done has acq
#7119 – TIEDOTE/Comptel: Comptel anc

Query result: 5 articles. Now displaying from: 1 to:5
#10357 – TIEDOTE/Nokia: Nokia's share capital increased in connection with
#10358 – TIEDOTE/Nokia: Nokia announces the
#10359 – TIEDOTE/Nokia: Nokia warrants exerci
#10360 – TIEDOTE/Done: Done has acquired alp
#10361 – TIEDOTE/Done: Done has acquired ide

Available Groups
IPTC Topics
Custom Groups
Story Types

Grouping Conditions

Story Types	IPTC	Custom Groups
Background Story	Company Information	Helsinki Stock Exch...
		Acquisition Stories

Keywords: All Words ENG
Published between: and
Issued between: 02 09 2002 and 26 09 2002
Author:
Similar to the selected articles

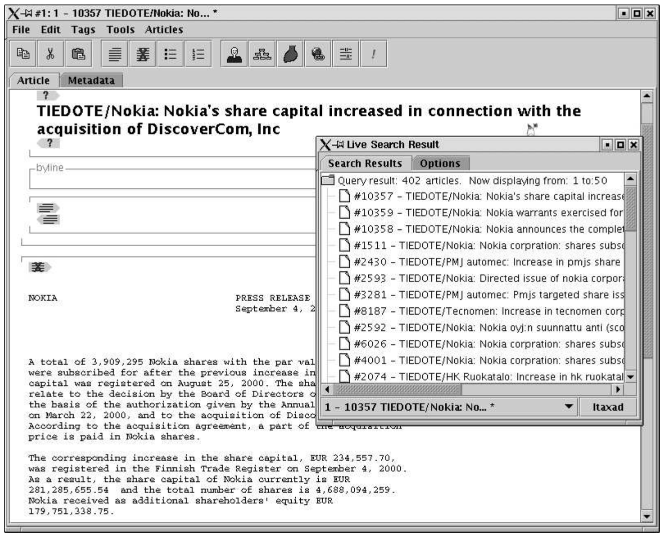
Fig. 5. Browsing the result.

Similarity queries. Similarity queries are used to retrieve stories based on their similarity with other user selected documents. The system builds an expanded query from the plaintext of the selected documents and runs that query against the entire collection. The query construction attempts to gather the most important terms (with the biggest discriminating value) from each selected story.

Here the preprocessing step is not necessary since the query is built directly from the term vectors of documents in the collection. These terms are already stemmed and filtered of stop-words during the saving process (see Section 4).

Browsing the results. One of the most important features of the query interface is to allow a very intuitive and flexible interaction with the retrieved stories. We have implemented several functions that ensure easy browsing of the results, grouping visualization, opening and previewing stories. Figure 5 depicts the interface ready for browsing.

All stories in the result list that belong to the various levels of grouping are marked accordingly. For example, in Figure 5, all stories belong to certain user-defined groups and are also part of IPTC categories. Users can easily access the user-defined groups: in our example, the first story in the result list is a part of the "Acquisition Stories" group. The other stories in this group can be browsed in the left panel. It is possible for a document to belong to more than one group. In this case, the groups will be sorted by the relevance score which is calculated with regard to the user query.



TIEDOTE/Nokia: Nokia's share capital increased in connection with the acquisition of DiscoverCom, Inc

```
NOKIA                                        PRESS RELEASE
                                             September 4, 2


A total of 3,909,295 Nokia shares with the par val
were subscribed for after the previous increase in
capital was registered on August 25, 2000. The sha
relate to the decision by the Board of Directors o
the basis of the authorization given by the Annual
on March 22, 2000, and to the acquisition of Disco
According to the acquisition agreement, a part of the acquisition
price is paid in Nokia shares.

The corresponding increase in the share capital, EUR 234,557.70,
was registered in the Finnish Trade Register on September 4, 2000.
As a result, the share capital of Nokia currently is EUR
281,285,655.54  and the total number of shares is 4,688,094,259.
Nokia received as additional shareholders' equity EUR
179,751,338.75.
```

```
Query result: 402 articles.  Now displaying from: 1 to:50
  #10357 – TIEDOTE/Nokia: Nokia's share capital increase
  #10359 – TIEDOTE/Nokia: Nokia warrants exercised for
  #10358 – TIEDOTE/Nokia: Nokia announces the complet
  #1511 – TIEDOTE/Nokia: Nokia corpration: shares subs
  #2430 – TIEDOTE/PMJ automec: Increase in pmjs share
  #2593 – TIEDOTE/Nokia: Directed issue of nokia corpor
  #3281 – TIEDOTE/PMJ automec: Pmjs targeted share iss
  #8187 – TIEDOTE/Tecnomen: Increase in tecnomen corp
  #2592 – TIEDOTE/Nokia: Nokia oyj:n suunnattu anti (sco
  #6026 – TIEDOTE/Nokia: Nokia corpration: shares subs
  #4001 – TIEDOTE/Nokia: Nokia corpration: shares subs
  #2074 – TIEDOTE/HK Ruokatalo: Increase in hk ruokatal
```

Fig. 6. Live search.

IPTC categories as well as the open, preview and delete functions can be easily accessed through context menus. In the case a user needs to browse the IPTC category that contains one of the stories in the result list, she will see only those stories in that category that are relevant to the her query.

Other facilities. The Itaxad interface provides some other tools that can be used to manage the user-defined groups of documents or to browse the IPTC categories. Users can define groups for organizing documents such that they fulfill various retrieval purposes. For instance, one may explicitly group related stories or all news stories written by a certain author. The stories written by one author may discuss different topics, hence they might be difficult to retrieve with good recall and might need a significant number of queries to be run. However, if they are part of a user-defined group, they can be all instantly browsed and retrieved. IPTC categories can also be browsed directly, independently of any query.

5.2 Live search

The Live search tool is designed to perform similarity searches simultaneously with news story editing. Users can enable the Live search tool while they are writing a story. The content of the story, including metadata, is used to generate queries with a customizable frequency. The result is that users can instantly see what other similar documents are already present in the collection. Figure 6 presents a Live search session.

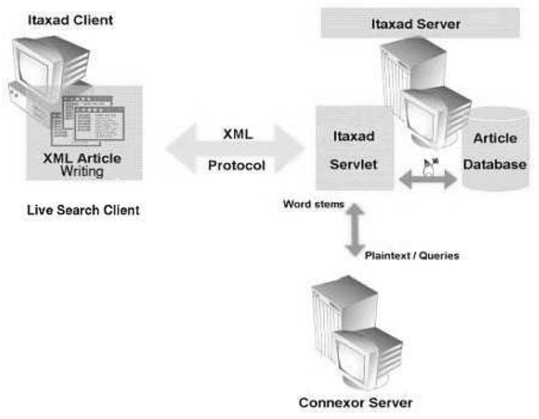


Fig. 7. Itaxad Architecture.

The list of retrieved similar stories can be forwarded to the Itaxad Query Interface, offering the user the possibility to interact with the results just as if they were generated in the interface. Live searches are performed in the background introducing very little overhead into a normal editing session. For further implementation details, see Section 6.

6 Technical implementation

The system is mainly implemented in two parts: the editor and the Itaxad retrieval system. The first prototypes were implemented in student projects [Dav01,Del01,HKL01], and has since been further developed within the TYTTI project. The editor is an ordinary Java Swing [JFC] application. Essentially the `Document`, `View` and `JTextComponent` classes of the `javax.swing.text` package are extended to display the document to the user. The Itaxad Retrieval System is essentially a distributed application. It consists of a collection of servers and clients that interoperate using an XML-based protocol. The architecture of the system can be observed in Figure 7.

The Itaxad Query Interface and the Live search tool are, in fact, clients providing a user interface for query building. They are written in Java and their role is to transform the user input into an XML request that the Itaxad server understands.

Beside the user interface, the Live search client contains a module for constructing queries from the story being edited. This module introduces a preprocessing step which is necessary considering that the source of the query construction consists of newly introduced text. Therefore, we have developed an algorithm for automatically detecting the language of the story (either English or Finnish). Once the language has been identified, the correct stemming tool is used to get the base form of the words in the story, the terms on the stop-list are eliminated and the query can finally be built. The Live search client also contains a module for communicating with the Itaxad server. This module is responsible for sending the query to the server and for forwarding the results to the user interface.

Live search itself is built following the "Producer Consumer" threading model [ACD00] with the query constructor module as the producer, the communication module as the consumer and the query itself as the shared buffer.

The collection of documents is stored in a MySQL [MySQL] database. A fulltext index is created for each of the searchable attributes.

The Itaxad server is responsible for processing the XML requests and returning the results to the clients. It is written as a collection of Java servlets [Jav]. A typical query processing session is handled by the servlets roughly following the steps below:

1. Preprocess the query. This step is actually not necessary for queries originating from the Live search tool.
2. Run an SQL query against the database to retrieve all relevant stories (in the form of title, unique ID and relevance score) and all their corresponding groups.
3. Process the results:
   (a) Assign each story to the groups that contain it.
   (b) Run an SQL query to retrieve the rest of stories in each group.
   (c) Calculate group similarities.
   (d) Save the group structure and the stories in a servlet session.
   (e) Return the first 50 stories to the client (we have experimentally determined that a cut-off of 50 is best for our system).

It can be observed that each user query is transformed in exactly two SQL queries. We have striven to keep the number of SQL queries as low as possible in our implementation because we have determined that the DBMS operations are the most resource expensive in the whole system.

7 Conclusion and future work

The first development phase of the Itaxad system has now been completed. The system provides the user with an interface for writing and saving stories as well as efficient retrieval of old stories from a database. The database contains at the moment about 10 000 news stories in Finnish and English and some even in Swedish (in some stories, the languages are mixed). Despite the large number of stories, retrieval is fairly efficient: in average it takes around 240 ms to execute a keyword query and about 1.2 s for similarity queries.

Two kinds of evaluation should be considered. How *effective* is the story retrieval, i.e., does the system find the most relevant stories? And, does the system give *relevant support* in writing and working with news? Is the user satisfied? During the last six months of the projects, we hope to find some answers to these questions, possibly with help of a graduate student who is writing her thesis on the subject of supporting journalist work with information technology [Kou03].

One of our goals is to provide detection of new and tracking of ongoing stories in a news feed. During the next phase of the development, we will add automatic *event detection and tracking* (EDT) to the system. The first tests seem promising, but some more work and inspiration is needed to reach useful results.

References

[All01] Tony Allday. *NewsML – enabling standards-led revolution in news publishing? XML Technologies in Broadcasting.* EBU Technical Review, June 2001. http://www.ebu.ch/trev_287-allday.pdf

[ACD00] Irina Athanasiu, Bogdan Constantinescu, Octavian Dragoi, Irina Popovici: *Limbajul Java: O Perspectiva Pragmatica* Computer Libris Agora 2000

[Con02] Connexor Machinese Syntax. http://www.connexor.com/m_syntax.html

[CoW01] Content Wire/Syndication. *Digital asset management system to provide massive XML-based news story archive.* November 2001. http://www.content-wire.com/Online/Syndication.cfm

[Dav01] Henri David: *Itaxad:S – An interface to an XML article database: storage part.* Technical documentation, June 2001.

[Del01] Antoine Delaunay: *Itaxad: R – An interface to an XML article database: retrieval part.* Technical documentation, June 2001.

[HKL01] Petri Hellemaa, Antti Koskinen, Lari Laine, Jarno Saarto, Eija Teitto, Harri Tuuloskoski. *JouTo – A journalist's tool.* Technical documentation, Version 1.0. April, 2001.

[IPTC] International Press Telecommunications Council. The Subject reference System. http://www.iptc.org

[Jav] Java servlet technologies. http://java.sun.com/products/servlet

[JFC] Java Foundation Classes. http://java.sun.com/products/jfc

[Kou03] Heli Koukku. *Toimittajan työn helpottaminen automaattisella tietojenkäsittelyllä.* Master's Thesis, to appear.

[MySQL] The homepage of MySQL. http://www.mysql.com/

[Net] The NetContent Content Management System. http://www.newsml.de

[NITF] News Industry Text Format. http://www.nitf.org

[Oasis] Oasis. Cover pages. http://www.coverpages.org

[RhM00] B. J. Rhodes and P. Maes. *Just-in-time information retrieval agents.* IBM Systems Journal **39**, *3&4* (2000), 685-704.

[Sal89] Salton, G., *Automatic text processing: the transformation, analysis, and retrieval of information by computer.* Addison-Wesley, Reading (MA) (1989).

[SaB97] Salton, G., and Buckley, C., *Improving Retrieval Performance by Relevance Feedback,* in: Spark Jones, K., and Willett, P., ed., Readings in Information Retrieval. Can Francisco, CA: Morgan Kauffman, 1997.

[Sey99] *WavePhore aims to simplify integration of news feeds for Web publishers.* The Seybold Report on Internet publishing, **3**, *9* (1999). http://www.seyboldreports.com/SRIP/free/0309/news1.htm

Perinnedatan automaattinen muotoilu XML-tekniikoin

Merja Ek, Heli Hakkarainen, Pekka Kilpeläinen, Tommi Penttinen

Kuopion yliopisto, Tietojenkäsittelytieteen ja sovelletun matematiikan laitos
PL 1627 (Microkatu 1 D), 70211 Kuopio
{merja.ek, heli.hakkarainen, pekka.kilpelainen, tommi.penttinen}@cs.uku.fi

Tiivistelmä. XML-tekniikat tukevat dokumenttitiedon moninaiskäyttöä: eri esitysmuodot tulostusta ja digitaalista mediaa varten voidaan tuottaa samasta XML-muodosta. XML-tekniikoiden soveltaminen edellyttää perinnedatan muuntamista XML-muotoon. Tähän on olemassa menetelmiä, mutta vastaavaa muotoilumääritysten muuntamista XML-muotoon ei juurikaan ole tarkasteltu. Esimerkiksi XSL on voimakas XML-dokumenttien muotoilukieli, mutta kymmenien erilaisten elementtityyppien käsittely vaatii silläkin kymmenien vastaavien muotoilusääntöjen kirjoittamisen. Mikäli perinnedataan on sovellettu eksakteja muotoilumäärityksiä, datan XML-muodolle olisi hyvä pystyä tuottamaan niiden perusteella muotoilu mahdollisimman automaattisesti. Tarkastelemme XML-tekniikoiden sovellusesimerkkinä muotoiltuna tulostettavan riviperustaisen aineiston muuntamista XML-muotoon. Esitämme tätä varten kehittämämme arkkitehtuurin, joka muuntaa ja muotoilee annetun aineiston sen rakennetta ja muotoilua kuvaavan kontrollitiedoston ohjaamana automaattisesti. Arkkitehtuuri perustuu käsiteltävien tiedostojen XML-muotoon konvertointiin kehittämällämme ”XML-käärintäkielellä” nimeltä XW. Kerromme myös järjestelmän toteuttamisesta saaduista XML-tekniikoiden soveltamiskokemuksista.

1 Johdanto

XML-tekniikat tukevat dokumenttitiedon moninaiskäyttöä: eri esitysmuodot esimerkiksi tulostusta, verkkolevitystä ja CD-ROM-versioita varten voidaan tuottaa samasta XML-muodosta. Perinnedatan ja sen muotoilumääritysten muuntaminen XML-muotoon on kuitenkin usein käytännössä työlästä. Datan XML-muotoon muuntamista voi helpottaa soveltamalla korkeantasoista, tarkoitusta varten kehitettyä kieltä. ”XML-käärintäkieli” XW [1, 2] on XML-perustainen kuvauskieli määrämuotoisen datan muuntamiseksi XML-muotoon. XW-kääre on XML-muotoinen hierarkkinen malli lähtöaineiston rakenteelle ja samalla sitä noudattavalle tulosaineiston rakenteelle. Lähtöaineiston osien peräkkäisyyttä kuvataan peräkkäisillä elementeillä ja sisäkkäisyyttä sisäkkäisillä elementeillä. Osien vaihtoehtoisuus, valinnaisuus ja toistuvuus sekä osia erottavat erotinmerkkijonot kuvataan tämän mallin osana omilla XW-elementeillään ja -attribuuteillaan.

Datan konvertointi XML-muotoon on yleisesti tunnistettu ja runsaasti tarkasteltu ongelma [3]. Sen sijaan vastaavaa olemassa olevien muotoilumääritysten konvertoimista XML-muotoon soveltuvaksi on tarkasteltu vähemmän. Vaikka esimerkiksi XSL [4] on voimakas XML-dokumenttien muotoilukieli, kymmenien erilaisten elementtien muotoileminen vaatii silläkin kymmenien vastaavien muotoilusääntöjen kirjoittamisen. Mikäli perinnedataan on sovellettu eksakteja muotoilumäärityksiä, myös vastaavien XML-dokumenttien muotoilu pitäisi pystyä tuottamaan niistä mahdollisimman automaattisesti.

Luvussa 2 kuvaamme esimerkkinä tarkastelemamme aineiston, riviperustaisen laskuaineiston sekä sitä kuvaavan kontrollitiedoston, joka kuvaa aineiston rakenteen ja muotoilun. Aineiston XML-konversio ja muotoilu voidaan automatisoida kontrollitiedoston ohjaamana. Luvussa 3 kuvailemme tämän mahdollisuuden toteuttavan arkkitehtuurin. Automatisointi perustuu kontrollitiedoston muuntamiseen XML-muotoon. Kontrollitiedoston XML-käärintä XW-kielellä esitellään luvussa 3.1. Myös varsinainen laskuaineisto voidaan konvertoida vastaavasti. Tarvittava kääreenkuvaus voi olla työläs kirjoittaa käsin, mutta XW-kielen yksinkertaisuuden takia se voidaan generoida automaattisesti kontrollitiedoston XML-muodosta (luku 3.2). Vastaavasti muotoilu laskuaineistolle voidaan tuottaa yleisellä muotoiluskriptillä, joka yhdistää kontrollitiedoston muotoilutiedot aineiston XML-esitykseen (luku 3.3). Tällainen automatisoidun muotoilun periaate on lupaava: sen avulla voi käsitellä eri aineistoja niiden kontrollitiedoston ohjaamana ilman rutiininomaisten muunnos- ja muotoilumääritysten kirjoittamista. Arkkitehtuuri on toteutettavissa olemassa olevilla XML-välineillä. Toteutusten laajamittaiseen käytettävyyteen liittyy kuitenkin tekniikoiden uutuuteen liittyviä ongelmia, joita käsitellään mm. suorituskykyarvioita sisältävässä luvussa 4.

2 Esimerkkiaineistot

Esimerkkiaineistona tarkastellaan rivipohjaista puhelinlaskuaineistoa. Aineisto koostuu yksittäisistä laskuista, jotka jakautuvat edelleen kolmeen osaan: laskun tunnisteosaan (A), erittelyosaan (B) ja maksutietoihin (C). Laskun maksutiedot sisältävät laskutusosoitteen, viitenumeron, eräpäivän ja loppusumman (kuva 1). Kunkin rivin merkitys osoitetaan sen alussa olevalla rivitunnisteella (`A1`, `A2` jne).[1]

Laskuaineistoon liittyy kontrollitiedosto (kuva 2), joka kuvaa aineiston rakenteen ja muotoilun. Kontrollitiedostossa on rivitunnisteittain kerrottu aineiston kunkin rivin merkitys ja muotoilu. Kontrollitiedosto sisältää laskuaineistoa vastaavasti tunnistetiedot-osan, erittelyn ja maksutiedot. Kontrollitiedoston tunniste- ja maksutietorivien rakenne on seuraava: ensimmäisenä on rivin tunniste, toisessa kentässä kyseisen tiedon tai elementin nimi, ja näitä seuraavat tulostukseen käytettävä fontti, pistekoko ja tuloskentän x- ja y-koordinaatit (kentän vasemman alanurkan sijainti).

Laskun erittely- eli B-rivit on kuvattu eri tavalla. Niille ei ole määritelty y-koordinaattia, sillä ne tulostetaan järjestyksessä peräkkäin. Erittelyrivit koostuvat soluista, joille on määritelty x-koordinaatti ja muita muotoiluja. Erilaisia soluja yhdistelemällä saadaan tuotettua erilaisia rivejä. Kontrollitiedostossa erittelyrivit kuvataan luettelemalla rivin tunnisteen ja nimen jälkeen rivin sisältösolujen nimet.

[1] Aineisto on pelkistys todellisesta laskuaineistosta, joka sisältää enemmän yksilöitävää tietoa.

```
A1|LASKU
A2|Laskunumero: 14045
A3|Asiakasnumero: 73052                         Tunnisteosa
A4|Pauli Puhelias
A5|Johdonmutka 24
A6|12345 LUURI
---------------------------------------------
B1|PUHELUERITTELY
B2|PÄIVÄ|SYKÄYKSIÄ|KESTO|NUMERO|HINTA
B3|2.8.2002|118|14 min|20010|5.02
B3|3.8.2002|139|15 min|12939|5.30             Erittelyosa
B3|4.8.2002|78|4 min|24978|1.01
B3|5.8.2002|90|5 min|44973|1.14
B4|6.8.2002|20203|0.30
B4|7.8.2002|12988|0.30
B3|8.8.2002|80|4 min|38271|1.06
---------------------------------------------
C1|Pauli Puhelias
C2|Johdonmutka 24
C3|12345 LUURI
C4|696224                                       Maksutiedot
C5|31.1.2002
C6|14.13
```

Kuva 1. Esimerkkilaskudata.

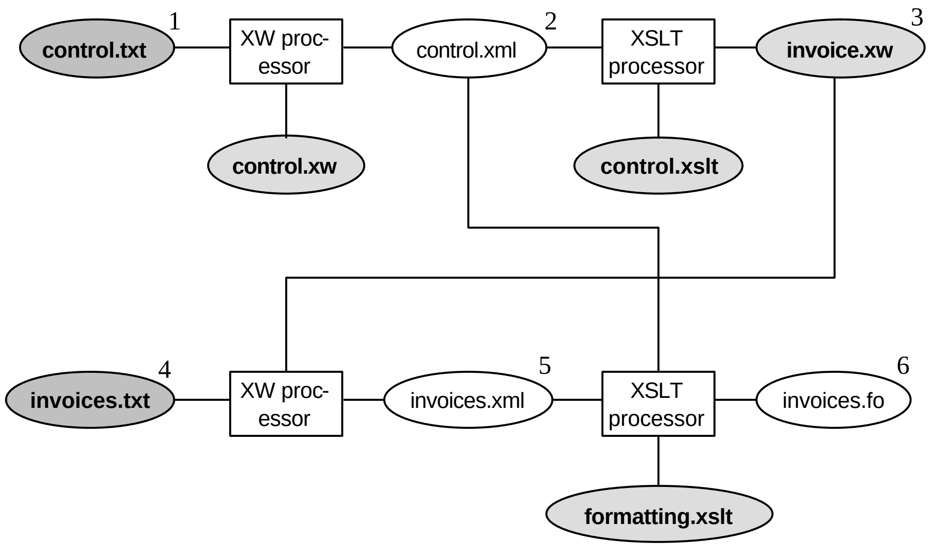
```
A1|Otsikko LASKU|Helvetica|12|76|65
A2|Laskunumero|Helvetica|10|432|65
A3|Asiakasnumero|Helvetica|10|432|80
...
B1|Otsikko PUHELUERITTELY|otsikkosolu1
B2|otsikkorivi|otsikkosolu1|otsikkosolu2...
B3|erittelyrivi puhelu|solu1|solu2|solu3|solu4|solu5
B4|erittelyrivi tekstiviesti|solu1|solu4|solu5
C1|Maksajan nimi|Courier|9|58|665
...
Erittelyn solut:
otsikkosolu1|Helvetica|12|40|left
otsikkosolu2|Helvetica|12|120|right
...
solu1|Helvetica|10|40|left
solu2|Helvetica|10|120|right
solu3|Helvetica|10|200|right
solu4|Helvetica|10|280|right
solu5|Helvetica|10|360|right
```

Kuva 2. Esimerkkiaineiston kontrollitiedosto.

Solujen muotoilu kuvataan kontrollitiedoston lopussa. Niihin liittyy muiden muotoilumääritysten lisäksi tieto siitä, tasataanko tieto solun oikeaan vai vasempaan reunaan (”right”/”left”).

3 Muotoiluprosessin automatisointi

Seuraavaksi kuvaamme, kuinka laskudatan XML-konversio ja muotoilu voidaan automatisoida dataa kuvaavan kontrollitiedoston ohjaamana. Kontrollitiedosto on prosessissa keskeisessä asemassa, koska se kuvaa datan rakenteen, (elementtien niminä käytettävät) osien nimet sekä muotoilutiedot. Kuva 3 esittää laskudatan käsittelyprosessia. Ensin kontrollitiedosto (1) muunnetaan käsittelyn helpottamiseksi XML-muotoon (2). Varsinaisen laskudatan XML-muotoon muuntava kääre (3) voidaan seuraavaksi generoida kontrollitiedoston XML-muodosta XSLT:llä. Tämän kääreen ohjaamana XML-muotoon (5) muunnettu laskuaineisto (4) voidaan lopulta muotoilla yleisen XSLT-skriptin (formatting.xslt) ohjaamana. Sama muotoiluskripti pätee eri aineistoihin, sillä se soveltaa aineistoa kuvaavan kontrollitiedoston (2) sisältämiä määrityksiä. Tuloksena syntyvä laskuaineiston XSL-FO-muotoinen muotoiltu esitys (6) voidaan lopulta tulostaa PDF:ksi, PostScriptiksi jne.



Kuva 3. Muotoiluprosessin automatisointi.

3.1 Kontrollidatan XML-käärintä

Muotoiluprosessin automatisointi perustuu kontrollitiedoston muuntamiseen XML-muotoon. Kuvaamme tämän suoraviivaisen muunnoksen XW-käärintäkielellä. Kuvassa 4 on esitetty XW-kääre laskuaineiston kontrollitiedoston muuntamiseksi XML-muotoon (kuva 5).

```
 1 <?xml version="1.0" encoding="ISO-8859-1"?>
 2 <xw:wrapper ...>
 3 <kontrollitiedosto>
 4    <tunnistetiedot xw:childterminator="\n">
 5       <rivi xw:starter="\^A"...>
 6          <tunniste>A<xw:collapse/></tunniste>
 7          <nimi xw:childseparator="\s">
 8             <xw:collapse xw:maxoccurs="unbounded">
 9                <xw:collapse/>_
10             </xw:collapse>
11          </nimi>
12          <fontti/> <koko/> <x/> <y/>
13       </rivi>
14    </tunnistetiedot>
15    <erittely xw:childterminator="\n">
16       <erittelyrivi xw:starter="\^B"...>
17          <tunniste>B<xw:collapse/></tunniste>
18          <nimi xw:childseparator="\s">
19             <xw:collapse xw:maxoccurs="unbounded">
20                <xw:collapse/>_
21             </xw:collapse>
22          </nimi>
23          <solu xw:maxoccurs="unbounded"/>
24       </erittelyrivi>
25    </erittely>
26    <maksutiedot xw:childterminator="\n">
27       <rivi xw:starter="\^C"...>
28          ...
29       </rivi>
30    </maksutiedot>
31    <erittelyn_solut xw:starter="\^Erittelyn solut:\n"...>
32       <solu xw:maxoccurs="unbounded" xw:childseparator="|">
33          <nimi/> <fontti/> <koko/> <x/> <a/>
34       </solu>
35    </erittelyn_solut>
36 </kontrollitiedosto>
37 </xw:wrapper>
```

Kuva 4. Kontrollitiedoston XML-muotoon muuntava XW-kääre.

Kääreessä kuvataan kontrollitiedoston (rivit 3-36) koostuvan neljästä osasta: tunnistetiedoista (rivit 4-14), erittelystä (rivit 15-25), maksutiedoista (rivit 26-30) ja erittelysolujen muotoilutiedoista (rivit 31-35). Kutakin osaa kuvataan XW-kääreessä elementillä. Nämä XW-nimiavaruuteen kuulumattomat tuloselementit tulostetaan kirjoitetussa muodossa tulokseen. Jos elementillä on kääreessä lapsielementtejä, tulostuvat samat lapsielementit tulokseen. Kääreen tyhjien elementtien sisällöksi tulee tulokseen lähtöaineiston vastaavan osan (teksti)sisältö. Elementtiä vastaavan osan aloittava tai lopettava merkkijono määritellään attribuuteilla `xw:starter` ja `xw:terminator`. Vaihtoehtoisesti osan aliosille voi määritellä esimerkiksi lopettavan merkkijonon sisältävää osaa vastaavassa elementissä attribuutilla `xw:childterminator` (esim. rivi 4). Osan toistuminen ilmaistaan XML Schema –tyyliin [5] attribuuteilla `xw:minoccurs` ja `xw:maxoccurs`. Elementti `xw:collapse` tuottaa tulosdokumenttiin itseään vastaavan syöteosan sisällön. Sen avulla esimerkiksi laskun osan nimeä kuvaavan kentän sisältö muutetaan tulosdokumenttiin `nimi`-elementin sisällöksi (kuva 4, rivit 8-10 ja kuva 5, rivi 6).

```
 1 <?xml version="1.0" encoding="ISO-8859-1"?>
 2 <kontrollitiedosto>
 3    <tunnistetiedot>
 4       <rivi>
 5          <tunniste>A1</tunniste>
 6          <nimi>Otsikko_LASKU_</nimi>
 7          <fontti>Helvetica</fontti> <koko>12</koko>
 8          <x>76</x> <y>65</y>
 9       </rivi>
10       ...
11    </tunnistetiedot>
12    <erittely>
13       <erittelyrivi>
14          <tunniste>B3</tunniste>
15          <nimi>erittelyrivi_puhelu_</nimi>
16          <solu>solu1</solu> <solu>solu2</solu>
17          <solu>solu3</solu> <solu>solu4</solu>
18          <solu>solu5</solu>
19       </erittelyrivi>
20       ...
21    </erittely>
22    ...
23    <erittelyn_solut>
24       <solu>
25          <nimi>otsikkosolu1</nimi>
26          <fontti>Helvetica</fontti> <koko>12</koko>
27          <x>40</x> <a>left</a>
28       </solu>
29       ...
30    </erittelyn_solut>
31 </kontrollitiedosto>
```

Kuva 5. Kontrollitiedosto XML-muodossa.

3.2 Käärekuvauksen automatisointi

Laskuaineisto on XML-käsittelyä varten muunnettava XML-muotoon. Tuotettava XML-muoto on esitetty kuvassa 6.[2] Tämä XML-muunnos voidaan tehdä XW-kääreenkuvauskielellä. Esimerkiksi laskuaineiston (kuva 1) ensimmäinen rivi

```
A1|LASKU
```

muunnetaan kuvan 6 rivin 5 esittämään muotoon. Kuvassa 7 on XW-kääre aineiston muuntamiseksi XML-muotoon. Muunnos tapahtuu tunnistamalla rivi tunnisteestaan ”`A1`” (kuva 7, rivit 7–13). Vastaavasti tunnistetaan ja muunnetaan XML:ksi myös muut laskuaineiston rivit, vaihtaen vain rivitunnistetta ja luotavan elementin nimeä.

Laskuaineistot saattavat olla rakenteeltaan rikkaita, ja muunnosmäärittelyn (esim. XW-kääreen) laatiminen aineistolle vaatii paljon rutiininomaista kirjoittamista. Muunnos voidaan automatisoida käyttäen XML-muotoista kontrollitiedostoa parametritiedostona, josta aineiston rivitunnisteet ja XML-muodossa käytettävien elementtien nimet haetaan. Näin kaikki luotavat elementit saadaan muodostettua samalla kaavalla.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <data>
3   <lasku>
4      <tunnistetiedot>
5         <Otsikko_LASKU_>LASKU</Otsikko_LASKU_>
6         <Laskunumero_>Laskunumero: 14045</Laskunumero_>
7         <Asiakasnumero_>Asiakasnumero: 73052</Asiakasnumero_>
8         ...
9      </tunnistetiedot>
10     <erittely>
11        ...
12        <erittelyrivi_puhelu_>
13           <solu1>2.8.2002</solu1> <solu2>118</solu2>
14           <solu3>14 min</solu3> <solu4>20010</solu4>
15           <solu5>5.02</solu5>
16        </erittelyrivi_puhelu_>
17        ...
18     </erittely>
19     <maksutiedot>
20        <Maksajan_nimi_>Pauli Puhelias</Maksajan_nimi_>
21        <Maksajan_katuosoite_>Johdonmutka 24</Maksajan_katuosoite_>
22        ...
23     </maksutiedot>
24  </lasku>
25 </data>
```

Kuva 6. Laskuaineisto XML-muodossa.

Kontrollitiedosto sisältää kaiken tarvittavan tiedon laskuaineiston muuntamiseksi, joten kun kontrollitiedosto on XML-muodossa, siitä voidaan generoida laskuaineis-

[2] Käytämme elementtiniminä kontrollitiedoston XML-käärinnän yhteydessä poimittuja tunnisteita kuten `Otsikko_LASKU_`.

```
 1 <?xml version="1.0" encoding="ISO-8859-1"?>
 2 <xw:wrapper xmlns:xw="http://www.cs.uku.fi/XW/2001"
 3       xw:inputencoding="ISO-8859-1" xw:outputencoding="ISO-8859-1"
 4       xw:sourcetype="text">
 5    <data>
 6       <lasku xw:maxoccurs="unbounded">
 7          <tunnistetiedot>
 8             <xw:collapse xw:starter="\^A1" xw:childseparator="|">
 9                <xw:ignore/> <!-- tunnistekentän (tyhjä) loppu -->
10                <Otsikko_LASKU_/>
11             </xw:collapse>
12             ...
13          </tunnistetiedot>
14          <erittely>
15             <xw:CHOICE xw:maxoccurs="unbounded">
16                ...
17                <erittelyrivi_puhelu_ xw:childseparator="|"
18                      xw:starter="\^B3">
19                   <xw:ignore/>
20                   <solu1/> <solu2/> <solu3/> <solu4/> <solu5/>
21                </erittelyrivi_puhelu_>
22                ...
23             </xw:CHOICE>
24          </erittely>
25          <maksutiedot>
26             <xw:collapse xw:childseparator="|" xw:starter="\^C1">
27                <xw:ignore/>
28                <Maksajan_nimi_/>
29             </xw:collapse>
30             ...
31          </maksutiedot>
32       </lasku>
33    </data>
34 </xw:wrapper>
```

Kuva 7. Laskuaineistolle automaattisesti muodostettava XW-kääre.

tolle XW-kääre. Tämä vaihe toteutettiin XSLT:llä [6]. XSLT-skripti saa syötteenä XML-muotoisen kontrollitiedoston. Edellä esitetty XW-kääreen osa (kuva 7, rivit 7–13) luodaan XSLT:llä seuraavasti:

```
<xsl:template match="rivi">
   <xw:collapse xw:starter="\^{tunniste}"
                xw:childseparator="|">
      <xw:ignore/>
      <xsl:element name="{nimi}"/>
   </xw:collapse>
</xsl:template>
```

Kontrollitiedoston `rivi`-elementin `tunniste`-lapsen sisältö otetaan siis attribuutin `xw:starter` arvoksi, ja sen `nimi`-lapsi tuottaa nimen luotavalle elementille. Kaikki XML-muotoisen kontrollitiedoston tunnistetietojen ja maksutietojen `rivi`-elementit voidaan käsitellä tällä samalla XSLT-kaavaimella.

Myös `erittelyrivi`-elementeistä luotavaan XW-kääreen osaan (kuva 7, rivit 17–21) tarvittavat nimi ja rivin aloittava tunniste tuotetaan XML-muotoisesta kontrollitiedostosta samalla tavalla:

```
<xsl:template match="erittelyrivi">
   <xsl:element name="{nimi}">
      <xsl:attribute name="xw:starter">\^<xsl:value-of
         select="tunniste"/></xsl:attribute>
      <xsl:attribute
         name="xw:childseparator">|</xsl:attribute>
      <xw:ignore/>
      <xsl:for-each select="solu">
         <xsl:element name="{.}"/>
      </xsl:for-each>
   </xsl:element>
</xsl:template>
```

Lisäksi jokaista erittelyrivin solua kohden tuotetaan kääreenkuvaukseen vastaavan niminen elementti (kuva 7, rivi 20).

Näin kaikki kääreen muodostamiseen tarvittavat tiedot saadaan kontrollitiedostosta, ja kääre voidaan muodostaa automaattisesti koko laskuaineistolle. Tuloksena saatu XW-kääre (kuva 7) muuntaa varsinaisen laskudatan XML-muotoon (kuva 6). Kontrollitiedostoa hyväksikäyttäen XW-kääre — joka saattaa todellisen laskuaineiston tapauksessa olla jopa kymmeniä sivuja pitkä — saatiin tuotettua kahdella yksinkertaisella ja lyhyellä muunnoksella.

3.3 Muotoilumäärittelyn automatisointi

Laskudatan XML-muotoon muuntamisen jälkeen sen muotoiluun voidaan käyttää XML-välineitä. Muotoilun automatisointia varten laskuaineistolle kirjoitettiin XSLT-skripti, joka kontrollitiedostoa hyväksikäyttäen muuntaa aineiston muotoilun sisältävään XSL-FO-muotoon.

Muotoiluskripti saa syötteenä XML-muotoisen laskuaineiston. Skripti toteuttaa muotoilun noutamalla elementtien muotoilutiedot (kirjasintyyppi ja -koko sekä sijainti) elementtinimen perusteella XML-muotoisesta kontrollitiedostosta ja muodostamalla niillä parametroidun muotoiluolion. Esimerkiksi tunnistetietojen lapsielementtien XSL-muotoilu on kuvattu kuvan 8 riveillä 3–20.

Yksittäisten elementtien tietoja on pystyttävä asettelemaan lopputulokseen mielivaltaisesti. Tähän asetteluun käytettiin XSL:n `block-container`-muotoiluoliota. Sille voidaan antaa koordinaatteina objektin sijainti muodostettavalla sivulla, kun attribuutin `position` arvo on `”absolute”` (kuva 8, rivit 15–20).

Maksutietojen elementit käsitellään samalla tavalla kuin tunnistetietojen elementit. Erittelyrivien muotoileminen on hieman haastavampaa. Erittelyrivien tiedoilla ei ole y-koordinaattia, sillä rivit tulostetaan vain järjestyksessä allekkain. Jokainen rivi voi koostua eri määrästä eri tavoin sisennettäviä soluja. Tämän takia erittelyrivien muotoilu toteutettiin käsittelemällä jokaista erittelyriviä omana taulukkonaan, jolloin jokaiselle riville voidaan määritellä solukohtaisesti sarakkeiden leveydet (kuva 9).

```
<xsl:template match="tunnistetiedot/*">
   <xsl:variable name="elementtinimi" select="name()"/>
   <xsl:variable name="fontti" select=
         "document($kontrollitiedosto)//
         tunnistetiedot/rivi[nimi=$elementtinimi]/fontti"/>
   <xsl:variable name="koko" select=
         "document($kontrollitiedosto)//
         tunnistetiedot/rivi[nimi=$elementtinimi]/koko"/>
   <xsl:variable name="x" select=
         "document($kontrollitiedosto)//
         tunnistetiedot/rivi[nimi=$elementtinimi]/x"/>
   <xsl:variable name="y" select=
         "document($kontrollitiedosto)//
         tunnistetiedot/rivi[nimi=$elementtinimi]/y"/>
   <fo:block-container height="1cm" width="20cm" top="{$y}pt"
         left="{$x}pt" position="absolute">
      <fo:block font-family="{$fontti}" font-size="{$koko}pt">
         <xsl:apply-templates/>
      </fo:block>
   </fo:block-container>
</xsl:template>
```

Kuva 8. Muotoiluskriptin tunnistetiedot muotoileva osa.

```
<xsl:template match="erittely/*">
<fo:table>
   <xsl:variable name="erittelyrivinimi" select="name()"/>
   <xsl:for-each select="document($kontrollitiedosto)//
      erittely/erittelyrivi[nimi=$erittelyrivinimi]/solu">
      <xsl:call-template name="sarake"/>
   </xsl:for-each>
   <fo:table-body>
      <fo:table-row><xsl:for-each select="*">
         <xsl:variable name="solunimi" select="name()"/>
         <xsl:variable name="fontti" select=
               "document($kontrollitiedosto)//
               erittelyn_solut/solu[nimi=$solunimi]/fontti"/>
         ...
         <fo:table-cell padding="0.5mm">
            <fo:block font-family="{$fontti}"
                  font-size="{$koko}pt" text-align="{$a}">
               <xsl:apply-templates/>
            </fo:block>
         </fo:table-cell>
      </xsl:for-each></fo:table-row>
   </fo:table-body>
</fo:table></xsl:template>
```

Kuva 9. Erittelyrivien muotoilu.

```
1  <xsl:template name="sarake">
2     <xsl:param name="max_pituus" select="450"/>
3     <xsl:variable name="solunimi" select="text()"/>
4     <xsl:variable name="x" select=
5           "document($kontrollitiedosto)//
6           erittelyn_solut/solu[nimi=$solunimi]/x"/>
7     <xsl:variable name="nextsolunimi" select=
8           "following-sibling::solu[1]"/>
9     <xsl:variable name="nextx" select=
10          "document($kontrollitiedosto)//
11          erittelyn_solut/solu[nimi=$nextsolunimi]/x"/>
12    <xsl:choose>
13       <xsl:when test="following-sibling::solu">
14          <fo:table-column column-width="{$nextx - $x}pt"/>
15       </xsl:when>
16       <xsl:otherwise>
17          <fo:table-column column-width=
18                "{$max_pituus - $x}pt"/>
19       </xsl:otherwise>
20    </xsl:choose>
21 </xsl:template>
```

Kuva 10. Erittelyrivien sarakkeiden määrittely.

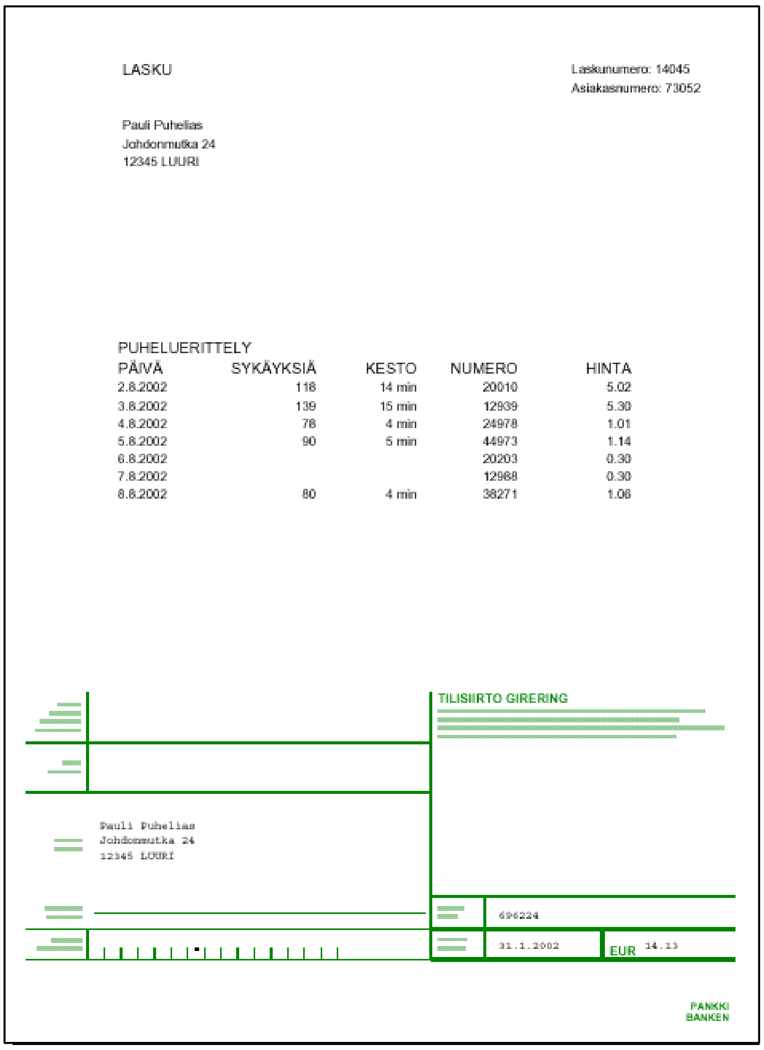
Taulukon muotoilu tapahtuu niin, että ensin tuotetaan sarakkeiden leveydet (kuva 9, rivit 4–7 ja kuva 10). Ne saadaan laskettua vähentämällä solua seuraavan solun x-koordinaatista solun oma x-koordinaatti (kuva 10, rivit 14 ja 17). Sen jälkeen erittelyrivin jokainen `solu` muotoillaan omaksi taulukon solukseen soveltaen kontrollitiedostosta saatavia fontti-, koko- ja tasaustietoja (kuva 9, rivit 9–21).

Muotoiluskripti tuottaa XSL-FO-muotoisen laskun, josta lasku voidaan tulostaa esimerkiksi PDF-muotoon (kuva 11).

4 Arviointia

Mittasimme laskuaineiston muuntamisen ja muotoilemisen vaatimaa suoritustaikaa kokeellisesti. Kontrollitiedoston käsittelyn vaatimaa aikaa ei testattu, koska se ei ole käytännössä oleellinen: kontrollitiedosto on kiinteän kokoinen, ja se käsitellään vain kerran yhtä aineistotyyppiä kohden.

Testaus suoritettiin SunOS 5.8 -käyttöjärjestelmässä 750 MHz prosessoria käyttävällä Sun Fire 280R -palvelimella ja Java-versiolla J2SE v1.4.0. XSLT-prosessorina käytettiin sekä Xalania [7] että Saxonia [8]. PDF-muotoilu suoritettiin Apachen FOP-prosessorilla [9]. Käytettyä prosessoriaikaa mitattiin Unixin `time`-komennolla. Lähtöaineistona oli kaksi tyypillistä laskua (neljä sivua) sisältävä 6,6 kilotavun tekstitiedosto ja siitä kopioimalla muodostetut monikerrat. Mittasimme erikseen laskuaineiston muuntamisen XW:llä XML:ksi ja sen muuntamisen edelleen PDF:ksi. XML–PDF-muunnos suoritettiin kahdella eri tavalla: yhdessä askeleessa FOPilla käyttäen sen sisäänrakennettua XSLT-prosessoria (Xalan) ja kahdessa askeleessa suorittaen XSLT-muunnos Saxonilla ja PDF-muotoilu FOPilla. Tulokset kuvassa 12.



LASKU

Laskunumero: 14045
Asiakasnumero: 73052

Pauli Puhelias
Johdonmutka 24
12345 LUURI

PUHELUERITTELY

PÄIVÄ	SYKÄYKSIÄ	KESTO	NUMERO	HINTA
2.8.2002	118	14 min	20010	5.02
3.8.2002	139	15 min	12939	5.30
4.8.2002	78	4 min	24978	1.01
5.8.2002	90	5 min	44973	1.14
6.8.2002			20203	0.30
7.8.2002			12988	0.30
8.8.2002	80	4 min	38271	1.08

TILISIIRTO GIRERING

Pauli Puhelias
Johdonmutka 24
12345 LUURI

696224

31.1.2002

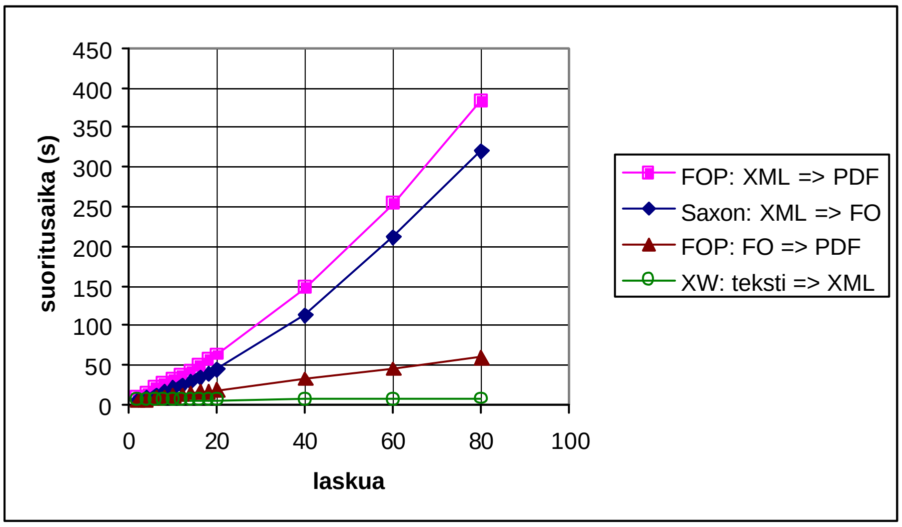
EUR 14.13

PANKKI
BANKEN

Kuva 11. Muotoiltu lasku PDF-muodon kautta tulostettuna

Tuloksista nähdään, että XW:llä suoritetun XML-konversion vaatima aika on murto-osa muotoilun vaatimasta ajasta. Tästä taas selvästi suurin osa kuluu XSLT-muunnokseen, jonka vaatima aika kasvaa lopullisen PDF-muotoilun ajantarvetta nopeammin. 80 laskun aineistolla suoritusaikojen ero on jo viisinkertainen. Käytetyllä XSLT-prosessorilla ei ollut merkittävää vaikutusta suoritusaikaan.

Nykyisellään kokeiltu käsittelytapa ei vaikuta käytännölliseltä vaihtoehdolta massatulosteiden muotoiluun: minuutissa muotoiltiin ainoastaan 20 laskua. Toisaalta pienivolyymisten verkkolaskuaineistojen käsittelyyn XSLT-prosessoreiden nykyinenkin suorituskyky voi olla riittävä.



Kuva 12. Laskuaineiston vaatimia muunnos- ja muotoiluaikoja

Automaattisen muunnoksen ja muotoilun perimmäinen tavoite on pystyä käsittelemään erilaisia aineistoja sisällöstä riippumatta samoilla muunnosohjelmilla räätälöimättä niitä eri aineistoille sopiviksi. Edellä esitetyt muunnoskuvaukset ovat pitkälti sisällöstä riippumattomia. Aineistoriippumattomien muotoiluskriptien käyttö aiheuttaa sen, että kaiken lopputulokseen halutun tekstin on sisällyttävä elementteihin. XML-aineistoihin tulee siksi esimerkiksi sellaisia kömpelön tuntuisia elementtejä kuin `<laskunumero>Laskunumero: 14045</laskunumero>`. Sisältöriippumattomuus mahdollistaa kuitenkin sen, että kerran kirjoitettuja muunnosohjelmia voidaan käyttää lukemattomien aineistojen käsittelyyn. Riittää, että aineisto on erotinmerkein eroteltua, ja sille on olemassa vastaava kontrollitiedosto. Kolmen suhteellisen lyhyen skriptin ohjaamana koko muunnos- ja muotoiluputki toimii automaattisesti.

XML-tekniikoiden käytössä on vielä toistaiseksi ongelmana niiden uutuus. Törmäsimme kokeilussa erityisesti XSL-prosessorien puutteisiin. Ilmeisesti täydellisiä XSL-toteutuksia ei ole.

Kiitokset

Edellä esitetty tutkimus on tehty XRAKE (XML-rajapintojen kehittäminen) –hankkeessa, jonka rahoittavat TEKES/EAKR, Deio Oy, Enfo Group Oyj, Järvi-Suomen Ohjelmapalvelu Oy, Kuopion yliopistollinen sairaala, Medigroup Oy, SysOpen Oyj ja TietoEnator Oyj.

Lähteet

1. Ek, M., Hakkarainen, H., Kilpeläinen, P., Kuikka, E., Penttinen, T.: Describing XML wrappers for information integration. In: Proc. of XML Finland 2001, Tampere, Nov. 2001, 38–51
2. Ek, M., Hakkarainen, H., Kilpeläinen, P., Penttinen, T.: Declarative XML wrapping of data. Report A/2002/2. Univ. of Kuopio, Dept. of Comp. Sci. and Applied Math., 2002
3. Waldt, D.: Getting data into XML: Data collection and conversion techniques. Abstract. In: Proc. of XML Europe, Barcelona, May 2002
4. Adler, S. et al, editors: Extensible Stylesheet Language (XSL) Version 1.0. W3C Rec., October 2001, http//:www.w3.org/TR/xsl/
5. Thompson, N., Beech, D., Maloney, M., Mendelsohn, N. editors: XML Schema Part 1: Structures. W3C Rec., May 2001
6. Clark, J. editor: XSL Transformations (XSLT) Version 1.0. W3C Rec., Nov. 1999
7. Apache Xalan version 2.2.D11, 2001, http://xml.apache.org/xalan-j/index.html
8. Saxon version 6.4.4, 2001, http://saxon.sourceforge.net/saxon6.4.4/index.html
9. Apache FOP version 0.20.4, 2002, http://xml.apache.org/fop/index.html

Aiming Towards Best Practices in XML techniques for Text Corpora Annotation

City of Helsinki Public Works Department Press Releases - A Case Study

Mikko Lounela

Research Institute for the Languages of Finland
mikko.lounela@kotus.fi

Abstract. The reusability of digital text resources collected for particular research purposes is a growing concern in the linguistic research community. For the collected and digitised texts to be maximally usable, both in view of their original research purposes and for the possible future uses, they have to be annotated in as rich and standard ways as possible. One of the goals of the project Digitised Archive Material in Culture Studies, funded by the Academy of Finland, is to find the best practices for organizing, annotating and productising different textual materials for research purposes. The work described here is part of the project.
The text materials are encoded in text the levels of text structure, morphosyntax, semantics, meta-information, and corpus structure. All the encoding has been conducted using the most widely recognised XML techniques.
This article demonstrates the structure and encoding of a small sample text corpus of press releases published by the City of Helsinki Public Works Department around years 1979 and 1999.

1 Introduction: corpus, subcorpus and text

A linguistic text corpus can be defined as a machine-readable text collection with certain qualifications. Some of the most important characteristics, cited in recent literature (see [1], [8], [3] and [2]), are:

– Authenticity - the texts should be real language,
– Representativeness - the texts should represent the language or language variety as accurately as possible,
– Size - the corpora should preferably be of finite, fixed size,
– Machine-readability - the texts should be in a standard format, that can be processed with a computer,
– Documentedness - the collecting, sampling and encoding principles should be thoroughly documented,
– Stability and publicity - the corpus should remain the same over time, and it should be available to researchers.

In addition to the characteristics above, the annotation of the text corpora is considered important. Many of the large text collections (tens or hundreds of millions of words) are at present automatically analyzed on the morphological or surface-syntactic level. The large corpora include The Language Bank of Finland (LBF) [14], The British National Corpus (BNC) [10], and The Bank of English (BE) [19]. On the morphosyntactic analysis of a large corpus, see [5].

In our sub-project of the project Digitised Archive Material in Culture Studies, we search for the best practices for annotating and organizing small text collections, and ways to exploit the resources. The texts are collected for current research needs, and are annotated as richly as possible, according to the most widely recognised formats. The annotation is done in cooperation with the researchers, according to their needs (but bearing in mind the possible future uses). Most of the processing is done semi-automatically, prioritising accuracy over speed. The characteristics of the individual collections will fall between large corpora and carefully hand-coded small text corpora such as the HKV-Corpus [4] or the Oulu Corpus [9].

Our example text collection have been collected by Salli Kankaanpää for her dissertation work. Her work will examine the textual changes in press releases issued by public authorities (see eg. [6]). The collection consists of 83 texts, i.e. approximately 54 000 words. All the texts are in Finnish. The texts are press releases published by the City of Helsinki Public Works Department around years 1979 and 1999.

The texts are annotated on the text-structure level as well as on the morphosyntactic, semantic and meta-information levels. The relations between the texts, text collections and relevant documents outside the collections are also expressed. All the encodings are in Extensible Markup Language (XML) [24] format, according to the following specifications, where possible: Text Encoding Initiative (TEI) [22] and Corpus Encoding standard (CES) [11], where TEI is not applicable. Resource Description Framework (RDF) [25] and recommendations of the Dublin Core Metadata Initiative (DCMI) [16] are used, e.g., for building a separate link-base. The link-base can be used eg. in building a searchable and browsable version of the text collection. For these purposes, some of the the specifications have been extended. All the extensions to the specifications have been documented and will be explained in this paper.

This paper will explore the encoding of the corpus from level to level, explaining the textual items and the specifications according to which their markup is carried out. The focus will be on the expression of the relations between text collections and individual texts.

2 Encoding the text structure level

The textual structure of a press release resembles somewhat that of a letter on the one hand, and that of a piece of news or a committee report on the other. It may have an opener, including information of the author, the date of the release, possibly information about the author, etc. The opener is followed by main information division, consisting of text paragraphs. The main division is rich in names, dates, etc. After the main information may come contact information, signatures, appendices, and other closing information. The press releases may also have appendices, such as maps, photographs etc.

2.1 The high-level structure

We have encoded The textual structure using the opener-element for the opening information and general (possibly nested) div-elements for the rest of the high-level text structure. TEI has an closer-element as well, but it proved too personal letter-like for our purposes. We have used the type-attribute to classify the divisions, and included closer in its values. The text division types, in addition to the distinct opener-element, are: covering note (which includes salutations and signatures before main information), main information (the information intended for the news), sub (subdivision), closer (salutations, signatures etc.), including contact information and appendix reference (references to the appendices).

A sample upper-level text structure looks like this:

```
<opener>
</opener>
<div type="main information">
</div>
<div type="closer">
 <div type="contact information">
 </div>
 <div type="appendix reference">
 </div>
</div>
```

2.2 The paragraph-level structure

Paragraph level (or middle level) text structure consists of paragraphs, subheaders, lists, tables and the like. The rendering information has been included in the rend-attribute of the paragraph level elements, if the rendering runs through the paragraph. the headers have been divided into main headers, which occur at the beginning of each text, and subheaders, each occurring at the beginning of its own text division.

Special cases on the middle level are the elements inside the opener-element and the elements inside the text division of type closer. They are mainly paragraph-level semantic elements, such as bylines, salutations, references to appendices, signatures, etc. In these cases the paragraph-level (appearance-based) markup is intermixed with the semantic level (interpretation-based) markup. Non-textual parts of the documents are encoded using a gap-element with a desc-attribute to describe the omitted material. An opener may look like this:

```
<opener>
```

```
<byline>HELSINGIN KAUPUNGIN RAKENNUSVIRASTO</byline>
<byline>Katurakennusosasto</byline>
<byline><docAuthor>A Taipale/RS</docAuthor></byline>

<title type="text class">LEHDISTÖTIEDOTE</title>

<gap desc="vertical line" />
</opener>
```

2.3 The low-level structure

By low-level text structure, we mean rendering inside the paragraph-level elements. Any text-rendering that differs from the paragraph-level rendering (eg. emphasis expressed as italics, etc.) has been encoded with a hi-element, rendering style being a value of the rend-attribute. Line breaks, column breaks and page breaks are marked with lb-, cb-, and pb-elements. A simple sample paragraph encoded in the text-level looks like this:

```
<p>Rajakylän paikallistie n:o 11631 (Maratontie)<lb />
Jakomäessä suljetaan ajoneuvoliikenteeltä<lb />
18.6.1979 alkaen noin puolentoista kuukauden<lb />
ajaksi Porvoon moottoritien ylittävän sillan<lb />
kohdalta eritasoliittymän rakentamisen vuoksi.<lb /></p>
```

3 Encoding the semantic level

By semantic level encoding we mean recognising and marking strings that refer to matters that require human interpretation, such as dates, places, people, organisations, measures, numbers etc. The elements have been classified further using the type-attribute.

The elements are represented in table 1 in an approximate order of frequency. Each element is listed together with its types, again in order of frequency. The personal names have been further typified according to sex and degree of abbreviation.

At the same stage with the semantic tagging, we have performed manual sentence-boundary tagging. After this, the hand-made tagging thus is complete, and a one-paragraph main information part of a press release looks like this:

```
<div type="main information">
<head type="main"><s>
  <name type="district">JAKOMÄEN</name> ERITASOLIITTYMÄN RAKENTAMINEN
</s></head>

<p><s><name type="road">Rajakylän paikallistie n:o <num type="road">11631</num>
</name>(<name type="road">Maratontie</name>)<lb />
<name type="district">Jakomäessä</name> suljetaan ajoneuvoliikenteeltä<lb />
<dateRange><date value="1979-06-18">18.6.1979</date> alkaen noin
```

Table 1. An example Table

Element	Types
name	communal (organization), person, street, region, place, district, enterprise, role, building, organization (of other type), country, proper, event, road, (piece of) work
num	undef, telephone, ordinal, time, street address, percentage, pagenum, money, velocity, transportation
abbr	
date	
rs (referring string)	occupation, email address, clarification
time	
measure	length, currency, area, weight, height, width, volume, thickness, depth, breadth
dateRange	
timeRange	
(surface mail) address	
street (as part of surface address)	

```
<num type="time">puolentoista</num> kuukauden<lb />
ajaksi</dateRange> <name type="road">Porvoon moottoritien</name> ylittävän sillan<lb />
kohdalta eritasoliittymän rakentamisen vuoksi.</s><lb /></p>
</div>
```

4 Encoding the morphosyntactic level

Morphosyntactic analysis means adding the lemmas of the words, the parts-of-speech, as well as other morphological features of the words in the text. It may also entail adding certain syntactic relations between words. For Finnish, there are some automatic analyzers available. Most of the analyzers (see e.g. [13], [20]) retain only the assumed correct analyses of the words. Removing the undesired analyses of the words within the context is called disambiguation. The parsers will sometimes make wrong choices in disambiguating the texts. For the press release material, we decided to use the TWOL-analysis (see [7]), and disambiguate it manually. We thought that this would minimize the amount of unwanted analyses.

The morphosyntactic analysis is added by wrapping each word in its own w-element. The element has three attributes. The lemma-attribute carries the word's dictionary-form as its value, the type-attribute bears the part-of-speech, and the msd-attribute includes the rest of the morphosyntactic analysis. Here is a part of the above text after the TWOL-analysis and manual disambiguation (a partial description of the TWOL tags can be found in [21]):

```
  <w lemma="(" type="delimiter">(</w>
  <name type="road">
```

```
    <w lemma="maratontie" type="N" msd="NOM SG">Maratontie</w>
  </name>
  <w lemma=")" type="delimiter">)</w>
<lb />
  <name type="district">
    <w lemma="jakomäki" type="N" msd="PROP INE SG">Jakomäessä</w>
  </name>
  <w lemma="sulkea" type="V" msd="PRES PSS PE4">suljetaan</w>
  <w lemma="ajoneuvoliikenne" type="N" msd="ABL SG">ajoneuvoliikenteeltä</w>
<lb />
```

5 Encoding the meta-level information

5.1 The TEI metadata

The meta-information of the texts are encoded following two distinct specifications. First, each subcorpus and each text within the subcorpus has its own TEI-header. The TEI-headers of the individual texts are kept as brief as possible, carrying information on the title of the text, people responsible for the manual encoding, the distributing organisation, and the type of the source text.

The TEI-metadata describing the subcorpus is much more elaborate, including data of the corpus itself, the origin of the texts, the corpus collecting project and process (collecting, scanning, ocr, linguistic analysis, markup), distributing organisation, sampling, languages used, etc. We have omitted the example because of the length of the TEI-header. The subcorpus metadata are intended to be as full and accurate as possible, following the TEI-header metadata guidelines (see [23]).

5.2 The Dublin Core metadata

The corpus is also described using the Dublin Core element set. The specification is implemented in RDF following the DCMI proposed recommendation [17]. The DCMI metadata set consists of 15 elements: Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, and Rights.

Most of the elements are employed simply for their intended use, when applicable for a text collection or an individual text within it. The relation-element is used to encode the relations between the different documents in the corpus and references to relevant documents outside the corpus. The link-base is produced using the relation-element.

6 Encoding the corpus structure

6.1 The corpus structure in TEI

The corpus consists of four levels: the corpus level, the subcorpus level (e.g. the press releases), and the level of the individual texts. The individual texts may also have subtexts, such as appendices. Each corpus, subcorpus and text, except for the image-appendices, has its own meta-description. What is problematic is that the TEI specification does not allow more than two layers of meta-information. That is why the text files have to be directly included in both the main corpus and the subcorpora they belong to, and the connection between the main corpus and the subcorpora is lost.

The dependencies are expressed as XML external entities. The entity lists are maintained in separate files on the subcorpus-level. One possible reference structure is expressed in figure 1. The arcs represent external entity references, that include the texts files in the corpora.

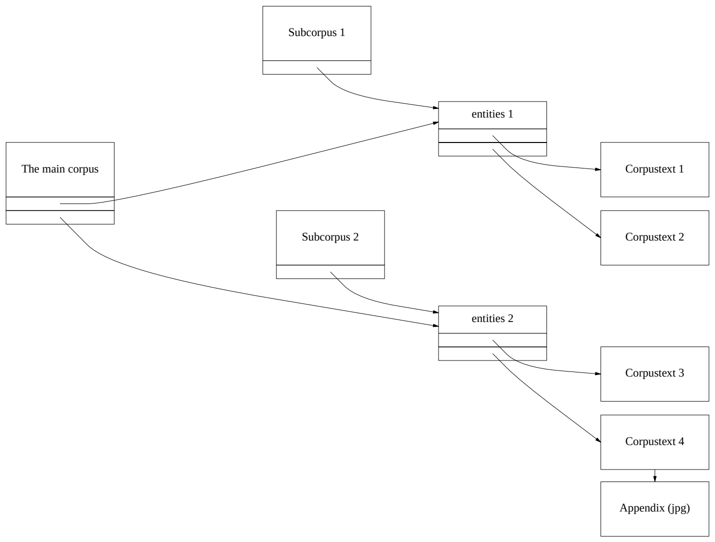


Fig. 1. Three-level corpus structure in TEI.

6.2 The corpus structure in DCMI/RDF

In addition to the TEI document structure, which is essentially restricted to the two-level structure, we have designed a more flexible link-base that would enable building the browsable corpus (and other applications later). The structure of the link-base follows two recommendations: DCMI gives a proposed recommendation for expressing the Dublin Core element set in RDF/XML [17]. The document includes a DTD, but the DTD does not permit the usage of RDF elements, such as sequences and bags, inside the Dublin Core elements. Because of that, we have crossbred the DTD with the RDF DTD by Dan Connolly [26].

In the resulting DTD, we have three namespaces: dc for the DCMI elements, rdf for the elements defined in the RDF DTD, and elias for the attributes defined by ourselves. We use the dc:relations-element for expressing the structure of the corpus. The classification is made according to the DCMI Relation Element Working Draft [18]. The classification is encoded in the elias:type attribute of the dc:relation-element. The possible values for the attribute are: IsPartOf, HasPart, IsVersionOf, HasVersion, IsFormatOf, HasFormat, References, IsReferencedBy, IsBasedOn, IsBasisFor, Requires and IsRequiredBy. In addition to elias:type, we defined two attributes. The first one is called elias:target, and its purpose is to refer to the ID of the subject in the other end of the relation, if the subject has its own entry in the link-base. The second one is elias:specifier, and its purpose is to specify further the nature of the subject in the other end. The possible values for the elias:specifier are: supercorpus, subcorpus, textcollection, document, appendix, information and unspecified.

With the above-explained machinery, we can achieve a satisfactory description of the relations between the different texts, text collections and subcollections within the corpora. The RDF description of the corpus level can be seen in the example below (other than dc:title and dc:relation-elements omitted). In the example, the corpus has no relations other than the ones it has with its parts.

```
<!-- The main corpus -->
<rdf:Description about="file:file:../../elias.xml" ID="elias">
  <dc:title>Elias Corpus Collection</dc:title>
  <!-- Relations -->
  <dc:relation elias:type="HasPart" elias:specifier="subcorpus">
    <rdf:bag>
      <rdf:li elias:target="elias-rakvir" />
      <rdf:li elias:target="elias-maakunta" />
    </rdf:bag>
  </dc:relation>
</rdf:Description>
```

On the subcorpus-level, we need to express the relations between the main corpus and the subcorpora, as well as those between the subcorpora and the texts they consist of. The subcorpora may also have references to relevant publications, terms of usage, and other relevant documents outside the actual corpus. The outside documents may have their own entries in the RDF link-base. The relations between the subcorpora are expressed only through the main corpus. The example below shows the relations between a subcorpus and two of its texts. It also illustrates references to some corpus-external documents.

```
<!-- Rakennusvirasto -->
<rdf:Description about="file:../elias_rakvir.xml" ID="elias-rakvir">
<dc:title>Elias Corpus Collection: City of
Helsinki Public Works Department press release subcorpus</dc:title>
<!-- Relations -->
<!-- up -->
<dc:relation elias:type="IsPartOf" elias:specifier="supercorpus"
  elias:target="elias" />
<!-- down -->
```

```
<dc:relation elias:type="HasPart" elias:specifier="document">
  <rdf:bag>
    <rdf:li elias:target="jakomaki-1-1979" />
    <rdf:li elias:target="talvipuhtaanapito-1980" />
  </rdf:bag>
</dc:relation>
<!-- other -->
<dc:relation elias:type="References" elias:specifier="information">
http://www.hkr.hel.fi/</dc:relation>
<dc:relation elias:type="IsReferencedBy" elias:specifier="information"
  elias:target="kankaanpaa_2000a" />
<dc:relation elias:type="IsReferencedBy" elias:specifier="information"
  elias:target="rakvir_terms_of_usage" />
</rdf:Description>
```

The individual texts are linked to the subcorpus and to their possible appendices. They are also linked to the image of the original text:

```
<rdf:Description about="file:../mallit/jakomaki-1-1979.xml" ID="jakomaki-1-1979">
<!-- Description -->
<dc:title>
CONSTRUCTION OF THE JAKOMäKI INTERCHANGE: electronic version</dc:title>
<!-- Relations -->
<!-- up -->
<dc:relation elias:type="IsPartOf" elias:specifier="subcorpus"
  elias:target="elias-rakvir" />
<!-- down -->
<dc:relation elias:type="References" elias:specifier="appendix"
  elias:target="jakomaki-1-1979-kartta" />
<!-- vertical -->
<dc:relation elias:type="HasVersion" elias:specifier="document"
  elias:target="jakomaki-1-1979-image" />
<!-- other -->
</rdf:Description>
```

The image of the text document has its own description, which only includes references to its XML counterpart and its appendix:

```
<rdf:Description about="file:../kuvat/jakomaki-1-1979.gif" ID="jakomaki-1-1979-image">
<dc:title>
CONSTRUCTION OF THE JAKOMÄKI INTERCHANGE: electronic version / image</dc:title>
<!-- Relations -->
<!-- down -->
<dc:relation elias:type="References" elias:specifier="appendix"
elias:target="jakomaki-1-1979-kartta" />
<!-- vertical -->
<dc:relation elias:type="IsVersionOf" elias:specifier="document"
elias:target="jakomaki-1-1979" />
</rdf:Description>
```

At the lowest level of the corpus structure we have the appendix files. The Jakomaki-1-1979 documents described above refer to an appendix - the map of the suburb Jakomäki. The description of the map is very simple, it only refers to (both, xml and jpeg, versions of) its referring document:

```
<rdf:Description about="file:../kuvat/jakomaki-1-1979-kartta.gif"
ID="jakomaki-1-1979-kartta">
<!-- Description -->
<dc:title>
MAP OF THE JAKOMÄKI INTERCHANGE: electronic version / image</dc:title>
<!-- Relations -->
<!-- up -->
<dc:relation elias:type="IsReferencedBy" elias:specifier="document"
elias:target="jakomaki-1-1979" />
<dc:relation elias:type="IsReferencedBy" elias:specifier="document"
elias:target="jakomaki-1-1979-image" />
</rdf:Description>
```

The multilevel corpus structure, including the main corpus (supercorpus), subcorpora, individual texts and appendices, can be seen in figure 2. The arcs represent the relations between the RDF descriptions. In the link-base the relations are expressed by the Dublin Core dc:relation-element. The IsPart- and HasPart -relations connect the corpora and the texts, and the IsVersionOf- and HasVersion -relations connect the texts in XML and JPG formats. The possible appendices are connected to the referring texts with the References- and IsReferencedBy -relations that also connect the subcorpus to external information.

7 Modifications to the specifications

We have augmented the TEI specifications by adding the msd-attribute to the w-element. We did this in order to include the morphosyntactic description of the words in the text. No other alternations to the TEI DTD were necessary for the encoding of the press release material. The attribute is adopted from the CES, and it has been recommended by the LBF (see [15]).

When building the DTD for the RDF/DCMI link-base, we took the W3C RDF DTD and added two extra namespaces on it, one for the Dublin Core elements and attributes, and the other for the elias-attributes.

```
<!ENTITY dcns 'http://purl.org/dc/elements/1.1/' >
<!ENTITY % dcnsdecl 'xmlns:dc CDATA #FIXED "&dcns;"' >
<!ENTITY eliasns 'file:/hankkeet/elias/dokumentit/dc-extension/' >
<!ENTITY % eliasnsdecl 'xmlns:elias CDATA #FIXED "&eliasns;"' >
<!ATTLIST %RDF; %dcnsdecl;>
<!ATTLIST %RDF; %eliasnsdecl;>
```

In the original RDF DTD, the rdf:li-element was not defined inside the elements rdf:sequence, rdf:bag, or rdf:alternative, so the definition had to be included (the following example covers only the sequence-element):

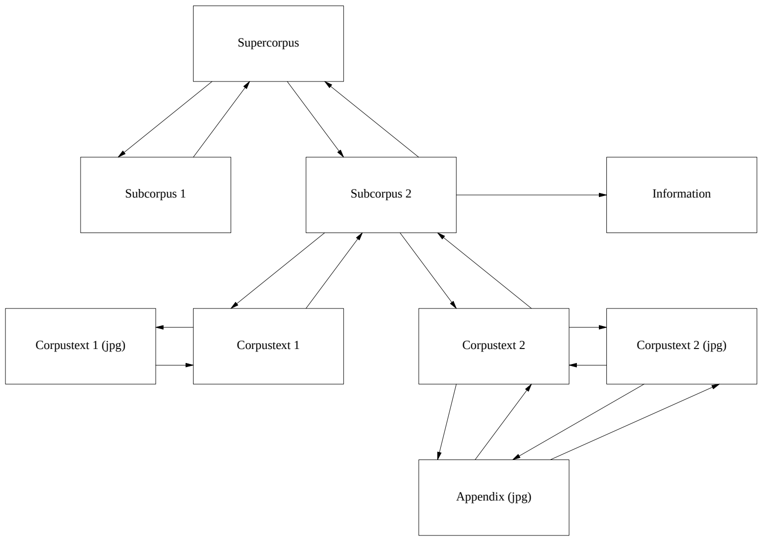


Fig. 2. Three-level corpus structure in RDF.

```
<!ENTITY % member %li;>
<!ELEMENT %sequence; (%member;)*>
<!ATTLIST %sequence;
            %idAttrOpt; >
```

The Dublin Core elements had to be included in the entity propertyEltstar, which is the RDF DTD's interface for the content elements.

```
<!ENTITY % dcmes "dc:title | dc:creator | dc:subject | dc:description |
dc:publisher | dc:contributor | dc:date | dc:type | dc:format |
dc:identifier | dc:source | dc:language | dc:relation | dc:coverage |
dc:rights" >

<!ENTITY % propertyEltStar "%dcmes;"> <!-- @@EXPORT! redefine this
                                if you want to validate your properties -->
```

Entity propertyDesc was created in order to include RDF containers (rdf:sequence, rdf:bag, and rdf:alternative) in the Dublin Core elements:

```
<!ENTITY % propertyDesc "#PCDATA |%container;">
```

And, finally, the Dublin Core elements were defined so that they may include PCDATA and RDF containers. The elias-attributes were included in the description of the dc:relation-element.

```
<!ELEMENT dc:title (%propertyDesc;)*>
<!ATTLIST dc:title rdf:resource CDATA #IMPLIED>
```

```
<!ELEMENT dc:relation (%propertyDesc;)*>
<!ATTLIST dc:relation rdf:resource CDATA #IMPLIED>
<!ATTLIST dc:relation elias:type ( IsPartOf | HasPart | IsVersionOf |
          HasVersion | IsFormatOf | HasFormat | References | IsReferencedBy |
          IsBasedOn | IsBasisFor | Requires | IsRequiredBy | Unspecified )
          "Unspecified">
<!ATTLIST dc:relation elias:specifier ( supercorpus | subcorpus | textcollection |
          document | appendix | information | unspecified ) "unspecified">
<!ATTLIST dc:relation elias:target IDREF #IMPLIED>
```

8 Conclusions

We have introduced specifications for adding structure to text material on different level. These specifications can be used in building multilevel research corpora. The basic encoding and structuring is expressed in TEI, slightly modified to include the morphosyntactic description. The multilevel corpus structure, however, cannot be satisfactorily expressed with the TEI, so we introduced another data structure. It is a directed, external RDF link-base, which we designed following recommendations given by the W3C and the DCMI.

The main motivation in building the maximally analysed TEI corpus and the directed RDF link-base is the need for tools for both quantitative and qualitative corpus linguistics. These may include tools for browsing the corpus and building arbitrary subcorpora from the text collections, programs for counting key figures on the basis of the TEI markup etc. The next phase in the project will consist of the building of prototypes of such tools and making the above specifications useful for real life linguistic research.

References

1. Biber, Douglas: Representativeness in corpus design. Literary & Linguistic Computing Vol 8, Number 4. Association for Literary and Linguistic Computing (1993)
2. Biber, Douglas, Conrad, Susan, Reppen, Randi: Corpus Linguistics. Investigating Language Structure and Use. Cambridge University Press, Gambridge (1998)
3. Garside, Roger, Leech, Geoffrey, McEnery, Anthony (eds.): Corpus Annotation. Linguistic Information from Computer Text Corpora. Addison Wesley Longman, Harlow, England (1997)
4. Hakulinen, Auli, Karlsson, Fred, Vilkuna, Maria: Suomen tekstilauseiden piirteitä: kvantitatiivinen tutkimus. Publications 6. Department of General Linguistics, University of Helsinki, Helsinki (1980)
5. Järvinen, Timo: Annotating 200 Million Words: The Bank of English Project. In COLING-94. The 15th International Conference on Computational Linguistics Proceedings, volume 1, Kyoto, Japan, pp. 565-568 (1994)
6. Kankaanpää, Salli: From letters to news reports. Diachronic changes in Finnish municipal press releases 1979 - 1999. In W. Vagle & K. Wikberg (eds): New Directions in Nordic Text Linguistics and Discourse Analysis: Methodological Issues.

Proceedings from the NordText Symposium, University of Oslo, January 7-9 2000. Oslo: Novus. S. 229 - 242. (2001)
7. Koskenniemi, Kimmo: Two-level Morphology: A General Computational Model for Word-Form Recognition and Production. University of Helsinki, Department of General Linguistics, Publications 11. Department of General Linguistics, University of Helsinki, Helsinki (1983)
8. McEnery, Tony, Wilson, Andrew: Corpus Linguistics. Edinburgh University Press, Edinburgh (1996)
9. Saukkonen, Pauli: Oulun korpus. 1960-luvun suomen yleiskielen tutkimusmateriaali. Oulun yliopiston suomen ja saamen kielen laitoksen tutkimusraportteja 1. Oulu (1982)
10. British National Corpus (online). http://www.hcu.ox.ac.uk/BNC/ Oxford University Computing Services
11. Corpus Encoding Standard (online). http://www.cs.vassar.edu/CES/
12. Corpus Encoding Standard, Directory of CES-Based Corpus Projects (online). http://www.cs.vassar.edu/CES/CES-P.html
13. Machinese Phrase Tagger (online). http://www.connexor.com/m_tagger.html Connexor
14. Kielipankki (online). http://www.csc.fi/kielipankki/ CSC
15. Language Bank Corpus DTD Documentation (online). http://www.csc.fi/kielipankki/aineistot/ohjeet/xml/dtd/DTD-HOME.phtml.en CSC
16. Dublin Core Metadata Initiative (online). http://www.dublincore.org/ DCMI
17. Expressing Simple Dublin Core in RDF/XML (online). http://www.dublincore.org/documents/2001/11/28/dcmes-xml/ DCMI
18. Relation Element Working Draft (online). http://dublincore.org/documents/relation-element/ DCMI
19. Collins Cobuild (online). http://titania.cobuild.collins.co.uk/ Department of English, Birmingham University
20. DCParser (online). http://www.kielikone.fi/english/demot/dcparser.shtml Kielikone Ltd.
21. Tags (Partial List) (online). http://www.lingsoft.fi/doc/fintwol/intro/tags.html Lingsoft Inc.
22. Text Encoding Initiative (online). http://www.tei-c.org/ TEI Consortium
23. The XML Version of the TEI Guidelines, 5 The TEI Header (online). http://www.tei-c.org/Guidelines/HD.html TEI Consortium
24. Extensible Markup Language (XML) (online). http://www.w3.org/XML/ W3C
25. Resource Description Framework (RDF) (online). http://www.w3.org/RDF/ W3C
26. Connolly, Dan: Valid RDF: Using Namespaces with DTDs (online). http://www.w3.org/XML/9710rdf-dtd/ W3C

Extracting More Relevant Document Descriptors using Hierarchical Information

Antoine Doucet[1,2]

[1] University of Helsinki,
Department of Computer Science,
P.O. Box 26 (Teollisuuskatu 23),
FIN-00014 University of Helsinki, Finland,
`antoine.doucet@cs.helsinki.fi`
[2] Université de Caen,
Département d'Informatique,
Campus Côte de Nacre,
F-14032 Caen Cedex, France,
`antoine.doucet@info.unicaen.fr`

Abstract. The emergence of a constantly growing quantity of semi-structured documents has provided considerable hierarchical information. However, most of the document processing techniques are still developed without regard to the hierarchical structure of the documents. This is especially true when the structure is only partial, in which case the preprocessing may simply consist in pruning the structure !

We present in this paper a method for adding structural information to text content descriptors. The resulting document descriptors take the form of word sequences, to which structural information has been added. The result of this process is a mapping from each document of the collection to a (possibly empty) set of descriptive text phrases. Following this ground work, many applications can be considered based on the extracted information.

This method was implemented and tested using a collection of French geographic articles. One example application was then tested, using the new set of descriptors: the discovery of co-occurring text phrases, based on classical algorithms for association rules discovery. An empirical comparison was then made between the results obtained via the "classical" and the proposed "structure-aware" method.

1 Introduction

Since a few decades ago, computer systems and networks have been exponentially growing. So has the amount of digital information stored in these systems: using automated data collection tools, massive amounts of data have been collected and stored in databases. Among this rising quantity of data, the proportion consisting of semi-structured text documents has exponentially increased. This growing number has provided numerous contextual information. Unfortunately, most text processing systems tend to ignore hierarchical information issued from semi-structured documents. Indeed, in most cases, a majority of the structure is irrelevant to the user (e.g., metadata), and it is therefore often entirely ignored. But since a few years ago, the emergence of the simple encoding standard XML [2] has brought valuable semantic information to the path leading to a text element.

In the case of very large text document collections, applying data mining techniques is a recent idea. Appearing in the second part of the nineties, the concept of data mining in text, known as *text mining*, has been of growing interest. It is especially appropriate when the user does not exactly know what he (or she) is looking for. In this context, the Doremi[3] research group (in the University of Helsinki, Finland) has developed a method for attaching compact content descriptors to full text documents: their maximal frequent sequences [4, 5]. A maximal frequent sequence is a sequence of words that occurs frequently in the document collection and moreover, that is not contained in any other longer frequent sequence. This permits the extraction of very compact content descriptors from the documents of the collection. Unfortunately, the method and experiments were based on plain text documents, rather than structured ones.

Nowadays, an increasing part of the newly created textual data is either semi- or fully structured. Hence, a natural improvement has been to meliorate the relevance of the results by taking the extra information provided by the document structure (be it full or partial) into account. This technique was then recently adapted to include hierarchical information, with a small influence on the algorithm's performance. The evidence of the empirical results has shown the extracted descriptors to have a better relevance. Once each of the documents has been associated with a bag of descriptors, the applications are numerous. In the result section we sketch one experiment we led, based on the discovery of association rules [3]. In this data mining application, we considered the sets of descriptors as the frequent itemsets, and discovered various relations between the descriptors.

In the remainder of this paper, we will summarize the existing method based on plain text documents in section 2. Section 3 will describe the new method: "Word to Pathword (W2PW)", adapted so as to take the documents' hierarchical structure into account. Then we will compare the data we extracted using the plain text method and the data we extracted using the enhanced method (section 4). In section 5, we sketch a comparison between association discovery results from the classical and from the proposed method. Finally, result analysis and a brief discussion will conclude this paper in section 6, opening issues regarding different ways to further develop this work.

[3] DOcument management, information REtrieval, text and data MIning

2 Maximal Frequent Sequences

The technique of extracting *Maximal Frequent Sequences* (MFS) from a document collection is extensively described in [4]. It exhibits details of the method used by the Doremi group for extracting document descriptors from a large plain text document collection. We will hereby summarize the main steps of that method and later remind of its specific strengths.

2.1 MFS: Definition and Extraction Technique

Three main steps are processed within the Doremi method. The general idea fits the main phases of KDD (Knowledge Discovery in Databases); that is, selection and cleansing of the data, followed by the use of core mining techniques, and a final postprocessing step intending to transform and select the results into an understandable knowledge.

Definition of MFS. Assuming S is a set of documents, and each document consists of a sequence of words...

Definition 1. *A sequence $p = a_1 ... a_k$ is a* subsequence *of a sequence q if all the items a_i, $1 \leq i \leq k$, occur in q and they occur in the same order as in p. If a sequence p is a subsequence of a sequence q, we also say that p* occurs *in q.*

Definition 2. *A sequence p is* frequent *in S if p is a subsequence of at least σ documents of S, where σ is a given frequency threshold.*

Note that only one occurrence of a sequence within a document is counted: whether a sequence occurs once or several times within the same document does not change its frequency.

Definition 3. *A sequence p is a* maximal frequent (sub)sequence *in S if there does not exist any sequence p' in S such that p is a subsequence of p', and p' is frequent in S.*

Phase 1: Preprocessing This first phase basically consists in 'clearing' the data of its useless information. Of course, whether an item is useful or not strongly depends on the intended use of the results. Usually, special characters (including, for example, punctuation and brackets) are pruned away. To avoid processing uninteresting items, a stopword list is also created. It includes articles, pronouns, conjunctions, common adverbs, and common forms of non-informative verbs (e.g., *is, are, be*). All of its elements are removed.

Typically, the two following text fragments:

```
...President of the United States Bush...
...President George W. Bush...
```

would result in:

```
...President United States Bush...
...President George Bush...
```

Phase 2: Extraction Technique (an overview).

Initial phase: Collecting all Frequent Pairs. In this initial phase, all pairs of words, such that their frequency is greater than a given threshold, σ (10 in the experiment), are collected. Two words form a pair if they occur in the same document, and if their distance is less than a given maximal gap. A gap of 2 was used in the experiment, which means that at most 2 other words can appear between the words forming a pair. Also, note that the pairs are ordered, i.e. the pairs (A,B) and (B,A) are different.

Expanding the frequent pairs to MFS's. For each step k, $Grams_k$ is the number of frequent sets of length k. Hence, the frequent pairs found in the initial phase form $Grams_2$. MFS's are found bottom-up by combining shorter frequent sequences (of length k) to form longer sequences (of length $k+1$). Each step also includes various pruning stages, so as to ensure computational efficiency.

Further details about the different phases of discovery would be highly technical and irrelevant for this paper's purpose. Please refer to [4] if you want to know more.

Phase 3: Postprocessing. However, by computing maximal frequent sequences, and by increasing the length of the selected sequences, the corresponding frequencies naturally tend to decrease toward the minimum frequency threshold. Sequences that are both shorter and more frequent have not been selected, even if they might carry more valuable information. This might be especially true if the sequence's frequency was much higher, by taking a few words out of it.

Therefore, more sequences are added to the final set of content descriptors (based on [6]), which will be used in the next phases. For each maximal frequent sequence, any of its subsequences responding to both of the following criterions will be selected:

- its frequency is bigger than the corresponding maximal frequent sequence's
- it is not the subsequence of some descriptive sequence having an equal frequency

This way, in the resulting sets of descriptive sequences, the sequences' sizes are optimized by the maximal frequent sequences, and the sequences' frequencies are optimized by the subsequences added afterwards.

Finally, as a result, an (eventually empty) list of content descriptors is attached to each document of the collection.

2.2 Main Strengths of the Method

The method efficiently extracts all the maximal frequent word sequences from the collection. From the definitions above, a sequence is said to be maximal if and only if no other frequent sequence contains that sequence.

Furthermore, a *gap* between words is allowed: in a sentence, the words do not have to appear continuously: a parameter g tells how many other words two words in a sequence can have between them. The parameter g usually gets values between 1 and 3.

For instance, if g = 2, a phrase "president Bush" will be found in both of the following text fragments:

```
...President of the United States Bush...
...President George W. Bush...
```

Note: Articles and prepositions were pruned away during the preprocessing.

This allowance of gaps between words of a sequence is probably the strongest specificity of the method, compared to the other existing methods for extracting text descriptors. This greatly increases the quality of the phrase, since processing takes the variety of natural language into account. The method is *style tolerant.* Even deficient syntax can be handled (and that is fairly common in newswires, for example).

The other powerful specificity of the Doremi technique is the ability to extract maximal frequent sequences of any length. This offers a very compact description. By example, by restricting the length of phrases to 8, the presence, in the document collection, of a frequent 25 words long phrase, would result in thousands of phrases representing the same knowledge as the one maximal sequence.

3 The "Word to PathWord" Method (W2PW)

The main idea of this structure-aware method is to modify the notion of a word. Each word of the collection is bound to its corresponding path in the structure's hierarchy. For example, in a recipes book, by applying this modification, potato may become either "/book/chapter/appetizers/recipe/ingredients/potato" or "/book/chapter/side_dishes/recipe/ingredients/potato". The point is that these 2 occurrences of the same word will be considered different, because they occur within different paths. This might be particularly helpful for homonyms (i.e., words sharing the same spelling but a different meaning).

The bad aspect of this approach is that it may be misleading in some cases: Thinking of an HTML document for example, being written under the style "bold" or with a different style (or without any) would make a word different. A difference on a lower level of the structure tree would also make a difference: A word would be considered different, depending on whether it is referred (by the structure) directly in a chapter or in a subchapter of a main chapter.

StopList of tags. Based on this example, if "potato" could indeed be considered different whether mentioned as an appetizer ingredient (e.g., crisps) or as a side-dish, on the other hand, one can complain that being included within a subchapter or not shouldn't make any difference:
"/book/chapter/appetizers/recipe/ingredients/potato" should be equivalent to "/book/chapter/subchapter/appetizers/recipe/ingredients/potato".

Therefore, the ability to create and use a stoplist of structure tags was naturally needed to separate relevant and non-relevant hierarchical information. This way, tags not carrying any important information, such as <chapter>, <subchapter> or <book> will not be considered. That pruning list also solves the "text style" problem: if we consider that the HTML tag <B> (for bold) brings a misleading information by making "<B>potato</B>" differ from "potato", then "<B>" should simply be included in the *"stoptag" list*.

Of course, as in any pruning step, the selection of the tags that should be included in the stoplist is strongly related to the intended use of the data (e.g., the font style might actually be what the user is interested in !). This feature makes the method highly adaptive to the user's interests.

Further Details. The switch from word to "path+word" is made by reading the data from the standard input and:

- storing the current path.
- printing each word preceded by the current path to the standard output.

This is clearly of linear complexity, since it only requires one pass over the data, and a single structure to store the current path. But we also need to take care of non-closed tags, such as
 (i.e., a newline in an HTML document).

To get rid of these single tags, a step was added to the preprocessing phase. Its principle is when a closing tag is met, then it is checked to determine whether it equals the last part of the *current_path*. If not, then this last part (i.e., the last opened tag) will never be closed, since we assume that the data is well-formed. Therefore, a backward search is made, so as to delete that never-closed tag.

This part of the algorithm can be considered as problematic in the case where numerous empty tags are present. But this is untrue in practice, since empty tags should be deleted during the preprocessing, via the stoplist of tags (as those empty tags cannot occur in any final result anyway). This process is thus done in linear time. This is why this backward search remains safe in practice: It is only used to deal with rare irregularities of the data.

4 Experiments

The Word to Pathword (W2PW) method was implemented in Perl. We tested it by comparing the result obtained by the former method (plain text) and the proposed W2PW method. This was done by using the French-written semi-structured document collection of the Cross-Channel Atlas [9], a 205 documents collection created through electronic sharing of geography articles, focused on the transborder relations between France and Great Britain, and fed by researchers from both countries. A sample of one of these semi-structured documents is shown in figure 1. The experiments focus on the content descriptors obtained via both techniques, comparing their amount, size and relevance.

```
<HEAD><TITLE>Situation des liaisons maritimes passagers (hiver 1998/1999)
</TITLE><DATE>1999-3-1</DATE><AUTHOR><NAME><LAST>Buléon
</LAST><FIRST>Pascal</FIRST></NAME><EMAIL>buleon@mrsh.unicaen.fr
</EMAIL></AUTHOR><AUTHOR><NAME><LAST>Lefevre</LAST><FIRST>Samuel
</FIRST></NAME><EMAIL>samuel.lefevre@criuc.unicaen.fr</EMAIL>
</AUTHOR><DESCRIPTOR>Fusion</DESCRIPTOR><DESCRIPTOR>compagnies
de ferries</DESCRIPTOR><DESCRIPTOR>liaisons maritimes</DESCRIPTOR>
</HEAD><BODY><FONT SIZE=2><P ALIGN="JUSTIFY"><CENTER>
<IMG SRC="lecteur.php?base=atlas&cmde=image&refimage=212">
</CENTER></P></FONT><P ALIGN="JUSTIFY"></P><P ALIGN="JUSTIFY">Les
<linkid>liaisons maritimes</linkid> passagers ont connu une
nouvelle évolution depuis l'automne 1998. Celle-ci est une
nouvelle conséquences de la <linkid>fusion</linkid> P&O et
prolonge celles survenues dans le cours de l'année 98.
(...)
```

Fig. 1. A sample of a document from the cross-channel atlas

4.1 Discovery of Content Descriptors

The stoplist of tags contains the common non-closed tags from the document collection, which are <BR> and <IMG>. It also includes those tags, that are not carrying relevant information, with respect to our interest. In this experiment, we considered the style of a document as irrelevant information. Therefore, such tags as <FONT>, <P>, <HEAD>, and <BODY> have been pruned away.

The whole preprocessing phase condensed the data from 74,754 words to 40,714 words. Note that each opening and closing tag is here counted as a word.

Number of maximal frequent sequences discovered. Table 1 gives a summary of the number of maximal frequent sequences discovered, depending on the method and frequency threshold.

Since the W2PW method makes some words different from each other, we could have expected that the number of content descriptors would be lower for this method, compared to the classical one, based on the same frequency threshold. However, it is actually quite similar. Besides that, one can observe that the number of content descriptors found by W2PW declines at a slower pace when the frequency threshold raises.

The reason for this difference is that many highly frequent words are found within the header of the documents, which is pruned away in the classical

Frequency Threshold	2	3	4	5	6	7	8	9	10	15	20
Classical Method	3,780	1,096	456	236	132	81	50	32	27	8	5
W2PW	3,816	1,137	493	260	162	113	75	53	42	20	9

Table 1. Number of maximal frequent sequences depending on method and frequency threshold

method. This fact compensates the average frequency reduction due to these newly different words. Confirmation occurs when the frequency threshold becomes unreasonably high. The number of maximal frequent sequences extracted through W2PW then becomes even larger than through the classical method. Indeed, the more the frequency threshold rises, the more those highly frequent words, occurring within the header, get proportionally numerous within the frequent ones.

During the next steps of the experiment, 4 was chosen as a frequency threshold. A larger number would lead to a lack of material for the following steps, whereas a lower number might select irrelevant descriptors, since we only consider a 205 documents collection.

Length of the Phrase	2	3	4-6	7-10	11-15	16-25	26+
Classical Method	444	9	1	1	1	0	0
W2PW	426	17	21	10	9	2	1

Table 2. Classical vs. W2PW, number of maximal phrases of various length

Length of the maximal frequent sequences. Looking at table 2, the main observation is that the longest sequences are found via the new method. This is due to the fact that long sequences are mostly found in the headers, and that a few words always occur together: for instance, the same last name with the same first name and the same email address. If a group of persons is often co-writing documents about similar themes, then the descriptive phrase can easily become quite long. For example, the phrase in figure 2 was found in 7 documents of the collection.

```
/date/NR   /author/name/last/loew-pellen   /author/name/first/frédérique
/author/email/loew@mrsh.unicaen.fr   /author/name/last/winter
/author/name/first/ansgar   /descriptor/emploi   /descriptor/indicateurs
/descriptor/population   /descriptor/ville   /some   /linkid/regional
/linkid/indicators   /about   /population   /and   /employment   /with
/map   /of   /travel   /to   /work   /areas
```

Fig. 2. An instance of a long descriptor (size 24).

It shows that Frédérique Loew-Pellen and Ansgar Winter are frequently co-writing documents, that they described by "emploi", "indicateurs", "population", and "ville" (entrance descriptors). These documents include links (i.e., exit descriptors) through the phrase "regional indicators". The plain text sequence "about population and employment with map of travel to work areas" then occurs within the body of the document.

5 An application: the Discovery of Associations

From these content descriptors, we can compute the association rules discovery, in a similar way as in [6].

This was done with various confidence and support threshold values. For both methods, the main difficulty was to pick a correct support threshold. Indeed, the collection of documents is not extremely large, and a 1% variation within the support differs the number of associations rules produced from 20 to thousands! This is also due to a large amount of phrases, occurring with similar frequencies, of type "region, city", such as "Hampshire, Southampton". Indeed, one of these typical association rules is:

```
/table/fareham_/table/hampshire
    =>
/table/hampshire_/table/southampton (1.00,0.04)
```

...meaning that when the phrase *(fareham, hampshire)* occurs in a document, then the phrase *(hampshire, southampton)* always occurs in the same document (confidence = 1 = 100%). Both of the phrases occur together in 4% of the documents of the collection. Also, observe that this association rule stems from the W2PW method. This offers additional information: these phrases were found in a table. The same association rule occurs, using the classical method. The only difference is that one piece of information is missing: nothing shows that this data is related to a table.

And indeed, this is the main difference between the 2 methods: W2PW effectively adds information to the results.

5.1 Extra Information is added...

Below is an even better example of what this hierarchical extra information can bring, comparing the classical method's association rules to the new one's.

- Classical method:
  `buléon_pascal  => transport_régional (0.86,0.03)`
  From this association rule, based on the classical method, the extracted information is that: *"Buléon Pascal" and "regional transport" are somehow related.* But we do not learn anything about the relation itself.

– Word to PathWord:
```
/author/name/last/buléon_/author/name/first/pascal
    =>
/descriptor/transport_/descriptor/regional (0.86,0.03)
```
Looking at the association rule based on the new method, the extracted information is now that: *Pascal Buléon is an author, "Pascal" is his first name, and "Buléon" his last. In addition, 86% of the documents he wrote are dealing with regional transportation.*

5.2 ...but a counterpart may exist

We can also notice that the support and confidence may differ between results related to the same phrases. For example:

– Classical method:
```
jersey_guernesey  => îles_anglo-normandes (0.78,0.05)
```
– Word to PathWord:
```
/jersey_/guernesey  => /îles_/anglo-normandes (0.88,0.04)
```

This means that one or more occurrences of "/îles_anglo-normandes" were found within a different path, and thus not counted together (because it is not considered as the same entity). A consequence is that some descriptors will not be found, using the W2PW, even though they were extracted by the classical method. This happens in the cases where that phenomenon pushes the frequency of a phrase below the threshold.

But it is not certain that the consequently unextracted descriptors should be regretted. Actually, it relies on the quality of the stoplist of tags. If the latter is appropriate, then those identical words that are differing only due to their paths should not be mixed, indeed. Based on a good stoplist of tags, this frequency lowering is actually very valuable.

6 Results and Discussion

We have successfully added structural information to the content descriptors of the documents, via the maximal frequent sequences discovery technique. A new method, the Word to PathWord, has been presented, implemented and tested. Some important observations can be formulated.

One of its strength is that it does not require the documents to be valid. In other words, no DTD is required. The only assumption is that the documents must be well-formed, that is, no overlap is allowed.

Also, hierarchical knowledge is efficiently added to the data, in the way that the structural information gives further details about the content descriptors, e.g., their structural context (table, item-list...), style context (bold, italic, emphasized...), or any other hierarchical information from the data.

As already underlined, the quality of the stoplist of tags is a crucial element, both for a better computational efficiency (by pruning empty tags) and for a better relevance (by dropping irrelevant tags). The latter implies that building the list requires a good knowledge of what the end user is interested in (i.e., what is relevant *to him/her*). That's why any use of the method on another collection of documents requires particular attention in building the stoplist of tags.

However, many further developments and ameliorations shall be considered. Among these, a way to take advantage of the information from the tag's attributes has to be added to the method features. Also, the computational efficiency of the method could be improved, by deepening the techniques, if they were to be adapted to specific data or for a specific purpose, whereas in this paper, the data spectrum was kept as broad as possible. The lack of a *very large* and publicly-available semi-structured document collection has also been problematic. Fortunately, such a collection has recently been released, through the "Initiative for the Evaluation of XML retrieval" (INEX) [8, 1]. This new dataset will permit further testing from a more significant amount of information.

It is clear that the extraction and binding of a set of content descriptors to each document of a collection gives various applicative prospects. These sets of descriptors representing the documents may be used so as to measure distances between the documents and thus form clusters of documents. Another possibility is to use these descriptors to dynamically create hyperlinks between the documents [7].

Acknowledgements

This work is supported by the Academy of Finland (project 50959; DoReMi - Document Management, Information Retrieval, and Text Mining).

References

1. Initiative for the evaluation of xml retrieval, Available at http://qmir.dcs.qmw.ac.uk/INEX/. [Cited 23 September 2002].
2. Extensible markup language (xml), Available at http://www.w3.org/XML/. [Cited 23 September 2002].
3. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining, Menlo Park, California, USA*, pages 307–328. AAAI Press, 1996.
4. H. Ahonen-Myka. Finding All Frequent Maximal Sequences in Text. In *Proceedings of the 16th International Conference on Machine Learning ICML-99 Workshop on Machine Learning in Text Data Analysis, Ljubljana, Slovenia*, pages 11–17. J. Stefan Institute, eds. D. Mladenic and M. Grobelnik, 1999.
5. H. Ahonen-Myka. Discovery of frequent word sequences in text. In *The ESF Exploratory Workshop on Pattern Detection and Discovery in Data Mining, Imperial College, London, UK*, pages 180–189, 2002.
6. H. Ahonen-Myka, O. Heinonen, M. Klemettinen, and A. I. Verkamo. Finding Co-occurring Text Phrases by Combining Sequence and Frequent Set Discovery. In *Proceedings of 16th International Joint Conference on Artificial Intelligence IJCAI-99*

Workshop on Text Mining: Foundations, Techniques and Applications, Bled, Slovenia, pages 1–9. ed. R. Feldman, 1999.
7. B. Crémilleux, M. Gaio, J. Madelaine, and K. Zreik. Discovering browsing paths on the web. In *Third International Conference on Human-System Learning, CAPS 2000, Paris (France)*. Europia, 2000.
8. N. Fuhr, N. Goevert, G. Kazai, and M. Lalmas. Inex: Intitiative for the evaluation of xml retrieval. In *ACM SIGIR Workshop on XML and Information Retrieval, Tampere, Finland*, 2002.
9. M. Gaio, M. Smurzlo, L. Thomazo, and C. Turbout. A collaborative editing system. In *GisPlanet'98, International Conference and Exhibition on Geographic Information, Lisbon, Portugal*, 1998.