Journal Subline

LNCS 5383

Journal on Data Semantics XI

Stefano Spaccapietra
Editor-in-Chief

Springer

Lecture Notes in Computer Science 5383

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK
Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
University of Surrey, Guildford, UK
Jon M. Kleinberg
Cornell University, Ithaca, NY, USA
Alfred Kobsa
University of California, Irvine, CA, USA
Friedemann Mattern
ETH Zurich, Switzerland
John C. Mitchell
Stanford University, CA, USA
Moni Naor
Weizmann Institute of Science, Rehovot, Israel
Oscar Nierstrasz
University of Bern, Switzerland
C. Pandu Rangan
Indian Institute of Technology, Madras, India
Bernhard Steffen
University of Dortmund, Germany
Madhu Sudan
Massachusetts Institute of Technology, MA, USA
Demetri Terzopoulos
University of California, Los Angeles, CA, USA
Doug Tygar
University of California, Berkeley, CA, USA
Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Stefano Spaccapietra   Jeff Z. Pan
Philippe Thiran   Terry Halpin
Steffen Staab   Vojtech Svatek
Pavel Shvaiko   John Roddick (Eds.)

Journal on Data Semantics XI

Springer

Volume Editors

Stefano Spaccapietra
EPFL-IC-IIF-LBD, Lausanne, Switzerland
E-mail: stefano.spaccapietra@epfl.ch

Jeff Z. Pan
The University of Aberdeen, UK; jeff.z.pan@abdn.ac.uk

Philippe Thiran
Namur University, Belgium; philippe.thiran@fundp.ac.be

Terry Halpin
Neumont University, Salt Lake City, UT, USA; terry@neumont.edu

Steffen Staab
Universität Koblenz-Landau, Germany; staab@uni-koblenz.de

Vojtech Svatek
University of Economics, Prague, Czech Republic; svatek@vse.cz

Pavel Shvaiko
TasLab, University of Trento, Povo, Italy; pavel@dit.unitn.it

John Roddick
Flinders University, Adelaide, Australia; roddick@cs.flinders.edu.au

Library of Congress Control Number: Applied for

CR Subject Classification (1998): H.2, H.3, I.3, C.2, H.4.3, D.3

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/
Web and HCI

ISSN	0302-9743 (Lecture Notes in Computer Science)
ISSN	1861-2032 (Journal on Data Semantics)
ISBN-10	3-540-92147-8 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-92147-9 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2008
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12585670 06/3180 5 4 3 2 1 0

The LNCS Journal on Data Semantics

Computerized information handling has changed its focus from centralized data management systems to decentralized data exchange facilities. Modern distribution channels, such as high-speed Internet networks and wireless communication infrastructure, provide reliable technical support for data distribution and data access, materializing the new, popular idea that data may be available to anybody, anywhere, anytime. However, providing huge amounts of data on request often turns into a counterproductive service, making the data useless because of poor relevance or inappropriate level of detail. Semantic knowledge is the essential missing piece that allows the delivery of information that matches user requirements. Semantic agreement, in particular, is essential to meaningful data exchange.

Semantic issues have long been open issues in data and knowledge management. However, the boom in semantically poor technologies, such as the Web and XML, has boosted renewed interest in semantics. Conferences on the Semantic Web, for instance, attract big crowds of participants, while ontologies on their own have become a hot and popular topic in the database and artificial intelligence communities.

Springer's LNCS *Journal on Data Semantics* aims at providing a highly visible dissemination channel for remarkable work that in one way or another addresses research and development on issues related to the semantics of data. The target domain ranges from theories supporting the formal definition of semantic content to innovative domain-specific application of semantic knowledge. This publication channel should be of the highest interest to researchers and advanced practitioners working on the Semantic Web, interoperability, mobile information services, data warehousing, knowledge representation and reasoning, conceptual database modeling, ontologies, and artificial intelligence.

Topics of relevance to this journal include:

- Semantic interoperability, semantic mediators
- Ontologies
- Ontology, schema and data integration, reconciliation and alignment
- Multiple representations, alternative representations
- Knowledge representation and reasoning
- Conceptualization and representation
- Multimodel and multiparadigm approaches
- Mappings, transformations, reverse engineering
- Metadata
- Conceptual data modeling
- Integrity description and handling
- Evolution and change
- Web semantics and semi-structured data
- Semantic caching

- Data warehousing and semantic data mining
- Spatial, temporal, multimedia and multimodal semantics
- Semantics in data visualization
- Semantic services for mobile users
- Supporting tools
- Applications of semantic-driven approaches

These topics are to be understood as specifically related to semantic issues. Contributions submitted to the journal and dealing with semantics of data will be considered even if they are not from the topics in the list.

While the physical appearance of the journal issues is like the books from the well-known Springer LNCS series, the mode of operation is that of a journal. Contributions can be freely submitted by authors and are reviewed by the Editorial Board. Contributions may also be invited, and nevertheless carefully reviewed, as in the case for issues that contain extended versions of the best papers from major conferences addressing data semantics issues. Special issues, focusing on a specific topic, are coordinated by guest editors once the proposal for a special issue is accepted by the Editorial Board. Finally, it is also possible that a journal issue be devoted to a single text.

The journal published its first volume in 2003 (LNCS 2800). That initial volume, as well as volumes II (LNCS 3360), V (LNCS 3870), VIII (LNCS 4380), IX (LNCS 4601), and this volume XI represent the annual occurrence of a special issue devoted to publication of selected extended versions of the best conference papers from conferences of the year before. Volumes III and VI were special issues on a dedicated topic. Volume III (LNCS 3534), coordinated by guest editor Esteban Zimányi, addressed Semantic-Based Geographical Information Systems, while volume VI (LNCS 4090), coordinated by guest editors Karl Aberer and Philippe Cudre-Mauroux, addressed Emergent Semantics. Volumes IV (LNCS 3730) and VII (LNCS 4244) were "normal" volumes, built from spontaneous submissions on any of the topics of interest to the journal.

The Editorial Board comprises an Editor-in-Chief (with overall responsibility), a Co-editor-in-Chief, and several members. The Editor-in-Chief has a four-year mandate. Members of the board have a three-year mandate. Mandates are renewable and new members may be elected anytime.

We are happy to welcome you to our readership and authorship, and hope we will share this privileged contact for a long time.

Stefano Spaccapietra
Editor-in-Chief
http://lbdwww.epfl.ch/e/Springer/

JoDS Volume XI

To foster the dissemination of the best ideas and results, the *Journal on Data Semantics* (JoDS) pursues a policy that includes annually publishing extended versions of the best papers from selected conferences whose scope encompasses or intersects the scope of the journal.

This initiative is motivated by the difference in goals between conferences and journals. Conferences usually have a faster turnaround and a focused audience, but they have to enforce space limitation and a fixed time frame, with no chances for improving a paper by producing multiple versions. In contrast, journals offer more space, room for debate and refinement, and are usually considered the real archival venue.

Therefore, the publication of an extended version of a conference paper is a much appreciated opportunity for researchers to widely disseminate a significantly improved presentation of their work, where they can develop the appropriate motivations, reasoning, results and comparative analysis. Moreover, by gathering the best papers from various conferences, JoDS special issues provide a unique opportunity for researchers to find in a single publication every year the best of ongoing research in the field of data semantics.

For this issue, papers from the following 2006 international workshops and conferences were invited:

- The Second International Workshop on Semantic Web-Enabled Software Engineering (SWESE 2006), held at the 5th International Semantic Web Conference (ISWC 2006). November 6, 2006, Athens, GA, USA. Guest editor: Jeff Z. Pan
- InterDB 2006, The Second International Workshop on Database Interoperability, Held in conjunction with ICDE 2006. April 3, 2006, Atlanta, USA. Guest editor: Philippe Thiran
- The Second International Workshop on Object-Role Modeling (ORM 2006), Montpellier, France, November 2–3, 2006, in conjunction with the On The Move Federated Conferences (OTM 2006). Guest editor: Terry Halpin
- EKAW 2006 - 15th International Conference on Knowledge Engineering and Knowledge Management - Managing Knowledge in a World of Networks, Podebrady, Czech Republic, October 2006. Guest Editors: Steffen Staab and Vojtech Svatek
- The International Workshop on Ontology Matching (OM-2006), co-located with the 5th International Semantic Web Conference (ISWC 2006), November 5, 2006, Athens, Georgia, USA. Guest editor: Pavel Shvaiko
- Third International Workshop on Conceptual Modeling for Geographic Information Systems (CoMoGIS 2006), and First International Workshop on Semantic Web Applications: Theory and Practice (SemWAT 2006), both in conjunction with the 25th International Conference on Conceptual Modeling (ER 2006). Guest editor: John Roddick

Papers were invited based on their quality, relevance and significance, and the viability of extending their results. Extended versions prepared by authors were subject to the traditional two-round scholarly review process, and the authors were required to respond to all concerns expressed by the reviewers before papers were accepted. Eight papers were eventually accepted for publication in this issue.

The selection of SWESE best papers eventually resulted in the acceptance of two papers.

The first paper "Experiences in the Design of Semantic Services Using Web Engineering Methods and Tools," by Brambilla, Ceri, Celino, Cerizza, Della Valle, Facca, Turati, and Tzviskou, shows how classical software engineering methods (such as formal business process development and automatic code generation) combine with semantic methods and tools (i.e., ontology engineering, semantic service annotation and discovery) to forge a new approach to software development for the Semantic Web. In the paper, the authors present their experience in the participation to the Semantic Web Service Challenge 2006, where the proposed approach achieved very good results in solving the proposed problems.

The second paper "Automatically Generated Model Transformations Using Ontology Engineering Space," by Roser and Bauer, presents an approach to using the semantic technologies to improve cross-organizational modeling by automated generation of model transformations. By automated generation of mappings it offers new possibilities for the integration of domain specific languages and 'legacy' models in a plug&play manner, making it easier for new organizations to join collaborations.

The extended version of the InterDB 2006 paper "The Harmony Integration Workbench" by Mork, Seligman, Rosenthal, Korb and Wolf presents a workbench for schema integration in which multiple tools share a common knowledge repository. The tool supports the interoperation of existing tools through a common view of schemas and mappings. The workbench is intended to help integration engineers to use and to select the best tool for an integration problem.

One extended article was selected from ORM 2006. The paper "Using ORM-Based Models as a Foundation for a Data Quality Firewall in an Advanced Generation Data Warehouse (Extended version)" by Piprani reports on the use of the Object-Role Modeling (ORM) approach to establish a data quality firewall architecture for an industrial data warehouse. The paper describes a metadata repository framework for extracting data from heterogeneous sources, cleansing and transforming the data for loading into the data warehouse, and then generating specific data marts. Techniques for performing detailed quality checks are described, along with auditing and control measures. The outlined approach has proven its value in a variety of industrial applications, typically resulting in 35-40% savings in overall project costs.

The EKAW 2006 conference contributed with an extended version of the paper "Discovering Semantic Sibling Groups from Web Documents with XTREEM-SG" by Brunzel and Spiliopoulou. The paper addresses a novel issue in ontology learning: the exploitation of term collections embedded in HTML mark-up structures. The particular targets are semantic siblings, i.e., co-hyponyms and co-meronyms in the taxonomic structure of an ontology. The advantage of this approach, compared to NLP-based ontology learning approaches, is relative independence on linguistic resources. Experimental evaluation of the sibling mining algorithm was made against a gold-standard data set, and several variations of input setting were systematically tested.

OM-2006 contributed its best paper, namely, “Exploring the Semantic Web as Background Knowledge for Ontology Matching” by Sabou, d'Aquin, and Motta. This work proposes to enhance ontology matching by combining appropriate background knowledge from multiple automatically identified online ontologies. Specifically, two strategies have been devised and implemented: (i) mappings within one ontology, that is when a mapping between the candidate concepts (of the ontologies to be matched) can be discovered in a single external ontology, and (ii) cross-ontology mapping discovery, that is when a mapping is derived from two or more (external) ontologies. The approach has been evaluated on real-life data from two large agricultural thesauri, such as AGROVOC and NALT.

The selection from the workshops held in conjunction with the ER 2006 conference resulted in two extended papers being accepted for JoDS: "Time-Aggregated Graphs for Modeling Spatio-Temporal Networks" by George and Shekhar, and "An Architecture for Emergent Semantics" by Herschel, Heese, Bleiholder and Czekay.

The first paper proposes a novel method for storing and efficiently querying spatio-temporal data. A detailed logical and physical structure is presented along with an analytical analysis of the storage requirements of the proposed model compared to other approaches. Following this a new algorithm is presented for finding the shortest path using the proposed data model. The algorithm is then proven correct and its worst-case runtime is formally analyzed.

The second paper introduces a universal architecture for emergent semantics using a central repository within a multi-user environment, based on linguistic theories. Based on this architecture, an implementation of an information retrieval system supporting term queries on standard information retrieval corpora is shown, which incorporates feedback on the retrieval results directly into the actual document representations, thus improving future retrievals.

SWESE 2006 Co-chair

Jeff Z. Pan	University of Aberdeen, UK

InterDB 2006 Co-chair

Philippe Thiran	Namur University, Belgium

ORM 2006 Co-chair

Terry Halpin	Neumont University, Salt Lake City, Utah, USA

EKAW 2006 Co-chairs

Steffen Staab	University of Koblenz-Landau, Germany
Vojtech Svatek	University of Economics, Prague, Czech Republic

OM-2006 Co-chair

Pavel Shvaiko	TasLab, Informatica Trentina, Italy[*]

ER Workshops 2006 Co-chair

John Roddick	Flinders University, Australia

Reviewers

We would like to express our gratitude to the following colleagues who helped in the review process by contributing detailed reviews of the submitted papers and provided invaluable feedback to the authors and editors:

Colin Atkinson	University of Mannheim, Germany
Nathalie Aussenac-Gilles	IRIT- CNRS Toulouse, France
Ken Baclawski	Northeastern University, USA
Philipp Cimiano	University of Karlsruhe, Germany
Philippe Cudre-Mauroux	EPFL, Switzerland
Denise de Vries	Flinders University, Australia
Ken Evans	ORM Foundation, UK
Dragan Gasevic	Simon Fraser University Surrey, Canada
Marc-Philippe Huget	University of Savoy, France
Michael N. Huhns	University of South Carolina, USA
Zakaria Maamar	Zayed University, UAE
David Martin	SRI International, USA
Sjir Nijssen	PNA Group, The Netherlands
Natasha Noy	Stanford University, USA
Christine Parent	University of Lausanne, Switzerland
Adrian Paschke	Technical University Dresden, Germany
Allison L. Powell	Corporation for National Research Initiative, USA
Erik Proper	Radboud University and Cap Gemini, The Netherlands
Gerald Reif	University of Zurich, Switzerland
Simone Santini	Universidad Autonoma de Madrid, Spain
Pavel Smrz	Technical University of Brno, Czech Republic
Heiner Stuckenschmidt	University of Mannheim, Germany
Michael Twidale	University of Illinois at Champaign, USA

[*] The work on this JoDS issue was performed in part while at the University of Trento, Italy.

Chad A. Williams	University of Illinois at Chicago, USA
Huimin Zhao	University of Wisconsin-Milwaukee, USA

JoDS Editorial Board

Co-editors-in-Chief

Lois Delcambre	Portland State University, USA
Stefano Spaccapietra	EPFL, Switzerland

Members of the Board

Carlo Batini	Università di Milano Bicocca, Italy
Alex Borgida	Rutgers University, USA
Shawn Bowers	University of California Davis, USA
Tiziana Catarci	Università di Roma La Sapienza, Italy
David W. Embley	Brigham Young University, USA
Jerome Euzenat	INRIA Alpes, France
Dieter Fensel	University of Innsbruck, Austria
Nicola Guarino	National Research Council, Italy
Jean-Luc Hainaut	FUNDP Namur, Belgium
Ian Horrocks	University of Manchester, UK
Arantza Illarramendi	Universidad del País Vasco, Spain
Larry Kerschberg	George Mason University, USA
Michael Kifer	State University of New York at Stony Brook, USA
Tok Wang Ling	National University of Singapore, Singapore
Shamkant B. Navathe	Georgia Institute of Technology, USA
Antoni Olivé	Universitat Politècnica de Catalunya, Spain
José Palazzo M. de Oliveira	Universidade Federal do Rio Grande do Sul, Brazil
Christine Parent	Université de Lausanne, Switzerland
Klaus-Dieter Schewe	Massey University, New Zealand
Heiner Stuckenschmidt	University of Manheim, Germany
Katsumi Tanaka	University of Kyoto, Japan
Yair Wand	University of British Columbia, Canada
Eric Yu	University of Toronto, Canada
Esteban Zimányi	Université Libre de Bruxelles (ULB), Belgium

Table of Contents

ER 2006 Workshops

Experiences in the Design of Semantic Services Using Web Engineering Methods and Tools

Marco Brambilla[1], Stefano Ceri[1], Irene Celino[2],
Dario Cerizza[2], Emanuele Della Valle[2], Federico M. Facca[1],
Andrea Turati[2], and Christina Tziviskou[1]

[1] Dipartimento di Elettronica e Informazione, Politecnico di Milano
P.za Leonardo da Vinci 32, I-20133 Milano, Italy
{mbrambil,ceri,facca,tziviskou}@elet.polimi.it
[2] CEFRIEL
Via Fucini 2, I-20133 Milano, Italy
{celino,cerizza,dellavalle,turati}@cefriel.it

Abstract. Although Semantic Web Services are expected to produce a revolution in the development of Web-based systems, very few concrete design experiences are available; only recently, Software Engineering methods and tools have started to embrace the deployment of Semantic Web applications. In this paper, we show how classical Software Engineering methods (i.e., formal business process development, computer-aided and component-based software design, and automatic code generation) combine with semantic methods and tools (i.e., ontology engineering, semantic service annotation and discovery) to forge a new approach to software development for the Semantic Web. In particular, we present our experience in the participation to the Semantic Web Service (SWS) challenge 2006, where the proposed approach achieved very good results in solving the proposed problems.

1 Introduction

Knowledge technologies have been admired in the past for their ability to address challenging problems, but only in a small scale. They have not been equally convincing in attacking large-scale problems, in which one has to consider complex requirements and to turn them into effective solutions.

The Semantic Web promotes the vision of an extended Web of machine-understandable information and automated services that allows knowledge technologies to reach Web-scale. The explicit representation of the semantics of the data and of the services will enable a new Web that provides a qualitatively new level of service. Automated services will improve in their capacity to assist humans in achieving their goals by “understanding” more of the content on the Web, and thus providing accurate filtering, categorization, and searches of information sources. Recent efforts around UDDI, WSDL, and SOAP are concentrating on making the Web more service-centric, allowing for on-the-fly software composition through the use of loosely coupled, reusable software components.

© Springer-Verlag Berlin Heidelberg 2008

However, more work needs to be done before the Web Service infrastructure can support the Semantic Web vision. Semantic Web Services (SWS) address the automation of *discovery* of services of interest; *mediation of data* exchanged between the different services; and *mediation of processes* performing service-enabled tasks.

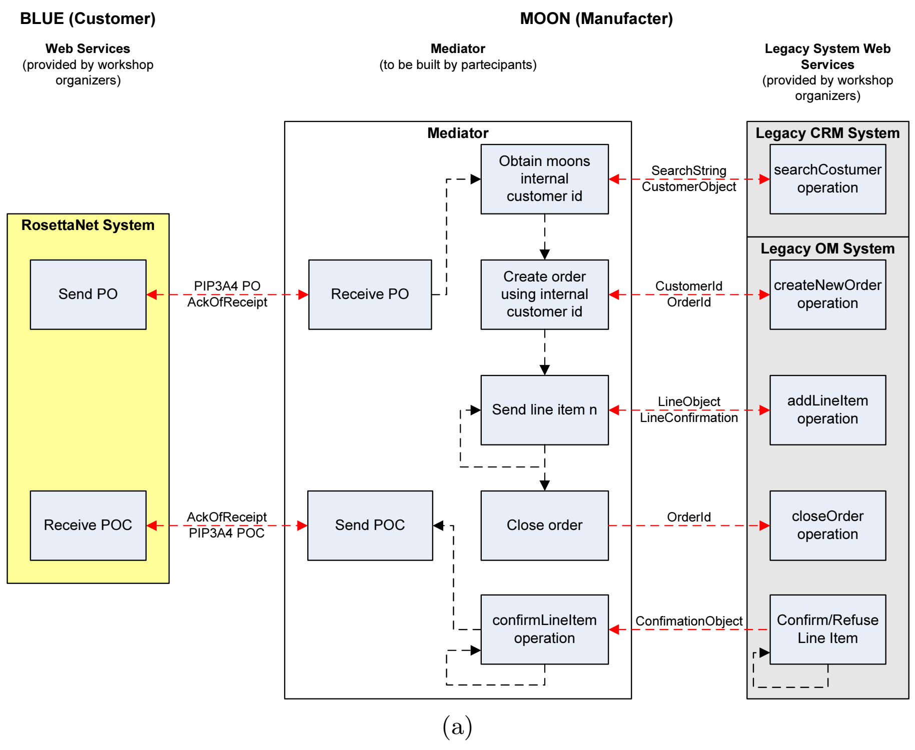
The emerging field of Semantic Web Services provides paradigms based on program annotation and self-descriptive implementation, to build cross-enterprise applications which favour flexibility, automatic resource discovery, and dynamic evolution. However, the development of applications based on Semantic Web Services is currently lacking a set of high level Software Engineering abstractions and tools that may push the spreading of such technology. Moreover, as Semantic Web and Semantic Web Services enter the real world, knowledge engineers discover the need of finding ways for structuring their work, thus steadily embracing Software Engineering techniques and methodologies. We believe this trend is fundamental to face the growing need of automated interactions between business partners. Indeed, this requires knowledge management and reasoning technologies to be complemented with large-scale project management approaches.

In this paper we report one of these cases in which two groups, one from the Web Engineering community and one from the Semantic Web community, got engaged together in building a structured solution to the design and implementation of Semantic Web applications. This experience started with the joint participation to the SWS challenge 2006 [1], that was scheduled as a 3-phase event throughout the year. In this paper we report about our work that addressed the first three phases of the SWS challenge 2006. Our efforts ended up in proposing the most complete solution for the second phase of the challenge[1] and one of the best solutions for the third phase[2]. At the time of the writing, the organization of the forth phase is ongoing..

The SWS challenge aims at employing semantics-based technologies on a set of problems. The goal of the challenge is to develop a common understanding of various technologies intended to facilitate the automation of mediation, choreography and discovery for Web Services using semantic annotations. Participants are not measured on computational speed, but rather on the way they operationally explore the semantic technology space. To fulfill the objectives of the challenge, the organizers set up two scenarios covering mediation (both for data and processes), and both static and dynamic discovery.

– In the *Mediation scenario*, a legacy system (Moon) is made compliant with the Blue system that uses the RosettaNet Purchase Order (PO) [2] (a complex XML specification as well as a simple protocol for exchanging messages about its processing); the scenario covers data and process mediation issues. While the Moon system requires to receive a set of messages that involves a complex interaction, the Blue system adopts the RosettaNet messaging protocol that relies on a simpler interaction. Furthermore the messaging format of the two

[1] Budva Evaluation: `http://sws-challenge.org/wiki/index.php/Workshop_Budva`
[2] Athens Evaluation: `http://sws-challenge.org/wiki/index.php/Workshop_Athens`



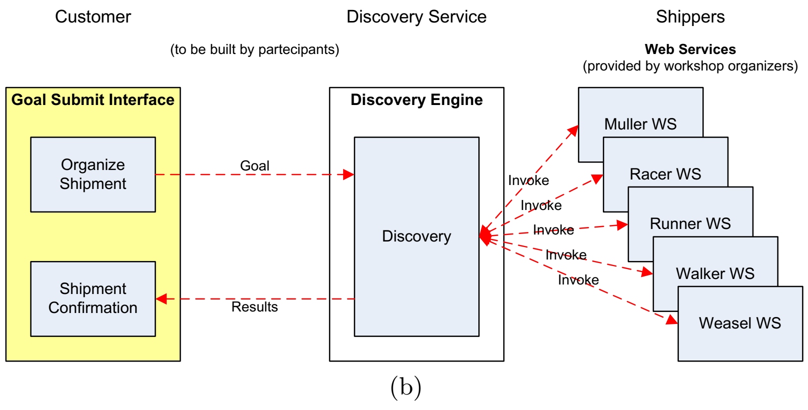


Fig. 1. SWS challenge scenarios - Mediation scenario (a) and Discovery scenario overview (b)

systems is different. Figure 1a reports the scenario overview proposed by the organizers.

– The *Discovery scenario* requires to automatically provide the best shipping option among different shipment services for the purchased goods (see Figure 1b). The set of shipment Web Services offers different destinations, delivery

Table 1. SWS Challenge Levels

Level	Description
0	Mediation Scenario (static).
1	Mediation Scenario (adopting to changes in systems).
1a	Data Mediation.
1b	Process Mediation.
2	Simple Discovery (single invocation based on service description).
2a	Shipment coverage (countries, cities semantics).
2b	Shipment price and weight calculations (arithmetic).
2c	Shipment constraints on pick-up time and delivery (temporal semantics).
2d	Shipment unit conversion (semantics of measures).
3	Composite Discovery (multiple invocations required for complete discovery).
3a	Discovery 2 including request for quote and a order operation.
3b	Discovery 3a including a request for multiple packages that has to be split.
3c	Discovery 3b including a dynamic currency conversion.

From: http://sws-challenge.org/wiki/index.php/SWS_Challenge_Levels

times, prices, and pick-up policies. Hence according to the shipment details, not all the services are suitable. In order to test the solutions, the organizers provided a set of heterogeneous shipment goals expressed in natural language. An example of such goal is: send one package (length 10, width 2, height 3 inches; weight 20 lbs) from California to Bristol (UK) for less than 120$.

The challenge requires participant to present their solutions, to freeze them, and to cope with a set of previously unknown changes to proof the flexibility of the initial solutions. The evaluation of the solutions is based on how much the participants have to alter their systems to respond to the released changes. Table 1 reports the levels used for the evaluation.

The problem is clearly neither a pure Semantic Web one, nor a pure Software Engineering one. Semantics is clearly needed to address in a flexible way the Discovery scenario, but Software Engineering tools and methods are the right ones to address in a flexible way the Mediation scenario.

For this reason (see Section 3) we adopt an original mix of Semantic Web and Software Engineering techniques: WSMO [3] as Semantic Web Service approach, Glue [4] as Semantic Web Service discovery engine, WebML [5] as Web engineering model for designing and developing semantically rich Web applications implementing Service Oriented Architecture, and WebRatio [6] as WebML CASE tool[3].

The combined approach introduces Semantic Web technologies in the development process of Web applications (see Section 2) and employs tools (i.e., WebRatio) developed for Web Engineering using Semantic Web languages and tools. As a result, it offers an application that combines Web Engineering techniques and Semantic Web tools and languages (see Section 4). Our experience introduces a significant contribution in the application of Software Engineering techniques to Semantic Web application design. Section 5 reports about the cross-fertilization between the two fields, resulting from intensive interaction

[3] Online demos and further material is available at: http://www.webml.org/sws-challenge.html

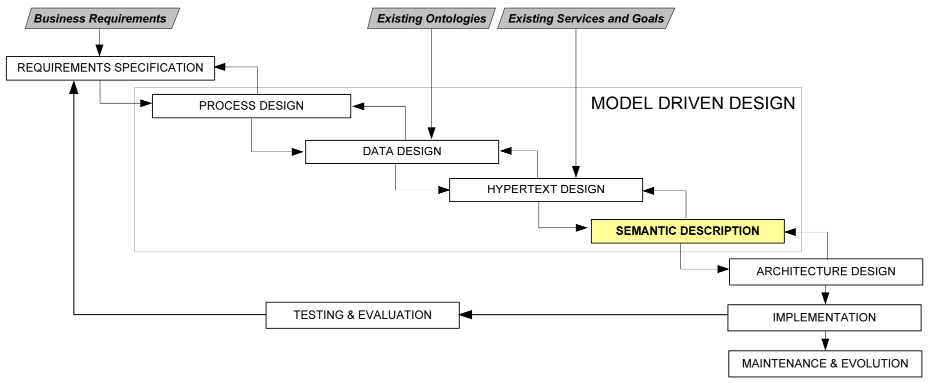


Fig. 2. Phases in the development process of data- and process-intensive Web applications

between two separate research groups. In Section 6 we review related work and solutions of other participants to the challenge. Finally, in Section 7, we summarize our experience and provide some forecasts on the design and development of model-driven Semantic Web Service application.

2 Design Methodology

The steps we adopted to solve the problems introduced by the two scenarios are shown in Figure 2. In line with the classic Boehm's Spiral model and with modern methods for Web and Software Engineering, the development phases must be applied in an iterative and incremental manner, in which the various phases are repeated and refined until results meet the business requirements.

– *Requirements specification* is focused at the collection and formalization of the essential information about the application domain and expected functions.
– *Process design* focuses on the high-level schematization of the (possibly distributed) processes underlying the application.
– *Data design* is the phase in which the data expert organizes the main information objects identified during requirements specification into a comprehensive and coherent domain model, that may comprise the importing of existing ontologies and the definition of new ones.
– *Hypertext design* is the activity that transforms the functional requirements identified during requirements specification into high level models, that include primitives for specifying user-browsable Web sites and machine-readable Web Services. These models let the architect specify how content elements are published within pages, how services provide information to requesters, and how hypertext elements are connected by links to form a navigable structure.

The core phase of the method is *Semantic description* of the application, a new design phase which is required to provide WSMO compatibility; it consists in a set of tasks, partially automated, that aim at providing a set of semantic specifications of the application to be implemented.

A detailed description of the different phases of Figure 2 is outside the scope of this paper. A comprehensive description of the non semantic ones can be found in [5], while details for the steps involving semantic descriptions can be found in [7].

3 Background Technologies

In the following we provide the required background on the languages and tools used for the challenge.

3.1 WSMO, WSML and WSMX

The WSMO initiative [3,8] aims at providing a comprehensive framework for handling Semantic Web Services which includes the WSMO conceptual model, the WSML language [9] and the WSMX execution environment [10].

The Web Service Modeling Ontology (WSMO) is an ontology for describing various aspects related to Semantic Web Services. WSMO defines four modeling elements (ontologies, Web Services, goals and mediators) to describe several aspects of Semantic Web Services, based on the conceptual grounding of the Web Service Modeling Framework (WSMF) [11].

Ontologies provide the formal semantics to the information used by all other components. They serve in defining the formal semantics of the information, and in linking machine and human terminologies.

Web Services represent the functional and behavioral aspects, which must be semantically described in order to allow semi-automated use. Each Web Service represents an atomic piece of functionality that can be reused to build more complex ones. Web Services are described in WSMO from three different points of view: *non-functional properties*, *capabilities* (describing functionalities), and *interfaces* (describing the behavior). A Web Service can have multiple interfaces, but it has one and only one capability.

Goals specify objectives that a client might have when consulting a Web Service. In WSMO [8], a goal is characterized in a dual way with respect to Web Services: goal's descriptions include the *requested capability* and the *requested interface.*

Finally, *mediators* provide interoperability facilities among the other elements. They aim at overcoming structural or semantic mismatches that appear between the different components that build up a WSMO description. For instance, a ggMediator acts as a mediator between two goals, a wgMediator mediates between a Web Service and a goal, and a wwMediator mediates between two Web Services with mismatching interfaces.

Web Service Modeling Language (WSML) [12] offers a set of language variants for describing WSMO elements that enable modelers to balance between expressiveness and tractability according to different knowledge representation paradigms. The most basic, and least expressive, variant is WSML-Core. WSML Core is separately extended in two different directions by the variants WSML-DL and WSML-Flight, respectively. WSML-Flight is based on a logic programming variant of F-Logic [13]. Web Service Execution Environment (WSMX) is a framework for the automation of discovery, selection, mediation, and invocation of Semantic Web Services. WSMX is based on WSMO and, at the same time, it is a reference implementation of it.

3.2 Glue, a WSMO-Compliant Discovery Engine

Glue [4,14] is a WSMO-compliant discovery engine that provides the basis for introducing discovery in a variety of applications; it provides efficient pre-filtering of relevant services and accurate discovery of services that fulfill a given requester goal.

In conceiving Glue, the model for WSMO Web Service discovery was refined explicating the central role of mediation:

– by making the notion of class of goals and class of Web Service descriptions explicit;
– by making wgMediators the conceptual element responsible for evaluating the matching; and
– by redefining the discovery mechanism as a composite procedure where the discovery of the appropriate mediators and the discovery of the appropriate services are combined.

Moreover, in designing Glue the authors refined the WSMX discovery engine architecture according to their refined WSMO discovery conceptual model, both in terms of components and execution semantics. In particular, the execution semantics of Glue implements a composite discovery procedure that:

1. given a class of goals and an XML message containing the information describing a specific goal, constructs an instance goal;
2. looks up the wgMediators that has the class of the goal as target;
3. filters the Web Services limiting the scope to those that are sources of the identified wgMediators; and
4. evaluates the rules in the wgMediator returning only those Web Services that semantically match the goal.

Glue implementation uses internally F-logic (semantically close to WSML Flight) and it is built around an open source F-logic inference engine called Flora-2 that runs over XSB, an open source implementation of tabled-prolog and deductive database system. Web Services and goal descriptions are represented in F-logic, like ontologies.

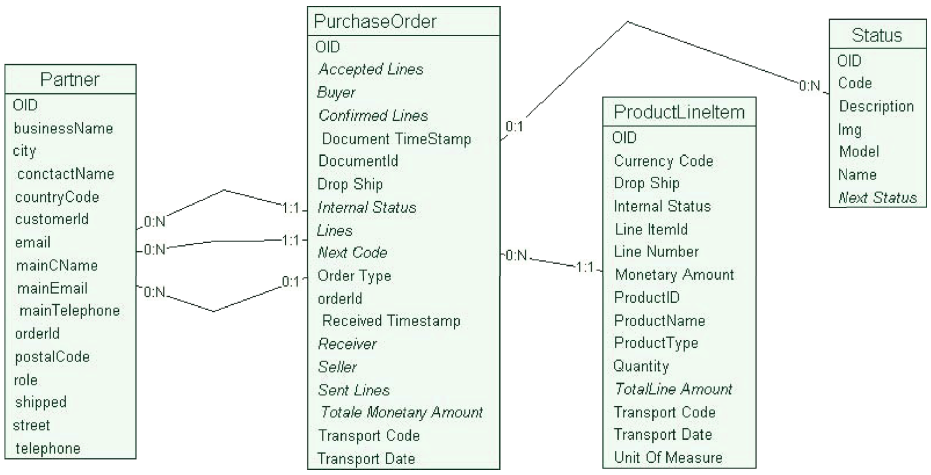


Fig. 3. The E-R diagram for the data model used of the initial Mediator

3.3 WebML and WebRatio

WebML [5] is a high-level notation for data- and process- centric Web applications. It allows specifying the conceptual modeling of Web applications built on top of a data schema used to describe the application data, and composed of one or more hypertexts used to publish the underlying data.

The WebML data model is the standard Entity-Relationship (E-R) model extended with an Object Query Language [15]. Figure 3, described in details in Section 4.2, is an E-R representation of the data needed for the Challenge scenario. The expressive power of the WebML E-R model can be compared to the WSML-Flight language. Table 2 shows the comparison of the expressive power of WebML in respect to OWL-DL and WSML-Flight extending the comparison presented in [7]. As WSML-Flight, the WebML data model comprises a rule language called WebML-OQL, which allows for calculating derived information, but not for defining constraints. WebML does not support the concepts Thing and Nothing (as WSML-Flight does), and it cannot deal with equality related constructors (i.e., EquivalentClasses, SameIndividual, DisjointClasses, DifferentIndividuals, and Fuctional/InverseFuctional properties); these constructors are instead supported by WSML-Flight. Regardless, the similarity between WebML extended E-R and WSML-Flight is sufficient to enable a partial extraction of WSMO descriptions for E-R schemes.

Upon the same data model, it is possible to define different hypertexts (e.g., for different types of users or for different publishing devices), called *site views*. A site view is a graph of *pages*, allowing users from the corresponding group to perform their specific activities. Pages consist of connected *units*, representing at a conceptual level atomic pieces of homogeneous information to be published: the content that a unit displays is extracted from an entity, and selected by means of a *selector*, testing complex logical conditions over the unit's entity. Units within a Web site are often related to each other through *links* carrying data from a unit to another, to allow the computation of the hypertext. WebML allows specifying also update *operations* on the underlying data (e.g., the creation, modification and deletion of instances of an entity, or the creation and deletion of instances of a relationship) or operations performing other actions (e.g. sending an e-mail). In [16] the language has been extended with operations supporting process specifications.

To describe Web Services interactions, WebML includes some Web Service primitives [17]. Web Services operation symbols correspond to the WSDL classes of Web Service operations: `Request-response`, `Response`, `Solicit`, and `One-way` units can be used in WebML for describing Web Service interactions.

Table 2. Expressive power comparison between ontology languages

OWL Abstract Syntax	DL syntax	WSML-F	WebML E-R and OQL
Axioms			
Class(A partial $C_1 \dots C_n$)	$A \subseteq C_i$	+	+ ($C_i \neq \bot$; $A \neq \bot$)
Class(A complete $C_1 \dots C_n$)	$A \equiv C_1 \cap \dots \cap C_n$	+	+ ($A, C_i \neq \bot$; T)
EnumeratedClass($A o_1 \dots o_n$)	$A \equiv \{o_1, \dots, o_n\}$	−	−
SubClassOf($C_1 C_2$)	$C_1 \subseteq C_2$	+	+
EquivalentClasses($C_1 \dots C_n$)	$C_1 \equiv \dots \equiv C_n$	+	−
DisjointClasses($C_1 \dots C_n$)	$C_i \cap C_j \subseteq \bot$	+	−
ObjectProperty(R super(R_1) … super(R_n)	$R \subseteq R_1$	+	+
domain(C_1) … domain(C_n)	$T \subseteq \forall R^-.C_i$	+	+ ($C_i \neq \bot$)
range(C_1) … range(C_n)	$T \subseteq \forall R^{\cdot} C_i$	+	+ ($C_i \neq \bot$)
[inverseOf(R_0)]	$R \equiv R_0^-$	+	+
[Symmetric]	$R \equiv R^-$	+	∼ (R and R^- with same name)
[Functional]	$T \subseteq \leq 1R$	+	−
[InverseFunctional]	$T \subseteq \leq 1R^-$	+	−
[Transitive])	Trans(R)	+	−
SubPropertyOf($R_1 \dots R_2$)	$R_1 \subseteq R_2$	+	+ (domain(R_i), range(R_i)$\neq T$)
EquivalentProperty($R_1 \dots R_n$)	$R_1 \equiv \dots \equiv R_n$	+	+ (domain(R_i), range(R_i)$\neq T$)
Individual(o type(C_1) … type(C_n)	$o \in C_i$	+	+
value($R_1 o_1$) … value($R_n o_n$))	$< o, o_i > \in R_i$	+	+
SameIndividual($o_1 \dots o_n$)	$o_1 \equiv \dots \equiv o_n$	−	−
DifferentIndividuals($o_1 \dots o_n$)	$o_i \neq o_j, i \neq j$	−*	−*
Descriptions (C)			
A (URI Reference)	A	+	+
owl:Thing	T	−	−
owl:Nothing	$\bot$	−	−
intersectionOf($C_1 \dots C_n$)	$C_1 \cap \dots \cap C_n$	+	rhs**
unionOf($C_1 \dots C_n$)	$C_1 \cup \dots \cup C_n$	lhs***	rhs**
complementOf(C_0)	$\neg C_0$	−	−
oneOf($o_1 \dots o_n$)	$\{o_1 \dots o_n\}$	lhs***	−
restriction(R someValuesFrom(C))	$\exists R, D$	lhs***	rhs**
restriction(R allValuesFrom(C))	$\forall R, D$	rhs**	rhs**
restriction(R value(o))	$\exists R, o$	+	rhs**
restriction(R minCardinality(1))	$\geq 1R$	−	rhs**
restriction(R minCardinality(n)) ($n > 1$)	$\geq nR$	−	rhs**
restriction(R maxCardinality(n))	$\leq nR$	rhs**	rhs**

+ The language fully support the related concept.
− The language does not support the related concept.
* Notice that already all individuals in an WSML Flight knowledge base and WebML repository are different; therefore, an explicit assertion would be superfluous.
** May only be used in partial class definitions and on the right-hand side (as the second argument) of SubClassOf.
*** May only be used on the left-hand side (as the first argument) of SubClassOf.

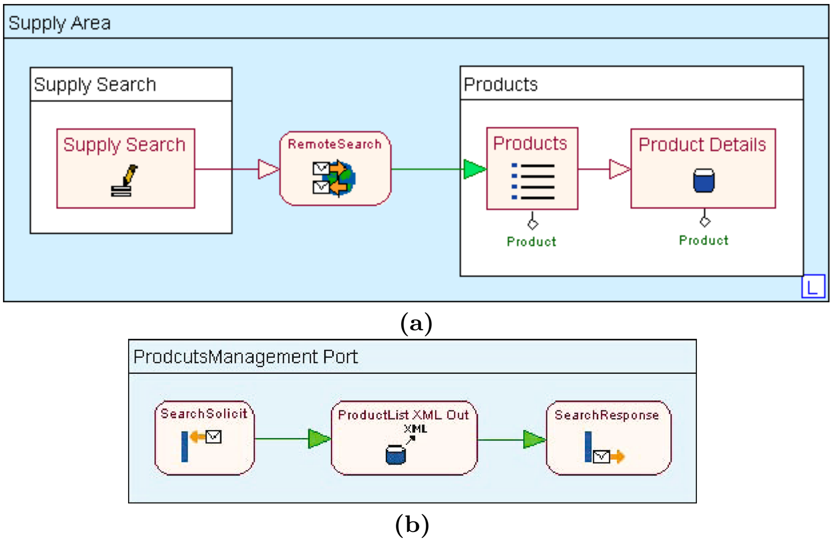


Fig. 4. Example of WebML hypertext model with invocation of remote service

`Request-response` and `Response` operations are triggered when the user navigates one of their input links; from the context transferred by these links, a message is composed, and then sent to a remote service as a request. In the case of a synchronous request-response, the user waits until the response message is received, then continues navigation as indicated by the operation's output link. Otherwise, navigation resumes immediately after the message is sent. `Solicit` and `One-way` are instead triggered by the reception of a message. Indeed, these units represent the publishing of a Web Service, which is exposed and can be invoked by third party applications. In the case of `One-way`, the WebML specification may dictate the way in which the response is built and sent to the invoker. Moreover, Web Services publishing units cannot have output links leading to pages, because there is no user interaction involved in the response to the caller. Another operation typically involved in Web Service interactions is the `Adapter` unit, which is able to apply any kind of XSLT transformation to a XML document. This unit is often used in conjunction with the `XML-In` unit or the `XML-Out` unit: the first is used to import canonic XML data (formatted according a particular XSD) into the database, the latter to extract database instances and convert them to the a canonic XML format.

Figure 4 shows an hypertext example that includes the model of a Web Service call and of the called service. *Supply Area* of Figure 4a is an area of a Web site for supply management. The employee can browse the *SupplySearch* page, in which the *SearchProducts* entry unit permits the input of search criteria. navigating the link outgoing the entry unit, a request message is composed and sent to the *RemoteSearch* operation of a Web Service. The user then waits for the response message, containing a list of products satisfying the search criteria. From these options, a set of instances of *Product* are created, and displayed to the user by means of the *Products* index unit in the *Products* page; the user may continue browsing, e.g., by choosing one of the displayed products and looking at its details. Figure 4b represents the model of the *RemoteSearch* service invoked by the previously described hypertext. The interaction starts with the *SearchSolicit* unit, which denotes the reception of the message. Upon the arrival of the message, an XML-out operation extracts from the local data source the list of desired products and formats the resulting XML document. The *SearchResponse* unit produces the response message for the invoking Web Service.

The WebML language is *extensible*, allowing for the definition of customized operations and units. It has been implemented in the CASE tool WebRatio [6], a development environment for the visual specification of Web applications and the automatic generation of code for the J2EE and Microsoft .NET platforms. The design environment is equipped with a code generator that deploys the specified application and Web Services in the J2EE platform, by automatically generating all the necessary pieces of code, including data extraction queries, Web Service calls, data mapping logics, page templates, and WSDL service descriptors.

4 Our Solution to the SWS Challenge

In this section we describe the general approach to the SWS challenge scenarios, and how we faced the three phases of the challenge. In the first place we provide an overview of the initial solution. Then we give some insight about how we coped with the changing requirements and about the effort required to adapt our solution to the subsequent new scenario specifications.

4.1 Integration of Web Engineering and Semantic Web Tools

Figure 5 summarizes our approach to the problem of designing Semantic Web Service-based applications. The blue blocks highlight the basic steps that have been derived from the methodology described in Section 2 for designing Web applications. The various steps produce some artifacts (BPMN models, WebML skeletons, data models, hypertext models), possibly enriched by imported ontological descriptions (on top of Figure 5). These "conventional" software engineering artifacts are exploited for deriving the set of WSMO specifications (at the bottom of Figure 5):

– the description of the mediator can be extracted from the hypertext describing the mediator;
– the Web Services capability description is derived from the hypertext model;
– the choreography information is derived from the Business Process (BP) model;
– and the user goals are derived from both the BP model and the hypertext model.

This approach seamlessly fits into traditional Software Engineering methods and techniques based on system modeling (i.e. Model Driven Design and Model

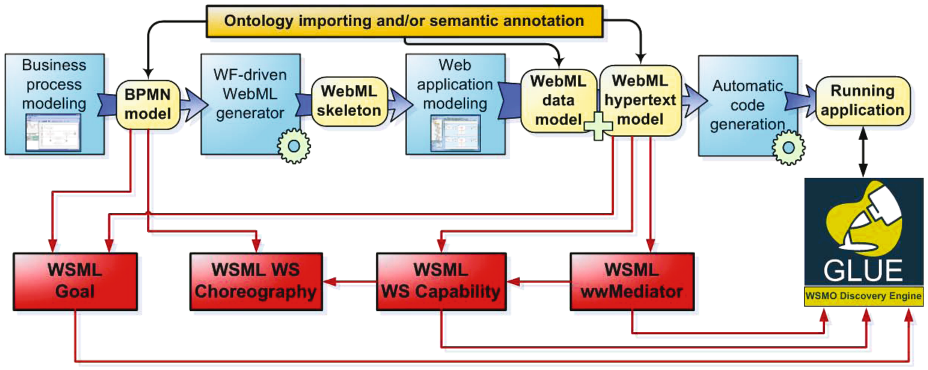


Fig. 5. Overall design methodology for Semantic Web Service-based applications

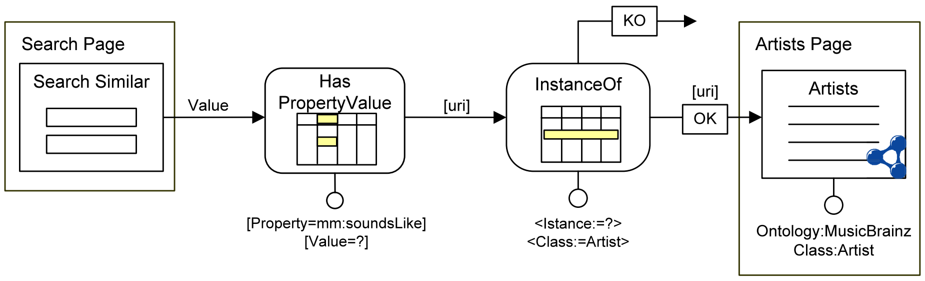


Fig. 6. Example of WebML model exploiting the Semantic Web units

Driven Architectures); therefore, existing CASE tool for Model Driven Design (MDD) can be easily extended for supporting the process. Details on the approach and relative examples can be found in [7].

In order to support the query over ontologies, new WebML units have been devised [18]. In particular we extended the WebML basic primitives provided by the hypertext model (e.g., Index and Data units) to support ontological data sources (e.g., RDF/OWL ontologies) and we provided a new set of primitives specifically designed to exploit ontologies characteristics and reasoning over them. These new units are aggregated primitives that, depending on the type of parameters, execute differently. The units (`SubClassOf`, `InstanceOf`, `HasProperty`, `HasPropertyValue`, `PropertyValue`, `SubPropertyOf`) aim at providing explicit support to advanced ontological queries. They allow to extract classes, instances, properties, values; to check existence of specific concepts; and to verify whether a relationship holds between two objects.

Figure 6 depicts a fragment a WebML application that allows to retrieve artists or albums whose names sound in a similar way to the name specified by the user. The ontology adopted in the example is the MusicBrainz ontology [19].

The value submitted in the form is passed to the `HasPropertyValue` unit that extracts a set of URIs of instances (albums or artists) that have value as value of the `mm:soundsLike` property. The set of URIs is then passed to the `InstanceOf` unit that checks if they are instances of the class Artist. In this case, the URIs are passed over through the `OK` link to an `Index` unit showing list of Artists, otherwise the URIs are passed on the `KO` link to publish a list of Albums (not shown in the figure).

In general, each WebML semantic unit can automatically extract a RDF description of its contents. The designer has to specify how he wants to use the RDF fragments; for instance, it is possible to aggregate the fragments of all the units in the page and publish the aggregate at the bottom of the page, as a global semantic annotation of the page itself; another option could be to maintain them separated and publish the RDF annotation for each unit in the page. For instance, annotations can be generated as RDF expressions [20].

Besides the units for ontological data query, we introduce also three new units working at a upper level: the `Set Composition` operation unit is able to perform classic set operations (i.e., union, intersection, difference) over two input sets of URIs, considering the hierarchy of the URIs involved; the `Import Ontological Source` unit adds a remote or local data source that must be consistent with ontological model of the web application (it's validated against it before being added to the ontology); and finally, the `Describe` unit returns the RDF description of an URI, thus enabling data exporting and semantic annotation of pages.

4.2 Initial Modeling

Mediation scenario. The modeling of the mediator started from the design of the data model. The RosettaNet message was analyzed and a corresponding WebML E-R diagram was obtained from it. We identified four main entities: PurchaseOrder, Partner, Status, and ProductLineItem as shown in Figure 3.

As showed by relationships in Figure 3, each PurchaseOrder instance has: one of more ProductLineItem instances, three Partner instances representing respectively the Buyer, the Seller and the Receiver. The entity Status tracks the status of each PurchaseOrder.

Once the WebML data model was completed, we started modeling the Web Service providing the mediation feature. An high level Business Process Modeling Notation (BPMN) model is created representing the mediator (see Figure 7), which formalizes the orchestration of the Moon Web Services, and defines states pertaining to the mediation process according to the scenario specification. Then, the BPMN model is used to automatically generate a WebML skeleton that is manually refined to complete the design of the mediator. The final model for the Blue to Moon mediator is reported in Figure 8a. Each row of the model depicted in the Figure corresponds to a specific step that the mediator must perform. Each of these steps comprises a set of specific operations on the received messages or on the local data. We exemplify in details the first two steps of the mediator, namely (i) the reception of the RosettaNet message and its forwarding to the legacy sys-

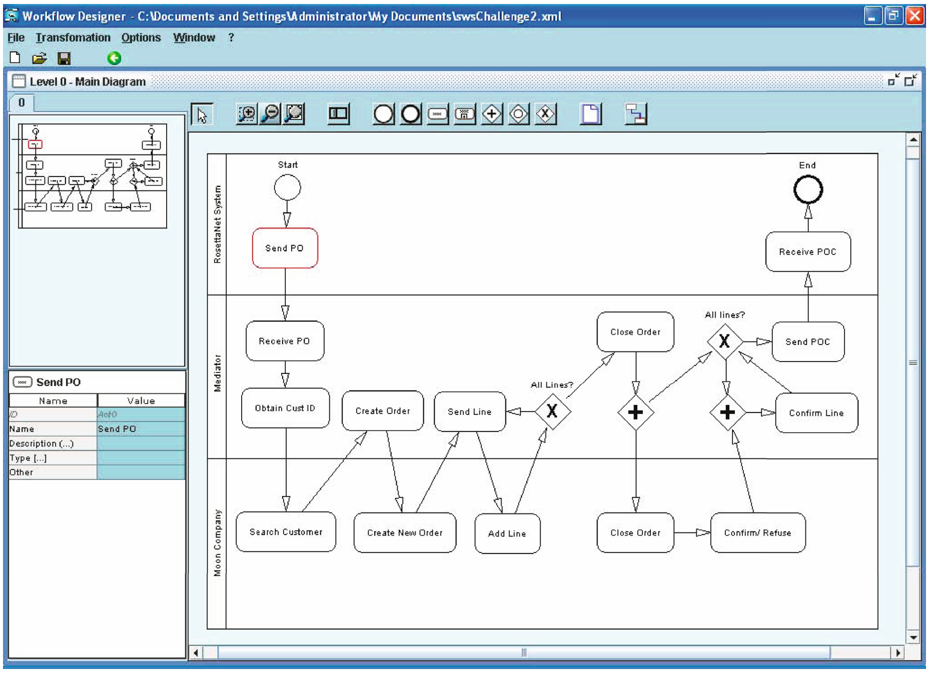


Fig. 7. The BPMN model of the Mediator from Blue-to-Moon

tem; and (ii) the selection of the Buyer Partner. First, we modeled the operation receiving the RosettaNet message and forwarding the order to the legacy system:

1. As soon as the order is received (`Pip3A4PurchaseOrderRequest` Solicit Unit), the Pip3APurchaseOrder is converted (`Lifting` Adapter Unit) and stored in the database (`StorePip3A4PurchaseOrder` Unit), the status of the current Pip3APurchaseOrder is set to "To Be Processed" (`SetProcessStatus` Connect Unit, that creates new relationship instances between objects) and the Acknowledge message is returned to the service invoker (`SendReceipt-Acknowledgement` Response Unit).
2. Next, the Buyer Partner is selected (`SelectBuyer` Selector Unit, that retrieves data instances according to a specified selector condition) and a message to query the Customer Relationship Management service (CRM) is created (`Lowering` Adapter Unit) and sent to the legacy system (`ObtainMoonCustomerID` Request-Response Unit). Once a reply has been received, the CustomerId is extracted from the reply message (`Lifting` Adapter Unit) and stored in the data model (`StoreCustomerID` Modify Unit). The status of the order is set to "CustomerId received" (`SetProcessStatus` Connect Unit).

Analogous operations are performed for the remaining steps (lines 3 to 5 in Figure 8a).

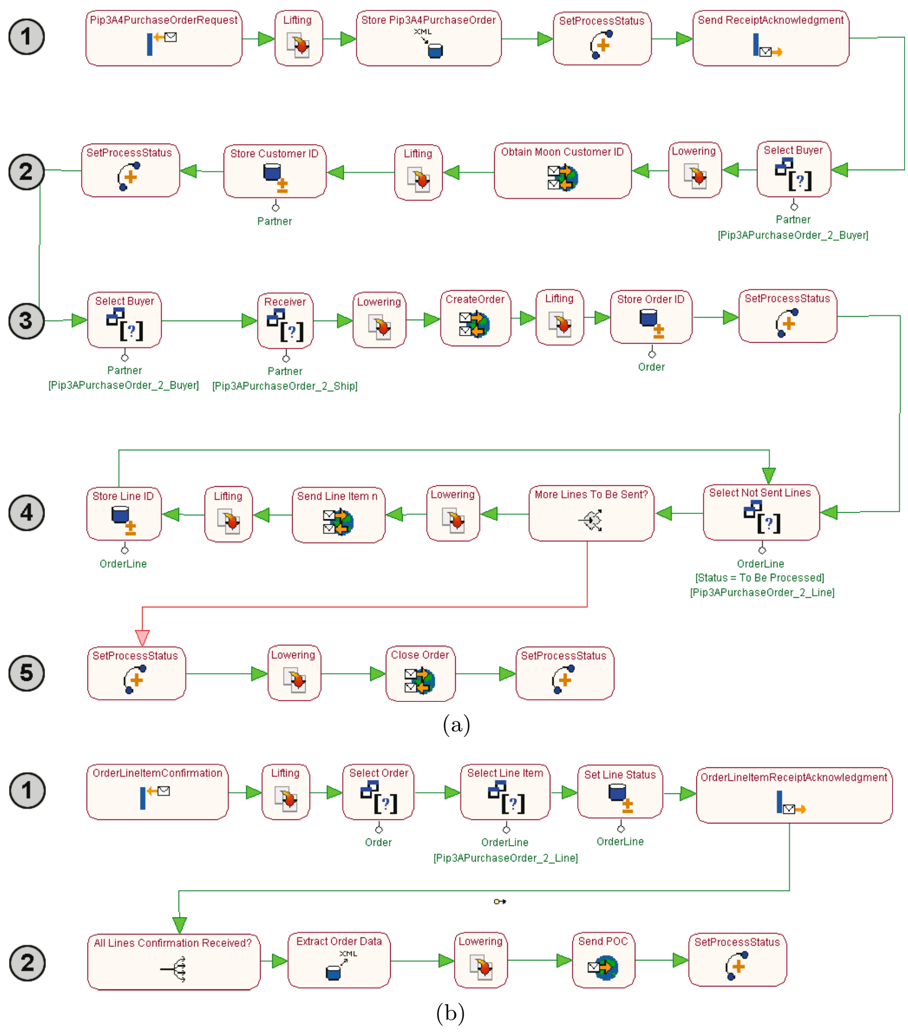


Fig. 8. The complete WebML model of the Mediator: (a) shows the Blue-to-Moon mediator, while (b) shows the Moon-to-Blue mediator

Figure 8b shows the corresponding process required at the legacy system for receiving order lines:

1. For each line confirmation (`OrderLineItemConfirmation` Solicit Unit), the status is extracted (`Lifting` Adapter Unit), the relative order and line stored in mediator database are selected (`SelectOrder` and `SelectLineItem` Selector Units), and the status of the stored line is modified according to the received confirmation (`SetLineStatus` Modify Unit). Eventually the Acknowledge message is returned to the service invoker (`OrderLineItemReceiptAck` Response Unit).

```
// temporal aspects
dateTime::instant[ date=>date, time=>time ].
before(X,Y) :− before(X.date,Y.date)); ((equal(X.date,Y.date), before(X.time,Y.time)).

// geographical aspects (i.e., taxonomy of continents, countries, nations and cities)
worldwide.
europe::worldwide.
italy::europe.
checkContainmentOfPickupLocation(Request, Provider) :−
  Request[pickupLocation−>X], Provider[pickupLocations−>>Y], (X=Y;X::Y).

// price aspects (e.g., dimensional weight)
calculateWeightForPricing(ActualGoodWeight,GoodDimension,CalculatedWeightForPricing) :−
  DimensionalWeight is (GoodDimension.l∗GoodDimension.w∗GoodDimension.h)/166,
  (
    (ActualGoodWeight > X , CalculatedWeightForPricing is ActualGoodWeight);
    (ActualGoodWeight = X , CalculatedWeightForPricing is ActualGoodWeight);
    (ActualGoodWeight < X , CalculatedWeightForPricing is DimensionalWeight)
  )
.
```

Listing 1.1. Ontologies modeled for the discovery scenario

2. When all the lines have been received (`AllLinesConfirmationReceived?` Switch Unit), the XML serialization of the data for the current Pip3A-Purchase
   Order is extracted (`ExtractOrderData` XML-Out Unit) and a RosettaNet Purchase Order Confirmation (POC) message is created (`Lowering` Adapter Unit) and sent to the RosettaNet client (`SendPOC` Request Unit) and the status of the order is set to "Rosetta PO Confirmation sent" (`SetProcessStatus` Connect Unit).

The SOAP messages transformation to and from the WebML data model are performed by proper WebML units (`Adapter` units) that apply XSLT transformations; XSLT stylesheets can be designed with an XML-to-XML visual mapping tool. A prototype tool is included in our toolsuite, but any other tool can be used (e.g., IBM Clio [21]).

Discovery scenario. First of all, we focused on the design of a set of ontologies. Each of them defines a set concepts within a domain and the relationships between those concepts. In particular, we modeled four ontologies including date-time, location, products and shipments. Listing 1.1 shows the definitions of some concepts (both geographical concepts, like `italy`, and temporal concepts like `dateTime`, that is an instant enriched with a date and a time) and axioms (like `calculateWeightForPricing`, which encodes the rules to compute the weight used to determine the price of a package). The development for the two scenarios was kept to the minimum necessary. For example, our date-time ontology is not expressive enough to model the generic notion of "business day".

Secondly, we worked in designing the goals. We defined two classes of goals, respectively for shipment and for purchasing. In both cases, we modeled post-conditions part of the capabilities. Listing 1.2 shows the class of goals for the

```
/* The Shipment Goal Class */
goalClass_Shipment::goalClass[
  capability=>capabilityGoal_Shipment::capabilityGoal[
    postcondition=>requestsShipmentService
  ]
].

requestsShipmentService[
  requestedPickupLocation=>location,
  requestedDeliveryLocation=>location,
  currentDateTime=>dateTime,
  requestedPickupDateTimeInterval=>dateTimeInterval,
  requestedDeliveryDateTime=>dateTime,
  requestedDeliveryModality=>deliveryModality,
  requestedGuarantee=>guarantee,
  goodWeight=>float,
  goodDimension=>dimension,
  requestedShipmentPriceInterval=>priceInterval
].

/* A Shipment Goal Class Instance */
goalInstance:goalClass_Shipment[
  capability->_#:capabilityGoal_Shipment[
    postcondition->_#:requestsShipmentService[
      requestedPickupLocation->stanford,
      requestedDeliveryLocation->sacramento,
      currentDateTime->_#:dateTime[
          date->_#:date[dayOfMonth->28,monthOfYear->4,year->2006],
          time->_#:time[hourOfDay->23,minuteOfHour->0,secondOfMinute->0]
      ],
      requestedPickupDateTimeInterval->_#:dateTimeInterval[
        start->_#:dateTime[...],
        end->_#:dateTime[...],
      ],
      requestedDeliveryDateTime->_#:dateTime[...],
      requestedDeliveryModality->letter,
      requestedGuarantee->guaranteeYes,
      goodWeight->10,
      goodDimension->_#:dimension[l->100,w->100,h->100],
      requestedShipmentPriceInterval->_#:priceInterval[start->0,end->1000]
    ]
  ]
].
```

Listing 1.2. The shipment goal class and an instance of it in F-logic

shipment (named `goalClass_Shipment`) and an instance of it, which is named `goalInstance` and requires the shipment of a good of given properties at a given time from Stanford to Sacramento.

Then, we focused on Web Services, whose functions are described in terms of capabilities stating what must be true about the input to the service (*precondition*) and the state of the world (*assumption*) in order for the service to be invoked; capabilities also state what will be true about the output of the service (*postcondition*) and the state of the world (*effect*) after the service invocation. Moreover, an interface of a WSMO Web Service contains a choreography description to describe how to invoke the service and an orchestration description to state which other services are used by this service in order to provide its functionality.

We modeled the classes of Web Services for shipment and purchasing. In both cases, we modeled all the restrictions that must hold in order to invoke the service

```
/* The Shipment Service Class */
wsdClass_Shipment::wsdClass[
  capability=>capabilityWSD_Shipment::capabilityWSD[
    assumption=>restrictionsOnShipmentService,
    postcondition=>providesShipmentService
  ]
].

restrictionsOnShipmentService[
  minNumOfHoursBetweenOrderAndPickup=>integer,
  maxNumOfDaysBetweenOrderAndPickup=>integer,
  maxNumOfDaysBetweenOrderAndPickup=>integer,
  maxNumOfDaysBetweenOrderAndDelivery=>integer,
  minPickupDTInterval=>integer,
  maxPickupDTInterval=>integer,
  maxGoodWeight=>float,
  weightToDimensionalWeightThreshold=>float
].

providesShipmentService[
  pickupLocations=>>location,
  deliveryLocations=>>location,
  pickupTimeInterval=>timeInterval,
  price=>>shipmentPricing
].


/* An instance of the Shipment Service Class */
wsdInstance_Shipment13:wsdClass_Shipment[
  nonFunctionalProperties->_#[ dc_publisher->'Muller'],
  capability->_#:capabilityWSD_Shipment[
    assumption->_#:restrictionsOnShipmentService[
      minNumOfHoursBetweenOrderAndPickup=>0,
      maxNumOfDaysBetweenOrderAndPickup=>2,
      maxNumOfDaysBetweenOrderAndPickup=>5,
      minPickupDTInterval=>7200,
      maxPickupDTInterval=>259200,
      maxGoodWeight=>50,
    ],
    postcondition->_#:providesShipmentService[
      pickupLocations->>{northAmerica,africa,asia,europe},
      deliveryLocations->>{northAmerica,africa,asia,europe},
      pickupTimeInterval->_#:timeInterval[...],
      price->>{ _#:shipmentPricing[
        location->worldwide,
        deliveryModality->deliveryModality,
        guarantee->guaranteeNo,
        basePrice->10,
        pricePerWeight->5],
      }
    ]
  ]
].
```

Listing 1.3. The shipment service class and an instance of it in F-logic

as assumptions, and the results provided by the service as postconditions. Listing 1.3 shows the class of shipment Web Services (named `wsdClass_Shipment`) and an instance of it, which is named `wsdInstance_Shipment13` and describes a service that can deliver goods in north and south America, Africa, Asia and Europe. Finally, we modeled the matching rules within a wgMediator, which allow the engine to check whether a specific Web Service satisfies a goal. In our case, we wrote the rules by using the F-logic syntax and they can be divided in

```
/* Rule that calculates intermediate results */
calculateShipmentPrice(ShipmentPricing,Location,DeliveryModality,Guarantee,GoodWeight,
      PriceCalculated) :-
  (Location::ShipmentPricing.location;Location=ShipmentPricing.location),
  (DeliveryModality::ShipmentPricing.deliveryModality;DeliveryModality=ShipmentPricing.
      deliveryModality),
  (Guarantee=ShipmentPricing.guarantee;Guarantee=guaranteeNo),
  PriceCalculated is (ShipmentPricing.basePrice + (GoodWeight-1)*ShipmentPricing.
      pricePerWeight).

/* Rule that evaluates assumptions */
checkRestrinctionOnMaxNumOfDaysBetweenOrderAndPickupInterval(RequestsShipmentService,
      RestrictionsOnShipmentService) :-
  RequestsShipmentService[
    currentDateTime->OrderDateTime,
    requestedPickupDateTimeInterval->_[
      start->PickupDateTimeStart,
      end->PickupDateTimeEnd
    ]
  ],
  RestrictionsOnShipmentService[
    maxNumOfDaysBetweenOrderAndPickupStart->MaxDaysForStart,
    maxNumOfDaysBetweenOrderAndPickupEnd->MaxDaysForEnd
  ],
  daysBetween(PickupDateTimeStart,OrderDateTime,X),
  (X<MaxDaysForStart;X=MaxDaysForStart),
  daysBetween(PickupDateTimeEnd,OrderDateTime,Y),
  (Y<MaxDaysForEnd;Y=MaxDaysForEnd).

/* Rule that encodes a necessary condition */
checkContainmentOfPickupAndDeliveryLocation(RequestsShipmentService,
      ProvidesShipmentService) :-
  RequestsShipmentService[requestedPickupLocation->X],
  ProvidesShipmentService[pickupLocations->>Y],
  (X=Y;X::Y),
  RequestsShipmentService[requestedDeliveryLocation->H],
  ProvidesShipmentService[deliveryLocations->>K],
  (H=K;H::K).
```

Listing 1.4. Matching rules used by the wgMediator

three groups: those that calculate intermediate results (such as the price), those that evaluate the restrictions in the assumption part of the description and those that describe the transactions in the postcondition part of the description.

Listing 1.4 shows an example of each type of rule:

– The first rule calculates the shipment price by summing a base price and an amount of money that is proportional to the weight of the good. The appropriate tariff (i.e., base price and the other coefficient) is chosen according to the delivery location, the required guarantee, and the delivery modality.
– The second rule verifies that there is at most a predefined number of days between the date in which the order has been created and the first day in which the pick up may occur (the same control is verified for the last day in which the pick up can occur). The maximum number of allowed days is included in the description of every service.
– The last rule checks whether the pick up and delivery locations that have been required in the goal are included in the list of locations in which a service operates (e.g., the `requestedPickupLocation` of the goal has to be the

Table 3. The ranking criteria for the shipment service discovery

Rank	Checked Constraints
1	All restriction checked.
2	Shipment price is not checked.
3	No check for pickup interval.
4	Only weight and location are checked.

same or a subclass of a `pickupLocation` declared in the service description); otherwise the service is discarded.

Rules serve also the purpose to rank services according to some predefined criteria. We defined four levels of ranking of the discovery results, allowing for choosing an alternative shipment solution when no exact match is found (see Table 3).

The WebML model shares the goal model with the discovery engine and allows the user to interact with it by providing a browsable front-end for defining and refining the requests (see Figure 9). As results of a request the user sees a list of Web Services which satisfy goal criteria, and he may choose among them the one to be invoked.

4.3 Second Phase of the Challenge

To test the flexibility of the solutions proposed by participants to the challenge, the SWS challenge organizers introduced some changes in the two scenarios. The separated approaches we initially used to solve the two scenarios permitted us to address the new changes in a easy way. This also proved that our initial choice of adopting a Software Engineering approach for the mediation scenario and a Semantic Web approach for the discovery scenario was good.

Mediator scenario. In the Phase II of the challenge, the new requirements imposed a change in the format of the exchanged messages and a change in the mediation process. The change in the format required an adjustment of the data model: we introduced a new relationship between the entity ProductLineItem and the entity Partner. Then we modified the mediator process: when the Stock Management system is unable to fulfill a request from the customer and replies

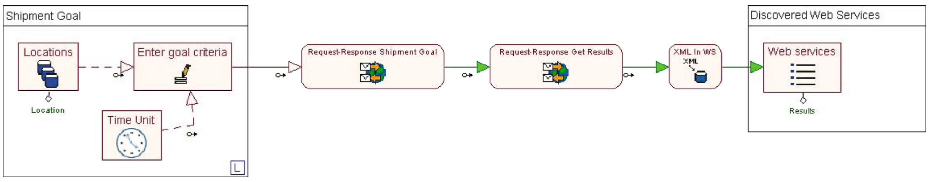


Fig. 9. The WebML model that provides the user interface towards the discovery engine

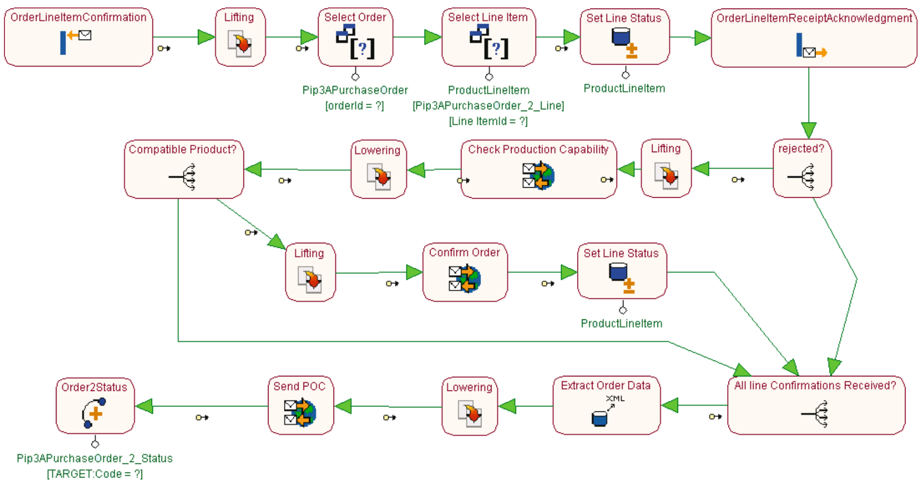

OrderLineItemConfirmation
Lifting
Select Order
Pip3APurchaseOrder [orderId = ?]
Select Line Item
ProductLineItem [Pip3APurchaseOrder_2_Line] [Line ItemId = ?]
Set Line Status
ProductLineItem
OrderLineItemReceiptAcknowledgment
rejected?
Lifting
Check Production Capability
Lowering
Compatible Prioduct?
Lifting
Confirm Order
Set Line Status
ProductLineItem
All line Confirmations Received?
Extract Order Data
Lowering
Send POC
Order2Status
Pip3APurchaseOrder_2_Status [TARGET:Code = ?]

Fig. 10. The WebML model of the modified portion of the Mediator (cfr. Figure 8b)

that the particular line item cannot be accepted, the new Mediator contacts the legacy Production Management system in order to search an alternative product. To fulfill this new requirement, we changed the mediator by introducing a chain of operations needed to query the new Production Management Web Service (Figure 10).

Discovery scenario. The new discovery scenario involved the capability of adding some instances to the ontologies (e.g. Oceania as a possible location), changing some instances of Web Services (e.g. updating latest pick up time, removing the distinction between `letter` and `wordwidePriorityExpress` as possible delivery modalities), and calculating the shipping price by means of more complex rules. Most of the changes required in the discovery scenario were addressed through minor modifications of the solution. The significant change was the introduction of the shipping price as a function on the weight, location, and delivery modality (see Listing 1.5). The calculated price is used by other rules to filter the results according to the maximal price expressed in the goal.

Generally speaking, because of the application of the Software Engineering methods and development processes, applying changes to the application proved to be relatively inexpensive in terms of development effort, especially if compared with pure Semantic Web-based approaches. Indeed, our solution allows faster development thanks to the use of high-level conceptual modeling. Most of the changes could be applied at a high level of abstraction, to the design specification of the application, instead of burdening with manual low level coding of the modified requirements, as typically needed for other approaches.

```
// Rule that calculate the price of the shipment
calculateShipmentPrice(ShipmentPricing, Location, DeliveryModality, Guarantee,
      CalculatedWeightForPricing, NumberOfPackages, PriceCalculated) :−
  // select the ShipmentPricing in accord with Location, DeliveryModality and Guarantee
  (Location::ShipmentPricing..location ; Location = ShipmentPricing..location),
  (DeliveryModality::ShipmentPricing.deliveryModality ; DeliveryModality = ShipmentPricing.
        deliveryModality),
  (Guarantee = ShipmentPricing.guarantee ; Guarantee = guaranteeNo),
  // calculate the price for one package
  PriceCalculatedForOnePackage is (ShipmentPricing.basePrice + CalculatedWeightForPricing ∗
        ShipmentPricing.pricePerWeight),
  (
    // if there is no a flat rate for a collection of packages, then the price for one package is
          multiplied for the number of packages
    (ShipmentPricing.additionalPricePerCollection = (−1),
    PriceCalculated is (PriceCalculatedForOnePackage ∗ NumberOfPackages));
    // if there is a flat rate for more than one package, than that rate is added to the price for
          one package
    (ShipmentPricing.additionalPricePerCollection > (−1),
    PriceCalculated is (PriceCalculatedForOnePackage + ShipmentPricing.
          additionalPricePerCollection))
  ).
```

Listing 1.5. The updated calculateShipmentPrice rule to calculate the shipment price

4.4 Third Phase of the Challenge

Phase III of the challenge did not introduce any substantial new requirement, therefore we concentrated on improving and refining our previous solution solving some of the open issues in our approach.

Mediation scenario. In the previous challenge edition we did not completely address some of the changes to the Mediation scenario. Among them we did not consider the process changes required in order to deal with the introduction of the optional shipment address for each line item. According to the modified scenario, line items must be grouped according to their shipment address and for each group an independent new order has to be sent to the Moon legacy system. We improved our mediator handling this requirement (see Figure 11): i.e., we introduced a loop over every shipment address associated to the incoming RosettaNet purchase order; inside the loop, a new shipment order for every different address is created, and each line item with that address is added to the new order; finally the order is closed, and the next address, if available, is processed.

Discovery scenario. The Glue discovery engine employed in Phase II assumes that every Web Service description is statically available. In general, Glue assumes no dynamic dependencies between Web Service descriptions and the goal. However, in many real scenarios a service exposes one or more operations for negotiating the service itself. This means that some operations of the services are intended to be invoked for retrieving dynamic information that may be used for the discovery and negotiation of the services. For instance, in the case of the SWS challenge, the shipping services provide an operation that calculates the price for shipping a product. This information can be exploited for a better

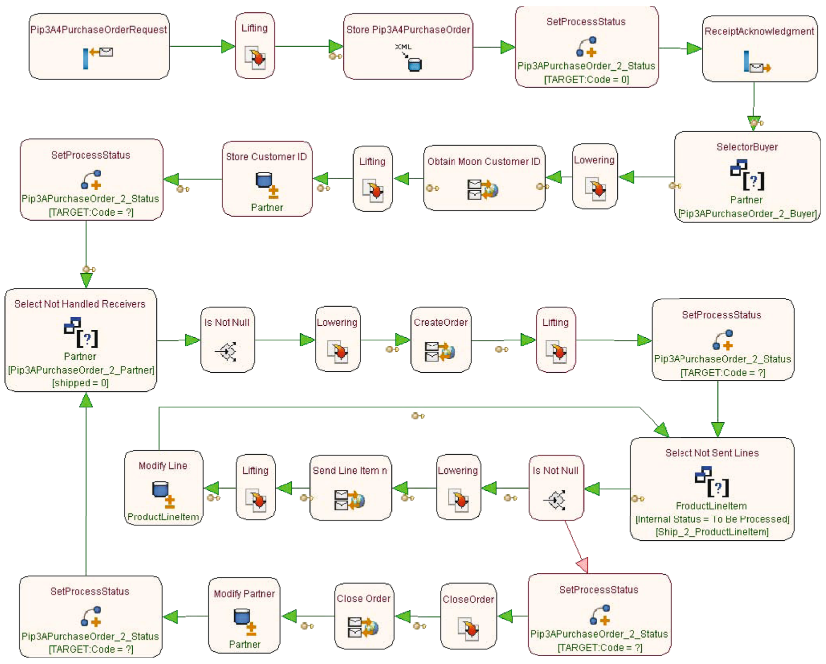


Fig. 11. The improved version of the Mediator (cfr Figure 8a)

discovery process that involves also negotiation of the service according to the price, which is dynamically calculated depending on the goods to be shipped.

In order to take this new requirement into consideration, several changes have been introduced in Glue. Indeed, the current implementation of Glue as standalone component only supports discovery; negotiation and selection are left to the application in which Glue is integrated. Nevertheless, by delegating invocation to an external component, we extended the version of Glue used in the challenge in order to support negotiation and selection.

At conceptual level we model both goal and Web Services making explicit differences among:

– *"discovery" capabilities*, which are static description of the service in terms of functional properties,
– *"selection" capabilities*, which are static or dynamic non functional descriptions;
– *"negotiation" capabilities*, which are description of the service that need to be evaluated by invoking one or more operation of the service (for describing the choreography we relay on WebML).

Not surprisingly the different modelling of goals and Web Services has some implication on Glue, but they are limited to the Communication Manager and the execution semantics. An important result is that the mediation centric approach adopted for the discovery covers also the need of negotiation and selection.

4.5 Handling the Requirements of the Future Edition of the SWS Challenge

The scenarios for the next series of the SWS challenge workshop have been already presented. In particular, a totally new discovery scenario is under development where a buyer of computer wants to choose among competing services offering computers and related accessories[4]. This challenges current technologies of the participants requiring dynamic services discovery and composition. We plan to solve this scenario by exploiting the concept of ggMediator to decompose the original goal of buying different products to a set of smaller goals that contain only a single product. In this way, we will be able to match the best provider to each single product.

5 Experience

In this Section we report on the joint work experience that allowed to build up our solution. Our results were contributed by two research groups with different expertise; the group from Politecnico contributed the know-how on Web Engineering, the group from Cefriel contributed expertise on Semantic Web technologies and techniques.

During the first meetings and for the period of time towards the first workshop of the challenge, the two teams were not confident about the results they could achieve combining their expertise, mainly because of the poor knowledge of the other party's technology. We went to the workshop at Stanford mainly for our curiosity about the current status of SWS research in other teams across the world. After the first workshop, our confidence on the possibility of reaching good results in the challenge totally changed. We had the chance to sit together for a longer time than in the previous meetings and to exchange our respective expertise: rapid prototyping and rule-based reasoning has lead to very effective software delivery.

We recall that we adopted an approach mainly based on the Software Engineering methods and techniques for the mediation scenario and an approach mainly based on the Semantic Web technologies for the discovery scenario. This allowed a good separation of tasks between the two groups: indeed, the Politecnico group developed the mediation scenario, while the Cefriel group focused on the discovery scenario. To improve the results and the timing, we assigned one person the role of being the interface between Cefriel and Politecnico. Later, we decided to mix our approaches; thus, we added process descriptions to the discovery engine and we started to use mediation rules. This improved the integration between the developed applications and lead also to a better exchange of knowledge between the two teams.

This proved to be really effective when the participants in the second phase of the challenge were required to freeze their solution and face changes to the

[4] http://sws-challenge.org/wiki/index.php/Scenario:_Discovery_II_and_Composition

scenarios in a limited amount of time in order to proof the flexibility of the initial solution.

Besides the achieved results, this experience allowed the two groups to extend their knowledge of the field of the other group, and to start a new challenging research effort between Cefriel and Politecnico.

6 Related Work

The Semantic Web is a new research area that in the last five years produced a great number of publications; However, few of them concern the systematic and methodological development of applications. Some early proposals (e.g., [22]) offered the definition of UML profiles for easily handling ontological definitions; however they haven't been adopted because of the lack of an overall methodology. A number of researches concentrated on the development of tools to support the generation of semantic descriptions for existing Web Services [23,24,25]. [26] presents an engineered approach to extraction of semantic annotations from XML schemas and documents to be published in dynamic Web applications. Most of these tools still require the learning of the annotation language used (e.g., OWL-S or WSMO) and hence do not rise the level of abstraction required from developers. Furthermore, they do not exploit the advantages of conceptual models of the Web Services to semi-automatically derive any part of the semantic descriptions.

Our research effort is more similar to the recent efforts of the Object Management Group (OMG)[5]. The OMG proposed the Ontology Definition Metamodel (ODM) [27] to define a suitable language for modeling Semantic Web ontology languages and hence Semantic Web applications in the context of the Model Driven Architecture (MDA) [28]. In [29] MIDAS, a framework based on MDA to model and develop Semantic Web applications, is introduced. The proposed framework focuses on the creation of Semantic Web Services and associated WSML descriptions using a UML model according to the MDA approach. This proposal inherits the limits of the MDA approach: the use of a UML model is not always fitting the Semantic Web needs, and often the model is too far from the implementation details to provide an effective automatic code generation. Furthermore, MIDAS does not provide a clear overall roadmap to the design of Semantic Web applications. The work of the W3C has created a Software Engineering Task Force dedicated to the Semantic Web[6] but its work is still under development.These proposals can be regarded as first contributions to the field, but they still do not provide a clear roadmap to the design of Semantic Web applications.

Other research efforts are converging on the proposal of combining Semantic Web Services (SWS) and Business Process Management (BPM) to create one consolidated technology, which we call Semantic Business Process Management (SBPM) [30]. This is based on the fact that mechanization of BPM can be

[5] http://www.omg.org/

[6] http://www.w3.org/2001/sw/BestPractices/SE/

addressed through machine-accessible semantics, that can be naturally provided by SWS frameworks (e.g., WSMO).

In the last years some efforts from the Web Engineering field have been redirected towards methodologies for developing Semantic Web Information Systems. Traditional Web design methodologies (like OOHDM [31]) and new approaches (like Hera [32]) are now focusing on designing Semantic Web applications. However, these methodologies are not supported by an effective CASE tool and they concentrate only on Semantic Web Portals instead of the development of Semantic Web Services.

6.1 Comparison to Other Approaches to the SWS Challenge

Mediation scenario. Other teams submitted interesting solutions to the mediation problem. In particular, [33] proposed a solution based on a SOA engineered framework called jABC/Jeti, that for some aspects is similar to our solution. [34] solved the process mediation using a BPEL engine, embedding it in the DIANE framework that provides for the data mediation; a similar solution was provided by the Meteor-S team [35]. The solution proposed by DERI [36], based on the WSMO/WSMX framework [3] is perhaps the most interesting solution and the most different with respect to ours. The DERI solution to the mediation scenario is purely semantic. The schema describing RosettaNet PIP 3A4, Customer Relationship Management (CRM), Order Management (OM) are ontologized. Next, Semantic Web Services for the CRM and the OM are generated together with the Goal templates for the service requester. The applied approach totally decouples the two partners involved in the mediation. Incoming requests by Blue are transformed to goal requests that are automatically matched against the best available service (only Moon in this case). The Moon Web Service description is associated with a choreography that specifies the legal interaction with the service. In WSMO, choreographies [37] are described through Abstract State Machines. In addition, a grounding must be defined from the semantic (WSMO) descriptions to the syntactic (WSDL) descriptions. Lifting and lowering has to be defined between the syntactic and semantic data models. WSDL descriptions are automatically generated by Axis and published on a Jetty server (internal to the WSMX). The WSMX runtime decomposes, according to provided mapping rules, the Blue message data to the different required Moon messages.

Discovery scenario. Only three teams provided solutions for the discovery scenario. The DERI WSMX-based solution from [36] is very similar to our solution. While in Glue WSMO is encoded in F-logic by means of Flora-2, the DERI solution used KAON2[7] interfaced via the WSML2Reasoner framework as internal reasoner to inference over the provided functionality of a service. KAON2 supports complex rules and arithmetic expressions, therefore the WSMX discovery component did not have to resort to an external arithmetic services but required calculations were carried out internally within the context of reasoner.

[7] http://kaon2.semanticweb.org

Similarly to Flora-2, KAON2 is a high precision reasoner giving explicit responses without the direct support for fuzziness or preferences. DERI provided simple support for preferences via non-functional properties within a goal which specify the selection criteria for services. The DERI submission directly supports service contracting required when the shipping price has to be dynamically obtained.

The solution provided by Jena university is based on DIANE [34], a system that uses its own ontology language, called DE (DIANE Elements) and DSD (DIANE Service Descriptions) [34]. Ontologies are very lightweight and the description elements of DSD used for ontologies can best be characterized as a small subset of F-logic without rules and quantifiers. In DSD, service offers are described as the set of effects that they can provide whereas service requests are described as the set of effects that are accepted by requesters. DSD is able to provide a more finegrained matching compared to Glue at the price of restricted expressiveness and limited compatibility to other semantic service frameworks.

7 Conclusions

This paper summarized our experience of applying Semantic Web Service and Web engineering techniques in the SWS Challenge 2006. We accomplished the coverage of all the requirement of the challenge by teaming up approaches best suited for each part of the challenge:

– we addressed the "*mediation scenario*" of the challenge with the WebML design and implementation of the wwMediator through the usage of the CASE tool WebRatio; and
– we addressed the "*discovery scenario*" by means of Glue WSMO discovery engine that was augmented in the deployed Semantic Web Service application and integrated with dynamic service invocation capabilities for providing negotiation.

Our approach extends the design flow supported for conventional Web applications [5] which leads the designer from the process modeling to the running Web application, by producing some intermediate artifacts (BPMN models, WebML skeletons, data models, hypertext models). Such models are enriched by imported ontological descriptions (on top of the figure) and are exploited for semi-automatically generating WSMO-compliant semantic (at the bottom of the figure): the ontology is derived from BP model, data model, and hypertext model; the Web Services capability description is derived from hypertext model; the choreography information is derived from BP model and hypertext model; the user goals are derived from the BP model.

The merits of our solution were twofold: the rooting in the tradition of Software Engineering and the use of sophisticated, up-to-date Web Enginnering technology. We have given the maximum attention to "the method", consisting of gathering requirements, producing formal specifications, and then adapting them to the new context of the Semantic Web. By using Webratio, models are automatically transformed into JSP scripts with embedded data management commands and with Web Service calls. Augmenting them so as to generate annotations and WSMO-compliant components has been possible, once the semantics of these components have been well understood in terms of classical E-R and BP models. Finally, the integration of WebRatio with Glue, the WSMO discovery engine, shows the advantages coming from the joint application of Software Engineering techniques and Semantic Web tools.

We believe that development methods and tools for the Semantic Web should not be different from the classic paradigms which are now dominating software design and deployment: habits of the software developer community should not change. Therefore, Semantic Web developers should adapt classic (UML-based) methods and tools hosted by classic tool frameworks (such as Eclipse). We fully agree with Charles Petrie's words: "*If semantic technology has a future — and I'm sure that it does — it's in Software Engineering*" [38]. Our participation to the challenge is a first attempt in this direction, where two groups of Web engineering experts and Semantic Web experts joined their culture, habits, and tool experience. The ability of our solution to adapt to changes is mostly the merit of our use of enhanced Software Engineering methods and platforms.

Our future work aims at implementing a complete extension of the WebML methodology towards the design of Semantic Web Services applications, supported by a design environment based upon WebRatio [6]. Other ongoing work at Politecnico di Milano and Cefriel can provide useful components to be used in the framework. We already extensively presented Glue, a discovery engine that can be used by Semantic Web applications with simple customization. In addition, we are building generic tool support for XML-2-XML mapping based on graphic notation and upon inference, that extends tools such as Clio [21] and makes such technology available as part of the WebRatio environment; transformations are developed by simple graphic drawings and are inferred whenever appropriate [39]. Such transformations are the basis for the semi-automatic development of "syntactic mediators" (i.e., those mediators doing simple format transformations) and are in general very helpful for the design and the implementation of arbitrary mediators.

References

1. DERI Stanford: Semantic web service challenge (2007), http://www.sws-challenge.org
2. RosettaNet: Purchace order (pip 3a4) (2007), http://www.rosettanet.org/PIP3A4
3. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer, New York (2006)

4. Valle, E.D., Cerizza, D.: The mediators centric approach to automatic web service discovery of glue. In: Hepp, M., Polleres, A., van Harmelen, F., Genesereth, M.R. (eds.) MEDIATE 2005. CEUR Workshop Proceedings, Amsterdam, The Netherlands, CEUR-WS.org, vol. 168, pp. 35–50 (December 2005), http://CEUR-WS.org/Vol-168/MEDIATE2005-paper3.pdf
5. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kauffmann, San Francisco (2002)
6. WebModels s.r.l.: Webratio site development suite (2007), http://www.webratio.com
7. Brambilla, M., Celino, I., Ceri, S., Cerizza, D., Della Valle, E., Facca, F.M.: A Software Engineering Approach to Design and Development of Semantic Web Service Applications. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273. Springer, Heidelberg (2006)
8. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web Service Modeling Ontology. Applied Ontologies 1(1), 77–106 (2005)
9. de Bruijn, J., Lausen, H., Polleres, A., Fensel, D.: The web service modeling language WSML: An overview. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011. Springer, Heidelberg (2006)
10. Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C.: WSMX - A Semantic Service-Oriented Architecture. In: Proceedings of the 2005 IEEE International Conference on Web Services (ICWS 2005), Washington, DC, USA, pp. 321–328. IEEE Computer Society, Los Alamitos (2005)
11. Fensel, D., Bussler, C.: The web service modeling framework WSMF. Electronic Commerce Research and Applications 1(2), 113–137 (2002)
12. de Bruijn, J., Lausen, H., Polleres, A., Fensel, D.: The web service modeling language: An overview. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011. Springer, Heidelberg (2006)
13. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. J. ACM 42(4), 741–843 (1995)
14. Valle, E.D., Cerizza, D.: Cocoon glue: a prototype of wsmo discovery engine for the healthcare field. In: Bussler, C., Fensel, D., Keller, U., Sapkota, B. (eds.) 2nd WSMO Implementation Workshop (WIW 2005). CEUR Workshop Proceedings, CEUR-WS.org, vol. 134, pp. 1–12 (2005), http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-134/cocoon-wiw05.pdf
15. Berler, M., Eastman, J., Jordan, D., Russell, C., Schadow, O., Stanienda, T., Velez, F.: The object data standard: ODMG 3.0. Morgan Kaufmann Publishers, San Francisco (2000)
16. Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process modeling in web applications. ACM Trans. Softw. Eng. Methodol. 15(4), 360–409 (2006)
17. Manolescu, I., Brambilla, M., Ceri, S., Comai, S., Fraternali, P.: Model-driven design and deployment of service-enabled web applications. ACM Trans. Internet Techn. 5(3), 439–479 (2005)
18. Facca, F.M., Brambilla, M.: Extending WebML towards Semantic Web. In: Proceedings of the 16th international conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12 (2007)
19. MusicBrainz: Musicbrainz project (2007), http://musicbrainz.org

20. W3C: Rdfa primer 1.0: Embedding rdf in xhtml (2007), http://www.w3.org/TR/xhtml-rdfa-primer/
21. Hernández, M.A., Miller, R.J., Haas, L.M.: Clio: a semi-automatic tool for schema mapping. SIGMOD Rec. 30(2), 607 (2001)
22. Djuric, D., Gasevic, D., Devedzic, V., Damjanovic, V.: Uml profile for OWL. In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004. LNCS, vol. 3140, pp. 607–608. Springer, Heidelberg (2004)
23. Elenius, D., Denker, G., Martin, D., Gilham, F., Khouri, J., Sadaati, S., Senanayake, R.: The OWL-S editor - a development tool for semantic web services. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 78–92. Springer, Heidelberg (2005)
24. Jaeger, M.C., Engel, L., Geihs, K.: A methodology for developing owl-s descriptions. In: Panetto, H. (ed.) Proceedings of the INTEROP-ESA, Workshops, Geneva, Switzerland, pp. 153–166. Hermes Science Publishing (2005)
25. Kerrigan, M.: D9.1v0.2 web service modeling toolkit (WSMT). Technical report, DERI (2005), http://www.wsmo.org/TR/d9/d9.1
26. Reif, G., Gall, H., Jazayeri, M.: Weesa: Web engineering for semantic web applications. In: Proceedings of the 14th International Conference on World Wide Web, pp. 722–729. ACM Press, New York (2005)
27. OMG: Ontology definition metamodel (2007), http://www.omg.org/cgi-bin/doc?ad/06-05-01.pdf
28. OMG: Model driven architecture (2007), http://www.omg.org/cgi-bin/doc?omg/03-06-01
29. Acuña, C.J., Marcos, E.: Modeling semantic web services: a case study. In: ICWE 2006: Proceedings of the 6th international conference on Web engineering, pp. 32–39. ACM Press, New York (2006)
30. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic business process management: A vision towards using semantic web services for business process management. In: ICEBE 2005: Proceedings of the IEEE International Conference on e-Business Engineering, Washington, DC, USA, pp. 535–540. IEEE Computer Society, Los Alamitos (2005)
31. Lima, F., Schwabe, D.: Application Modeling for the Semantic Web. In: 1st Latin American Web Congress (LA-WEB 2003), Empowering Our Web, Sanitago, Chile, November 10-12, pp. 93–102. IEEE Computer Society, Los Alamitos (2003)
32. Vdovjak, R., Frasincar, F., Houben, G.J., Barna, P.: Engineering Semantic Web Information Systems in Hera. J. Web Eng. 2(1-2), 3–26 (2003)
33. Kubczak, C., Steffen, B., Margaria, T.: The jabc approach to mediation and choreography. In: 2nd Semantic Web Service Challenge Workshop (June 2006)
34. Küster, U., König-Ries, B.: Discovery and mediation using diane service descriptions. In: Third Workshop of the Semantic Web Service Challenge 2006 - Challenge on Automating Web Services Mediation, Choreography and Discovery, Athens, GA, USA (November 2006)
35. Wu, Z., Harney, J.F., Verma, K., Miller, J.A., Sheth, A.P.: Composing semantic web services with interaction protocols. Technical report, LSDIS Lab, University of Georgia, Athens, Georgia (2006)
36. Zaremba, M., Vitvar, T., Moran, M., Hasselwanter, T.: WSMX discovery for sws challenge. In: Third Workshop of the Semantic Web Service Challenge 2006 - Challenge on Automating Web Services Mediation, Choreography and Discovery, Athens, GA, USA (November 2006)

37. Roman, D., Scicluna, J.: Ontology-based choreography of wsmo services. Wsmo final draft v0.3, DERI (2006), http://www.wsmo.org/TR/d14/v0.3/
38. Petrie, C.J.: It's the programming, stupid. IEEE Internet Computing 10(3), 95–96 (2006)
39. Brambilla, M., Ceri, S., Comai, S., Tziviskou, C.: A visual data mapping tool for software components interactions in service-oriented architectures. In: Kokol, P. (ed.) IASTED Conf. on Software Engineering, pp. 33–38. IASTED/ACTA Press (2006)

Automatic Generation and Evolution of Model Transformations Using Ontology Engineering Space*

Stephan Roser and Bernhard Bauer

Programming of Distributed Systems
Institute of Computer Science, University of Augsburg, D-86135 Augsburg, Germany
{roser,bauer}@ds-lab.org

Abstract. Model-driven software development facilitates faster and more flexible integration of information and communication systems. It divides system descriptions into models of different view points and abstraction levels. To effectively realize cross-organisational collaborations, it is an important prerequisite to exchange models between different modelling languages and tools. Knowledge is captured in model transformations, which are continuously adjusted to new modelling formats and tools. However, interoperability problems in modelling can hardly be overcome by solutions that essentially operate at syntactical level. This paper presents a novel approach using the capabilities of semantic technologies in order to improve cross-organisational modelling by automatic generation and evolution of model transformations.

1 Introduction

New challenges arise with the development of increasingly complex systems across enterprises. Systems must be more adaptable to changing requirements. It has to be possible to compose systems from existing components, to replace parts of systems, and to integrate existing systems. Model-driven software development (MDSD) is a young but promising approach to deal with these challenges. However, to enable an efficient development of flexible cross-organisational information and communication systems one needs to support interoperability in modelling enterprises and applications. As enterprises often apply different methodologies, they need to share their enterprise models and knowledge independent of languages and tools. Therefore, one needs to develop mappings between different existing enterprise modelling formalisms based on an enterprise modelling ontology as well as tools and services for translating models (IDEAS analysis - gap 12 [21]). The IDEAS network stated in its vision for 2010 [22] requirements to enable enterprises to seamlessly collaborate with others. According to this, it is necessary to integrate and adapt ontologies in architectures and infrastructures to the layers of enterprise architecture and to operational models. This can be done by applying mappings between different enterprise model formalisms based on an enterprise modelling ontology. Heterogeneous business models can be semantically enriched by ontologies to achieve a shared understanding of the enterprise domain.

* This paper is a revised and extended version of [39].

© Springer-Verlag Berlin Heidelberg 2008

[10] proposes a rather abstract interoperability framework for MDSD of software systems, which supports the business interoperability needs of an enterprise. Mutual understanding on all levels of integration, conceptual, technical, and applicative level, has to be achieved. One uses the conceptual reference model to model interoperability, whereas metamodels and ontologies are used to define model transformations and model mappings between the different views of an enterprise system. Still, solutions like [5], [27], and [41], that aim at improving such kind of interoperability, address the problems of different representation formats, modelling guidelines, modelling styles, modelling languages, and methodologies at the syntactical level. Their focus is on metamodels' abstract and concrete syntax. Approaches providing interoperability solutions based on ontologies and automated reasoning lack key features for modelling [16]. For example they do not store trace information of transformation executions in order to enable transactions or incremental updates [37].

In this work we propose the approach of *ontology-based model transformation* (ontMT). It integrates ontologies in modelling by utilising different technological spaces [26] (namely MDA and Ontology technological space) to automate the generation and evolution of model transformations. Interoperability in modelling is fostered by employing automated reasoning technologies from the ontology engineering technological space for the generation of model transformations. We present how the ontMT approach can be realized as a *semantic-enabled model transformation tool (Sem-MT-Tool)* in a semantic-enabled modelling and development suite (see [2]). This tool applies technology bridging MDA and Semantic Web approaches and makes use of the capabilities and benefits of both approaches.

This paper is organized as follows: After introducing background information in Section 2, we provide a problem description in Section 3. The approach of ontology-based model transformation is presented in Section 4. Section 5 provides insights into the components of a semantic-enabled modelling and development tool realizing the ontMT approach. Section 6 contains a detailed case study. Section 7 discusses the ontMT approach and Section 8 provides related work. Finally, we conclude with a short summary and outlook in Section 9.

2 Background and Context

Model-driven Software Development (MDSD): MDSD, as a generalization of OMG™'s Model Driven Architecture paradigm (MDA®), is an approach to software development based on modelling and automated transformation of models to implementations [15]. In MDSD models are more than abstract descriptions of systems: they are the key part in defining software systems, as they are used for model- and code generation. Largely automated model transformations refine abstract models to more concrete models or simply describe mappings between models of the same level of abstraction. As model transformations play a key role in MDSD, it is important that transformations can be developed as efficiently as possible [16].

Models: The definition of the mega-model[1] presented in [14] describes a model as a system that enables us to give answers about a system under study without the need to

[1] Models about modelling are called mega-models, while metamodels are models of modelling languages.

consider this system directly. In short, a model is a *representation of* a system, whereas systems can be physically observable elements or more abstract concepts like modelling languages. A modelling language is a set of models, and models are *elements of* a modelling language. Models *conform to* a model of the modelling language, i.e. a metamodel. Those metamodels can be used to validate models. For one modelling language multiple (meta)models can exist, which can differ in the language they are described in.

Model transformations: Model transformations (MTs) are specified between metamodels. The execution of a model transformation transforms models conforming to the source metamodel into models conforming to the target metamodel. *Vertical model transformations* refine abstract models to more concrete models, while *horizontal model transformations* describe *mappings* between models of the same abstraction level. With the MOF 2.0 Query/View/Transformation specification [37] (QVT) the OMG provides a standard syntax and execution semantics for transformations used in a MDSD tools chain. The QVT relations language allows to specify relationships between MOF models declaratively. It supports complex object pattern matching and implicitly traces the transformation execution. A relational transformation defines how a set of models can be transformed into another. Relations in a transformation declare constraints that must be satisfied by the elements of the candidate models. Domains are part of a relation and have patterns. Patterns can be considered as templates for objects and their properties that must be located, modified, or created in a candidate model that should satisfy the relation. A domain pattern consists of a graph of object template expressions (OTEs) and property template items (PTIs). An OTE specifies a pattern that matches model elements. It is uses a collection of PTIs to specify constraints on the values of the properties of model elements.

```
relation PackageToSchema {
  domain uml p:Package {name=pn}
  domain rdbms s:Schema {name=pn}
}
```

Listing 1: QVT sample relation

Listing 1 depicts a relation specified in QVT relational syntax [37, p.13]. Two domains are declared that match elements of the *uml* and *rdbms* models respectively. Each domain specifies a pattern: a *Package* with a name, and a *Schema* with a name. Both *name* properties being bound to the same variable *pn* implying that they should have the same value.

Ontology: Ontologies are considered a key element for semantic interoperability. They act as shared vocabularies for describing the relevant notions of application areas, whose semantics is specified in a (reasonably) unambiguous and machine-processable form [7]. According to [33] an ontology differs from existing methods and technologies in the following way: (i) the primary goal of ontologies is to enable agreement on the meaning of specific vocabulary terms and to facilitate information integration across individual languages. (ii) Ontologies are formalized in logic-based representation languages. Thus, their semantics is specified in an unambiguous way. (iii) The representation languages come with executable calculi enabling querying and reasoning at run time. *Application ontologies* contain the definitions specific to a particular application [19]. *Reference ontologies* refer to ontological theories, whose focus is to clarify the intended meanings of terms used in specific domains.

Technological Spaces: Kurtev et al. [26] introduce the concept of *technological spaces* (TS) aiming to improve efficiency of work by using the best possibilities of different technologies. A technological space is, in short, a zone of established expertise and ongoing research. It is a working context together with a set of associated concepts, body of knowledge, tools, required skills, and possibilities. Initially five technological spaces (MDA TS, XML TS, Abstract Syntax TS, Ontology TS, DBMS TS) have been presented in [26], of which the *MDA TS* and the *Ontology TS* are important for our work. In the MDA TS models are considered as first-class citizens, that represent particular views on the system being built. The Ontology TS can be considered as a subfield of knowledge engineering, mainly dealing with representation and reasoning. The ontology engineering space performs outstanding in traceability, i.e. in the specification of correspondences between various metamodels, while the MDA TS is much more applicable to facilitate aspect or content separation. With the Ontology Definition Metamodel (ODM) [35] the OMG issues a specification defining a family of independent metamodels and mappings among these metamodels. These metamodels correspond to several international standards for ontology definition. ODM comprises metamodels for RDF(S), OWL, common logic (CL), topic maps (TM), and as a non normative part description logic (DL). Metamodels for RDF(S) and OWL represent more structural or descriptive representations, which are commonly used in the semantic web community. ODM further defines transformations between the UML2 metamodel and the OWL metamodel defined in ODM.

Semantics: The notion of the term semantics depends on the context it is used in and varies by the people using it. As the root of the problem Harel and Rumpe [20] identify insufficient regard for the crucial distinction between syntax and true semantics. Thus we clarify a few terms that have particular significance to this work.

- **Syntax:** *Syntax* N_L is the notation of a language *L*. It is distinguished between the concrete syntax, the textual or graphical representation of the language, and an abstract syntax or metamodel, being the machine's internal representation. A metamodel is a way to describe the syntax of a language [20].
- **Semantics:** Semantics is the meaning of language, that is expressed by relating the syntax to a semantic domain. The description of a *semantic domain S* (its notation is N_S) can vary from plain English to mathematics. Semantics is defined by a *semantic mapping* $M: L \rightarrow S$ from the language's syntax to its semantic domain [20].
- **Ontological:** According to [35] 'an ontology defines the common terms and concepts (meaning) used to describe and represent and area of knowledge'. Talking about 'ontological' we mean technology of the Ontology TS. That is to say technology based on logic like RDF(S) or OWL, which is used by the semantic web community to describe e.g. vocabularies or ontologies.

3 Problem Description

To enable collaboration in enterprise and systems modelling, enterprises have to be supported by interoperability solutions for model sharing and model exchange independent of modelling languages and tools. Also the evolution of model transformations has to be considered. To maintain and reuse existing model transformations, model transformations have to be adjusted to new modelling languages or styles. This section illustrates these challenges via a MDSD scenario. It further discusses the problems that possible automation solutions face.

3.1 A MDSD Scenario

Figure 1 illustrates the application of MDSD to cross-organisational business process development. The vertical dimension distinguishes the different layers of abstraction applied in MDSD, and the horizontal dimension represents the collaborative modelling between two enterprises A and B. Models of enterprises A and B have to be shared at different levels of abstraction in order to agree on and develop cross-organisational business processes.

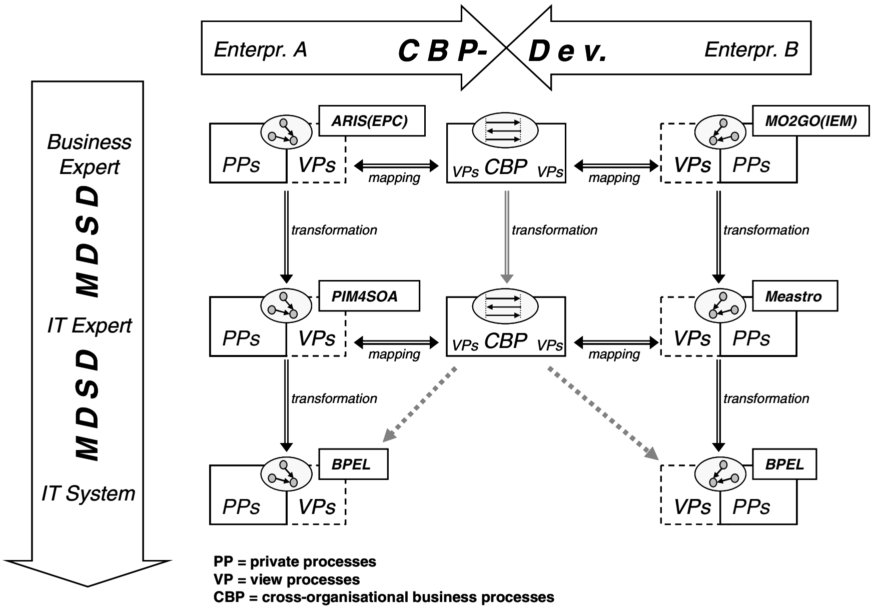


Fig. 1. Scenario realizing cross-organisational business process modelling and execution

A concrete scenario, implementing cross-organisational business process modelling and execution like show in Figure 1, has been developed in the ATHENA project (more details can be found in [18]). Enterprises A and B develop models for their processes (privates processes (PPs), view processes (VPs), and cross-organisational business processes (CBPs)) at three levels of abstraction, i.e. business expert, IT expert, and IT-system level. Vertical transformations, like presented in [1], encode knowledge about the architecture and the platform in order to transform models from higher to lower abstraction level. For example ARIS models (eEPCs [25]) are transformed to models conforming to PIM4SOA [3]. Enterprises A and B use different modelling tools and languages at the various abstraction levels. To develop cross-organisational business processes, both enterprises have to provide public parts of their models as a basis for discussion during collaborative modelling.

However, some issues prevent a smooth realization of such a MDSD scenario:

- **Exchange of models:** Models are shared across inter-organisational relationships. Hence, mappings have to be developed between the various enterprises' modelling languages and tools. This is necessary to achieve a shared understanding of cross-organisational business processes and to enable collaborative MDSD.
- **Evolution of model transformations:** Over a period of time enterprises will apply new (versions) of modelling languages, metamodels, and modelling styles. Therefore, existing transformations have to be maintained, adjusted, or redeveloped.

3.2 Problem Statement

Managing and developing model transformations are error-prone and long lasting tasks. Since model transformations are a kind of metaprogramming, they require a deep knowledge of all the underlying modelling technology, which is, in most cases, quite hard to learn. Thus, it is beneficial to provide support with a solution that automates model transformation development and adjustment tasks. Despite the multiplicity of model transformations and application scenarios, the core principles of modelling (i.e. representing information about real world things in models) and problems of such automation solutions remain the same. The core barriers to model exchange and maintenance of model transformations are multiple representation formats and different modelling styles, serving the particular application.

- **Different representation format:** The trend towards the use of domain specific languages (DSLs) leads more and more people to create their own domain specific models (DSMs). This naturally results in a variety of different languages and metamodels. To exchange models that conform to these various metamodels (abstract syntax), model transformations have to be developed. Often there are multiple model transformations for the same modelling language. Also time and again new versions of metamodels, e.g. the metamodels for UML 1.x and UML 2.x, are released. Whenever new versions replace the old ones, new model transformations have to be developed and existing model transformations have to be adjusted. Though visual representations (concrete syntax) should be decoupled from internal representation (abstract syntax), different concrete syntax is often considered in model transformations to provide e.g. views on models.
- **Different semantics:** Since the semantics of modelling languages' concepts is rarely formally specified (in the UML specification this is plain English), different people and organisations can associate different semantics with the same concepts used in the metamodel. This is often done by applying special modelling styles and representation guidelines. Again, model transformations have to be specified enabling sensible exchange of models according to the respective interpretations.

4 Ontology Supported Model Transformations

OntMT facilitates methods to generate and adjust model transformations despite of structural and semantic differences of metamodels. Different representation formats and different semantics (as described in Section 3.2) are overcome by applying semantic web technology of the Ontology TS. In ontMT metamodels are annotated through the elements of a reference ontology (RO) and reasoning is applied to the RO and the annotations. OntMT allows to generate and adjust common model transformations automatically in order to apply MDSD in the MDA TS.

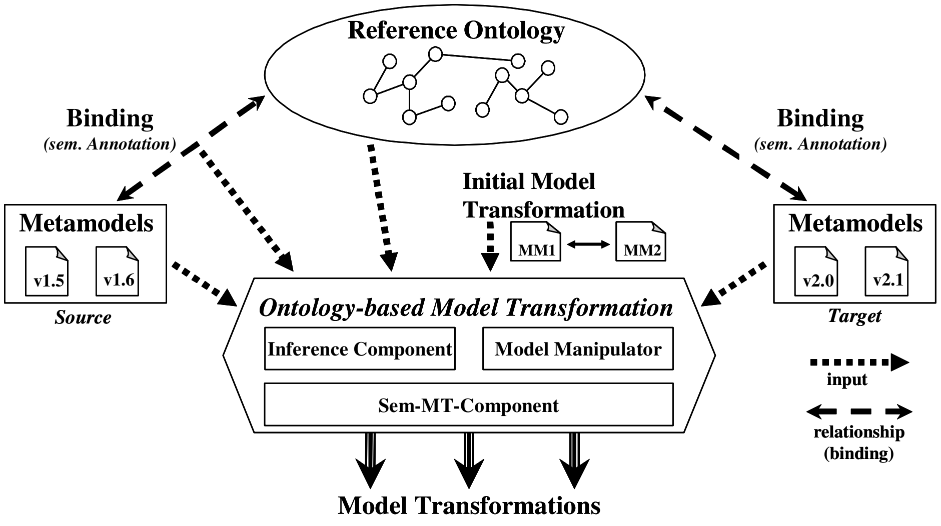


Fig. 2. Ontology-based model transformation – overall approach

Figure 2 depicts the overall approach of ontMT. Different versions of metamodels are bound to a reference ontology of a certain domain. Bindings (semantic annotations) specify the semantic mapping of metamodels to the semantics of their concepts, i.e. to the reference ontology. To generate model transformations for various model transformation languages and to adjust existing model transformations, ontMT makes use of reasoning mechanisms. The metamodels and the reference ontology are given, while the bindings of the metamodels to the reference ontology have to be specified. Finally, an initial model transformation is needed. For the evolution of model transformations the initial model transformation is the model transformation that shall be reused or adjusted (see Section 4.2). The initial model transformation (e.g. from metamodel v1.5 to metamodel v2.0) encodes transformation rules and especially the semantics of the model transformation. If for example the metamodel v2.0 is replaced with a version 2.1 only the delta between these metamodel has to be considered to adjust the existing model transformation. The new model transformation is generated by substituting the concepts of metamodel v2.0 with the concepts of metamodel v2.1 in the initial model transformation. In the case of automated mapping generation, a bootstrapping algorithm generates an initial model transformation (see Section 4.1).

4.1 Generation of Model Transformations

Model transformations between various modelling languages can be automatically derived and generated with the ontMT approach. In this section we describe the procedure to generate mappings, i.e. semantically identical model transformations, between two modelling languages *A* and *B*. We illustrate the procedure via a strongly simplified example, where *A* and *B* both consist of two concepts: *A={Process, Task}* and *B={EPC, EPCElement}*.

For both languages exists an abstract syntax N_A/N_B in various technological spaces: A has (like B) an abstract syntax in the MDA TS $N_{A\text{-}mda}$ and the Ontology TS $N_{A\text{-}ont}$ which are synchronized. Thus, one can work with the syntax and the capability of that technological space, that is better suited for solving a problem (see Figure 3). The semantics of the concepts is described by the means of the semantic domain *SD* and its notation in a reference ontology N_{RO} (e.g. OWL) respectively. Semantics of the languages is defined by semantic mappings from the languages to the semantic domain: M_A*: A* $\rightarrow$ *SD* and M_B*: B* $\rightarrow$ *SD*. In this example, the semantic domain is given as *SD={Activity, Action}*, while the semantic mappings are M_A*={Process* ≅ *Activity, Task* ≅ *Action}* and M_B*={EPC* ≅ *Activity, EPCElement* ≅ *Action}*[2].

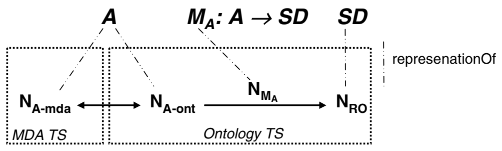


Fig. 3. Modelling language, semantic mapping, semantic domain and their representations

The ontological grounding[3] is a notation of the semantic mapping from $N_{A\text{-}ont}$ to N_{RO}. The goal of the transformation to generate is to define 'identity' relationships between the concepts of *A* and *B*. The model transformation MT_{mapAB}: *A* ↔ *B* between *A* and *B* has the following semantics: $M_{MTmapAB}$: MT_{mapAB} $\rightarrow$ *id*, where *id* is the identical mapping. The generation procedure works on the model of the model transformation and the models of the modelling languages. It exploits the ontological grounding to the reference ontology. On the basis of reasoning results gained in the Ontology TS (*{Process* ≅ *EPC, Task* ≅ *EPCElement}*), modification operations are called to obtain the new model transformation working solely on the model of the model transformation. To generate the model transformation MT_{mapAB}, the following steps are performed (see Figure 4a):

- ⓪ A bootstrapping algorithm generates the model transformation MT_{mapAA}*: A* ↔ *A*, which is a mapping of *A* on itself. This bootstrapping step is necessary to obtain a first model of the model transformation (transforming N_A to N_A') [4], which only has to be adjusted by modifications operations. Assuming the same ontological grounding for N_A and N_A', the bootstrapping model transformation is an *id*: M_{MTma}

[2] ≅ stands for equivalence.

[3] The definition of the ontological grounding is a semantic annotation comprising static semantics of the metamodels, i.e. the semantics of the concepts and an ontology respectively.

[4] Such a mapping can be generated on the basis of a metamodel in the MDA TS. The appropriate mapping rules are generated by traversing the metamodel via its composite aggregation (in short composition) relationships.

$_{pAA}$: $MT_{mapAA} \rightarrow id$. In our example the model transformation relations identified by the bootstrapping are $MT_{mapAA}\{Process \leftrightarrow Process,\ Task \leftrightarrow Task\}$.

- ① The inference engine derives relationships between N_A' and N_B in the Ontology TS. This is possible, since both N_A' and N_B are mapped to the same reference ontology N_{RO}. It is automatically computed, how the concepts of N_A' can be substituted by semantically identical concepts of N_B ($\sigma(MT_{mapAA})=MT_{mapAB}$). Those relationships can be transferred to the MDA TS as the modelling languages A and B have synchronous representations in both MDA TS and Ontology TS. The substitutions computed for our example are *[EPC/Process]* and *[EPCElement/Task]*.
- ② Finally, the concepts of N_A' are substituted with the concepts of N_B in the model of MT_{mapAA} and we obtain a model of the model transformation MT_{mapAB} with $M_{MTmapAB}$: $MT_{mapAB} \rightarrow id$. The substitution is performed via modification operations on the model of the model transformation MT_{mapAA} in MDA TS. In the example the following model transformation relations are generated: $MT_{mapAB}\{Process \leftrightarrow EPC,\ Task \leftrightarrow EPCElement\}$.

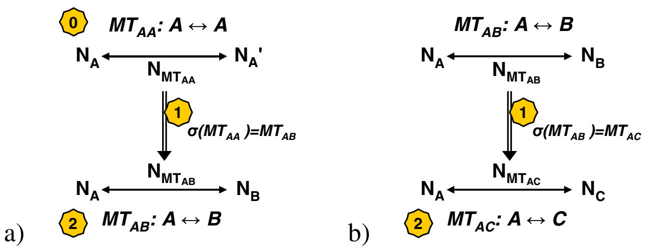


Fig. 4. Procedure of a) automated mapping generation, b) model transformation evolution

4.2 Evolution of Model Transformations

OntMT also fosters the evolution and reuse of existing model transformations. Instead of performing the bootstrapping step, the procedure for model transformation evolution takes the model transformation that shall be reused as input (see Figure 4b). This initial model transformation MT_{mapAB}: $A \leftrightarrow B$ encodes knowledge about how modelling language A is translated into B. The steps ① and ② are the same as for automated mapping generation. In step ①, a substitution $\sigma(MT_{mapAB})=MT_{mapAC}$ is computed on the basis of inference results. Step ② applies this substitution and generates a new version of the initial model transformation MT_{mapAC}: $A \leftrightarrow C$. The bootstrapping step helps to extend ontMT to scenarios where existing model transformations are adjusted. Avoiding to derive model transformations directly from ontologies results in a more flexible and well-structured architecture. OntMT can both generate new model transformations and reuse knowledge encoded in existing transformations. Issues concerning the model transformation, like checking if its model conforms to the QVT metamodel or considering the cardinality of associations' ends, are all dealt within the MDA TS. The Sem-MT-Component invokes modifications operations on the basis of the reasoning results and the application of heuristics.

5 Realization of OntMT

This section presents the components and concepts of ontMT realized as a tool for a semantic-enabled modelling and development suite, its parts and functionality.

5.1 Components of a Sem-MT-Tool

OntMT, as part of our vision of a semantic-enabled modelling and development suite, is realized as Sem-X-Tool (see Figure 5) [2]. The infrastructure provides basic functionality including a bridge ⓪ between models of the MDA TS and application ontologies of the Ontology TS (like it is described in [6]) and an inference component, which can be individually configured and used by Sem-X-Tools registered at the infrastructure. Sem-X-Tools, like the Sem-MT-Tool presented in this paper, are built on top of the infrastructure. They consist of a model manipulator, a Sem-X-Component, and a rule set. The model manipulator reads, creates, modifies and deletes models of the model repository ①. It delivers information about models to the Sem-X-Component ② and provides interfaces for model manipulation ③. The Sem-X-Component implements the core functionality of a Sem-X-Tool. It makes use of the reasoning results gained by inferring ontologies and computes the respective model manipulation ④. Since Sem-X-Tools are based on different relationships between the ontologies' elements, each Sem-X-Tool has its own set of reasoning rules.

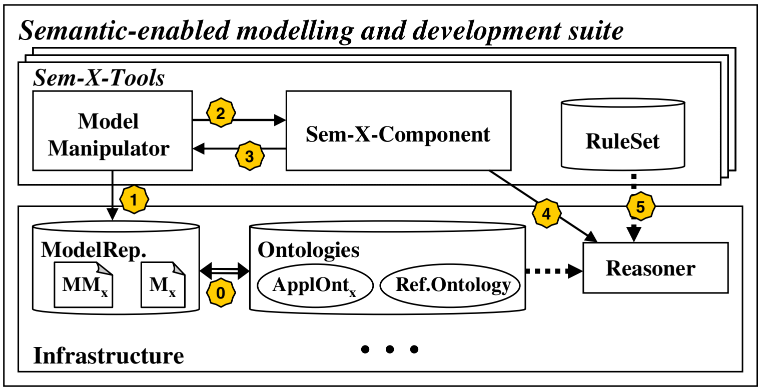


Fig. 5. OntMT as part of a semantic-enabled modelling and development suite

Figure 6 shows the architecture of the components building the Sem-MT-Tool which is an instantiation of the Sem-X-Tool. The model manipulator provides functionality via three interfaces: one that identifies the concepts of a metamodel that have to be substituted in a model transformation, one that performs a substitution of a metamodel's concepts in the model transformation, and one that provides validation functionality for the generated model transformation. The inference component provides an interface for accessing the reasoning results, i.e. the relationships between the metamodel elements. The Sem-MT-Component is the component of the Sem-MT-Tool, which connects the inference results of the Ontology TS to concrete modification actions on the models of the model transformation in the MDA TS.

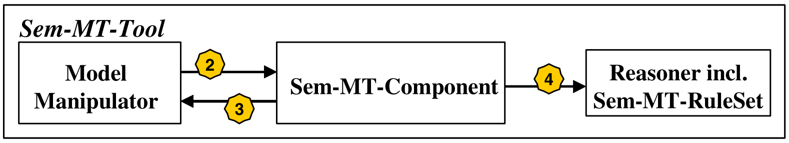


Fig. 6. Sem-MT-Tool component architecture

5.2 Architecture of OntMT

To validate our approach we have implemented a prototype that realizes the crucial parts of ontology-based model transformation. The following section provides more details about the architecture and the implementation. Therefore the architectural figures additionally depict the technologies we used to implement our prototype.

5.2.1 Inference Component

Figure 7 depicts a detailed architectural view on the inference component of ontMT. The inference component consists of a knowledge base and a reasoner. The base graph contains all facts of the knowledge base before the reasoning, i.e. the reference ontology, application ontologies[5], and the ontological groundings. The reasoner is triggered by rules specific to the Sem-MT-Tool, and computes the inference graph on the basis of the base graph. As the result of the reasoning, the knowledge base contains information about all relationships that are important for ontMT. These are the relationships between the application ontologies.

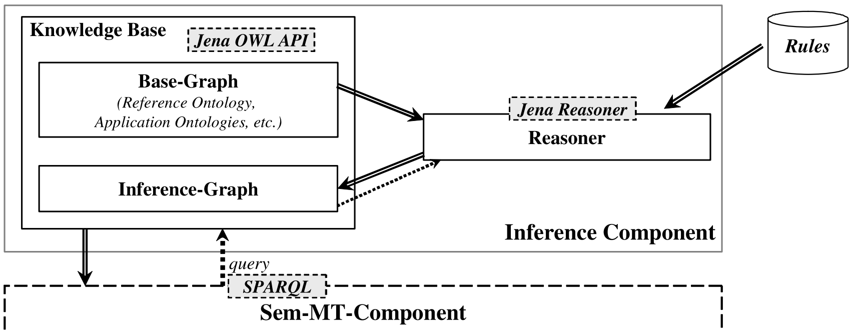


Fig. 7. Inference component

In [9], [29], and [40], *equivalence*, *containment*, and *overlap* are described as the main relationships for mapping ontologies. The inference component identifies (for ontMT) these relationships between the ontology elements. The relationships are also used for the ontological groundings by specifying mappings between the application ontologies and reference ontologies. This is possible, since the model elements are represented in application ontologies via the UML to OWL mapping described in the ODM standard [35, p.201ff].

[5] An application ontology corresponds to a metamodel in the Ontology TS.

- *"Equivalence"* (≡) means that the connected elements represent the same aspect of the real world. An element of an application ontology corresponds to an element in the reference ontology or can be precisely expressed by a composition of elements. Later, we will refer to this relationship by the relationship type <*equal*>.
- *"Containment"* (A,B) states that the element in one ontology represents a more specific aspect of the world than the element in the other ontology. The relationship can be defined in one or the other direction, depending on which concept is more specific. When an element is not sufficiently refined, i.e. it does not match the accuracy level of the ontology, we use the relationship <*general*>. When an element is described at a level of refinement that does not match the level of refinement of the other ontology we use the relationship <*special*>.
- *"Overlap"* (*o*) states that the connected elements represent different aspects of the world, but have an overlap in some respect. This relationship is of the type <*overlap*>.

Implementation

In our current prototype we use the Jena ontology API[6] to create and handle ontologies. The inference is realized through rules deployed to the rule engine included in Jena. Jena also uses this rule engine to provide (partial) RDFS and OWL reasoning[7]. The rule 2 in Listing 2 for example states, that if A overlaps B and B is an intersection of C and D then A overlaps C and D. The inference results are obtained with SPARQL, which queries the knowledge base for the relationships between the application ontologies.

rule 1: $A\ o\ B \wedge B \sqsubseteq C \rightarrow A\ o\ C$
rule 2: $A\ o\ B \wedge B \equiv C \sqcap D \rightarrow A\ o\ C \wedge A\ o\ D$

Listing 2: Sample reasoning rules

The decision to use the Jena framework and its rule based reasoning support for the prototype implementation was mainly based on two decisions. First, it better met our requirements, which were mainly a combination of TBox reasoning, rule support, and good documentation, than other open source projects. Second, the Jena framework provides the possibility to integrate other reasoners like Pellet[8] or future implementations of ontology mapping approaches using local domains like C-OWL [8].

5.2.2 Model Manipulator

The model manipulator provides modification operations on model transformations. It implements a language for model transformation modification that is used by the Sem-MT-Component to trigger the modification of the model transformations via modification programs. The semantics of this model transformation modification language treats model transformations as models. The fact that model transformation languages like QVT are represented through metamodels and modeltransformation programs are

[6] http://jena.sourceforge.net/tutorial/RDF_API/
[7] http://jena.sourceforge.net/inference/
[8] http://pellet.owldl.com/

models allows higher-order transformations, like transformations taking other transformations as input and producing transformations as output [4].

Due to the gap between the concepts of DSLs and metamodels implementing these DSLs, the semantics of the model transformation modification language needs to provide mechanisms to allow the Sem-MT-Component to adapt a modification program to the best possible solution. Hence the semantics is divided into a *modification semantics* and a *checking semantics* (see Figure 8).

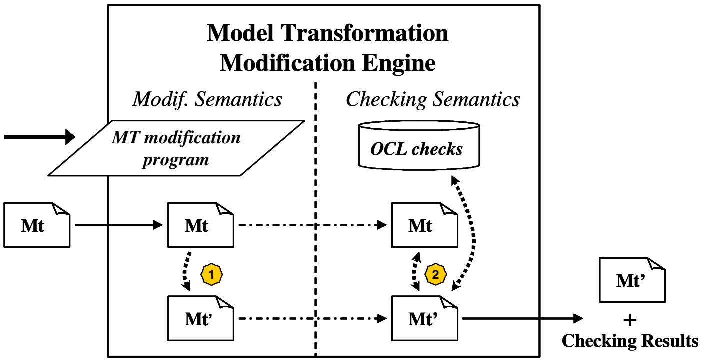


Fig. 8. Semantics of model transformation modification

Modification semantics

The modification semantics defines how the modification of model transformations, which is specified in modification programs, is executed. A simplified picture that helps to work with the model transformation modification language is to imagine the modification program as a substitution. The elements of the modification program's source metamodel are substituted with the elements of the target metamodel. The detailed implementation realizing the semantics is encapsulated in a separate component of the model manipulator. Currently realized substitution operators provide functionality for one-to-one, one-to-many, and removal substitutions of both classes and properties. In the following we give a short outline of the substitution operators' functionality via short examples.

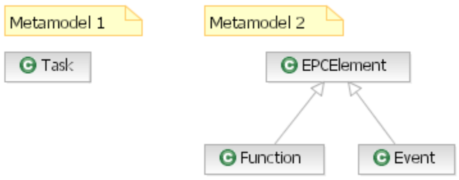


Fig. 9. Two example metamodels

```
relation rule {
  checkonly domain l_mm var1:Task { };
  enforce domain r_mm var1':Task { };
}
```

Listing 3: Example model transformation specification (notation similar to QVT)

- **One-to-one substitution:** If the *Task* in the sample model transformation rule shall be substituted by *EPCElement* in the right-hand model, then the one-to-one substitution for classes has to be applied: [*EPCElement*/*Task*]. The result of applying this one-to-one substitution to the transformation rule of Listing 3 is as follows:

```
relation rule′ {
  checkonly domain l_mm var1:Task { };
  enforce domain r_mm var1′:EPCElement { };
}
```

Listing 4: Model transformation after applying one-to-one substitution

- **One-to-many substitution:** If the *Task* in the sample model transformation rule (see Listing 3) shall be substituted by *Function* and *Event*, then the one-to-many substitution for classes has to be applied: [{*Function*,*Event*}/*Task*]. The result of the one-to-many substitution is not so obvious like the result of the one-to-one substitution since the model transformation rule has to be duplicated. Details about one-to-many substitutions can be found in the case study in Section 6.

```
relation rule′_a {
  checkonly domain l_mm var1:Task { };
  enforce domain r_mm var1′:Function { };
}

relation rule′_b {
  checkonly domain l_mm var1:Task { };
  enforce domain r_mm var1′:Event { };
}
```

Listing 5: Model transformation after applying one-to-many substitution

- **Removal substitution:** A removal substitution is sensibly applied when an element of the source metamodel cannot be substituted by any element of the target metamodel. If e.g. a removal substitution [-/*Task*] is applied to the *Task*, the whole transformation rule of Listing 3 would be removed from the model transformation.

Checking semantics

The checking semantics represents the second part of the model transformation modification language's semantics. It tests the generated model transformations for so-called problems, which can occur by applying the modification semantics. One set of problems affects the consistency of model transformation programs with respect to the model transformation language, i.e. the generated model transformations are not valid and cannot be executed. Another kind of problems is caused, when knowledge encoded into the original model transformation is not preserved or lost. This is the case when modifications and substitutions are applied to relations where they (normally) do not make sense. To detect the second kind of problems the generated model transformation has to be compared with the original model transformation. In general, problems are detected via OCL [34] constraints. Only for a few problems, where additional information about the execution of the modification is needed, we extend this mechanism with information from the modification execution. The following list describes the main problem types and provides some sample OCL constraints:

- *The substitution of property failed (PropertySubstitutionFailed):* This problem occurs, when the model transformation modification program did not specify a substitution for a property that is used in the model transformation. Since in such cases the modification semantics sets the value of the property to *UNDEFINED*, the OCL constraint in Listing 6 checks whether the value of the property (*referredProperty*) is not empty.

```
package qvttemplate
  context PropertyTemplateItem
    inv referredProperty_must_be_set: self.referredProperty→notEmpty()
endpackage
```

Listing 6: OCL constraint: substitution of property failed

- *The substitution of class failed (ClassSubstitutionFailed):* This problem occurs, when the model transformation modification program did not specify a substitution for a class that is used in the model transformation.
- *A property is not part of class (PropertyNotPartOfClass):* The generated model transformation would require a property to be part of a class, what is not the case in the respective metamodel (model types as described in [41] do not match). The OCL invariant *property_part_of_class* in Listing 7 is used to check this fact. The constraint is satisfied, if the OTE's referred class (cp. QVT relational description in Section 2) contains the property or if the OTE does not reference a class. The second case occurs when the class of the OTE could not be substituted.

```
package qvttemplate
  context PropertyTemplateItem
    inv property_part_of_class:
      self.objContainer.referredClass.hasProperty(self.referredProperty)
      or self.objContainer.referredClass→isEmpty()
endpackage

package emof
  context Class
  def: hasProperty(property: Property): Boolean =
    self.ownedAttribute→exists(p: Property | p = property)
    or self.superClass→exists(c: Class | c.hasProperty(property))
endpackage
```

Listing 7: OCL constraint: property part of class

- *A property has not the type of class (PropertyNotTypeOfClass):* This constraint checks, whether a property of the generated model transformation has only types in the model transformation that are compatible with the types of the respective metamodel. The constraint is not satisfied, when the types of the properties inferred from the model transformation and the metamodel do not match (model types as described in [41] do not match).
- *Granularity of initial model transformation is not appropriate (InitialMTGranularityNotAppropriate):* This problem occurs when a one-to-many substitution has to be applied to top level relations. Since the granularity of the initial model transformation does not match the level of refinement of the new metamodel, i.e. the

model transformation relations are too coarse grained, the substitution operators (especially one-to-many) cannot be sensibly applied.[9]

- *Pattern removal (PatternRemoval):* A loss of information occurs when a class or a property could not be substituted and therefore the respective patterns had to be removed from the model transformation.

Architecture and Implementation

The model manipulator component is divided in a front and a back end (see Figure 10). The front end primarily conducts tasks that depend on the source language, while the back end deals with all issues specific to the target language. The metamodels and the bootstrap model transformation are brought into an intermediate representation format by the scanner and the parser. The substitution algorithm performs the substitutions proposed by the Sem-MT-Component. The validator checks whether any performed substitution leads to problems in the new model transformation.

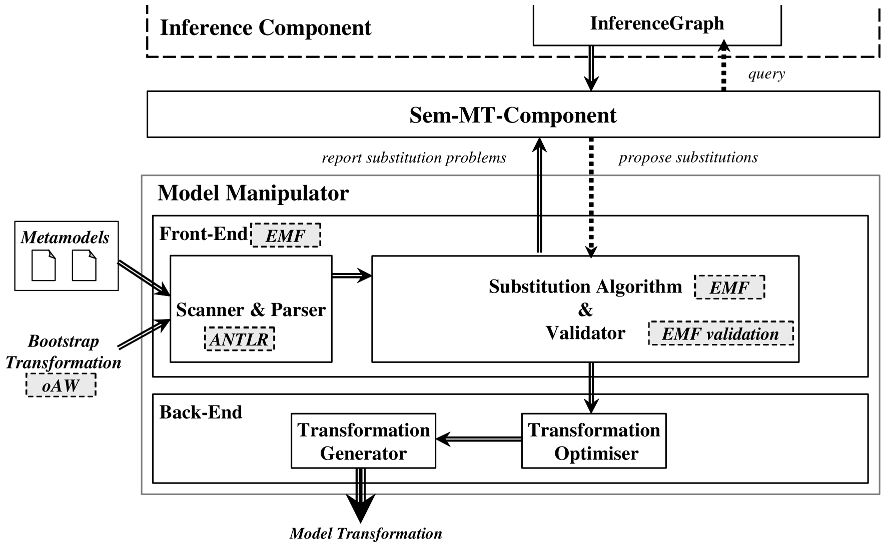


Fig. 10. Model manipulator

Our prototypical implementation of the model manipulator is based on Eclipse. It uses the Eclipse Modeling Framework (EMF). EMF allows the model manipulator to treat the metamodels and the model transformations with a single model manipulation API. This reflective API allows to handle EMF objects regardless of which metamodel they are (EMOF, QVT, OCL, etc.) generically. The metamodels are instantiations of the EMF EMOF implementation and the model transformation models are treated as instantiations of the EMF QVT relational implementation. Since the first final adopted version of the QVT standard [36] contains some inconsitencies we had to make some adjustments which are documented in our implementation.

[9] As we can see in the case study example in Section 6, an automatic bootstrapping algorithm can avoid this problem, if the refinement levels of the metamodels do match.

- Parser: The implementation of the parser makes use of the ANTLR parser generator [38] and parses a QVT relational textual syntax into EMF QVT relational models. It has been made available under GPL via the QVT Parser project[10].
- A prototype of the model manipulator implementation is part of the OntMT project[11]. The substitution algorithm is totally based on the EMF API. The validation component uses the EMF validation framework to check the EMF model of the generated QVT relational transformation with OCL constraints. Since we use the Eclipse Modeling Framework (EMF) [11], the Eclipse Validation Framework [12] is a consequent choice for implementing and performing the OCL checks. The results of EMF validation lend themselves very well to determine the exact position of problems or inconsistencies in a model transformation. OCL constraints, checking whether a model transformation is syntactically correct, can be automatically generated from the QVT metamodel. It is checked whether the model transformation conforms to the grammar that the QVT metamodel was generated from. With further manually implemented OCL constraints the model manipulator checks whether the generated model transformation is valid and can be executed or whether knowledge has been lost through the substitution.
- The bootstrapping generates from a metamodel expressed in MOF a QVT relational model transformation. It is implemented with templates expressed in the oAW[12] expand-language and available via the OntMT project. The bootstrapping is well integrated in the model manipulator, since EMF models can be used for oAW code generation. In fact the same metamodels that are used by the QVT-Parser and the model manipulator are also used for the bootstrapping.

5.2.3 Sem-MT-Component

The Sem-MT-Component implements the core part of the ontMT approach. It provides the main functionality of the Sem-MT-Tool. It makes use of the inference results of the Ontology TS and computes modifications programs for the generation and evolution of model transformations in the MDA TS. Listing 8 illustrates the algorithm implemented by the Sem-MT-Component to generate the new model transformations.

The Sem-MT-Component takes as input an initial model transformation, the metamodel which has to be substituted in the model transformation, and the new metamodels. In a first step (1) it requests the model manipulator to compute a set of all classes and properties that have to be substituted in the initial model transformation. Second, it invokes the inference component to obtain possible substitutions for all classes and properties to substitute (2). Next the computation of a substitution proposal begins. A substitution proposal contains the model transformation modification program, the problems that occur by applying the substitutions to a model transformation, and a rating of the performed substitutions. After a substitution proposal is calculated by the Sem-MT-Component (3), the model manipulator performs the substitution (4) and validates (5) the generated model transformation. Then the substitution proposal is rated by the Sem-MT-Component (6). The Sem-MT-Component tries to compute alternative substitution proposals until their application does not lead

[10] http://sourceforge.net/projects/qvtparser/
[11] http://sourceforge.net/projects/ontmt/
[12] http://www.eclipse.org/gmt/oaw/

to any problems, or no further substitution proposals can be found. Finally the Sem-MT-Component chooses the substitution proposal with the best rating from all computed substitution proposals (7) and the new model transformation is generated on the basis of this substitution proposal (8).

```
input:   initialMT: the initial model transformation
         subMM: metamodel to substitute in the model transformation
         newMM: new metamodel
output:  newMT: new generated model transformation

begin
    let sCS = set of concepts to substitute in the model transformation
    let sCRS = set of tuples with concepts to substitute in the model
               transformation and their possible substitutions
    let subProp = current substitution proposal
    let setSubProp = set of substitution proposals

(1) sCS := identifyConceptsToSubstitute(initialMT, subMM, newMM);

    foreach c ∈ sCS
(2)   sCRS := sCRS ∪ {(c, findPossibleSubstitutionsForConcept(c))};
    end foreach

    do
(3)   subProp := calculateSubstitutionProposal(cCRS);

(4)   tempMT := performSubstitution(subProp);

(5)   subProp := validateSubstitution(tempMT, subProp);

(6)   subProp := rateSubstitution (subProp);
    until subProp.problems == ∅ OR NoOtherSubProposalsPossible end do

(7)   subProp := chooseSubPropWithBestRating(setSubProp);
(8)   newMT := performSubstitution(subProp);

    return newMT
end
```

Listing 8: Algorithm to compute new model transformation (Sem-MT-Component)

Rating substitutions proposals

The choice of the substitution proposal, which is used to generate the new model transformation, is based on the ratings of the substitution proposals. A rating of a substitution proposal is a measure of the generated model transformation's quality. The rating is based on factors that are measured for each substitution proposal:

- **Problems occurring in the substitution proposal**: This measure counts the problems that were detected by the validator in the generated model transformation. The measure distinguishes between the different kinds of problems and assigns different weights to the various problem types according to the severity.
- **Number of concepts that could be substituted**: This measure counts how many class and properties could be substituted. From this measure can be derived how many concepts could not be substituted.

- **Relationships used for substitution**: This measure counts and rates the relationships that are used in the modification program of the substitution proposal. In general a substitution derived from an <*equal*> relationship gets a better rating than substitutions derived from other relationships.
- **Position of the problems and used relationships in the model transformation**: This is an optional factor that can influence the three other ratings describe above. The assumption for this factor is, that some relations of the model transformation are more important to the overall result than others.

Concrete ratings depend always on the purpose ontMT is used for. For the different application scenarios separate metrics are defined. A metric, which was developed for automated mapping generation, will put more emphasis on an executable model transformation than on the relationships used for substitution. If ontMT is used to support developers in adjusting their model transformations, ontMT will only make suggestions to the developer. Hence, the metric puts more emphasis on exact substitutions of metamodel elements than on the execution of the new model transformation.

Implementation

The OntMT project currently provides a simple implementation of the correlation algorithm, which is described in Listing 8. However, an automated synchronisation of the modelling and the reasoning world (see ⓪ in Figure 5) is not yet fully integrated. We are developing a prototype that synchronizes EMF and Jena OWL models and allows to answer SPARQL like queries on EMF models with reasoning support. The synchronization mechanism makes use of the UML to OWL mapping described in the ODM standard [35, p.201ff]. However, we plan to replace our prototype with an implementation of the Eclipse EODM project[13]. This projects aims to provide inference capabilities for OWL models implemented in EMF and model transformations of RDF/OWL to other modelling languages such as UML.

6 Case Study about Automated Mapping Generation

This section provides further insights about how the Sem-MT-Tool works. It illustrates the automated mapping generation application scenario of ontMT that has been introduced in Section 4.1. A mapping between two metamodels (Figure 11 and 12) for process modelling is generated. The first metamodel *Process* is an excerpt of a metamodel for process orchestration in a service-oriented environment. The second metamodel *EPC* is also for process modelling.[14]

The reference ontology in this example (see Figure 13) is an excerpt of the Web Ontology Language for Services (OWL-S). For the ontological grounding we use a notation similar to SMAIL [29] (Semantic Mediation and Application Interoperability Language). '=:' stands for a lossless annotation, where the annotation fully captures the intended meaning. '>:' denotes an overspecification, where the level of refinement

[13] http://www.eclipse.org/modeling/mdt/?project=eodm

[14] The QVT model transformations of this case study together with the EMOF metamodels, and a few test models have been executed with ModelMorf [31] beta version 2.

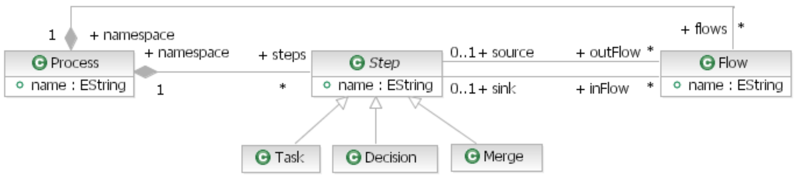


Fig. 11. Metamodel *Process*

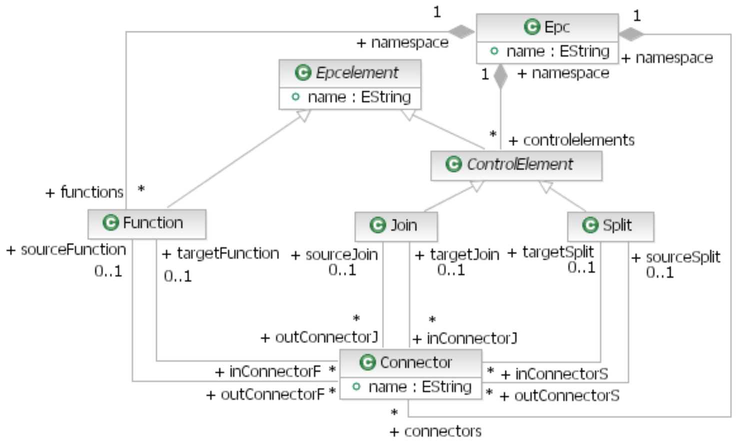


Fig. 12. Metamodel *EPC*

of the annotated element is greater than the level of refinement of the concepts in the reference ontology.

- In a first step, the bootstrapping generates the initial model transformation by traversing metamodel *MM1* via its composition (see Listing 9). The bootstrapping works as follows:
- For each class in the *Process* metamodel one *top relation* mapping rule is generated for the bootstrapping model transformation (*Process*, *Flow*). If there occurs inheritance, mapping rules for the concrete leaf classes (*Task*, *Decision*, *Merge*) are generated instead of mapping the abstract superclass (*Step*). This enhances the granularity of the model transformation specification. The mandatory properties are specified as part of the top relation (in the example the *name* property). Optional properties would be outsourced to separate relations, which are used to further constrain the *top relations* via *where*-statements.
- Composition associations are realized in the initial model transformation as properties of the contained elements (*namespace*). These properties constrain the *top relations* via *when*-statements (e.g. in the *FlowToFlow* relation).
- Other associations are realized via separate *relations* in the initial model transformation (*StepToStep_out*, *StepToStep_in*).

Table 1 and 2. Ontological Grounding of MM1 and MM2

Process [name,steps,flows]	=:	CompositeProcess [name,composedOf,composedOf]
Step [name,outFlow,inFlow,namespace]	=:	ProcessComponent [name,connected,inverseOf(followed), inverseOf(composedOf)]
Task [name,outflow,inFlow,namespace]	=:	AtomicProcess [name,connected,inverseOf(followed), inverseOf(composedOf)]
Flow [sink,source]	=:	FollowedBy [followed,inverseOf(connected)]
…	…	…
EPC [name,connectors]	=:	CompositeProcess [name,composedOf]
EPC [name,functions,connectors]	>:	CompositeProcess [name,composedOf,composedOf]
EPC [name,controlelements,connectors]	>:	CompositeProcess [name,composedOf,composedOf]
EPCElement [name]	=:	ProcessComponent [name]
Function [name,outConnectorF,inConnectorF, namespace]	=:	AtomicProcess [name,connected,inverseOf(followed), inverseOf(composedOf)]
Connector [sinkFunction,soureFunction, sinkJoin,sourceJoin, sinkSplit,souceSplit]	>:	FollowedBy [followed,inverseOf(connected), followed,inverseOf(connected), followed,inverseOf(connected)]
…	…	…

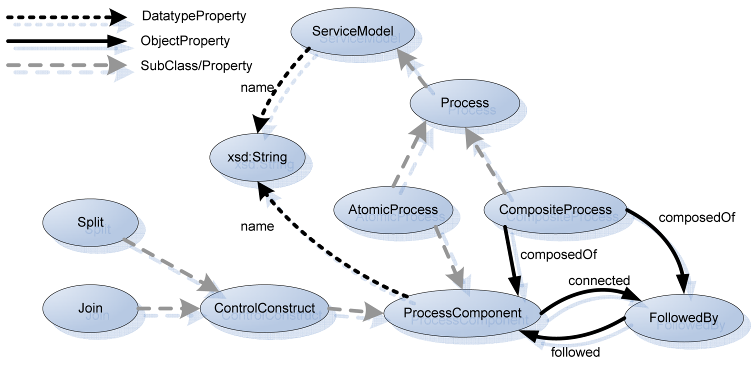


Fig. 13. Reference Ontology

- The model of the initial model transformation serves as input for the model manipulator. The first task of the model manipulator is to determine the classes and properties of the right-hand metamodel, which have to be substituted in the relations of the input model transformation.

```
transformation ProcessToProcess(prc_1:processMM_1; prc_2:processMM_2) {

key processMM_2::Process {name};
key processMM_2::Task {name, namespace};
key processMM_2::Decision {name, namespace};
key processMM_2::Merge {name, namespace};
key processMM_2::Flow {name, namespace};

top relation ProcessToProcess {
  pn: String;
  checkonly domain prc_1 p_1:Process { name=pn };
  enforce domain prc_2 p_2:Process { name=pn };
}

top relation TaskToTask {
  tn: String;
  checkonly domain prc_1 t_1:Task { namespace=p_1:Process {}, name=tn };
  enforce domain prc_2 t_2:Task { namespace=p_2:Process {}, name=tn };
  when { ProcessToProcess(p_1, p_2); }
  where {  StepToStep_out(t_1, t_2); StepToStep_in(t_1, t_2); }
}

top relation DecisionToDecision {
  dn: String;
  checkonly domain prc_1 d_1:Decision { namespace=p_1:Process {},
                                        name=dn };
  enforce domain prc_2 d_2:Decision {namespace=p_2:Process {}, name=dn};
  when { ProcessToProcess(p_1, p_2); }
  where {  StepToStep_out(d_1, d_2); StepToStep_in(d_1, d_2); }
}

top relation MergeToMerge {
  mn: String;
  checkonly domain prc_1 m_1:Merge { namespace=p_1:Process {}, name=mn};
  enforce domain prc_2 m_2:Merge { namespace=p_2:Process {}, name=mn};
  when { ProcessToProcess(p_1, p_2); }
  where {  StepToStep_out(m_1, m_2); StepToStep_in(m_1, m_2); }
}

relation StepToStep_out {
  fn: String;
  checkonly domain prc_1 s_1:Step { outFlow=out_1:Flow { name=fn } };
  enforce domain prc_2 s_2:Step { outFlow=out_2:Flow { name=fn } };
}

relation StepToStep_in {
  fn: String;
  checkonly domain prc_1 s_1:Step { inFlow=in_1:Flow { name=fn } };
  enforce domain prc_2 s_2:Step { inFlow=in_2:Flow { name=fn } };
}

top relation FlowToFlow {
  fn: String;
  checkonly domain prc_1 f_1:Flow { namespace=p_1:Process {}, name=fn };
  enforce domain prc_2 f_2:Flow { namespace=p_2:Process {}, name=fn };
  when { ProcessToProcess(p_1, p_2); }
}
}
```

Listing 9: The initial model transformation in QVT relational syntax

- For each concept to substitute the Sem-MT-Component queries the inference component for relationships. The inference component searches the knowledge base for triples like *<Step:Process>* *<?>* *<?:EPC>*[15]. The result for *<Step:Process>* is:

  <Step:Process > *<equal>* *<EPCElement:EPC>*
  <Step:Process > *<general>* *<Function:EPC>*
  <Step:Process > *<general>* *<ControlElement:EPC>*
  <Step:Process > *<general>* *<Split:EPC>*
  <Step:Process > *<general>* *<Join:EPC>*

With this input (metamodels, reference ontology, ontological groundings and initial transformation and possible substitutions) the computation of the substitution of N_A' with N_B can start:

- In the first substitution proposal, the Sem-MT-Component considers only facts with the predicate *<equal>*, in order to find the best possible substitution. Since for the ObjectProperty *<outFlow(Flow):Process>* and *<inFlow(Flow):Process>* no substitution in the context of *<Epcelement:EPC>* is possible, this ObjectProperty is omitted in the substitution, in the hope that this does not affect the model transformation. Thus the substitutions of the first substitution proposal are:

Table 3. Substitution proposal *SP1*

Process[name]	→	EPC[name]
Step[outFlow,inFlow]	→	Epcelement[---,---]
Task[name,namespace]	→	Function[name,namespace]
Decision[name,namespace]	→	Join[name,namespace]
Merge[name,namespace]	→	Split[name,namespace]
Flow[name,namespace]	→	Connector[name,namespace]

The model manipulator generates a new model transformation on the basis of the first substitution proposal.

```
transformation ProcessToEpc_v1(prc_1:processMM_1; epc_1:epcMM_1) {

...

top relation TaskToFunction {
  tn: String;
  checkonly domain prc_1 t_1:Task { namespace=p_1:Process {}, name=tn };
  enforce domain epc_1 f_1:Function { namespace=e_1:Epc {}, name=tn };
  when {
    ProcessToEpc(p_1, e_1);
  }
  where {
    StepToEpcelement_out(t_1, f_1);
    StepToEpcelement_in(t_1, f_1);
  }
}

...
```

[15] The facts of in the knowledge base are of the form *<subject>* *<predicate>* *<object>*.

```
relation StepToEpcelement_out {
  fn: String;
  checkonly domain prc_1 s_1:Step { outFlow=out_1:Flow { name=fn } };
  enforce domain epc_1 e_1:Epcelement { };
}

relation StepToEpcelement_in {
  fn: String;
  checkonly domain prc_1 s_1:Step { inFlow=in_1:Flow { name=fn } };
  enforce domain epc_1 e_1:Epcelement { };
}

...

}
```

Listing 10: Model transformation generated from substitution proposal *SP1*

- The model transformation is validated by the model manipulator, which detects two *PropertySubstitutionFailed* problems for the *inFlow* and *outFlow* properties of the class *Epcelement*.
- Thereon the Sem-MT-Component searches for an alternate substitution, which also considering facts with predicates other than <*equal*>. For a substitution decision it applies a hierarchy, where the predicate <*equal*> is better than <*special*> and <*special*> is better than <*general*>. The facts provided by the inference component are:

  <*outFlow:Process*> <*general*> <*outConnectorF:EPC*>
  <*inFlow:Process*> <*general*> <*inConnectorF:EPC*>
  <*outFlow:Process*> <*general*> <*outConnectorJ:EPC*>
  <*inFlow:Process*> <*general*> <*inConnectorJ:EPC*>
  <*outFlow:Process*> <*general*> <*outConnectorS:EPC*>
  <*inFlow:Process*> <*general*> <*inConnectorS:EPC*>

  Based on its history of previously proposed substitutions[16] and the fact, that no facts with the predicates <*equals*> or <*special*> exist, the Sem-MT-Component computes a new substitution proposal *SP2*. This substitution proposal proposes to substitute the *outFlow* property with the three different *outConnector* properties:

Table 4. Substitution proposal *SP2*

Process[name]	→	EPC[name]
Step[outFlow,inFlow]	→	Epcelement [outConnectorF&outConnectorJ&outConnectorS, inConnectorF&inConnectorJ&inConnectorS]
Task[name,namespace]	→	Function[name,namespace]
Decision[name,namespace]	→	Join[name,namespace]
Merge[name,namespace]	→	Split[name,namespace]
Flow[name,namespace]	→	Connector[name,namespace]

[16] The Sem-MT-Component has a history of its previous substitution proposals, so that it will not make the same proposal a second time and the search for substitutions terminates.

- The model manipulator generates a model transformation from the new substitution proposal:

```
transformation ProcessToEpc_v1(prc_1:processMM_1; epc_1:epcMM_1) {

...

relation StepToEpcelement_out_F {
  fn: String;
  checkonly domain prc_1 s_1:Step { outFlow=out_1:Flow { name=fn } };
  enforce domain epc_1 e_1:Epcelement {
                            outConnectorF=out_2:Connector { name=fn } };
}

relation StepToEpcelement_in_F {
  fn: String;
  checkonly domain prc_1 s_1:Step { inFlow=in_1:Flow { name=fn } };
  enforce domain epc_1 e_1:Epcelement {
                            inConnectorF=in_2:Connector { name=fn } };
}

relation StepToEpcelement_out_S {
  fn: String;
  checkonly domain prc_1 s_1:Step { outFlow=out_1:Flow { name=fn } };
  enforce domain epc_1 e_1:Epcelement {
                            outConnectorS=out_2:Connector { name=fn } };
}
...
}
```

Listing 11: Model transformation generated from substitution proposal *SP2*

The model transformation is validated by the model manipulator, which detects six *PropertyNotPartOfClass* problems, since e.g. the property *outConnector* is part of the class *Function* and not part of the class *Epcelement*.

- Thus the Sem-MT-Component calculates an alternative substitution proposal *SP3*, where a *Step* is substituted by *Function*, *Join*, and *Split*:

Table 5. Substitution proposal *SP3*

Process[name]	→	EPC[name]
Step[outFlow,inFlow]	→	Function[outConnectorF,outConnectorJ] Split[outConnectorS,inConnectorF] Join[inConnectorJ,inConnectorS]
Task[name,namespace]	→	Function[name,namespace]
Decision[name,namespace]	→	Join[name,namespace]
Merge[name,namespace]	→	Split[name,namespace]
Flow[name,namespace]	→	Connector[name,namespace]

- The model transformation generated on the basis of *SP3* is as follows:

```
transformation ProcessToEpc_Trans(prc_1:processMM_1; epc_1:epcMM_1) {

key epcMM_1::EPC {name};
key epcMM_1::Function {name, namespace};
key epcMM_1::Split {name, namespace};
key epcMM_1::Join {name, namespace};
key epcMM_1::Connector {name, namespace};

top relation ProcessToEpc {
  pn: String;
  checkonly domain prc_1 p_1:Process { name=pn };
  enforce domain epc_1 e_1:Epc { name=pn };
}

top relation TaskToFunction {
  tn: String;
  checkonly domain prc_1 t_1:Task { namespace=p_1:Process {}, name=tn };
  enforce domain epc_1 f_1:Function { namespace=e_1:Epc {}, name=tn };
  when { ProcessToEpc(p_1, e_1); }
  where {
    StepToEpcelement_out_F(t_1, f_1);
    StepToEpcelement_in_F(t_1, f_1);
  }
}

...

relation StepToEpcelement_out_F {
  fn: String;
  checkonly domain prc_1 s_1:Step { outFlow=out_1:Flow { name=fn } };
  enforce domain epc_1 e_1:Function {
                          outConnectorF=out_2:Connector { name=fn } };
  when { FlowToConnector(out_1, out_2); }
}

relation StepToEpcelement_in_F {
  fn: String;
  checkonly domain prc_1 s_1:Step { inFlow=in_1:Flow { name=fn } };
  enforce domain epc_1 e_1:Function {
                          inConnectorF=in_2:Connector { name=fn } };
  when { FlowToConnector(in_1, in_2); }
}

...

top relation FlowToConnector {
  fn: String;
  checkonly domain prc_1 f_1:Flow { namespace=p_1:Process {}, name=fn };
  enforce domain epc_1 c_1:Connector { namespace=e_1:Epc {}, name=fn };
  when { ProcessToEpc(p_1, e_1); }
}
}
```

Listing 12: Model transformation generated from substitution proposal *SP3*

- The validator of the model manipulator comes to the result, that this substitution proposal leads to a new model transformation in which none of the problems mentioned above occur.
- Thus the Sem-MT-Component stops computing new substitution proposals and compares the ratings of the substitution proposals already tested:

Table 6. Rating of the substitution proposals

Substitution Proposal	***Problems occurred***	***Substituted Concepts***	***Used Relationships***
SP1	2x *PropertySubstitutionFailed*	5x class 8x property	13x <*equal*>
SP2	6x *PropertyNotPartOfClass*	5x class 10x property	13x <*equal*> 6x <*general*>
SP3	---	5x class 10x property	12x <*equal*> 9x <*general*>

- The Sem-MT-Component decides to use the substitution proposal *SP3* to generate the new model transformation. This is based on the consideration that a model transformation generated from *SP2* will not be able to be executed due to typing problems. Furthermore *SP3* can substitute more concepts of the original model transformation than *SP1*. The model transformation between metamodel N_A (*Process*) to N_B (*EPC*) generated by ontMT is listed in Listing 12.

7 Assessment of OntMT

This section discusses the ontMT approach with respect to its partical application, its limits, and possible weaknesses.

7.1 Application Areas

Ontology-based model transformation fosters the exchange of models and the evolution of model transformations. Model exchange scenarios are build on the generation of new model transformations, while model transformation evolution scenarios aim at reusing model transformations. As introduced in Section 2, one can distinguish between horizontal and vertical model transformations. Horizontal model transformations are mappings between models at a certain abstraction level, where no information is lost and no additional information is added. Vertical model transformations are refinements that add additional information to the generated model about e.g. architecture or platform. Thus, the target model of a refinement is more detailed than the source model.

	Mapping	***Refinement***
Model Exchange	autom.gen. (+ man.)	n/a
MT Evolution	autom.gen. / autom.mod.	autom.mod.

Fig. 14. Application of ontMT to model exchange and model transformation evolution

Figure 14 categorizes the support that ontMT can provide to the described application scenarios and the different types of model transformations.

- To exchange models between different DSLs, metamodels and modelling styles, ontMT is able to automatically generate mappings. However, the level of automation depends on how different the DSLs and their modelling approaches are. It may be necessary to provide additional mapping information through an initial model transformation, which cannot be inferred from the ontologies.
- OntMT supports the evolution and reuse of existing mappings. The new model transformation can be either generated from scratch or obtained through adjusting the existing mapping. The more individual features, which are different to the core structure of the metamodels, are encoded in existing mappings, the more preferable it is to adjust existing mappings. The generation of new mappings is better, if the new metamodel provides extensions to the old one or a new modelling style specifies a fundamentally different composition of modelling elements.
- For the evolution and reuse of refinements, ontMT provides the possibility of automated modification and adjustment of existing model transformations. Refinement model transformations cannot be generated without human interaction, since they contain individual knowledge about software architecture or the platform, e.g. patterns like broker, model-view-controller, etc..

7.2 Evaluation

The ontMT approach adjusts initial model transformations in order to generate or maintain model transformations. Since mapping knowledge is captured in bindings of the metamodels to the reference ontology, one could favour an approach that derives model transformation rules directly from these bindings. This may very well work for model exchange scenarios. However, in model transformation evolution scenarios the model transformation itself would have to be encoded in the bindings. In our opinion, it is better to encode this transformation knowledge in an initial model transformation, i.e. the model transformation to reuse.

The level of automation that ontMT can provide highly depends on how different metamodels, DSLs, and modelling approaches are. If for example two DSLs totally differ in their modelling approaches, their metamodel bindings will be two mostly unconnected sets of the reference ontology. OntMT does not add real transformation knowledge that changes the semantics of model transformation. It depends on the results that are inferable via the ontologies that are used to adjust the syntax of model transformations.

We also made scalability considerations and tests for ontMT in terms of memory requirements, runtime, and size of model transformations that can be processed. This was done for the three components of ontMT (see Section 5.1) separately. Memory requirements and runtime of the model manipulator rise linear to the number of rules a model transformation contains. We tested this with model transformations that contain up to 200 rules. In ontMT reasoning has only to be performed once at runtime. Its memory requirements and runtime depends on the size and the complexity of the ontologies. Since the application scenarios of ontMT do not have hard real-time constraints, we do not see problems in practice concerning memory requirements, runtime, and size of model transformations for the model manipulator and the inference component.

However, the Sem-MT-Component can be seen as the ‘bottleneck’ of the ontMT approach. This component has to combine the reasoning results to a sensible input for the model manipulator. For this combination the size of the solution space grows exponentially with the relationships that are inferred for each concept. The size of the solution space is c^n, where c is the number of concepts in a metamodel and n is the number of relationships inferred for each concept. We try to solve this problem by restricting the solution space. As exemplified in the case study we apply heuristics that first guess an ‘ideal’ solution and the try to solve problems locally in the solution space, i.e. where the problems in the generated model transformation were detected.

7.3 Discussion

The ontMT approach assumes the existence of an appropriate reference ontology. However, developing or agreeing on a reference ontology is a non-trivial task. For example there may exist different versions of (reference) ontologies, what would transfer the problem of heterogeneous models from the MDA TS to the ontology TS. In those cases techniques for matching and merging ontologies, like linguistic, schema-based, or probabilistic approaches, combined with human intervention have to be applied to obtain a suitable reference ontology. Ontology alignment, matching, and mapping approaches can be also very useful to discover and define bindings from the metamodels to the reference ontology. [28] describes an approach and a conceptual framework for mapping distributed ontologies. It can provide the basis for an interactive and incremental mapping process that is needed for developing the bindings in ontMT. In such a process the SKOS mapping vocabulary [42] could be used to specify mappings between concepts from different ontologies. For this vocabulary a search algorithm has been developed [17] that can discover potential candidates for substitutions in ontMT.

To provide ontological groundings and to find reference ontologies may require investing a lot of effort. Depending on the concrete application scenario, this effort may not be justifiable with the generation and evolution of model transformations. Developing or adjusting model transformations by hand may be cheaper. Hence, the goal is to reuse reference ontologies and ontological groundings with other applications that are part of a semantic-enabled modelling and development suite (see Section 5.1).

A totally automated solution may also have to cope with acceptance problems of software engineers. Software engineers will probably not be willing to give up overall control of model transformation to an automated tool, which makes its choice based on metrics and heuristics. Hence, the majority of application scenarios will be of such a form, that the Sem-MT-Tool makes suggestions with a change and problem history to the software engineer. The engineer has the possibility to accept, correct, or reject the suggestions.

8 Related Work

In [41], the authors introduce model typing as extension of object-oriented typing and propose an algorithm for checking the conformance of model types. It is presented, how model typing permits more flexible reuse of model transformations across various metamodels while preserving type safety. This approach improves the reuse of model transformations whenever small changes to metamodels occur, like altering the cardinality of an association. In case of major change in models' representation formats or modelling guidelines model transformations still have to be modified manually. Furthermore automatic mapping generation is not provided.

The ATLAS Model Weaver (AMW) tool implements the model weaving approach introduced in [5]. It enables the representation of correspondences between models in so-called weaving models, from which model transformations can be generated. Model weaving aims to improve efficiency in the creation and maintenance of model transformations. Nevertheless, creating weaving links is not automatic and weaving models have to be adjusted whenever changes DSLs, metamodels or model guidelines of source and target models are made.

The work described in [13] presents an approach to semi-automate the development of transformations via weaving models of the model weaving approach. It describes an iterative and incremental procedure of weaving link generation, similarity calculation, and weaving link selection.

The Model-based Semantic Mapping Framework (Semaphore) [27] follows a similar approach as the ATLAS Model Weaver. By aiming to support mappings between domain models, it supports (graphically) specification of mappings between DSLs. These specifications are saved in mapping model, which for example can be used to generate code to transform the models. Like in AMW mappings have to be specified manually and have to be adjusted when ever changes to the model representations or modelling guidelines occur.

The ATHENA Semantic Suite provides tools for improving interoperability between organisations. In this approach e.g. XML Schemas can be annotated by a reference ontology and reasoning rules can be specified, so that reasoner can convert XML documents. The reasoner can be used as mediator transforming messages at runtime. This approach could be extended to modelling by transforming XML serialisations of models. The problem is that there would be no traceability of transformation executions between models. However, this is a key feature for MDD [16] and cross-organisational modelling. Since this is provided by model transformation languages it is also supported by our approach.

The ModelCVS project [23] provides a framework for semi-automatic generation of transformation programs. By explicitly representing the concepts modelling language in ontologies, the goal is to derive *bridgings* (transformations) between the original metamodels from the mapping between the ontologies. The approach focuses on mappings in order to foster tool interoperability [24]. For reusing existing (refinement) model transformations, knowledge about the transformation would have to be captured in the ontologies or ontology mappings.

9 Summary and Outlook

The approach of ontology-based model transformation provides technology that fosters interoperability in model exchange and the evolution of model transformations. It integrates ontologies in MDSD and makes use of the reasoning capabilities of the

Ontology TS. By automated generation of mappings it offers new possibilities for the integration of domain specific languages and ‘legacy’ models in a plug&play manner. This makes it easier for new organisations to join collaborations. OntMT also supports organisations evolving their modelling techniques like using new and more advanced versions of modelling languages. It yields more efficient reuse of model transformations and the knowledge that is captured in those transformations. Nevertheless, ontMT uses additional information, which has to be provided by the people developing metamodels and domain specific languages.

Future work to extend and improve ontMT is manifold. The model manipulator will be extended with more expressive substitution mechanisms, which for example allow more complex pattern matching. The metrics and heuristics of the Sem-MT-Component have to be tested and improved via more case studies and application scenarios from various domains. Other reasoners and implementations of ontology mapping approaches have to be integrated in the inference component. Finally, the ontMT approach has to provide or adopt methodologies, which support discovery and development of reference ontologies and bindings between metamodels and ontologies.

References

1. Bauer, B., Müller, J.P., Roser, S.: A Decentralized Broker Architecture for Collaborative Business Process Modelling and Enactment. In: 2nd International Conference on Interoperability of Enterprise Software and Applications (2006)
2. Bauer, B., Roser, S.: Semantic-enabled Software Engineering and Development. In: 1st International Workshop on Applications of Semantic Technologies. LNI (2006)
3. Benguria, G., Larrucea, X., Elvesæter, B., Neple, T., Beardsmore, A., Friess, M.: A platform-independent model for service-oriented architectures. In: 2nd International Conference on Interoperability of Enterprise Software and Applications (2006)
4. Bézivin, J.: On the unification power of models. Software and System Modeling 4(2), 171–188 (2005)
5. Bézivin, J., Jouault, F., Valduriez, P.: First Experiments with a ModelWeaver. In: OOPSLA & GPCE Workshop (2004)
6. Bézivin, J., Devedzic, V., Djuric, D., Favreau, J.M., Gasevic, D., Jouault, F.: An M3-Neutral Infrastructure for bridging model engineering and ontology engineering. In: 1st International Conference on Interoperability of Enterprise Software and Applications (2005)
7. Borgo, S., et al.: OntologyRoadMap. WonderWeb Deliverable D15 (2002), `http://wonderweb.semanticweb.org`
8. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: C-OWL: Contextualizing Ontologies. In: Fensel, D., Sycara, K.P., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 164–179. Springer, Heidelberg (2003)
9. Brockmanns, S., Haase, P.: A Metamodel and UML Profile for Networked Ontologies. In: 2nd Workshop on Semantic Web Enabled Software Engineering (2006)
10. Elvesæter, B., Hahn, A., Berre, A.-J., Neple, T.: Towards an Interoperability Framework for Model-Driven Development of Software Systems. In: 1st International Conference on Interoperability of Enterprise Software and Applications (2005)
11. Eclipse Modeling Framework (EMF), `http://www.eclipse.org/modeling/emf/`

12. Eclipse Validation Framework (VF), `http://www.eclipse.org/emft/projects/validation/`
13. Fabro, M.D.D., Valduriez, P.: Semi-automatic model integration using matching transformations and weaving models. In: 22nd ACM Symposium on Applied Computing, Model Transformation Track (2007)
14. Favre, J.M.: Foundations of Meta-Pyramids: Languages vs. Metamodels, Episode II: Story of Thotus the Baboon (2004)
15. Frankel, D.S.: Model Driven Architecture – Applying MDA™ to Enterprise Computing. Wiley, Chichester (2003)
16. Gardner, T., Griffin, C., Koehler, J., Hauser, R.: A review of OMG MOF 2.0 QVT Submissions and Recommendations towards the final Standard. In: MetaModelling for MDA Workshop (2003)
17. Gasevic, D., Hatala, M.: Searching Web Resources Using Ontology Mappings. In: K-CAP 2005 Workshop on Integrating Ontologies (2005)
18. Greiner, U., et al.: Designing and implementing cross-organizational business processes - Description and Application of a Modeling Framework. In: 2nd International Conference on Interoperability of Enterprise Software and Applications (2006)
19. Guarino, N.: Understanding, Building, and Using Ontologies. International Journal of Human-Computer Studies 46(2-3), 293–310 (1997)
20. Harel, D., Rumpe, B.: Meaningful Modeling: What's the Semantics of "Semantics"? IEEE Computer 37(10), 64–72 (2004)
21. IDEAS: A Gap Analysis (2003), `http://www.ideas-roapmap.net`
22. IDEAS: The Vision for 2010 (2003), `http://www.ideas-roapmap.net`
23. Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., Schwinger, W., Wimmer, M.: Lifting Metamodels to Ontologies: A Step to the Semantic Integration of Modeling Languages. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006. LNCS, vol. 4199, pp. 528–542. Springer, Heidelberg (2006)
24. Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., Schwinger, W., Wimmer, M.: On Model and Ontologies – A Layered Approach for Model-based Tool Integration. In: Proceedings of Modellierung (2006)
25. Klein, R., Kupsch, F., Scheer, A.-W.: Modellierung inter-organisationaler Prozesse mit Ereignisgesteuerten Prozessketten. In: Scheer, A.-W (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Nr. 178 (2004)
26. Kurtev, I., Bézivin, J., Aksit, M.: Technological Spaces: An Initial Appraisal. In: Int. Federated Conference (DOA, ODBASE, CoopIS) (2002)
27. Limyr, A., Neple, T., Berre, A.-J., Elvesæter, B.: Semaphore – A Model-Based Semantic Mapping Framework. In: Workshop on Enterprise and Networked Enterprises Interoperability at BPM 2006 (2006)
28. Maedche, A., Motik, B., Silva, N., Volz, R.: MAFRA - A Mapping Framework for Distributed Ontologies. In: 13th European Conference on Knowledge Engineering and Knowledge Management (2002)
29. Missikoff, M., et al.: A Controlled Language for Semantic Annotation and Interoperability in e-Business Applications. In: Workshop on Semantic Integration (2003)
30. MDDi - Eclipse project, `http://www.eclipse.org/mddi/`
31. ModelMorf, `http://www.tcs-trddc.com/ModelMorf/index.htm`
32. MODELWARE project, `http://www.modelware-ist.org/`
33. Oberle, D.: Semantic Management of Middleware. Springer, Heidelberg (2005)
34. OMG: Object Constraint Language, Version 2.0, formal/2006-05-01
35. OMG: Ontology Definition Metamodel, ad/2006-05-01

36. OMG: MOF QVT Final Adopted Specification, ptc/05-11-01
37. OMG: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification - Final Adopted Specification, Final Adopted Specification, ptc/07-07-07
38. Parr, T.: ANTLR 3.0 (2007), `http://antlr.org/`
39. Roser, S., Bauer, B.: An Approach to Automatically Generated Model Transformation Using Ontology Engineering Space. In: 2nd Workshop on Semantic Web Enabled Software Engineering (2006)
40. Serafini, L., Stuckenschmidt, H., Wache, H.: A formal investigation of mapping language for terminological knowledge (2005)
41. Steel, J., Jézéquel, J.-M.: Model Typing for Improving Reuse in Model-Driven Engineering. In: Briand, L.C., Williams, C. (eds.) MoDELS 2005. LNCS, vol. 3713, pp. 84–96. Springer, Heidelberg (2005)
42. W3C: SKOS Mapping Vocabulary Specification, `http://www.w3.org/2004/02/skos/mapping/spec/`

The Harmony Integration Workbench

Peter Mork, Len Seligman, Arnon Rosenthal, Joel Korb, and Chris Wolf

The MITRE Corporation
McLean, VA, USA
{pmork,seligman,arnie,jkorb,cwolf}@mitre.org

Abstract. A key aspect of any data integration endeavor is determining the relationships between the source schemata and the target schema. This schema integration task must be tackled regardless of the integration architecture or mapping formalism. In this paper, we provide a task model for schema integration. We use this breakdown to motivate a workbench for schema integration in which multiple tools share a common knowledge repository. In particular, the workbench facilitates the interoperation of research prototypes for schema matching (which automatically identify likely semantic correspondences) with commercial schema mapping tools (which help produce instance-level transformations). Currently, each of these tools provides its own ad hoc representation of schemata and mappings; combining these tools requires aligning these representations. The workbench provides a common representation so that these tools can more rapidly be combined.

1 Introduction

Schema integration is an integral aspect of any data integration endeavor. The goal of this paper is to organize the strategies and tools used in schema integration into a consistent framework. Based on this framework, we propose an open, extensible, integration workbench to facilitate tool interoperation.

We view the development of a data integration solution to consist of three main steps: schema integration, instance integration and deployment. This paper focuses on schema integration, which generates a transformation that translates source instances into target instances.

Schema integration first involves identifying, at a high level, the semantic correspondences between (at least) two schemata, data models, or ontologies, a task we refer to as *schema matching*. Second, these correspondences are used to establish precise transformations that define a *schema mapping* from the source(s) to the target.

Researchers have built many systems to semi-automatically perform schema matching [1, 2]. Schema mapping tools generally provide the user with a graphical interface in which lines connecting related entities and attributes can be annotated with functions or code to perform any necessary transformations. From these mappings, they synthesize transformations for entire databases or documents. These tools have been developed by commercial vendors (including Altova's MapForce, BEA's AquaLogic, and Stylus Studio's XQuery Mapper) and research projects (such as Clio [3], COMA++ [4] and the wrapper toolkit in TSIMMIS [5]).

© Springer-Verlag Berlin Heidelberg 2008

Currently, an integration engineer can choose to embrace a specific development environment. The engineer benefits from the automated support provided by that vendor but cannot leverage new tools as they become available. The alternative is to splice together a number of tools, each of which has its own internal representation for schemata and mappings. In one case, we needed four different pieces of software to transform a mapping from one tool's representation into another.

By adopting an open, extensible workbench, integration engineers can more easily leverage automated tools as they become available and choose the best tool for the problem at hand.

1.1 Contributions

First, we discuss the information likely to be available to integration engineers: 1) contrary to conventional wisdom, many real-world schemata are well documented, so linguistic processing of text descriptions is important, 2) in several real-world scenarios, schema integration must be performed without the benefit of instance data, and 3) domain values are often available and could be better exploited by schema matchers.

Second, we establish a task model for schema integration based on a review of the literature and tools and on observations of engineers solving real-world integration problems. We have presented our task model to three experienced integration engineers to verify that the model includes all of the subtasks they have encountered.

The task model is important because it allows us to make comparisons: Among integration problems, we can ask which of the tasks are unnecessary because of simplifying conditions in the problem instance. Among tools, we can ask what each tool contributes to each task and quantify the impact in realistic settings.

Third, we describe how the task model and pragmatic considerations guide the development of a specific integration tool, in our case Harmony, a prototype schema matcher, which bundles a variety of match algorithms with a graphical user interface.

Fourth, we articulate the need for data integration among schema integration tools—our community can benefit in insight and utility by practicing what we preach. We propose a candidate collection of interfaces that constitute an integration workbench, which allows multiple integration tools to interoperate and provides a common knowledge repository for schemata and mappings. One outcome of the integration workbench is that integration engineers can more easily choose which match algorithms (or suites thereof [6]) to use when solving real integration problems.

In this expanded version of our previous work [7], we add two new contributions: Our fifth contribution is to demonstrate the integration workbench by describing how several schema integration tools can be instantiated within the workbench. We introduce a general model for matching tools that accounts not only for the extent to which the available evidence suggests the existence of a semantic correspondence (as is traditionally done), but also the *amount* of evidence. Thus, the results generated by multiple matching tools can be combined based on the amount of evidence considered by each approach. In our experiences, the resulting match scores correspond more closely to the intuitions of integration engineers about the "goodness" of a match than traditional methods.

Our sixth contribution is a discussion of the lessons we have learned from our users' experiences with the integration workbench. We describe how the modular architecture allows these users to utilize Harmony to meet their specific schema integration needs including situations in which they have needed to introduce new tools to accomplish their tasks. We conclude by describing how the integration workbench simplifies integration of our schema matching tools with a commercial schema mapping tool (BEA's AquaLogic) that generates global-as-view (GAV) [8] mappings.

1.2 Outline

This paper is organized as follows: Section 2 contains our observations regarding schema integration efforts performed on behalf of the federal government. In Section 3 we describe a task model for integration problems. In Section 4 we present design desiderata based on the task model and describe how the Harmony schema matching tool addresses these desiderata. Section 5 describes the interfaces that constitute the integration workbench. In Section 6 we describe a set of schema matching tools that we have plugged into the integration workbench. Section 7 describes the lessons we have learned from interviewing our users about their experiences with Harmony. Finally, we discuss related work in Section 8 and future work in Section 9.

2 Integration in Large Enterprises

Conventional wisdom suggests that schema matching should focus on data instances because instances are common and documentation is sparse (or even incorrect). Whereas these phenomena may be observed in some settings, particularly web-based sources, it is often not the case for schemata developed for or by the US federal government (or, we suspect, other large enterprises).

From the perspective of an integration engineer, data instances may be extremely hard to obtain (the data exist, but are not available to the engineer) for at least two reasons.

- **Security/sensitivity**: Data instances are often more sensitive than their corresponding schemata—e.g., in defense applications, an integration engineer may have access to schemata but may lack sufficient clearances to access instances. Sometimes, an agency that owns the data is willing to share them with another agency, but not with the contracting integration engineers responsible for developing the initial mappings. Wider release of schema information is less problematic.
- **Integrating to a future system:** One may begin creating important mappings to and from a new system, even before it has any data or running applications. For example, the U.S. Federal Aviation Administration developed a mapping of some of its systems to a conceptual model for the new European Air Traffic Control System before that system was implemented or had any instance data. As a general phenomenon, when one builds a data warehouse, the mappings from data sources are the actual means for populating it.

Thus, we have observed that it is not safe to assume that instance data will be available to integration tools. Instead, schema integration tools must use whatever information is available. Instance data, thesauri, etc. are sometimes available and sometimes not.

While instance data are often unavailable, we have found that many government (and probably many other enterprises') schemata are well documented. Evidence for this claim will now be presented.

We obtained a collection of 265 conceptual (ER) models from the Department of Defense metadata registry (which contains schemata only, no instances!). This repository contains 13,049 elements (entities or relationships) and 163,736 attributes. As indicated in Table 1, the vast majority of these items contain a definition of roughly one sentence.

Table 1. Frequency and length of documentation in the DoD Metadata Registry

Item	Item Count	# With Definition	% With Definition	Word Count	Words/ Item	Words per Definition
Element	13,049	12,946	~99%	143,315	~11.0	~11.1
Attribute	163,736	135,686	~83%	2,228,691	~13.6	~16.4
Domain	282,331	282,128	~100%	1,036,822	~3.67	~3.68

This registry also explicitly enumerates domain values for which documentation is also available. A domain introduces a list of codes, each of which has particular semantics. A domain is a reusable schema construct that can be referenced by multiple attributes. For example, a common domain is the list of two-character state codes (such as VA or MD). In a shipping order, this domain might be referenced by both the shipping entity and the billing entity. Domains and their associated documentation facilitate schema integration even in the absence of instance data. Unfortunately, this documentation is often lost when a logical schema is converted into SQL. The standard approach is to store each coding scheme in its own relation, and each code as a string or integer value, *sans* documentation.

This approach is good for referential integrity, but bad for integration efforts. A better solution would be to define semantic domains for each coding scheme so that integration tools could more easily identify domain correspondences. In fact, when we asked integration engineers to describe how they approach an integration problem, a recurring pattern emerged. They first identify obvious top-level entity correspondences. But then, instead of proceeding to sub-elements or attributes, they then manually inspect the domain values to find correspondences. From this low-level, they then work their way up the schema hierarchy to attributes, sub-elements, and finally back to top-level entities. Our task breakdown is designed to support this pattern.

3 Task Model for Data Integration

To better understand how schema integration tools assist an integration engineer, we enumerated the subtasks involved in schema integration. We started with a task model that we created and that was acceptable to 147 survey participants familiar with schema integration from a research or practical perspective [9]. We extended that model to include the subtasks addressed by a variety of systems ([4, 5, 10-16]) and then presented it to three experienced integration engineers for validation. Based on their feedback, we extended the model to include subtasks not directly supported by any system.

At a high level, we consider 13 fine grained integration tasks, grouped into five phases: schema preparation, schema matching, schema mapping, instance integration and finally system implementation. During schema preparation, the source and target schemata are identified so that a set of correspondences can be identified during the matching phase. These semantic correspondences are formalized in the third phase as explicit logical mappings. Once schema integration is complete, instance integration reconciles any remaining discrepancies. In the final phase the integration solution is deployed.

In this section, we describe each phase in detail and describe how we evaluated the task model's completeness. Throughout this section we refer to the following terms: a schema is a collection of schema elements, each of which is either an entity or an attribute. An entity represents a collection of related instances, and an attribute represents a relationship between an entity and another entity or a datatype. An instance belongs to a particular entity and it instantiates values for that entity's attributes. In many cases, the ultimate goal of data integration is to transform source instances into valid target instances.

3.1 Schema Preparation

The first phase of schema (or data) integration captures knowledge about the source and target schemata, to facilitate the subsequent matching and mapping phases. It identifies the target schema, and organizes the source schemata. The specific subtasks are:

1) Obtain the source schemata. This step gathers available documentation and imports the source schemata into the integration platform. If the source schemata are not in a format compatible with the platform, this step also includes any necessary syntactic transformations.

2) Obtain or develop the target schema. If performed, this step is analogous to the previous step. In many cases, the target schema is defined by the problem specification (e.g., translate data into the following message format). In other cases, the target schema must be developed based on the queries to be supported, or to combine the data from multiple sources. This step is optional because the target schema may be derived from the correspondences identified among the source schemata, as is assumed in [11].

In both cases, one may enrich the schemata, e.g., by defining coding schemes as domains, or documenting constraints that are not documented in the actual system, either because the system does not support the needed constructs, or because nobody took the time to do so. Thus, the integration platform may enable richer descriptions than the underlying systems. One also needs a means to keep the metadata in synch as the actual systems change.

3.2 Schema Matching

The second phase establishes high-level correspondences among schema elements. There is a semantic correspondence between two schema elements if instances of one schema element imply the existence of corresponding instances of the other [17]. We avoid a more precise definition of a semantic correspondence because the nature of a correspondence depends on the overall goal of schema integration. For example, in the case of a data exchange system, these correspondences imply the existence of a logical transformation that can convert instances of the source element into instances of the target. However, when the integration goal is to generate a consensus vocabulary for a particular community, a semantic correspondence may indicate that the set of source instances overlaps with the set of target instances (i.e., their intersection is non-empty).

If a target schema has been identified, these correspondences establish relationships between each source schema and the target. As noted in [11], in the absence of a target schema, correspondences can also be established between pairs of (or across sets of) source schemata.

For example, to publish data stored in a relational database into an XML message format, some correspondences indicate that tuples from the source relation will be used to generate XML elements. Additional correspondences indicate which attributes will be used to generate data values. For example, multiple relations might correspond to a single element because a join is needed to populate the element's attributes, or a single relation may correspond to multiple elements to match nesting present in the target.

3) Generate semantic correspondences. This step determines which schema elements loosely correspond to the same real world concepts. These correspondences establish a weak semantic link in that they indicate that instances of one element can be used to generate instances of the other.

Whereas this phase consists of a single step, we consider matching to be its own phase because of its importance and the research attention it has received. The exact transformations implied by a correspondence are detailed in the mapping phase.

3.3 Schema Mapping

The schema mapping phase establishes, at a logical level, the rules needed to transform instances of the source schemata into instances of the target. The mappings must generate results that adhere to the target schema (or the target must be modified to reflect accurately the transformed data).

These mappings are often expressed as queries expressed in a language applicable to the source or target schema. For example, in [8] mappings are expressed as Datalog queries and in [18] mappings are expressed using XQuery (even though the source schema is relational). However, in [19] the mappings are expressed in SQL even though the transformed data are expressed as XML.

The first four subtasks below establish piecemeal transformations, and are not performed in a particular order. Each transformation indicates the precise mechanism by which source data is used to generate target data. Note that at times these transformations cross the schema/instance boundary [20]. Once transformations have been established for each schema element, they are aggregated into a logical mapping and verified.

4) Develop domain transformations. For each pair of corresponding domains, a transformation must be developed that relates values from the source domain to values in the target domain. In the simplest case, there is a direct correspondence (i.e., no transformation is needed). However, it is often the case that an algorithmic transformation must be developed, for example, to convert from feet to meters, or from first- and last-name to full-name. In the most detailed case, the transformation can best be expressed using a lookup table (e.g., to convert from one coding scheme to a related coding scheme). Context mediation techniques can then be applied [21, 22].

5) Develop attribute transformations. The previous step handled the case where the same property was encoded using different domains. This step deals with properties that are different but derivable. Sometimes one provides a transformation from source to target values, either scalar (e.g., Age from Birthdate), or by aggregation (e.g., AverageSalaryByDepartment from Salary). Other transforms we have seen include pushing metadata down to data (e.g., to populate a type attribute or timestamp), and populating a comment (in the target) to store source attribute information that has no corresponding attribute. Finally, it may be necessary to convert a single attribute into a composition of attributes (in the local-as-view (LAV) [8] formalism) or vice versa for GAV.

6) Develop entity transformations. The next step is to determine the structural transformations necessary to generate instances of the target schema. In the simplest case, a direct 1:1 mapping can be established. Alternatively, multiple entities may need to be combined to generate a single target entity. This combination may require a join operation if the source schema vertically partitions information across multiple entities or a union operation if the source schema horizontally partitions information across entities that are subclasses of the target entity. Additionally, a single entity may need to be split into multiple entities (e.g., based on the value of some attribute), which effectively elevates data in the source to metadata in the target.

7) Determine object identity. For each entity in the target, the next step is to determine how unique identifiers will be generated. In the simplest case, explicit key attributes in the source can be used to generate key values in the target. This may include populating implicit keys (such as those inherited from a parent entity), or correctly establishing parent/child relationships (such as in a nested meta-model). For arbitrarily assigned identifiers (such as internal object identifiers), Skolem functions are commonly employed (see, for example, [3]).

These four subtasks interact with schema matching because establishing transformations is an iterative process. For example, in the first pass, we might establish a transformation from Professor to Employee (since instances of the former are also instances of the latter). While working on the Course/Grade sub-schema, we might realize that, in some cases, Students are also Employees. This new insight requires us to refine the Employee mapping. In other words, the previously identified correspondences may be both imprecise and incomplete.

The remaining mapping subtasks produce an executable mapping.

8) Create logical mappings. The next step is to aggregate the piecemeal mappings, which all concern individual elements, into an explicit mapping for entire databases or documents. Humans may need to specify additional information (e.g., to distinguish join from outerjoin) before automated tools can sew the pieces together. In most cases, this requires writing a query (over the source schemata) that generates instances of the target schema, although in LAV [8] the source schemata are expressed as views over the target schema.

9) Verify mappings against target schema. If the integration task included a specific target schema, the final step is to verify that the transformations are guaranteed to generate valid data instances (i.e., all constraints are satisfied). In some cases, the only solution may be to modify the target schema to remove constraints that we cannot satisfy. If a target schema is not specified, the final step is to generate the target schema based on the logical mappings.

3.4 Instance Integration

At this point, the tasks involved in schema integration are complete, and we turn our attention to instance integration.

10) Link instance elements. Two source instances (with different unique identifiers) may represent the same real-world object. This subtask merges these instances into a single instance or creates an association between the instances. See [23] for an overview of the algorithms involved.

11) Clean the data. This subtask removes erroneous values from instances. A value may be erroneous because it violates a domain constraint or because it contradicts information from a more reliable source. For example, we may know that a person should have a single value for the height attribute, but the available sources might provide differing values for this attribute value. See [24] for more information about this subtask.

3.5 System Implementation

Finally we are ready to develop and deploy a system that addresses operational constraints—factors external to schema and instance elements. Examples include determining the frequency and granularity of updates and the policy that governs exceptional conditions.

12) Implement a solution. In this phase the system developers must first gather any operational constraints and then design an integration system that satisfies these constraints. The significance of the operational constraints on real-world integration systems is stressed by the integration engineers who have reviewed the task model. For example, operational constraints such as the volume of data involved, the freshness of results, and security factors strongly influence whether a federated database or data warehouse should be developed.

13) Deploy the application. This step does not receive much research attention, but ease of deployment is an important concern. Many of the commercial data integration tools place particular emphasis on this subtask. Once deployed, system engineers must maintain the application, but a task model for application maintenance exceeds the scope of this paper.

This task model guided our development of the Harmony schema matching tool.

4 Harmony

Harmony is a schema matching tool that combines multiple match algorithms with a graphical user interface for viewing and modifying the identified correspondences. The architecture for Harmony is shown in Fig. 1. Harmony's contributions include adding linguistic processing of textual documentation to conventional schema match techniques, learning from the input of a human in the loop, and GUI support for removing clutter and iterative development, as discussed in following sections.

Harmony currently supports XML schemata, entity-relationship schemata from ERWin, a popular modeling tool, and will soon support relational schemata. Schemata are normalized into a canonical graph representation.

The Harmony match engine adopts a conventional schema integration architecture [6, 25-27]. It begins with linguistic preprocessing (e.g., tokenization, stop-word removal, and stemming) of element names and any associated documentation. Then, several *match voters* are invoked, each of which identifies correspondences using a different strategy. For example, one matcher compares the words appearing in the elements' definitions. Another matcher expands the elements' names using a thesaurus. For each [source element, target element] pair, each match voter establishes a confidence score in the range (–1, +1) where –1 indicates that there is definitely no correspondence, +1 indicates a definite correspondence and 0 indicates complete uncertainty.

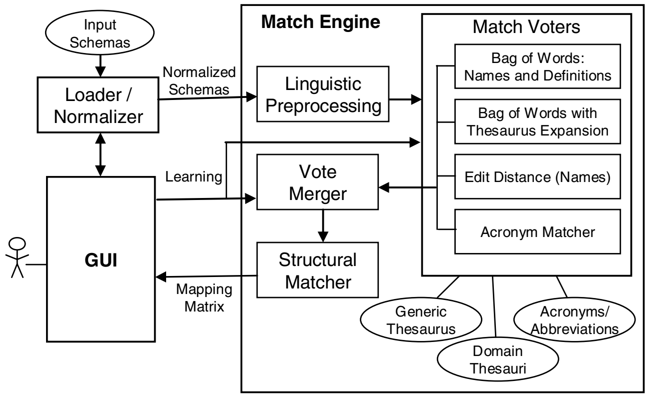


Fig. 1. Architectural Overview of Harmony

Given *k* match voters, the vote merger combines the *k* values for each pair into a single confidence score. The vote merger weights each matcher's confidence based on its magnitude—a score close to 0 indicates that the match voter did not see enough evidence to make a strong prediction.

A version of similarity flooding [28] adjusts the confidence scores based on structural information. Positive confidence scores propagate up the schema graph (e.g., from attributes to entities), and negative confidence scores trickle down the schema graph. Intuitively, two attributes are unlikely to match if their parent entities do not match.

Finally, these confidence scores are shown graphically as color-coded lines connecting source and target elements. The GUI provides various mechanisms for manipulating these lines, based on our design desiderata.

4.1 Design Goals

The statistics presented in Section 2 suggest that schema matching algorithms should not assume the absence of usable documentation. Many of the candidate matchers in the Harmony engine perform natural language processing and comparisons on this documentation. In our experience, these matchers have good recall, although their precision is less impressive.

The task model in Section 3 suggests additional design desiderata. First, the integration engineer needs to be able to focus at different levels of granularity. For example, a common first step is to establish correspondences among conceptual sub-schemata. In the air traffic flow management domain, these sub-schemata might include facilities (airports and runways), weather, and routing. Note that the hierarchical and decomposable nature of XML Schema makes it easier to identify sub-schemata.

After establishing these high-level correspondences, the integration engineer focuses on one sub-schema at a time and delves into the details of the domains appearing in that sub-schema. The engineer wants to be distracted neither by correspondences pertaining to other sub-schemata nor those at intermediate levels of granularity.

A related goal is that the software tools must support iterative refinement. This desideratum is one of our motivations for developing the integration workbench described in Section 5. If data cannot flow freely among components, the engineer has little control over the order in which tasks will be completed.

The final desideratum is that all sub-tasks involved in schema integration must be supported. The commercially available tools naturally take this requirement more seriously than do research tools, such as Harmony. Whereas it is an interesting research problem to identify semantic correspondences, this contribution alone does not greatly assist the integration engineer. Because Harmony by itself does not currently support schema mapping, we defer further consideration of this desideratum to Section 5. We now consider how Harmony addresses the remaining desiderata.

4.2 Filtering

The Harmony GUI supports a variety of filters that help the integration engineer focus her attention. These filters are loosely categorized as link filters and node filters. A link filter is a predicate that is evaluated against each candidate correspondence to determine if it should be displayed. A node filter determines if a given schema element should be *enabled*. An enabled element is displayed along with its links; a disabled element is grayed out and its links are not displayed.

Harmony currently supports three link filters. First, a confidence slider filters links based on the confidence assigned to a link by the Harmony engine. Only links that exceed the slider-set threshold are displayed. Links that were drawn by the integration engineer, or were explicitly marked as correct, have a confidence score of +1. Similarly, links explicitly rejected have a score of –1.

The second filter determines if a link should be displayed based on whether it is human-generated or machine-suggested. The final filter displays those links with maximal confidence for each schema element (usually a single link, but ties are possible).

The node filters include a depth filter and a sub-tree filter. The former enables only those schema elements that appear at a given depth or above. For example, in an ER model, entities appear at level 1, while attributes are at level 2. In XML schemata, arbitrary depths are possible. Thus, using this filter, the engineer can focus exclusively on matching entities.

The sub-tree filter enables only those elements that appear in the indicated sub-tree. For example, this filter can be used to focus one's attention on the 'Facility' sub-schema. By combining these filters, the engineer can restrict her attention to the entities in a given sub-schema.

4.3 Iterative Development

Harmony supports iterative refinement through two mechanisms. First, the engineer can rerun the Harmony engine, which can learn from her feedback. Second, the engineer can mark sub-schemata as complete. We now describe these two mechanisms.

When the Harmony engine is invoked after some correspondences have been explicitly accepted or rejected (i.e., set to +1 or –1), this information is passed to the engine and used in two ways. First, each candidate matcher can learn from the user's choices and refine any internal parameters. For example, a matcher that weighs each word based on inverted frequency increases or decreases word weight based on which words were most predictive. Second, the vote merger weights the candidate matchers based on their performance so far. Learning new weights must be done carefully, though. Each candidate matcher focuses on a particular form of evidence, such as elements' names. If the engineer based her first pass on exactly that form of evidence, the corresponding candidate matcher will appear overly successful.

In addition to accepting and rejecting specific links, the engineer can mark a sub-tree as complete. This action has several effects. First, it accepts every link pertaining to that sub-tree as accepted (if currently visible), or rejected (otherwise). Once a link has been accepted or rejected, the engine will not try to modify that link. This ensures that links do not mysteriously disappear or appear should the user subsequently invoke the Harmony engine.

Second, it updates a progress bar that tracks how close the engineer is to a complete set of correspondences. This feature was introduced at the request of integration engineers working on large schema integration problems that involve several dozen iterations.

Once all schema elements have been marked as complete, the final set of correspondences could be used to guide the generation of a more detailed mapping. Harmony provides neither a mechanism for authoring code snippets, nor a code generation feature; these would duplicate commercial capabilities. Instead, we are developing the integration workbench to couple our matching tools (and GUI) with commercially-available mapping products.

5 Integration Workbench

Our attempts to integrate Harmony with other schema integration tools revealed a key barrier to interoperability. Whereas schema integration experts trumpet the advantages of a modular, federated architecture that presents a unified view of multiple data sources, we (as a community) have not applied that same insight when we develop our own systems and tools.

As a concrete example, we recently received a collection of XML files from a colleague. Each file described a schema mapping between a source and target schema.

However, before we could use these files, we needed to transform them into a structure compatible with Harmony. To effect this transformation we used one tool to reverse engineer the schema assumed by our colleague. We then matched that schema to the Harmony schema (using Harmony). We recreated the match in AquaLogic to generate a suitable XQuery for transforming a single XML file. Finally, we wrote a Perl script to apply the XQuery to each XML file. A modular architecture would facilitate tool interoperability.

While some vendors (such as IBM and BEA) may be moving in this direction internally to support integration of their own tools, they have not published their approaches or interfaces. There are obvious advantages to user organizations and small software companies to developing a *standard* framework for combining schema integration tools. We propose the following as a way to initiate discussion that could lead toward development of such a standard.

At the core of our workbench proposal is an integration blackboard, which is a shared knowledge repository. Mediating between the blackboard and the various schema integration tools is a workbench manager. The manager provides several services including transaction management, event services and query evaluation. The following sections describe the blackboard and manager.

5.1 Integration Blackboard

The integration blackboard (IB) is a shared repository for information relevant to schema integration that is intended to be accessed by multiple tools, including schemata, mappings, and their component elements. We propose using RDF [29] for the IB, because: 1) it is natural for representing labeled graphs, 2) one can use RDF Schema to define useful built-in link types while still offering easy extensibility, 3) it is vendor-independent, and 4) it has significant development support.

The basic contents of the IB are schema graphs and mapping matrices (an approach also taken in [25]). However, in RDF, any element can be annotated; we use this feature to enrich the graphs and matrices with additional information. We predefine certain annotations using a controlled vocabulary (these terms appear in sans serif).

5.1.1 Schemata

The IB represents a schema as a directed, labeled graph. The nodes of this graph correspond to schema elements. In the relational model, these elements include relations, attributes and keys. In XML, they include elements and attributes.

The edges of a schema graph correspond to structural relationships among the schema elements. These edges are object properties whose subject and object are both schema elements. For example, in the relational model contains-table edges are used to link a database to the tables it contains. Tables are linked to attributes via contains-attribute edges. In XML, elements are linked to sub-elements via contains-element edges, and to attributes via contains-attribute edges. For many schema languages, the edge-types are specified by the modeling language, but with ontologies they are extensible.

Whereas schema elements can be annotated arbitrarily, we identify three edge labels of particular importance to schema importing and matching utilities: name, type

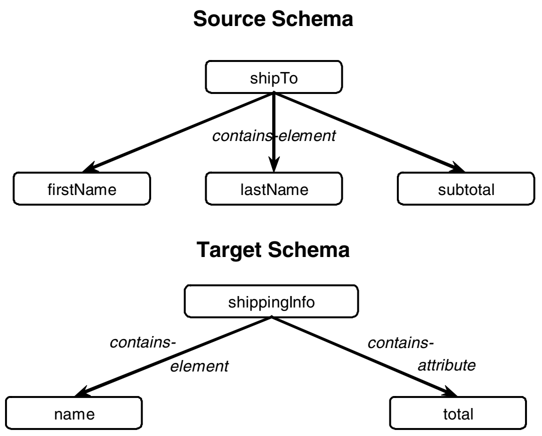


Fig. 2. Sample schema graphs

and documentation. Import tools populate these metadata so that they can be used by schema matchers to identify potential correspondences.

Sample schema graphs appear in Fig. 2. In the next section we present a sample mapping from the source schema to the target schema.

5.1.2 Mappings

Inter-schema relationships can be represented conceptually as a *mapping matrix*. This matrix consists of headers (describing source and target elements) plus content (a row for each source element and a column for each target element). Note that whereas the structure can easily be interpreted as a matrix, we store this matrix using RDF.

code= `let $shipto := $purchOrd/shipTo return <shippingInfo total = "{ data($shipto/subtotal) * 1.05 }"> { for $fName in $shipto/firstName, $lName in $shipto/lastName return <name>{ concat($lName, concat(", ", $fName)) }</name> } </ShippingInfo>`	**shippingInfo** is-complete=`false` code=	**name** is-complete=`false` code= `concat($lName, concat(", ", $fName))`	**total** is-complete=`false` code= `data($shipto/subtotal) * 1.05`
shipTo is-complete=`false` variable=`$shipto`	confidence=`+0.8` user-defined=`false`	confidence=`−0.4` user-defined=`false`	confidence=`−0.6` user-defined=`false`
firstName is-complete=`true` variable=`$fname`	confidence=`−1` user-defined=`true`	confidence=`+1` user-defined=`true`	confidence=`−1` user-defined=`true`
lastName is-complete=`true` variable=`$lname`	confidence=`−1` user-defined=`true`	confidence=`+1` user-defined=`true`	confidence=`−1` user-defined=`true`
subtotal is-complete=`true` variable=`$shipto/subtotal`	confidence=`−1` user-defined=`true`	confidence=`−1` user-defined=`true`	confidence=`+1` user-defined=`true`

Fig. 3. Sample mapping matrix in which every component has been annotated

For example, the mapping matrix for the schemata in Fig. 2 contains four rows and three columns, as shown in Fig. 3. Each cell in the mapping matrix describes a potential correspondence between a source element and a target element.

Mapping elements are also annotated. First, each cell is annotated with confidence-score, which ranges from –1 (definitely not a match) to +1 (definitely a match), and is-user-defined. This latter annotation is true for any correspondence provided by the user (for example, by drawing a link between two elements), and the associated confidence-score is either +1 or –1 (for rejected links). When a match algorithm is executed, is-user-defined is false, and the confidence-score falls in the range (–1,+1).

Each row is further annotated with a variable-name. Each column is annotated with code that references these names. Finally, the matrix as a whole has a code annotation, which represents the mapping from source to target. Additional annotations are possible; for example, Harmony annotates rows and columns with is-complete to track progress. The relationship between these annotations and the mapping matrix appears in Fig. 3.

5.1.3 Integration Blackboard Enhancements

We currently assume that the blackboard captures information about the source and target schemata, as well as the current state of the mapping that relates the source(s) to the target. Future goals include the following.

- The blackboard should maintain a library of mappings, partly to facilitate mapping reuse, but also as a resource for some matching tools.
- Schemata inevitably change; the blackboard should track schemata across versions.
- Mappings are also refined over time, especially once they are tested on real data. The blackboard should maintain mapping provenance.
- Based on Section 4.2, the blackboard should allow contextual information, such as focus on a particular subschema, to be shared across tools.
- The blackboard should be shared across multiple workbench instances.

5.2 Workbench Manager

All interaction with the IB occurs via the workbench manager, which coordinates matchers, mappers, importers, and other tools. The manager provides several services: First, it provides transactional updates to the IB. Second, following each update, it notifies the other tools using an event. Third, the manager processes ad hoc queries posed to the IB.

A single-user version of the workbench architecture appears in Fig. 4. Ultimately, we envision there to be one IB for each community of interest—i.e., a set of stakeholders “who must exchange information in pursuit of their shared goals, interests, missions, or business processes” [30]. Each integration engineer would have her own instance of the integration workbench containing a single manager and multiple tools.

5.2.1 Tools

We focus on four kinds of tools: loaders, matchers, mappers and code-generators. The first two tools support the first two phases of schema integration. Given the complexity of schema mapping, we separate out steps 4)–7), in which the mapping is produced piecemeal, from steps 8) and 9), in which code is generated.

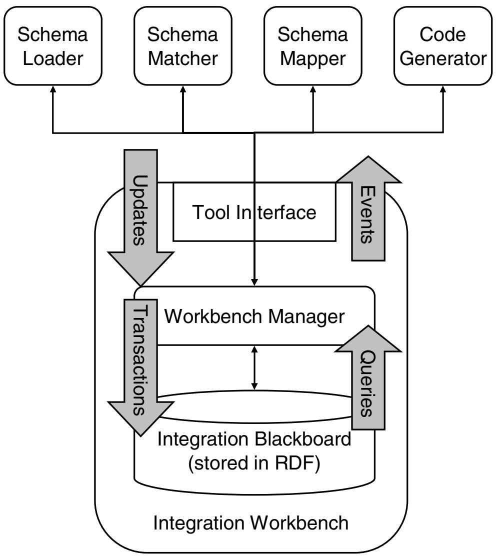


Fig. 4. Workbench Architecture

Loaders are used during schema preparation to parse a schema from a file, database or metadata repository (including ancillary information such as definitions from a data dictionary) into the internal representation used by the IB. When the user invokes a loader, that tool places the new objects in the IB, which extends the mapping matrix accordingly and advises the other tools via an event.

Schema matching can be performed manually, as is the case for most commercial tools, or semi-automatically. (Harmony supports both approaches.) A match tool updates the cells of the mapping matrix. When correspondences are generated automatically, all of the interactions with the IB are wrapped in a transaction; no events are generated until the mapping matrix has been updated.

Schema mapping can also be performed manually or automatically [31], although we are not aware of any commercial automatic mapping tools. A mapping tool updates the code associated with each column. Both matchers and code generators may need to listen for these events to update their internal state.

Finally, a code-generator assembles the code associated with each column into a coherent whole. Thus, the code-generator must understand how to assemble code snippets based on the structure of the target schema graph (e.g., Clio [3]).

This enumeration of tools is by no means complete. Another tool might attempt to enforce domain-specific constraints on the mapping matrix. Or, a tool might annotate a schema with information culled from external documentation. All that is required is that a tool implements the tool interface.

The tool interface defines two methods. First, a tool must provide an invoke method. The implementation of this method might launch a GUI (for mapping), invoke a match algorithm, or display a file selection dialog (to load). Second, when the workbench starts, each tool has the option of implementing an initialize method. Generally, this is done when a tool needs to register for events.

5.2.2 Events

Tools generate events whenever they make any change to the contents of the IB. The workbench manager propagates these events to allow any tool to respond to the update. A different type of event is generated for each major component of the IB so that a tool can register for only those events relevant to that tool.

A schema loader generates a *schema-graph event* when it imports a schema into the workbench. Any tool with a GUI listens for these events and refreshes the display.

A *mapping-cell event* is generated when a user manually establishes a correspondence. Multiple such events are triggered by an automatic matching tool. A mapping tool can listen for these events to propose a candidate transformation, such as a type conversion.

Conversely, when a mapping tool establishes a transformation, it generates a *mapping-vector event*. Match tools listen for these events to synchronize the mapping cells with the updated row or column. A code generation tool similarly listens for these events to synchronize the assembled mapping. The code generation tool, in turn, generates a *mapping-matrix event* when the user manually modifies the final mapping.

Additional interactions are possible, but generally speaking, a tool listens for events immediately upstream or downstream in the task model. It is necessary to listen in both directions given the iterative behavior described in Section 4.3 and illustrated by the schema matching tools we have developed.

6 Sample Schema Matching Tools

Our research has focused on the development of two types of schema matching tools. The role of a match voter is to consider some source of evidence to generate a match score for a particular (source element, target element) pair. The match score is a function of the amount of evidence observed that suggests the pair of elements match (the positive evidence) and the total amount of evidence available. The standard approach for generating a match score is to compute the ratio of positive evidence to total evidence.

However, this approach ignores the fact that, as the amount of evidence increases, the impact of that evidence is greater. In this section, we first formalize the roles of positive and total evidence. We then describe how to apply this theory to various match voters.

Within the integration workbench, multiple match voters might be available. The role of a vote merger is to combine the match scores generated by a suite of match voters into a single confidence score to be stored in the mapping matrix. To derive a confidence score the vote merger assigns a weight to each match voter and combines the match scores based on the amount of evidence observed by each match voter.

In this section we describe each component in greater detail. As a preliminary, we briefly describe how we normalize the available documentation. We then describe how *any* match voter can compute a match score based on numeric scores for positive and total evidence. Next, we describe a specific match voter in which the evidence is based on the extent to which the words appearing in the schema documentation for two elements overlap. Finally, we describe how the scores generated by multiple match voters can be combined into a single value.

6.1 Text Normalization

For some match voters, several pre-processing strategies are required. First, we tokenize all text strings in the source and target schemata, splitting those phrases that are not divided by spaces into distinct words. Because of the frequency with which upper-case letters are used to indicate word boundaries (sometimes called CaMeL case), whenever an upper-case letter is immediately followed by a lower-case letter, we break the text into separate words at that boundary (e.g., 'firstName' becomes 'first Name'). Tokenization also removes all punctuation. Following tokenization, the text contains only letters, numbers and white-space.

Second, we replace all capital-letters with lower-case letters. Third, we remove plural suffixes and verb conjugations. For example, 'reading books' becomes 'read book'. Fourth, we remove any words that appear on a pre-defined list, (such as 'a' and 'for'). These *stop-words* are too common to be useful for linguistic processing. We refer to the output of these four steps as *normalized* text.

During pre-processing, we also count the frequency of each normalized word appearing anywhere in a source or target schema element. Generally speaking, words that are rarely used are more significant that words that appear frequently. The word frequency function *freq(wd)* maps each word *wd* to the number of times it appears in normalized text:

$$freq(wd) \rightarrow N \tag{1}$$

The weight associated with each word is inversely proportional to the number of times it appears in the source and target schemata. In the ideal case, a word appears exactly once in the source and once in the target, or twice total. Based on this observations, the weight function *wt(wd)* is:

$$wt(wd) = \frac{2}{freq(wd)} \tag{2}$$

As an ongoing example, we will consider two schema elements drawn from the domain of military tracking. In this domain, it is important to know how a particular set of coordinates were obtained so that human experts can gauge the reliability of the information. Hence, we will consider source element s = "How: provides a hint about how the coordinate was obtained," and target element t = "TargetSource: indicates how the latitude and longitude were obtained." Whereas these elements are not identical, they are similar in nature and should be matched (and ultimately mapped) to one another.

After normalization, these elements are simplified to "how provide hint about how coordinate obtain" and "target source indicate how latitude longitude obtain," respectively. For simplicity, let us assume "how" appears sixteen times in the source and target schemata and that the remaining words appear twice each. Thus, the wt("how") = 0.125 and $wt(wd)$ = 1 otherwise. We will return to this example in section 6.3 when we describe our bag-of-words match voter. But first we describe (in abstract terms) our match score framework from the perspective of positive and total evidence.

6.2 Match Scores

Our match score framework expects that each match voter will assign a single score to each pair of source and target elements. This match score is generated by considering some collection of evidence (*toe* for total observed evidence) of which a subset suggests a correspondence between the pair of elements (*poe* for positive observed evidence). For example, in the preceding example, the total evidence consists of the words used to describe *s* and *t* and the positive evidence consists of the words they share. Other sources of evidence might include the datatypes assigned to these elements or the data values used to instantiate them. In this section we describe how any match voter can combine *toe* and *poe* to generate a match score that ranges from –1 to +1.

The intuition behind these match scores is that a score of 0 should indicate that, based on the observed evidence, the likelihood of a match is impossible to determine. As the ratio of positive evidence to total evidence increases, the match score should increase. For a fixed evidence ratio, as the total evidence increases, the match score should also increase.

Based on this intuition, we can establish some theoretic bounds. If there is an infinite amount of positive evidence, the match score should equal +1. However, if there is no positive evidence, but an infinite amount of total evidence, the match score should equal –1. Finally, if there no evidence (of either type), the match score should be 0.

Formally, for a given (source element, target element) pair, let *poe* represent the amount of positive observed evidence, and *toe* represent the total observed evidence. However, before observing this evidence, there is some small probability *x* that two elements (chosen at random) match. Thus, we must factor in this prior probability to calculate the (combined) positive evidence *pe* and total evidence *te*.

$$pe = x + k \times poe \tag{3}$$

$$te = 1 + k \times toe \tag{4}$$

In equations (3) and (4), *k* is a scaling factor that indicates how much we want to weigh the observed evidence. Now, we calculate the evidence ratio *er* as the ratio of positive evidence to total evidence.

$$er = \frac{pe}{te} \tag{5}$$

The weighted evidence ratio *wer* scales the evidence ratio from the interval [0, 1] to the interval [1, e]. When the weighting factor *j* is one, this is a linear transformation. Large values of j generate a sub-linear transformation.

$$wer = er^{1/j}(e-1)+1 \tag{6}$$

The evidence factor *ef* measures the amount of evidence considered by mapping the positive evidence from the interval [0, ∞) to the interval [e, 1].

$$ef = (1+pe)^{1/pe} \tag{7}$$

The match score *ms* is the natural log of the ratio between *wer* and *ef*.

$$ms = \ln\left(\frac{wer}{ef}\right) \tag{8}$$

Table 2. Relationship between evidence (positive and total) and match scores for extreme values. The final column provides insight into equations (3)–(8).

pe	te	er	wer	ef	ms	=
0	∞	0	1	e	–1	$\ln\left[\frac{1}{e}\right]$
0	0	1	e	e	0	$\ln\left[\frac{e}{e}\right]$
∞	∞	1	e	1	1	$\ln\left[\frac{e}{1}\right]$

Finally, the match score is guaranteed to fall in the interval (–1, +1) as demonstrated by a limit analysis (see Table 2) as the positive and total evidence approach 0 and positive infinity. All that remains is to determine suitable values for the parameters *j*, *k*, and *x*. We choose *x* such that in the absence of direct evidence, the match score evaluates to 0. Whereas we have not found a closed solution for *x* in terms of *j*, for certain values of *j* we have observed the following:

$$x \approx e^{\frac{-\ln j}{1.5}} \quad \text{when} \quad j \geq 7 \tag{9}$$

The values of the remaining two parameters depend on the match voters under consideration. Generally speaking, *j* controls how much positive evidence is required for *ms* to generate a match score greater than zero, and *k* amplifies the observed evidence. We recommend suitable values for these parameters for match–voters based on algorithms developed for measuring the similarity between two natural language documents.

6.3 Sample Linguistic Match Voters

The preceding section described a match voter at an abstract level. We now turn our attention to specific match voters based on natural-language processing (NLP). In this section, we describe how to quantify the observed evidence for NLP-based match voters. We then establish reasonable values for the constants described above.

In the domain of document retrieval, one strategy for determining the similarity of two documents is to determine the extent to which the pair of documents has words in common. We apply this approach to schema matching by treating each schema element as a document. For a given schema element, the corresponding document contains the normalized text appearing in the element's documentation and name[1]. This document is then reduced to a bag-of-words (i.e., a set of words in which a given word can appear multiple times). The evidence represented by bag-of-words B_s is computed as follows, where the weight function was defined in equation (2), above.

$$ev(B) = \sum_{wd \in B} wt(wd) \tag{10}$$

[1] Because of the importance of an element's name, we actually add that normalized text to the document twice.

Table 3. Relationship between evidence (positive and total) and match scores for three different values of positive observed evidence.

$B_s \cap B_t$	*poe*	*toe*	*pe*	*te*	*er*	*wer*	*ef*	*ms*
{"how", "obtain"}	1.125	10.125	3.6	32	0.11	2.4	1.5	0.44
{"how"}	0.125	10.125	0.59	32	0.019	2.2	2.2	–0.019
{}	0	10.125	0.22	32	0.0068	2.0	2.5	–0.19

For a given (source-element, target-element) pair, the positive evidence is based on the intersection of the corresponding bags, and the total evidence is based on the union.

$$poe(s,t) = ev(B_s \cap B_t) \tag{11}$$

$$toe(s,t) = ev(B_s \cup B_t) \tag{12}$$

In our ongoing example ("How" vs. "TargetSource"), the positive observed evidence is based on the bag {"how", "obtain"} and the total observed evidence on the bag {"about", "coordinate", "hint", "how", "how", "indicate", "latitude", "longitude", "obtain", "provide", "source", "target"}. Given the previously assigned word weights, *poe*(*s*,*t*) = 1.125 and *toe*(*s*,*t*) = 10.125.

Harmony also supports the inclusion of evidence external to the source and target schemata. A second match voter uses a bag-of-words augmented with a thesaurus. For each word in B_s, if that word appears in the thesaurus, its synonyms are added to the bag. Once the bags have been augmented with synonyms, the weight function in equation (2) must be re-evaluated. Otherwise, the thesaurus-based bag-of-words match voter is identical to the normal bag-of-words match voter.

All that remains is to establish values for *j* and *k*. In our experience, *j*=20 seems to work well in practice. Given the trade-off between precision and recall, we prefer to err on the side of recall because it is easier for an integration engineer to reject false matches, than to identify false non-matches. We found *k*=3 to work well for the basic bag-of-words matcher, and *k*=1 to work well when using a thesaurus. The intuition behind using a smaller *k* is that we expect to see more total evidence with the thesaurus, and therefore do not need to amplify the effect of the observed evidence.

To illustrate how the total and positive observed evidence is used to calculate a match score, we will return to our ongoing example in which the positive observed evidence value is 1.125. Let us also consider similar scenarios in which the common words are {"how"} and {}. (This example assumes *k*=3, but *j* is set to 10 because the documentation strings are so short). Table 3 shows how the match scores are derived in each of these scenarios. We have deliberately chosen *j* and *k* such that the match score will be relatively large whenever a pair of elements share even a small number of uncommon words. Moreover, very low scores cannot be generated without a huge amount of total evidence. In our experience, based on real-world schemata, positive evidence is a much stronger indicator than negative evidence. By incorporating this intuition into our match scores, multiple sources of evidence can be combined by the vote merger.

6.4 Vote Merger

Within Harmony, several match voters are run in parallel, each of which generates a match score for each pair of source and target elements. In this section we describe how to combine these values into a single score for each pair. We begin by describing how to merge match scores assuming each match voter were also to return an evidence score in addition to a match score (for each pair). We then describe how to merge match scores without imposing this additional requirement.

The vote merger is responsible for combining multiple match scores into a single confidence value. This combination is based on multiple factors including the weight assigned to each match voter, the amount of evidence available to that match voter, and the positive evidence observed by that match voter. For each (source element, target element) pair, the match voter generates a single confidence value.

The basic vote merging algorithm is simply a weighted average of the match scores generated by each match voter. If we assume that the weight of a given match voter v is $wt(v)$, and that the weight associated with the evidence observed by that match voter is ew_v, then the confidence score is the weighted average of match scores as follows, where V is the set of all match voters.

$$conf = \frac{\sum_{v \in V} wt(v) \times ew_v \times ms_v}{\sum_{v \in V} wt(v) \times ew_v} \tag{13}$$

When the weights associated with each match voter are equal, the confidence score is simply the weighted average of the match scores, based on ew_v. Thus, we need to determine how to compute evidence weights.

In general, the evidence weight needs to scale from zero (in the absence of evidence), to one (given infinite evidence). Thus, any function that maps te to the interval [0, 1] fulfills this conditions. For example, the following function is an analogue of equation (7).

$$ew = (1 + \frac{1}{te})^{te} \tag{14}$$

Note that equation (14) requires that we preserve multiple values for each match voter. However, the match score calculated in equation (8) is close to zero when there is little total evidence, and close to ±1 when the amount of total evidence is large. Given this observation, we use the absolute value of the match score as the evidence weight. Assuming equal match voter weights, the confidence score simplifies to the following.

$$conf = \frac{\sum_{v \in V} | ms_v | \times ms_v}{\sum_{v \in V} | ms_v |} \tag{15}$$

Intuitively, equation (15) uses the match score returned by each match voter as its weight. This simplification works because a match score of zero indicates insufficient evidence to determine if the source element and target element match[2]. A score close

[2] As a special case, if the denominator is zero, the confidence score generated is also zero.

to ±1 indicates strong evidence either in support of a match, or against a match. Thus, by scaling each match voter as described in the previous section, we can easily merge match scores based on the strength of each match score.

In our ongoing example, the bag-of-words match voter generated a match score of 0.44. The bag-of-words with thesaurus match voter generated a match score of 0.55, and a match voter based on the edit distance between "how" and "targetsource" generated a match score –0.21 (the schema element names share only the letter "o"). Based on equation (15), the final confidence score is 0.38. The bag-of-words match voters are weighted more heavily because their match scores are more decisive. Recall that the match voters can generate large positive scores more easily than large negative scores. Given the behavior of the match voter, as long as any match voter suggests a match, the final confidence score will likely be positive.

We have incorporated all three match voters and the vote merger into the Harmony integration workbench, along with the Harmony GUI. Our customers and colleagues have been using this package for roughly one year. In the next section, we report on their experiences with the tool suite.

7 User Experiences

We released the original version of Harmony (including the GUI, match engine, and integration workbench) in November 2006. Since that time, the package has been used to support several government projects. We followed up with a half-dozen Harmony users to assess the extent to which Harmony has met their needs. In this section, we describe the lessons learned from these interviews. We first enumerate the questions that we have asked. We then provide a summary of these users' interactions with Harmony, both in terms of the GUI and the integration workbench. We conclude the section with a description of how the integration workbench has simplified the integration of Harmony with BEA's AquaLogic tool.

7.1 Background

We contacted several Harmony users, of which a half-dozen provided feedback on the tool suite. In each interview we asked the following questions.

- What can you tell us about the schemata in your application domain?
- What (if any) were the benefits of using Harmony over manual integration or other tools?
- Of which UI features were you aware, and which did you use?
- What issues or limitations did you experience?
- Did you interact with the integration workbench? If so, how difficult was it to use this framework?

In almost all cases, the schemata in question were very large, containing several thousand distinct schema elements. In one case, the schemata were OWL ontologies, one of which contained nearly 100,000 concepts.

The application domains ranged widely. For example, one scenario involved mapping XML message formats to a smaller "community of interest vocabulary"—i.e., a set of terms with text definitions all directly connected to a root node. The goal of this project was not to create an executable mapping, but instead to establish a data dictionary describing the elements common in the domain, including alternate formulations of these elements. A second scenario involved mapping the same set of source schemata to a collection of target schemata using just the match engine (i.e., without human intervention) to determine which target schema best covered the source schemata. A third scenario involved aligning a small highly-technical ontology with a large general-purpose ontology. The goal of this project was to merge the technical ontology into the general-purpose ontology to provide better domain coverage. Note that none of these projects were trying to generate executable transformations to generate target instances from source instances, which is the typical motivating application for schema integration research.

7.2 Match Engine and GUI Experiences

Not surprisingly for a research prototype, the Harmony match engine was unable to handle source and target schemata containing thousands of schema elements. Because the match engine evaluates a confidence score for every possible [source element, target element] pair, the match engine was unable to generate a complete mapping matrix in less than 24 hours (and in some cases would run out of memory). This limitation stresses the importance of match algorithms that do not need to consider all possible pairs (e.g., [32]).

As a workaround, in all but one case, the users identified external tools that could partition the schemata into smaller, more manageable, pieces. From their experiences, we can draw two conclusions. First, new tools should be added to the workbench that can partition a schema into smaller sub-schemata. Second, the GUI should make it clear that only the nodes currently selected (e.g., using the sub-tree filter) would be fed to the match engine. Only one user was aware of this strategy for handling large schemata.

The users with whom we spoke did use most of the GUI filters to explore the mapping matrix. In particular, we heard that the sub-tree filter was very helpful in focusing one's attention on a particular context. This feature allowed the integration engineer to verify the proposed matches, specifically within that context because the validity of a match depended on the context.

To identify these contexts, the integration engineer used a combination of the depth filter and the confidence filter. The depth filter eliminated the low-level details, leaving only high-level concepts used to establish a context. The confidence filter identified those contexts for which good matches could quickly be identified. Thus, our intuition that schema matching is an iterative process in which the integration engineer alternates between high-level and detailed views of the problem was validated.

7.3 Integration Workbench Experiences

In three cases, the users with whom we spoke modified Harmony directly. In the first case, a new match voter was created that parallelized the generation of match scores. This match voter also discarded any score that fell below a user-defined threshold to avoid the overhead of maintaining these scores in the blackboard. Once implemented, it was trivial to add this new match voter to Harmony, largely because the interactions between a match voter and the workbench were well-established.

To support ontology alignment, the GUI was extended to introduce an additional mapping cell annotation: relationship indicates the nature of the relationship between the source schema element and target schema element. This annotation could be used to indicate that the source element was more specific than (a subclass of), equivalent to, or compatible with the target element. In effect, this annotation made each mapping cell a reified relationship linking the source to the target. It took the integration engineer roughly 40 hours to extend the GUI and to link the new tool into the integration workbench. The integration engineer responsible indicated that he was quite pleased to see that the workbench correctly saved and loaded the new annotations along with the built-in annotations.

Finally, to determine which target schema best covered the source schemata, the integration engineers needed a new tool to display a mapping matrix in summary form. This tool generates a pie chart that indicates the percentage of source elements for which a good match (confidence score ≥ 0.75), weak match (≥ 0.25), or no match was found. Implementing this tool and linking it into the integration workbench took an integration engineer roughly 20 hours.

Although our experiences are limited, we believe that the integration workbench has proven to be an effective mechanism for adding new tools to the suite. This capability is particularly important because many of our users were not interested in generating executable code. In fact, several of them reported that they were unable to use commercial schema integration tools because the only possible end product generated by these tools is executable code. Our users' needs were more varied than could be supported by off-the-shelf tools.

However, we recognize that in many cases, the goal of schema integration is to generate an executable mapping. Towards that end, we have teamed with BEA to integrate the Harmony match engine with BEA's AquaLogic tool via the integration workbench.

7.4 Matching + Mapping

In [33] we describe our efforts to combine the Harmony match engine with BEA's AquaLogic tool, which we summarize here. Briefly, AquaLogic "employs a declarative foundation to enable a user to design, develop, deploy, and maintain a framework that understands both the logical and semantic heterogeneity of data sources." In the context of the integration workbench, AquaLogic provides a graphical interface so that an integration engineer can manually indicate semantic correspondences. The tool automatically proposes mapping snippets (largely type-conversions) based on the semantic correspondences. It then assembles these snippets into an executable transformation, optionally deploying this transformation in a service-oriented architecture.

Given the potential synergy between Harmony and AquaLogic, we have begun a joint effort to combine these tools. In the resulting product, the Harmony match engine will propose candidate matches; AquaLogic is responsible for providing a graphical user interface and for generating mappings/transformations. Moreover, given a library of source schemata and a target schema, Harmony can suggest source schemata that are likely to be relevant.

To make Harmony accessible to AquaLogic, we needed to implement two new functions. The first takes, as input, a source schema element and a target schema element, and computes the mapping matrix for the corresponding schema sub-trees. The function returns the top *k* [target element, confidence] pairs for each source element such that the confidence score exceeds some threshold. (The intention is to limit the amount of information presented to the user.) Implementing this functionality using the integration workbench required only four lines of code: 1) invoke the Harmony match engine, 2) determine which confidence scores to compute, 3) filter out any results that do not exceed the confidence threshold, and 4) add the top *k* matches for each source element to the result.

The second new function allows AquaLogic to indicate which correspondences have been accepted by the integration engineer. This tells Harmony which mapping cells should not be modified by future invocations of the match engine and could potentially be used to tune the algorithmic parameters. Implementing this method required three lines of code: 1) iterate over the set of manually identified matches, 2) lookup the corresponding cell of the mapping matrix, and 3) update the confidence score for that cell.

At this time, BEA is extending their graphical interface to display the results generated by the Harmony match engine. However, the ease with which the necessary information could be extracted from the blackboard via the integration workbench offers further proof that the workbench is an effective mechanism for integrating schema integration tools and that our task model correctly captures activities common to data integration.

8 Related Work

The data integration task model is an extension [7] of our prior work presented in [9]. The improved model includes additional subtasks addressed by real integration systems and identified as being important by three experienced integration engineers. A task model of schema integration also appears in [11], but that work predates the data integration industry and does not benefit from the insights of practitioners.

In [34], Haas describes a task model similar to ours. In her model, Haas includes four basic tasks: First, the integration engineer must understand the schemata (subtasks 1–2, above). Second, the integration engineer must standardize the underlying sources. This includes establishing a standard schema that specifies the syntax, structure and semantics of the information (subtasks 3–9). She also emphasizes the importance of determining how to a) identify information that pertains to the same subject (subtask 10) and b) handle missing or inconsistent information (subtask 11). Third, the developers must specify the execution engines and produce the executable (subtask 12). Finally, the solution must be executed (subtask 13).

Of the tasks pertaining to schema integration (subtasks 1–9), most of the research, including our own, has focused on subtask 3, schema matching (e.g., [4, 6, 25-28]). Overviews of the common approaches appear in [1] and [2]. Based on Rahm and Bernstein’s hierarchy [1], the match engine (as a whole) is a composite matcher that composes the vote merger with a structure-level matcher. The vote merger, in turn, is a hybrid matcher that combines the match scores generated by a collection of element-level linguistic matchers (the match voters). However, we are aware of only one prior schema matching algorithm that exploits textual definitions [35], which uses a simple approach based on a commercial information retrieval tool. Harmony adds more sophisticated linguistic pre-processing (e.g., stemming), a thesaurus and scoring algorithms that consider the amount of evidence available. In [36] similar linguistic pre-processing techniques to ours are used, but are applied only to element names. In addition, instead of doing bag-of-word comparisons across elements of different schemata, they use natural language techniques to translate each name into a logical formula and then compare the logical formulae to perform match. This approach is complementary with techniques in use in Harmony.

Harmony provides the first GUI that supports an iterative development cycle. This GUI is the first to allow the integration engineer to filter the match results based on a variety of criteria.

The integration workbench is far from the first schema integration toolkit to adopt a modular architecture. A similar approach is used by both schema matching prototypes such as COMA++ [4] and Protoplasm [25] and commercial schema mapping tools such as those offered by IBM and BEA. For example, Protoplasm allows the integration engineer to string together match voters and vote mergers in arbitrary ways. This modularity allows the research group or commercial entity to adapt or extend their software. However, the integration workbench that is proposed in this paper is unique in that it is based on a common blackboard using open standards so that independently developed tools can interoperate.

9 Conclusions and Future Work

Data integration is a widely researched problem. However, we described ways in which enterprise data integration differs from the situations usually encountered in the research literature (e.g., documentation is widely available, instance data less so). Other pragmatic comments discussed how best to represent coding schemes so they can be leveraged by integration tools.

We also enumerated the subtasks involved in data integration, partitioned to reflect the behavior of integration engineers and the support provided by existing tools. This task analysis is intended to guide tool development and to enable comparisons across tools and integration problems.

Based on our observations and task modeling, we identified important design goals for integration tools. Specifically, we articulated the need to support all of the tasks involved in schema integration. One approach to meeting this need is to bring multiple tools to bear.

Unfortunately, assembling several tools to solve a particular integration problem is daunting. Our community needs to adopt the principle of assembling systems from modular components and integrating existing components. To facilitate tool interoperation, we proposed an open, extensible integration workbench. This architecture provides a unified view of schemata and mappings so that integration tools can more easily communicate. We believe that both tool vendors and database researchers benefit from this arrangement. We hope that this proposal will generate discussion that ultimately could lead to standards (e.g., for mapping matrices) for data integration tool interoperation.

Since our overarching goal is to improve the lives of integration engineers, our next task is to perform a usability analysis of the Harmony integration suite. We will measure the extent to which software tools save time on each of the schema integration subtasks.

Acknowledgements

We would like to thank Ken Samuel for his work developing the Harmony match voters and vote merger, Mike Carey and Sachin Thatte for their input, Jeff Hoyt and Michael Morse for their editorial comments and the reviewers for their feedback.

References

[1] Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. The VDLB Journal 10, 334–350 (2001)

[2] Shvaiko, P., Euzenat, J.: A Survey of Schema-Based Matching Approaches. Journal on Data Semantics 4, 146–171 (2005)

[3] Miller, R., Hernández, M.A., Haas, L.M., Yan, L., Ho, C.T.H., Fagin, R., Popa, L.: The Clio Project: Managing Heterogeneity. SIGMOD Record 30, 78–83 (2001)

[4] Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, MD (2005)

[5] Hammer, J., Garcia-Molina, H., Nestorov, S., Yerneni, R., Bruenig, M.M., Vassalos, V.: Template-Based Wrappers in the TSIMMIS System. In: Proceedings ACM SIGMOD International Conference on Management of Data, Tucson, AZ (1997)

[6] Doan, A., Domingos, P., Halevy, A.Y.: Learning to Match the Schemas of Databases: A Multistrategy Approach. Machine Learning 50, 279–301 (2003)

[7] Mork, P., Rosenthal, A., Seligman, L.J., Korb, J., Samuel, K.: Integration Workbench: Integrating Schema Integration Tools. In: InterDB 2006 Second International Workshop on Database Interoperability, Atlanta, GA (2006)

[8] Ullman, J.D.: Information Integration Using Logical Views. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186. Springer, Heidelberg (1997)

[9] Seligman, L.J., Rosenthal, A., Lehner, P.E., Smith, A.: Data Integration: Where Does the Time Go? IEEE Database Engineering Bulletin 25, 3–10 (2002)

[10] Ashish, N., Knoblock, C.A.: Wrapper Generation for Semi-structured Sources. SIGMOD Record 26, 8–15 (1997)

[11] Batini, C., Lenzerini, M., Navathe, S.B.: A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys 18, 323–364 (1986)

[12] Cluet, S., Delobel, C., Siméon, J., Smaga, K.: Your Mediators Need Data Conversion! In: SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, Seattle, WA (1998)

[13] Florescu, D., Levy, A.Y., Mendelzon, A.O.: Database Techniques for the World-Wide Web: A Survey. SIGMOD Record 27, 59–74 (1998)

[14] Pan, A., Raposo, J., Álvarez, M., Hidalgo, J., Viña, Á.: Semi-Automatic Wrapper Generation for Commercial Web Sources. In: Engineering Information Systems in the Internet Context, Kanazawa, Japan (2002)
[15] Papakonstantinou, Y., Gupta, A., Garcia-Molina, H., Ullman, J.D.: A Query Translation Scheme for Rapid Implementation of Wrappers. In: Ling, T.-W., Vieille, L., Mendelzon, A.O. (eds.) DOOD 1995. LNCS, vol. 1013. Springer, Heidelberg (1995)
[16] Popa, L., Velegrakis, Y., Miller, R., Hernández, M.A., Fagin, R.: Translating Web Data. In: VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, Hong Kong, China (2002)
[17] Fagin, R., Kolaitis, P., Miller, R., Popa, L.: Data Exchange: Semantics and Query Answering. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572. Springer, Heidelberg (2003)
[18] Fernandez, M.F., Tan, W.-C., Suciu, D.: SilkRoute: Trading between Relations and XML. In: Ninth International World Wide Web Conference, Amsterdam, The Netherlands (2000)
[19] Rys, M.: Bringing the Internet to Your Database: Using SQL Server 2000 and XML to Build Loosely-Coupled Systems. In: Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany (2001)
[20] Wyss, C.M., Robertson, E.L.: Relational Languages for Metadata Integration. ACM Transactions on Database Systems 30, 624–660 (2005)
[21] Goh, C.H., Bressan, S., Madnick, S.E., Siegel, M.: Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. ACM Transactions on Information Systems 17, 270–293 (1999)
[22] Sciore, E., Siegel, M., Rosenthal, A.: Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems. ACM Transactions on Database Systems 19, 254–290 (1994)
[23] Koudas, N., Sarawagi, S., Srivastava, D.: Record Linkage: Similarity Mesaures and Algorithms. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, IL (2006)
[24] Johnson, T., Dasu, T.: Data Quality and Data Cleaning: An Overview. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, CA (2003)
[25] Bernstein, P.A., Melnik, S., Petropoulos, M., Quix, C.: Industrial-Strength Schema Matching. SIGMOD Record 33, 38–43 (2004)
[26] Do, H.H., Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. In: VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, Hong Kong, China (2002)
[27] Madhavan, J., Bernstein, P.A., Rahm, E.: Generic Schema Matching with Cupid. In: VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, Rom, Italy (2001)
[28] Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm. In: Proceedings of the 18th International Conference on Data Engineering, San Jose, CA (2002)
[29] Brickley, D., Guha, R.: RDF Vocabulary Description Language 1.0: RDF Schema. In: World Wide Web Consortium (W3C®) (2003)
[30] Stenbit, J.P.: Department of Defense Net-Centric Data Strategy (2003)
[31] Ilyas, I.F., Markl, V., Haas, P.J., Brown, P., Aboulnaga, A.: CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France (2004)

[32] Mork, P., Bernstein, P.A.: Adapting a Generic Match Algorithm to Align Ontologies of Human Anatomy. In: Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, Boston, MA (2004)

[33] Carey, M.J., Ghandeharizadeh, S., Mehta, K., Mork, P., Seligman, L.J., Thatte, S.: AL$MONY: Exploring Semantically-Assisted Matching in an XQuery-Based Data Mapping Tool. In: International Workshop on Semantic Data and Service Integration, Vienna, Austria (2007)

[34] Haas, L.M.: Beauty and the Beast: The Theory and Practice of Information Integration. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353. Springer, Heidelberg (2007)

[35] Clifton, C., Housman, E., Rosenthal, A.: Experience with a Combined Approach to Attribute-Matching Across Heterogeneous Databases. In: Data Mining and Reverse Engineering: Search for Semantics, IFIP TC2/WG2.6 Seventh Conference on Database Semantics (DS-7), Leysin, Switzerland (1997)

[36] Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-Match: an Algorithm and an Implementation of Semantic Matching. In: Bussler, C.J., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053. Springer, Heidelberg (2004)

Using ORM-Based Models as a Foundation for a Data Quality Firewall in an Advanced Generation Data Warehouse (Extended Version)

Baba Piprani

SICOM, Canada
babap@attglobal.net

Abstract. Data Warehouses typically represent data being integrated from multiple source systems. There are inherent data quality problems when data is being consolidated in terms of data semantics, master data integration, cross functional business rule conflicts, data cleansing, etc. This use case demonstrates how multiple Object Role Models were successfully used in the establishment of a Data Quality Firewall architecture to define an Advanced Generation Data Warehouse. The ORM models represented the realization of the 100% principle in ISO TR9007 Report on Conceptual Schemas, that were then successfully transformed into attribute-based models to generate SQL DBMS schemas. These were then subsequently used in RDBMS code generation for an 100% automated implementation for the Data Quality Firewall checks based on the described advanced generation Data Warehouse architecture. This same Data Quality Firewall approach has also been successfully used in implementing multiple web based applications, characteristically yielding a representative savings of 35-40% savings in development costs. The intent of the paper is to explain how ORM can be successfully used in the eventual implementation of a data quality firewall, including the details of the architecture of the data quality firewall in an enterprise data warehouse to enable data quality assurance. It is not within the scope of this paper to address the use or merits of alternative modelling paradigms in this regard.

Keywords: data quality firewall, advanced generation data warehouse, data quality assessment, data quality threshold, conceptual schema, ORM.

1 Introduction

Data Warehouses typically represent data being integrated from multiple source systems. The term data warehousing generally refers to combining the data from many different databases across an entire enterprise, into a data warehouse database.

The effort in bringing together data from heterogeneous sources in itself requires enormous amounts of time and resources. Creating a multi-subject, consolidated information store requires reconciling the different data models involved.

Data quality is one of the most overlooked challenges. It is not as simple as dumping all the data into one big bucket without an integration scheme and voila---you have a data warehouse.

© Springer-Verlag Berlin Heidelberg 2008

1.1 Common Approaches towards Development of Data Warehouses

There have been several approaches that have been described for the development of data warehouses---from simple hastily assembled data marts, to poorly normalized collection of SQL tables requiring extensive navigation, and to attribute based data models resulting in highly normalized SQL tables---all describing the makeup of the data warehouse with a build-it-and-they-will-come attitude.

The score card for success rates on the data warehouses based on the above approaches remain in the dismal to mediocre range, respectively, based on which approach can guarantee data quality in the data warehouse.

Data warehouse success measures, or more appropriately stated, “failure rates or limited acceptance rates” have been in the range of 92% (back in late 1990s) to greater than 50% for 2007 [1]----a dismal record indeed.

Just like any engineering initiative, improperly designed data warehouses tend to experience significant rework to keep up with the changing business needs. The question to be raised is “why is there not a solid but flexible infrastructure in place at the foundation of the DW effort?”

Let’s face it. Any piece of computer code can be “made” to work in some fashion, eventually. In general, these DW initiatives are simply thrown together and “made” to work after several iterations. The costs of the re-work and infrastructure costs may surpass the initial budgeted costs by 2 to 3 times, indeed since these costs essentially increase exponentially.

While lack of data quality appears to be the major culprit being singled out by Gartner [1], let us examine some of the issues involved in the “failure” of a data warehouse project.

A data warehouse project is yet another Information Technology project, albeit specialized in the sense that it is responsible for “semantic integration” of multitude of “business concepts” within an organization.

The meaning of the term “failure” has been amplified by the Standish Group [8] with the interpretation that the “success” of the project refers to the project being completed on time and on budget with all features and functions as initially specified; or the project “challenged” referring to the project is completed and operational but over-budget, over the time estimate, and offers fewer features and functions than originally specified; and “impaired” referring to the project being cancelled at some point during the development cycle.

According to the Standish Group’s 2003 CHAOS report, 15% of the IT projects “failed” and another 51% were considered “challenged”, while 82% of the IT projects experienced significant schedule slippage with only 52% of required features and functions being delivered. For 2004, results show that 29% of all projects succeeded i.e. delivered on time, on budget, with required features and functions; 53% were “challenged”; and 18% failed i.e. cancelled prior to completion or delivered and never used. A staggering 66% of IT projects proved unsuccessful in some measure, whether they fail completely, exceed their allotted budget, aren't completed according to schedule or are rolled out with fewer features and functions than promised [7].

Several more data warehouse and IT project failure rates and metrics are available in [9].

Let us examine some of the specific factors contributing to data warehouse pitfalls.

In its simplest incarnation, the data warehouse incorporates a direct feed from multiple sources, and often using surrogate keys or system generated identifiers regardless of the source. This “consolidation” results in the data, especially duplicate data, being assembled in a “shipping trunk” like environment where, it’s all there, but, requires complex search algorithms for retrieval---since there is no model. This dumping of the data produces reconciliation nightmares.

Other approaches use multiple SQL-unions producing near-Cartesian joins from a set of SQL tables that are grouped together in a “loosely-modelled” environment. This form of environment generally does not contain any integrity rules or data quality aspects involving data cleansing.

These forms of data warehouse models generally result in dependence on complex navigation and heavy-duty number crunching requirements.

Another factor contributing to a data quality setback is the dependence of data integrity of the source system models. It is “assumed” that the source data is sacrosanct and consistent. This may be so (although taken with not a pinch of salt but a large bag of salt), but data brought together from multiple sources more often than not, is inconsistent. This is simply because there could be conflicting business rules across system models pertaining to the shared or overlapping data.

1.2 Taking Advantage of the Disciplined Process for Deriving Business Rules in ORM vis-à-vis Data Quality

The situation is exacerbated when the source system data models are attribute-based, because, in general, a good portion of the business rules that are “routine” in ORM based models [12] are simply non-existent in the attribute based data models---simply because there is not a defined discipline or process for the ‘discovery’ of these business rules. ORM [14] on the other hand, has a disciplined approach to be used so that these hidden business rules (usually the very obvious ones) surface and are formally declared in a semantic model.

This paper does not deny that good models cannot be built using other attribute based modelling approaches like ER, UML etc., nor does this paper purports to weigh or stack one modelling paradigm against the other. That topic is out of scope of this paper.

Instead, this paper focuses on the usage and experience of using NIAM [13] and ORM over the past 10 years on a multitude of Data Warehouse projects that have had a 100% success rate (as compared to over 50-92% typical failure rates), at times totally replacing the incumbent attribute based ER models, including popularly espoused Data Warehouse modelling approaches---essentially rescuing these projects from certain failure!

The reader is referred also to the ISO TR9007 Conceptual Schema report [2], that took an example scenario and applied Entity Relationship, NIAM, and Interpreted Predicate Logic approaches to solve this scenario so as to demonstrate some measurability in terms of semantic and expressiveness match of the modeling approaches. The solutions clearly demonstrated the ability of NIAM in supporting a much larger number of axioms and constraints in its modelling paradigm than does the ER approach.

Focusing on achieving a higher level of data quality for the enterprise data warehouse model, Table 1 below highlights how data quality characteristics derived from [10] are addressed by NIAM and ORM based modelling:

Table 1. Data Quality Characteristics vis-a-vis NIAM and ORM

Data Quality Characteristic	Description	NIAM/ ORM Example Application
Accuracy	Degree of agreement between a set of data values and a corresponding set of correct values	he population diagrams with sample data value populations related in a fact type that was associated with a business "concept" provided a less cluttered view to be able to discourse with the user, e.g. helped establish min and max ranges for data values, helped identify inconsistencies between risk measures used in a sub business domain and another like business domain at the lower granularity level of a fact type because of a common derived fact type
Completeness	Degree to which values are present in the attributes that require them	Nullability vs. totality as applied to a concept based fact type along with incomplete sentence populations in a population diagram assisted in helping users complete their vision of the business process, e.g. helped identify missing information required for calculation or derivation of a risk measure, because the data was being imported from another application. The derived fact type in NIAM ORM required the use of a missing fact that was not part of a particular data stream being imported
Consistency	Agreement or logical coherence among data that free them from variation or contradiction	Using a "business concept" based focus, natural language sentences with sample real data values formed the basis of agreement. Expanding the natural language sentence constructs with variations and contradictions in population that broke business rules helped confirm the absurdity of the data quality and requirement for a strong business rule e.g. death date cannot be before birth date
Relatability	Agreement or logical coherence that permits rational correlation in comparison with other similar or like data	Using a "business concept" based focus, natural language sentences with sample real data values formed the basis of agreement and logical coherence amongst the users to identify pattern similarities, e.g. risk measures used for different insurance companies based on insurance company groupings turned out to be identified as being common across.
Timeliness	Data item or multiple items that are provided at the time required or specified	The granular level of the NIAM and ORM models provided easy cross correlation with data sets to be examined for timeliness in terms of availability and for the derivation of sequencing requirements for Extract Transform and Loading of data warehouse data. This aspect directly influenced the Audit and Control Model
Uniqueness	Data values that are constrained to	This aspect was built into every sentence type and population diagram example for each concept and fact

Table 1. (*continued*)

	a set of distinct entries—each value being the only one of its kind	in NIAM and ORM. In effect, this aspect was one of the easiest for the user to comprehend and relate to, e.g. "I want to list only once the cities I have visited", which makes the combination of the person and city unique in the population diagram
Validity	Conformance of data values that are edited for acceptability—reducing the probability of error	The lower level granularity of a fact type associated with a business concept and the population diagrams helped define the acceptability and validity of data, In particular, the requirement of a "business key" for all artificial identifiers (surrogate keys) to enable strict enforcement of the availability of correct data, e.g. the validation of a financial return from an institution with multiple versions being selected for processing. This aspect also helped define the audit and control architecture details

In other words, NIAM and ORM contributed extensively to mitigating the risks in the approach to data quality assurance and definition of the data warehouse enterprise model.

Citing the Success/Failure profiles in the Standish Group Report 1995 [10], it is interesting to see how NIAM and ORM addresses the major factors involved in the "success' projects, "challenged" projects and "Impaired" projects as per Table 2 below:

Table 2. IT Project factors that contribute to success and failure and the influence of NIAM and ORM

IT Project Factors	**Success %**	**Challenged %**	**Impaired %**	**How NIAM ORM influences this – examples**
User Involvement	15.9	-	-	NIAM and ORM use natural language sentences to arrive at a fact type, including population diagrams. The users are the main players involved in defining the natural language sentence, supplying values to the sentences, and to pass correctness approval and including variations/violations of those sentences. There is no concept of normalization, keys, nulls etc. The sentences relate one or more pairs of objects with realistic data values to form a business statement of a fact. All other business rules are against this established statement of the business fact.
Executive Management Support	13.9	-	9.3	Because the communication medium between the users (including management) is natural language sentences, it is easy to see why executive management support is readily available. In a matter of speaking, the higher level objects and subject area domains stated in natural language facts will be readily identifiable and enables management to easily relate to the business scope. Citing an example in Canada of a situation with over 1000 objects involved, it was easy to isolate 16 major sentence types

Table 2. (*continued*)

				that would relate these mainstay objects---which essentially constituted the scope of the system to be defined along with examples of some business facts and rules. The non-IT management (1 star military generals and assistant ministers) were able to recognize and validate the scope of the requirements and provide support. It was in this project that over 50 military staff personnel (non-IT) were able to define the entire functional specifications with the help of NIAM and ORM analysts as scribes. Executive management are able understand and appreciate that these NIAM and ORM fact statements portray the facts of the business (i.e. WHAT of the business) that are least susceptible to change vs. business processes (HOW) which can be heavily influenced by any change in business related factors e.g. regulatory, compliance etc.
Clear Statement of Requirements	13.0	-	-	The natural language statements and population diagrams are able to very quickly help define the boundaries and the precise fact statements that are involved, which enables the pegging down of the scope and objectives. This would clearly demarcate the “hand waving” facts that cannot be defined vs. the precise formulation of the fact statements. In a recent example financial analysis project for BASEL II, the attribute based models (ER) had vague requirements in several areas, which duplicated the same dimensions. The use of NIAM and ORM quickly identified the haziness involved and the similarities because of the requirement to define a natural language sentence for those connected facts.
Clear Vision and Objectives	2.9	-	-	The involved business objects and subject area domains stated in natural language facts are readily identifiable and enables management and users to easily relate to the business scope using natural language sentences with realistic values, and population diagrams to help identify business rules
Lack of user Input or involvement	-	12.8	12.4	Users are heavily involved in the definition of natural language sentences, providing realistic sample populations, filling in population diagrams, and validating the business rules using business language terms and statements, at the same time validating in their mind their own business process models---which are certainly not 100% defined at the time of derivation of functional specifications. In a particular scenario, two pairs of similarly classed fact types were shown to the user across separate domains, it was determined that it was the same analysis process that was being performed but across different domains of data within the same insurance sector.
Incomplete Requirements & Specifications	-	12.3	13.1	Users are heavily involved in the definition of natural language sentences, providing realistic sample populations, filling in population diagrams, and validating the business rules using business language terms and statements, at the same time validating in their mind their own business process models. Users are quick to recognize any

Table 2. (*continued*)

				incomplete sentence connections or identify inability to achieve required sets of fact statements. This item directly relates to the inability of a programmer in being able to navigate across an SQL database due to lack of the ability to join or access required data because of missing functionality. NIAM and ORM are able to trap this in the analysis phase.
Changing Requirements & Specifications	-	11.8	8.7	Since NIAM and ORM essentially model the facts of the business via natural language sentences resulting in established fact types and business rules on the facts themselves, this in itself isolates the data structures derived from NIAM and ORM from major changes. In other words, a NIAM or ORM based model is able to more readily portray the facts of the business in natural language terms, and that these business facts (WHAT) are reasonably insulated from change. It is the HOW of the business process that is more involved in the change. e.g. sequencing, or changing of a criteria set involved in a checkout (but consisting of an already established fact or the requirement to add a new fact).
Unrealistic Expectations	-	5.9	9.9	The natural language sentences of the NIAM and ORM fact types enable the identified boundaries and scope of the available facts in the enterprise model. Facts that are unable to be derived or formulated are readily identified. Sentence types that do not make sense are also highlighted as being invalid at the start instead of discovering this during the coding process.
Unclear Objectives	-	5.3	-	The natural language sentences of the NIAM and ORM fact types enable the identified boundaries and scope of the available facts in the enterprise model. Facts that are unable to be derived or formulated are readily identified. Sentence types that do not make sense are also highlighted as being invalid at the start instead of discovering this during the coding process.
Controllable via Enterprise Modelling	**45.7**	**48.1**	**53.4**	**The basic procedures and processes involved in the derivation of the enterprise model using NIAM and ORM modeling paradigm are able to influence 100% of the controllable project factors related to the enterprise model.**
Project Management related + Other	**54.3**	**51.9**	**46.6**	Outside the scope of modelling the enterprise model
Total	**100**	**100**	**100**	

Perhaps it is something to do with the engineering type nature of ORM that opens up the vista for the declaration of business rules (and the facts of the enterprise model), which of course, need to be transformed and mapped to attribute based or driven SQL schemas for implementation. I would not even venture to track the multitude of mappings required to transform and implement these rules using Object Oriented or XML based database technologies.

In the author's 28-year experience in teaching and implementing over several dozens of NIAM and ORM based data models, the declaration and conformance to a business rule set has been the most critical factor contributing to a successful implementation of an application.

The absence of these business rule declarations in most of current implementations essentially contributes to the instability of the Data Warehouse when data is brought together from heterogeneous sources.

1.3 100% Principle in ISO TR9007

With work being contributed from over 45 experts worldwide beginning in 1978, the ISO TR9007:1987 Technical Report on Concepts and Terminology for the Conceptual Schema and the Information Base [2], was a landmark effort in formalizing the role and content of a metadata driven implementation. Examining the guidelines for the conceptual schema, the report focused on the importance of declaring 100% of the rules of the Universe of Discourse. The Reference Model of Open Distributed Processing [3] took this a step further through the declaration of multiple viewpoints that are involved in implementations for a distributed environment. As seen in the conceptual schema report [2], a NIAM/ORM model comes closer to be able to fulfill the 100% principle declaration in a semantic model than any other commonly used modeling approach (with the exception of limited Interpreted Predicate Logic (IPL) based trial implementations). See the solved scenario case in ISO TR9007 [2] that is solved by NIAM, ER and IPL modelling approaches and the associated business rules that can be portrayed in each modelling paradigm.

Why is the 100% principle important for the data warehouse? Along with the declaration of an enterprise model that would depict the "Conceptual Schema" of an enterprise, the declaration of business rules is a requirement for the completion of the conceptual schema (Conceptualization principle + 100% principle). This conceptual schema then is the blueprint for any further implementation mechanisms for the enterprise data warehouse. This means that all the declared rules (in terms of propositions of the conceptual schema) will require to be mapped to the respective SQL, Object Oriented or other coding paradigms to be incarnated in their respective universes. It is here that a judicious evaluation needs to be conducted so as to ensure that a maximally optimizable vehicle is chosen for the enforcement of the business propositions or rules declared in the conceptual schema. See Chapter 3 of ISO TR9007 [2] on the relevance of mappings between the conceptual schema and the information processor for the implementation of the propositions or business rules. By maximally optimized, it is meant that all the rules of the conceptual schema must be enforceable by the DBMS or central kernel processor via rules declared at the information processor enforcer schema level---in this case the database schema level---and certainly not in the application code.

This would eliminate the need to write any hand-crafted application programs to process any business rules for the data, and instead enable the automated generation of the code modules---whose only function in life would be to pass data from the data store to the user/agent and vice versa, i.e. a no-brainer interface that can be generated.

In one case of a successfully running data warehouse real life scenario for a financial application designed using NIAM and ORM and transformed into the in-house toolset ERWIN using Oracle and SQL, there are over 400 tables with nearly 3000 business rules that are declaratively enforced by the SQL schema. Nearly half a million lines of code is generated every night for a data warehouse run that handles the interface, including loading and refreshing. A first pass generates the required metadata, and the second pass uses this generated metadata in conjunction with other business rules metadata to generate the interface programs. This project was achieved ahead of schedule, on time within budget and was salvaged from an incumbent attribute based model based schema including star schemas involving several surrogate key identifiers.

It is this concept of 100% principle + Conceptualization principle together formulate the fundamental basis and content of the advanced generation data quality firewall against which measures and probes can be introduced to evaluate the sanity and quality of the data presented to the data quality firewall.

The process of combining databases in an integrated DW involves semantic integration of information from multiple sources and calls for an environment that is able to bring together not only common data representations, but addresses the data's meaning and its relationships to other data and information, including associated business rule propositions.

In other words, there needs to be a conceptual schema for the data warehouse to describe the enterprise model.

1.4 Implementation via ISO 9075 Database Language SQL

In the pre-DBMS days and in the early days of DBMSs, almost 100% of the business rules were written into the application code. With a typical early DBMS (hierarchical, network, CODASYL era), only 20% or so of the business rules could be enforced through the DBMS. Today, with a strong ISO SQL [4] complement, 100% of the rules are directly enforceable by a fully conforming ISO SQL DBMS. But in reality, many of the SQL DBMS implementers do not yet fully conform to all the available features in ISO SQL, leaving us with a 'good-enough-nearly-100%' enforcement of these business rules translated into native declarative SQL integrity rules, along with some overhead of centrally enforceable procedural code in the form of triggers and user defined functions.

ISO SQL today is able to declaratively implement the majority of the ORM declared data business rules when converted to an attribute based data model. This of course is watered down to the ability of the current RDBMSs to implement a subset of the ISO SQL declarations, based on their adoption of the ISO SQL standard.

So the question was raised by the author as to why these principles cannot be used to define a foundation architecture for a data warehouse? The task here was to define an integration architecture to set the necessary standards for data integration, consistency and a topology to support dynamic changes in the business environment.

1.5 An Advanced Generation Data Warehouse Architecture

The result was an advanced generation data warehouse architecture, which has as its fundamental data model defined using ORM, and navigates the information flows through a series of Data Quality Firewalls.

Of course, the implementation is in RDBMS environment via attribute based models feeding a rich SQL based schema set of declarations. The implementations generally resulted in a maximally automated environment using dynamic SQL for the modules constituting the handling of the flow of application data. Since 100% of the rules were declared and implemented in the SQL schema environment (with minimal support from supporting coded modules in the form of triggers and user defined functions), the code for information flow interface applications was automatically generated.

The foundation for the Advanced Generation Data Warehouse Architecture consists of the 6 building blocks as in Table 3:

Table 3. Building Blocks for an Advanced Generation Data Warehouse

#	Building Block	Description	How Achieved
1	Data Quality/ Data Validation	Inspect data for errors, inconsistencies, redundancies and incomplete information;	Declare a strong ORM based data model with business rules, transform to an attribute based SQL schema, establish Data Quality Firewall, and validate data for conformance to business rules
2	Data Correction	Correct, standardize and verify data	Declare an ORM based mapping and transformation schema to assist the data quality firewall, transform to an attribute based SQL schema and Initiate correction process to data where necessary, or improve/modify return content
3	Data Integration	Match, merge or link data from a variety of disparate sources	Declare an ORM based semantic data model, transform to an attribute based SQL schema and consolidate the different data across the various classes of data from multiple sources to a unified set ensuring data integrity through the data quality firewall
4	Data Augmentation	Enhance data using information from internal and external data sources where appropriate and necessary	Declare an ORM based interface data model with dynamic business rules for behaviour, transform to an attribute based SQL schema, and establish event driven business processes triggered through schema instance populations
5	Data Composition	Data requirements change in response to external changes, trends in industry, or governance	Declare an ORM based data model with dynamic business rules for behaviour, transform to an attribute based SQL schema, address changes via event driven business processes triggered through schema instance populations
6	Persistent Data Chronology	Collect history and control data integrity over time	Declare an ORM based temporal data model, transform to an attribute based SQL schema, and maintain a time series of data for history, and to track and maintain changes of returns over time

1.6 Constituent Data Models of the Advanced Generation Data Warehouse—the ORM and Attribute Based Data Models

At the heart of the advanced generation data warehouse is an ORM based data model. This ORM based model is transformed to an attribute based data model, and subsequently defined in an SQL schema.

The resulting attribute based data model contains the logical data structures of the business representing the abstraction of the facts, relationships and includes all the business rules as declared in the ORM model.

The resulting attribute based data model as derived from the rich semantic ORM model reflects the logical integration of the selected applications or functions into a normalized-to-the-highest level model---using a combination of Third Normal Form, Boyce-Codd Normal Form, and the Fifth Normal Form---in order to materialize a maximally optimized database schema.

This normalized model essentially contains a mix of normal forms (3rd, 4th, 5th and the Boyce-Codd Normal Form) that represents an enterprise's business data model in a neutral form, optimized to house strong semantics along with a mix to achieve a balance towards implementation efficiencies that does not include denormalization.

The thrust here is to eventually achieve the highest degree of the declaration of schema level statements at the physical level that map to the business rules contained in the ORM business model, with the least number of procedural declarations that may result in stored procedures or triggers to enforce these business rules. In other words, maximize the schema declaration statements and minimize the procedural declarations.

Normalization to the highest degree or at least the 3rd normal form is mandatory to ensure proper enterprise level integration of the various source system data semantics.

1.7 Constituent Data Models of the Advanced Generation Data Warehouse—the Physical SQL Data Model

In the advanced generation data warehouse physical data model, the ORM-derived logical model entities are mapped to corresponding tables; attributes are mapped to corresponding columns. The relationships and business rules are mapped to appropriate primary key, unique constraint, foreign key, CHECK, and default column value constraints. Other remaining non-directly-mappable business rules are implemented via SQL stored procedures, user defined functions or triggers.

The data warehouse physical model may consist of additional tables, pre-formed queries or procedures and, additional constructs to support the data warehouse architecture, especially to support data cleansing and data scrubbing activities in a tight data quality controlled environment.

The data warehouse physical database schema DDL will partially be the product of automatic generation of schemas using some data modelling tool's forward engineering facilities. Additional constructs may need to be added to the generated models. These may include additional access path indexes, temporarily disabling constraints and the like.

2 The Advanced Generation Data Warehouse Framework

Figure 1 below shows a Data Warehouse Framework that defines the components involved in the Advanced Generation data warehouse.

The source system data is extracted into a staging area that simply mirrors the data structures of the source applications. The data structures of the staging area should not contain any integrity constraints and may represent a full or subset of the source base tables or source viewed tables from which the data is being extracted. The source-to-staging extract generally undergoes some data transformation or selection, making the data ready for integration into the data warehouse

Data then moves from the staging area to the data warehouse area, again with possible data transformation or selection, but this time the target data structures represent the normalized integrated data warehouse data model including a full set of integrity constraints resulting from ORM based business rules data model and analysis.

In selected cases, data can be extracted directly from the source area into the data warehouse area.

The integrated data from the data warehouse is then extracted into data marts, again undergoing the necessary transformation and selection, to suit the Business Intelligence data requirements.

No other extraction or transformation paths other than shown in the advanced generation data warehouse architecture, either explicit or implicit, are permitted.

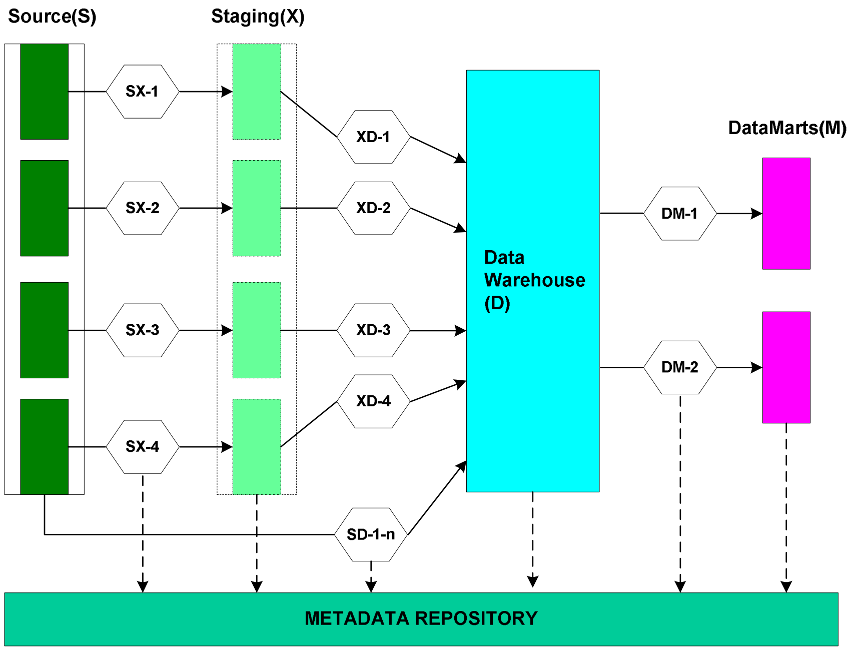


Fig. 1. Advanced Generation Data Warehouse Architecture

Considering there are several mappings involved amongst several components like source systems, staging areas, the ORM based logical data model, Data Warehouse, and Data Marts, it is necessary to establish a data warehouse framework that needs to be adopted and implemented as shown below.

Legend:
S - Source Systems
X - Staging Area
D - Data Warehouse
M - Data Marts

Mapping Schemas are shown as:
SXn: Source to Staging Area Mapping for Source system <n>
XDn: Staging Area to Data Warehouse Mapping for Staged system <n>
DMn: Data Warehouse to Data Mart Mapping for Data Mart <n>
SD1-n: Source to Data Warehouse Mapping for Source Systems 1to <n>
The advanced generation data warehouse architecture positions the various components of the data warehouse in relationship with the associated mappings.

2.1 Data Warehouse - Transformation Mappings and Rules

To enable the data warehouse populations and to make them operational, it is necessary to maintain and track the mappings between each of the source-target model pair components.

The use of a Meta Data repository that has built in mechanisms is encouraged for providing several of the cross mappings between repository objects and data lineage.

However, specific mappings need to be designed for cross model mapping via mapping schemas, which include the associated rules for selection, filtering and transformation. These mapping schema tables need to be defined in the repository database and populated accordingly from the row metadata values of the inputted data models. These of course, are driven and designed using ORM models.

Mapping necessarily takes the following forms:

a. 1:1 (one to one)
b. 1:n (one to many)
c. n:1 (many to one)
d. m:n (many to many)
e. 0:1 (zero to one)
f. 1:0 (one to zero)

where the left-hand side represents the source, the right hand side represents the target. Examples are, a column say aircraft_id in the staging area may be mapped to 2 tables in the holding area e.g. a given column and row in the ALL_AIRCRAFT table and its sub-table CIVIIAN_AIRCRAFT (1:n). Another example, a source row may contain legal names, carrying-on-business name, incorporation name for an organization which are all located in the same row, and may map to just one column in a normalized table in the holding area in a alias_name_id with a name_type_code containing types ‘legal’, ‘carrying-on-business’, ‘incorporation’ (n:1).

In other words, data from the source system may realize itself in the target system via a transformation applying the above rules, plus, any accompanying filtering criteria on each.

The mapping schema definition must represent a generic specific source-target combination and be insulated from any particular instance of a specific source-target combination. In other words, one mapping schema definition should be able to contain one or more source-target combination mapping value instantiations and not hard-coded to contain a specific mapping instance.

In Figure 1, mapping schema SX1 represents the mapping between source system to staging area for source system 1. Mapping schema XD1 represents the mapping between the staging area for source system to Data Warehouse for source system 1.

Mapping schema SD1 represents the mapping between Source system to Data Warehouse for source system 1.

The Metadata Repository is logically portrayed as containing the Data Warehouse metamodel within the repository's native data model, along with supporting model schema structures like the mapping schema, transformation schema, data cleansing schema etc.

The metadata repository in effect, represents the information schema of the Data Warehouse and its associated components. The mapping schemas contain the mappings between the various components. The physical realization of the metadata constructs in the extended information schema tables can then be suitably materialized using a RDBMS.

2.2 Mapping Source System to Staging Areas

It may not be practical or feasible to simply capture the data models for all existing application systems as a stand-alone task.

The opportunity to capture the source system data model components will present itself more justifiably when a candidate source system has been selected to provide source data for the BI application in the advanced generation data warehouse, or for review to derive an Enterprise Data Model.

ORM driven attribute data models are to be defined for the portion of the source system that is of interest for data extraction, and for the staging area. In addition, the associated mapping criteria are to be established between the two.

In general, the staging area data model will in general, be a full source system model or a subset of the source system mode, more typically the latter. The table structures of the staging area would normally be representative of the source table structures. The data sizes and data formats of the attributes of the source system will be maintained accordingly.

In some cases, particularly involving ERP based systems, the source system data may be available through a pre-defined view. The staging area may then either reflect the data structure definitions as per the pre-defined view or a subset thereof, or another view involving pre-joined data. In these cases, it is best to define the staging area as being representative of the un-joined data structures, providing more freedom for any debugging that may be necessary to unravel undesired Cartesian products.

The mapping between source-to-staging will be the driver for the Extract Transformation Loading (ETL) facility to extract data from source for populating into the staging areas.

Staging area tables must be defined as SQL tables, following source system naming conventions. This will allow for the classification of these tables as external tables, and be treated as Foreign Tables as per SQL2003 in the future when these features are available in an SQL implementations.

The source-to-staging area mappings are to be defined and declared in the repository, with the staging area schema definitions being declared in an SQL-schema.

2.3 Mapping Staging Areas to Data Warehouse

The mapping between the staging area and the integrated data warehouse environment will consist of CAST definitions for data type transformations, cleansing algorithms, or any filtering criteria where applicable. The mapping schema should contain entries for the mapping between the source staging area and the target integrated data warehouse area.

2.4 Data Filtering, Data Transformation Rules

The mapping row entries in the mapping schema should contain any CASTing rules i.e. transformation or conversion of data types in SQL2003 format, any filtering or merging/splitting criteria in SQL2003 syntax, to take into account the different mapping forms listed above.

Data filtering, data transformation rules are applicable at all stages of transfer. Specific rules may be applicable to data from the source system to staging area tables.

2.5 BI Requirements - Data Mappings from DW to Data Marts

Mappings between the DW data model and the data mart models should also be maintained.

The following information is relevant for the use of Business Intelligence purposes and should be defined for each BI requirement:

1) Business descriptions of target data elements, e.g. for report columns, screen columns
2) Description of transformations from the data warehouse to data marts, e.g. algorithms, consolidation, aggregation
3) Description of data mart data load validation and control parameters e.g. control dates, min-max boundaries
4) Business descriptions of preformatted reports

This set of metadata may be difficult to deliver in environments where intelligent interfaces amongst metadata repository schemas and Business Intelligence tool schemas do not exist.

2.6 Cleansing and Scrubbing Rules

Data cleansing essentially is concerned with ensuring the integrity and quality of data. These include rules for restructuring records or data elements, column value decoding and translation, rules for supplying missing data element values, and rules for data integrity and consistency checking.

These rules need to be maximally defined in the DW schema declarations. The following section addresses the issue of enforcement of data quality.

3 Data Warehouse Data Quality

Data quality and data integrity form the essential foundation of a successful data warehouse. The entire data warehouse initiative can be severely jeopardized in the absence of adequate measures being taken to protect the DW from data corruption. Experience has shown that an ORM driven attribute based data model is one of the most critical factors in the definition of data quality aspects for a data quality firewall.

The advanced generation DW Framework provides a stable platform to enable sustainable mechanisms for the establishment and maintenance of mappings. This same framework is now used to establish and position a data quality firewall as shown in Figure 2 below.

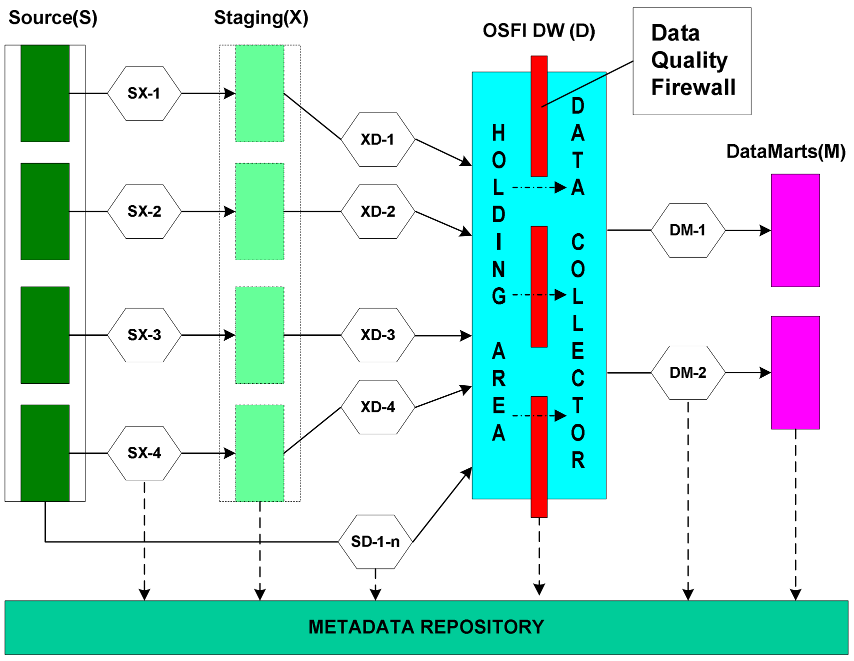


Fig. 2. Data Quality Firewall in the advanced generation DW architecture

3.1 An Advanced Generation DW Data Quality Firewall

As seen in Figure 2 the Advanced Generation Data Warehouse Architecture contains a Data Quality Firewall as its backbone for establishing data quality and data integrity by the judicious use of two data areas---a holding area and the data collector area within the data warehouse.

The holding area does not enforce any business rules or SQL integrity constraints. Data is simply transformed into the holding area from the staging area tables.

The data collector area contains schema declarations for all applicable integrity rules, as portrayed by the spirit of a 3rd normal form database structure, which essentially forms the backbone of the Data Quality Firewall.

3.1.1 Holding Area

The role of the holding area is to act as a transit point for the data to migrate from the staging area to the data collector area.

Each row of data being loaded into the data warehouse is first brought into the holding area tables from the staging area. Each row is given a unique row identity, and is tagged by a batch run identifier. There are no integrity checks other than data type checks made at this time for the holding area.

Any required transformations or selections or re-formatting to meet the data structuring requirements of the fully normalized data structures of the collector from the incoming source data structures as in the staging area, are done en route from the staging area into the holding area. Additional transformations may also be conducted while the data is in the holding area for convenience as necessary.

The objective of a row being in the holding area is that it is being prepared for a full data integrity check cycle at the data quality firewall.

3.1.2 Data Collector

The role of the Data Collector is to act as a data collection point where rigorous business rules can be enforced via SQL schema declarations (as derived from an ORM based conceptual schema).

The data collector area tables form the focal point for the enablement of integrity rule mechanisms.

The modules that make up the data quality firewall are used to selectively screen the data in the holding area for SQL integrity constraint violations before transferring the data from the holding area to the data collector.

3.2 Determining Data Quality

In terms of user terminology, business rules are a set of conditions that govern a data attribute. For the ORM analyst, the routine process of ORM modeling becomes the mechanism to capture and formally declare these business rules.

Business rules are prescribed so as to constrain the set of permissible values in that "data attribute" (as seen from the user's viewpoint) in a way that obeys the business model and is acceptable to be business (or customer).

A business rule is a statement that defines or constrains some aspect of the data belonging to that ORM fact type that maps to one or more attributes. There may be several business rules that pertain directly to the fact type or to its relationship to other fact types.

Each ORM fact type (mapped to one or more data attributes) must have at least one business rule associated with it; otherwise it does not belong in the data warehouse model.

Each data attribute and associated business rule must have been transformed from an ORM data model; otherwise it does not belong in the data warehouse attribute based model.

Business rules form the basis of the SQL declared integrity constraints. Declared SQL integrity constraints are directly enforceable by the SQL database management system environment that the DW resides in.

Data quality in the data warehouse is reflected by the degree to which the data in the data warehouse conforms to the enforcement of the corresponding SQL integrity constraints to the declared business rules.

SQL integrity constraints are essentially derived from the consolidation of user requirements, business rules, and data models as derived from ORM modeling.

Audit and Control of data quality is driven directly by the declaration of the SQL integrity constraints that are named and declared in the SQL database schema.

3.3 Categories of Business Rules

To facilitate a user-driven pre-ORM modeling analysis phase, 3 Categories of business rules were declared and a form was handed over to the user whenever there was a requirement seen for a 'new data item' in the enterprise. This form was based on the concepts embodied in ISO 11179-3 Metadata Registries standard [6], which provides a requirement for basic attributes of data elements.

The business rules were categorized using casual non-modelling terminology into the following 3 types to assign responsibility---so as to help drive the constraint violation notification process in the modeling and implementation phase. For example Category 1 business rule notifications would go to end-users, whereas category 2 and 3 notifications would be funneled to the Data Administrators for further action during operation.

Possible Category 1 types:

1) Required: e.g. Mandatory or Optional;
2) Range: e.g. Between 2 and 200;
3) Valid Domain: e.g. List of acceptable values i.e. 1 = Performing, 2 = Non-performing 3 = Unknown;
4) Future or Past Date: e.g. Date of recovery cannot be in the future;
5) Date Range: e.g. maturity date must be greater than origination date;
6) Dependency: e.g. if country is Canada or USA then there must be a valid province or state id;
7) Validation Calculation: e.g. Net amount is less than or equal to gross amount, e.g. The maximum amount that a facility can draw must be equal to total authorized multiplied by the maximum percentage that a facility can draw;
8) Unique Value:e.g. Unique code assigned to identify a uniquely occurring property;
9) Unique Role In A Relationship: e.g. Unique code assigned to identify a common risk; and

10) Valid Result: e.g. maximum amount a financial organization can draw must be equal to total authorized multiplied by the maximum percentage that an organization can draw.

Category 1 business rules are rules that are business subject area specific and are used to identify the business rule data violators to provide data quality and integrity. Integrity violations on the data will be reported on using this category classification.

Category 2 business rules: Contains rules that are database structure specific directed towards database technical personnel. Category 1 business rules are translated to Category 2 business rules to be declared in the SQL database schema, for example Primary Key, Foreign key, Alternate Candidate Key and CHECK integrity constraints.

Category 3 business rules contain formulae, expressions or rules used in the derivation of this data attribute, where applicable. Examples are: calculation of averages, formula for ratios, formulas for peer averages, etc. Category 3 business rules are translated to Category 2 business rules to be declared in the SQL database schema where possible, for example SQL expressions, functions, predicates, user defined data types, triggers, or re-usable and or called stored procedures.

3.4 Data Quality Checks for the Load Cycle

The staging area data structures mirror the data structures of the source data structures.

The holding area data structures mirror the data collector data structures with some exceptions.

Data quality is enforced in the modules that make up the Data Quality Firewall as data is loaded from the holding area to the data collector area.

Table 4 below shows the applicable criteria and data quality properties being enforced at each of these stages in the load cycle.

Table 4. Properties of Staging, Holding Area and Data Collector Tables

#	Property	Staging Area Tables	Holding Area Tables	Data Collector Tables
1	Data Structure	Same or subset/view of source	Same as Collector but without PK or FK	Full Third Normal Form (3NF) normalized
2	Table Naming Standards	Based on source + X (for external Tables)	Based on Data Collector + RPR (holding area for repair)	Local Table naming standard
3	Column Naming Standards	Same as source column	Same as Data Collector column	Local Column naming standard
4	Data type checks	Yes	Yes	Yes
5	Assign unique row id for each row	No	Yes	Carry over from Holding Area

Table 4. (*continued*)

6	Assign batch run id	Optional	Yes	Carry over from Holding Area
7	Primary Key	No key	Row IDENTITY	Natural Key
8	Unique Constraints	No	No	Yes
9	Foreign Keys	Optionally yes (for internal housekeeping batch run id)	No (except for internal housekeeping batch run id)	Yes
10	Data integrity checks	Optional (within same row only)	No	Yes
11	Data Validated	Optional (within same row only)	No	Yes
12	Usage during loading	Local incremental extract	Intermediary for transformation and pre-check	Merge/Insert
13	Usage after loading	Archived upon acceptance of batch run	Emptied except for violator rows	Fully validated (except for Soft Inserts)

3.5 Process Flow

Figure 3 below depicts the typical process flow of information through the advanced generation data warehouse. By providing the staging area, holding area and the DW collector area, there are several windows of opportunities for applying transformations, business rule checks etc.---thus insulating each of the stages from any errors, and more importantly, allowing any validation to be run in lower granularities without causing major disruptions.

For example, a business rule engine was developed to automatically handle pre-checks of the business rules between the holding area and the collector area. These business rules were originally declared in the ORM model and are to be realized in implementation via the SQL DBMS based business rule engine modules that can be selectively applied at different stages.

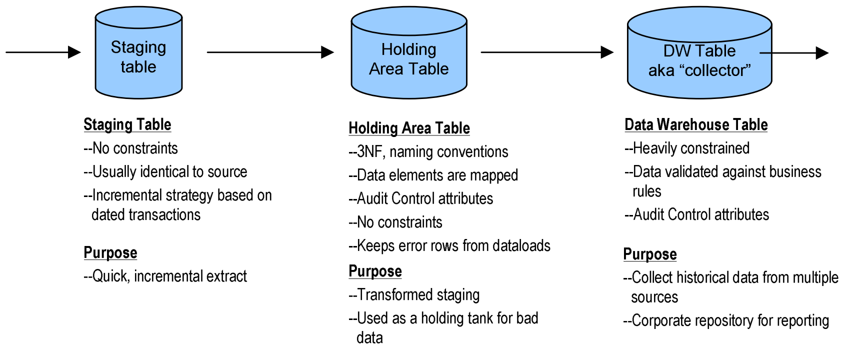

Staging table
Holding Area Table
DW Table aka "collector"
Staging Table
--No constraints
--Usually identical to source
--Incremental strategy based on dated transactions
Purpose
--Quick, incremental extract
Holding Area Table
--3NF, naming conventions
--Data elements are mapped
--Audit Control attributes
--No constraints
--Keeps error rows from dataloads
Purpose
--Transformed staging
--Used as a holding tank for bad data
Data Warehouse Table
--Heavily constrained
--Data validated against business rules
--Audit Control attributes
Purpose
--Collect historical data from multiple sources
--Corporate repository for reporting

Fig. 3. Process Flow through the Advanced Generation DW Architecture

3.6 Data Quality Firewall Modules

It is expected that business rules are identified during the business analysis process as per the ORM data model. Each of these rules are mapped to one or more SQL integrity constraint declarations that are to be declared in the SQL Schema.

The data quality firewall enforces data quality checks during the data load process for the data collector using 5 distinctly separable mechanisms as declared in the SQL schema in the order shown below:

a) Primary Key constraint
b) Uniqueness constraint
c) Foreign Key constraint
d) CHECK constraint
e) Trigger.

These constraints are declared using SQL Schema statements at the table constraints level for the data collector tables. As indicated earlier, these integrity constraints are derived from business rule declarations and subsequent data modeling requirements.

Each constraint is named as per the local constraint naming standard and is mappable to a business rule as declared in the business requirements ORM model.

Each of the data quality firewall enforcement modules except the trigger module, is generic and designed such that it is generatable automatically.

Except for the triggers, each module consists of dynamic SQL statements containing parameterized entries for the qualified table name.

Trigger modules are individually scripted to address special and specific situations of integrity constraints, including inter-table assertions.

3.7 Data Quality Firewall Modules - Generic Operations

Each of the modules, except for the trigger module, essentially has the same processing set of operations.

Each of the modules, except for the trigger, receive the <table name> and a <run id> as input parameters. The module performs the following operations:

1) Determine the applicable SQL integrity constraints by interrogating the extended Information Schema tables in the Audit Control schema.
2) Scan the corresponding holding area table populations with a dynamic query that is constructed from each of the applicable constraint type for the <table name> and <run id>. The constructed query interrogates the incoming rows to check the validity of the applicable constraint as noted below:
    a) For a primary key PK, the constructed search condition would query the incoming holding area rows to determine if there is a corresponding row in the data collector table with the same column values that form the Primary Key constraint.
    b) For a uniqueness constraint, the constructed search condition would query the incoming holding area rows to determine if there is a corresponding row in the data collector table with the same column values that form the uniqueness constraint.

c) For a foreign key constraint, the constructed search condition would query the incoming holding area rows (forming the referencing column values) to determine if there is a corresponding row in the referenced table in the data collector with the same column values that form the referential integrity constraint.
d) For a CHECK constraint, the constructed search condition would query the incoming holding area rows to determine if the search condition of the CHECK clause for that row is satisfied with the values for the columns that form the CHECK constraint.

3) For each constraint violation that has been identified satisfying the above queries, entries are recorded in the Audit & Control tables for each row and each constraint that it violates, flagging each <row id> of the affected row with an error condition.
4) A summary is produced for each constraint that is violated denoting the count of the rows.
5) A summary is produced for each table to denote the counts of rows that were loaded vs. violated regardless of the number or type of constraints.
6) Load the data collector tables with only the rows that do not contain any violations as identified by the Audit & Control table for constraint violations.
7) Delete the corresponding rows in the holding area that have been loaded in the data collector. A row cannot exist in both tables with the exception of a temporary ‘soft insert’ condition.
8) In a ‘soft insert’ condition, the violated row is to be loaded in the data collector notwithstanding the constraint violation. This is done by disabling the constraint on the data collector table, loading the row(s), and re-enabling the constraint with a NOCHECK to prevent the DBMS from determining any violations for existing rows. Note this can only be preformed for CHECK constraint violations. Other constraint violations cannot be circumvented by a ‘soft insert’. In the case of the ‘soft insert’ a corresponding row will exist in the holding area until the constraint violation condition has been removed.

Regardless of the number of tables involved, there is only one module for a given type of integrity constraint check, i.e. one each to handle a PK, Unique (AK), FK and CHECK constraint---in total having 4 modules. The same module will be able to address PK and AK constraints.

4 Data Warehouse Audit and Control

This section primarily addresses the subject of audit and control during data loading in the advanced generation data warehouse environment.

In particular, the emphasis is on monitoring data integrity and data quality issues during the loading process over the stages as defined in the advanced generation Data Warehouse architecture.

Audit and Control essentially addresses the following aspects to enable decision making in the loading process:

- Detection of the occurrence of bad quality data
- Prevention of the occurrence of bad quality data

The following sections provide mechanisms to enable maximum detection of the occurrence of bad data.

Prevention is achieved mainly via non-automated means by the user analyst or designated personnel reviewing the results of the Audit and Control data and enabling decisions to proceed further in the loading process.

It is necessary to maintain an audit trail of the rows being loaded at each Extract-Transform-Load (ETL) process in the DW environment,

4.1 Audit Control Data Collection

The audit control data to be collected varies from simple number counts; to sanity counts; to validation counts---depending on the processes involved in the transformation or load cycle.

At a minimum, the basic audit control information required for each data load at any given point in the load life cycle, consists of:

- number of occurrences loaded
- number of occurrences rejected

and, includes such statistics as

- start date-time of load run
- end date-time of load run
- identification of a load run
- user identification of load operator
- type of run (whether incremental, full, periodic load, iinitial, test load etc.)

Additional information is also collected for the data validation, or data cleansing runs, pre-BI aggregation or pre-join runs. This is discussed in the following section.

The facilities of metadata repository are used to define a supplemental support model in the extended Information Schema, which will contain constructs to capture these additional metadata occurrences. This model is to be realized via an external SQL based database.

4.2 DW Audit and Control Model

An ORM based load control model (and transformed to SQL schema) is to be defined to support the required statistics and audit control information during the load process. The load control model is defined via the extended information schema of entities/tables that contains metadata describing the audit and control requirements. The extended information schema series of tables hold data for the audit and control, and are to be populated for every load operation in the data warehouse.

4.3 Collection Load Statistics at Audit Control Points

Statistics are collected for every load run for any given table, qualified by a given database, at every Audit Control Point (ACP). An Audit Control Point is established at every ETL run as shown in Fig.4 below.

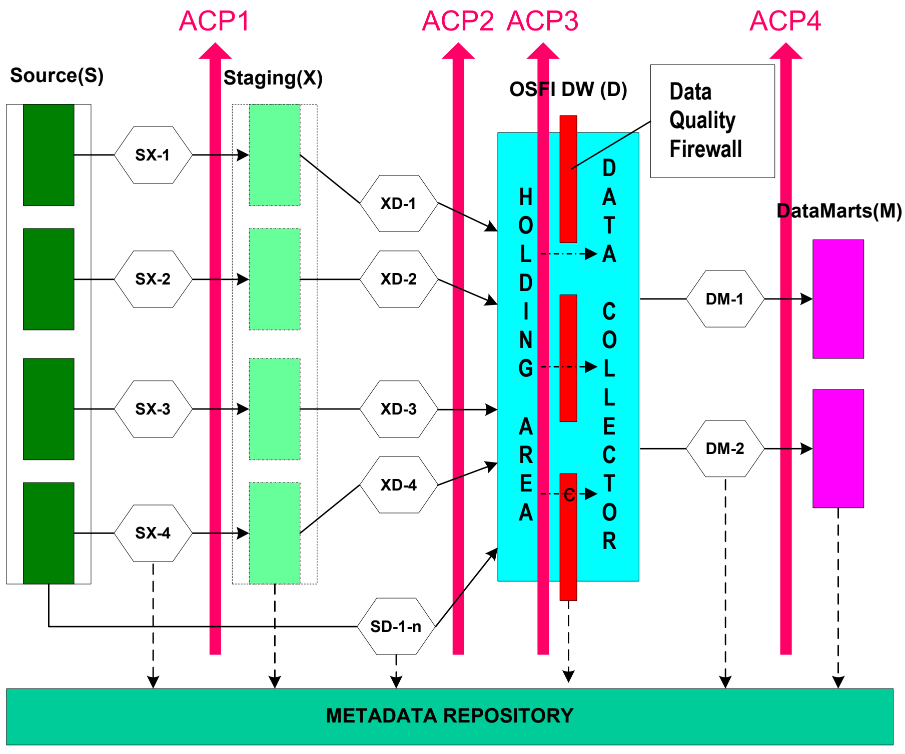


Fig. 4. Audit Control Points in The DW Framework

The attributes and statistics being measured vary by each ACP, starting from simple counts-in, counts-out, to more detailed data quality violations at each progressive ACP.

These attributes and statistics have been grouped into attribute groups as shown below.

Table 5. Correlation of Audit Control Points and Required Attribute Groups

Audit Control Point	Required Attribute Group at ACP
ACP1	Basic Counts, Input Format Check, Go-Nogo Threshold Check
ACP2	Basic Counts, Go-Nogo Threshold Check
ACP3	Basic Counts, Enhanced Threshold Check, Sanity Check, Validation Counts, Integrity Violations By Row
ACP4	Basic Counts, Go-Nogo Threshold Check, Sanity Check, Validation Counts, Integrity Violations By Row

Attribute groups have been designed to measure groups of applicable attributes depending on differing levels of data loadings and data quality availability over the data warehouse load process.

Table 5 shows the correlation of Audit Control Points and the respective required attribute group information in order to perform standard Audit and Controls for the advanced generation Data Warehouse.

4.3.1 Basic Counts

The Basic count group essentially provides for the fundamental level of row counts for the table being loaded for a given run.

A row count is to be recorded for the rows input from the source for the run and table being loaded.

A row count after the run is to be recorded for the table being loaded providing the count of rows loaded.

4.3.2 Input Format Check

The Input Format check is an accuracy check to confirm any required formatting or rule validation of columns within the same row of the input table.

This check includes data type checking, left padding with zeros, date format checking, no commas/dots in the numeric string, min/max rules, mandatory column range rules etc. The purpose of the check is to ensure the data is received as intended, and is conforming to defined formatting, business rules or business requirements.

4.3.3 Go-Nogo Threshold Check

The Go-Nogo threshold check is a test to confirm whether pre-defined minimum conditions have been met to proceed to the next stage of loading. This has to be tailor-made for sets of tables based on definable parameters. A Go-Nogo threshold check is based on row occurrence counts or a combination thereof for rows in a given run in a given table.

A generic example is threshold checks performed on say, the number of items received where a minimum set has been established. Failures in the transfer of data could be brought to the surface using these checks.

At ACP2 for example, the Go-Nogo threshold check could provide information on whether certain reference table information was loaded and whether it is worth continuing to the next loading stage into the Data Warehouse.

4.3.4 Enhanced Threshold Check

The Enhanced threshold check is a superset of the Go-Nogo threshold check, and contains all the checks as the Go-Nogo threshold check as the base. In addition, the Enhanced threshold check provides data integrity/data quality thresholds like number of constraint violations for rows in a given run in a given table.

Whereas a Go-Nogo threshold check is based on row occurrence counts or a combination thereof, the Enhanced threshold check includes constraint violation counts in addition, i.e. some measure of data quality involved to check out the threshold for the next stage.

It is the responsibility of the subject area analyst to define the conditions and corresponding enhanced threshold checks to be defined, along with appropriately modelled tables to hold the conditions.

At ACP3 for example, the Advanced threshold check could provide information on whether there were serious constraint violations on particular tables (e.g. invalid financial coding information created massive reject violations in other tables), and as such, it would not be worthwhile continuing on to the Data Mart stage to produce any reports or cubes.

Another example would be a case where there were 51,000 rows on input for a particular table, and if there were 50,000 integrity constraint violations, then it would not be worthwhile producing a data mart.

4.3.5 Sanity Check

Sanity Checks, also known as reasonability checks, go beyond data quality row occurrence counts into checking for established or pre-defined column or aggregate value totals for reasonableness for rows in a given run in a given table.

These sanity checks are essentially “good practice checks” being performed to highlight or reject absurdities and nonsense conditions.

A classical example would be total number of hours worked on multiple projects, including overtime, cannot exceed 24 hours for the day. Another example would be the total hours flown by an aircraft cannot exceed 8760 hours (24hrs x 365 days).

4.3.6 Validation Counts

Validation counts record details on the row counts of integrity violations and associated statistics for a given run on a given table.

For a given table, this includes the number of rows input, number of rows loaded, number of rows violated on the load. The number of rows violated denotes the rows that were rejected due to violations of declared integrity constraints of Primary Key, Uniqueness constraints, Check constraints, Foreign Key constraints, and Assertion constraints (inter table) that are summed up in a gross total validation count.

In addition, the number of rows violated (gross), for each named integrity constraint is also noted.

4.3.7 Integrity Violations by Row

The Integrity violations by row group tracks the individual rows tagged as being in violation for a named integrity constraint for a given run, along with the error status. A given row may be simply tagged as being in error until its status changes to being fixed, or ignore etc.

4.4 Automating and Tracking Data Cleansing/Data Quality Issues

Some proponents in data warehousing state that the data cleansing activity could very easily consume up to 80% of the loading schedule times.

It will be more productive to track the data going through the data cleansing process, particularly, when many precise steps have been defined.

Data cleansing issues are often relegated to being a labour intensive after-load process, or, in some cases treated as part of the loading process in an arbitrary fashion. This approach does not leave room for addressing the data in error and the follow-on actions. Data cleansing algorithms are usually applied manually through discrete ad-hoc processes, back-door approaches/procedures.

Data quality and data consistency needs to be dealt with on multiple levels. It is simply not sufficient to deal with this issue during a load stream, on a stand-alone basis. Data is normally being integrated in a data warehouse from multiple sources. Data consistency is to be sought immediately after the integration has occurred. It is futile to spend resources on an individual loading stream and then to hope that the data is going to be consistent when integrated. For example, data could be consistent in each source system within its individual boundaries. But, when integrated, this very same data could become inconsistent when brought together.

This issue could be handled manually, after integration to review consistency, and to follow up via hand- written application procedures to further cleanse the data. Issues that still need to be addressed is "what to do with culprit data?", "how does one follow up with feedback to source system and awaiting clarifications or update to cleanse already loaded data?"

An intelligent data cleansing architecture uses the facilities and features of the modern day Database Management System based on established International SQL standards.

All data have to adhere to some form of pre-defined set of rules based on formal propositions regardless of their residency---be it in source systems, in the data warehouse or in data marts.

The advanced generation data warehouse architecture defines a fully integrated normalized data structure, strongly supported by structurally enforced business rules and constraints using formal syntax and semantics of ISO SQL, for the collector database where source system data are being integrated. Facilities can also be made in the architecture to house the culprit occurrences that need to be further cleansed, and to define minimal "integrity sets" to pass through the data that is consistent as a set but not otherwise.

4.5 Data Cleansing Support Tables

The automated approach to data cleansing requires each business rule (also cleansing rule) to be named and identified. It is also necessary to be able to identify each incoming row of data in a given table. These are declared in the extended information schema.

The loading process could be designed to then associated or "pass through" a rule conformance check of the incoming data row with each applicable cleansing rule.

Additional tables can be defined in extended information schema to track the violators of the rules by row identifiers and rule identifiers.

Details are collected on a given row that violates one or more rules, the status assigned to the row for cleansing for a given run, e.g. ERROR, UNDER REVIEW, PENDING, OK IGNORE, FIXED etc.

Statistics are collected by table, rule violated, giving the number of rows violated in a given run.

5 Conventions and Standards for Implementation

The following section provides the base architectural requirements in terms of conventions and standards for the successful implementation of the advanced generation data warehouse supported by a strong data quality firewall.

5.1 ORM Model

The base ORM model in itself shall be developed using a strong set of modeling standards for each of its constituent modeling constructs, e.g. each fact type, object type, constraint (declarative and text), role etc., needs to have an identified and standardized unique identifier so that traceability and mappings to an attribute based model (e.g. Entity Relationship (ER) Logical model) can be defined and maintained. This topic is not covered here since the focus of this section is more on implementation conventions and standards for the data quality firewall in the data warehouse environment.

5.2 Data Warehouse Logical Data Model(DW LDM)

The DW LDM shall reflect the integration of the semantics of selected applications or business activities into a single enterprise-wide optimally normalized data warehouse logical data model.

The following General Rules are applicable so as to be able to completely define the semantics required for the data quality firewall rules in the DW LDM:

1) Mappings from the ORM model to the DW LDM to be defined and maintained, preferably through automated means using suitable ORM software tools.
2) The DW LDM shall be developed based on some locally defined standards and guidelines and conventions for the business data model objects using ER or equivalent attribute modelling constructs
3) The DW LDM shall represent the semantics of the Business Model for the particular subject area based upon the enterprise viewpoint to enable enterprise wide semantic integration
4) The DW LDM shall be formulated in at least the Third Normal Form
5) The DW LDM shall contain the maximum possible declarations of integrity constraints as definable within attribute based modelling paradigm at the logical level.
6) For those integrity constraints in the DW LDM that are beyond the expressive power of the syntax or semantics of attribute based modelling paradigm, clear specifications in English language text shall be defined and documented in the entity property of definitions for incorporation in the physical database schema.
7) The DW LDM shall represent the Business Model for the particular subject area incorporating history capabilities using temporal date-time tracking of data that has been identified as a target for reporting or data marts
8) The DW LDM shall incorporate the necessary attributes and data integrity as required in support of the Audit & Control specifications for the DW effort, based on local or federal governance requirements
9) The DW LDM shall incorporate attributes / columns in every DW entity / table for accommodating traceability by each row occurrence for Audit & Control purposes
10) The DW LDM shall incorporate attributes / columns in every DW entity / table for accommodating traceability by each batch load or run stream for Audit & Control purposes.

It is important to note that there could very well be overlapping semantics resulting from this exercise between the DW LDM and the source system data models, including a re-statement or as it usually turns out, a complete statement of the business rules involved in the particular subject area.

A successful implementation of the data quality firewall absolutely mandates that all business semantics and rules applicable to the subject area be mapped from the ORM model, and suitably defined in the DW LDM (and perhaps sometimes bypassed to the Physical Data Model due to attribute modelling limitations for semantic expressability) , following the TR9007 Conceptual Schema 100% principle [2].

5.3 Data Warehouse Physical Data Model(DW PDM)

The DW PDM shall reflect the SQL schema version of the DW LDM maintaining the consistent data model semantics as defined in the DW LDM.

The following General Rules are applicable so as to be able to derive and implement the data quality firewall rules in the DW PDM:

1) The DW PDM shall represent a truthful mapping of the entire set of semantics as declared in the ORM model (subsequently mapped to the DW LDM), with the semantics maximally defined through declarative SQL schema statements, and minimally via supporting triggers or any stored procedures.
2) The DW PDM shall follow some locally defined standards and database guidelines as for schema objects
3) The DW PDM shall represent the DW LDM of the Business Model based on permissible physical schema mappings to enable enterprise wide semantic integration
4) The DW PDM shall follow the data naming standards for tables, columns, integrity constraints, triggers
5) The DW PDM shall contain the maximum possible list of integrity constraints as declarable within SQL. This list shall include the following declarations:
   a) Primary Key
   b) Unique Constraint (AK) where applicable
   c) Foreign Key
   d) Data Type
   e) Data Size
   f) Nullability (only for the Primary Key or Unique Constraint attributes)
6) All integrity constraints shall be named
7) To avoid any cascade deletes, the referential integrity properties for DW tables shall be set via the REFERENCES clauses of the CREATE TABLE and ALTER TABLE statements with ON UPDATE and ON DELETE as follows:
   a) ON DELETE NO ACTION (default if ON DELETE is not specified)
   b) ON UPDATE NO ACTION (default if ON UPDATE is not specified)

5.4 Data Mart Logical Data Model(DMLDM)

The DM LDM reflects a selected functional specialization of the Data Warehouse Logical Data Model as per the following rules:

1) The DM LDM shall be developed based on some locally defined standards and guidelines and conventions The DM LDM shall follow the standards and guidelines as defined in SDLC021 for the business data model objects.
2) The DW LDM shall represent the semantics of a functional subset of the DW Business Model for the particular subject area based upon the requirements of the BI output product(s).
3) The DM LDM shall contain the maximum possible list of integrity constraints as declarable within the attribute model paradigm at the Logical Level. This is needed so that these constraints can to be validated to provide assurance against any loss of semantics in data transfers from the collector DW to the Data Mart.
4) The DM LDM shall incorporate the necessary standard datetime attributes for temporal handling of searches and BI reporting where necessary.
5) The DM LDM shall incorporate the necessary attributes and data integrity as required in support of the Audit & Control specifications for the DW effort, based on local or federal governance requirements.
6) The DM LDM shall incorporate attributes / columns in every DM entity / table for accommodating traceability by each row occurrence for Audit & Control purposes.
7) The DM LDM shall incorporate attributes / columns in every DM entity / table for accommodating traceability by each batch load or run stream for Audit & Control purposes.

6 User Experiences

The ORM based advanced generation data warehouse data firewall approach has been proven and used successfully in the development of very large data warehouses, and web based applications very productively. Previous incarnations of these applications either minimally existed (or struggled to exist) and were developed on poorly defined attribute-based data models. The production environment of these struggling applications essentially was a glorified prototype in an attempt to complete the attribute based data modeling exercise.

The use of ORM as the fundamental basis for the development of semantic models, and subsequent transformation to attribute based models using CASE tools, with final implementation based on strong ISO SQL adaptations of RDBMSs, has resulted in a 100% success rate---including some with award winning implementations [5]. More often than not, the data warehouses and applications developed using ORM were completed ahead of schedule, with measured savings of 35-40% being realized over the entire project costs. More importantly, the implementation worked the first time around with a 95-98% correctness of the data model fit to user requirements---simply because of the discipline and rigour inherent in the ORM approach.

In the author's 36 years experience of systems and applications development, and having reviewed 100s of data models, I have yet to come across an application designed with attribute based data modelling that has in it a complete definition of all the business rules readily defined and implemented! The term "complete definition" uses an ORM yardstick of 100% rule set, because, most of the time the analysts and developers comment that they would never have even "thought" of the business rules that were being discovered via ORM.

The above advanced generation data warehouse architecture and its associated data quality firewall modules described in this paper has been implemented 100% over the years from 1997 onwards. ORM was the foundation used in the development of the components and the stages constituting the design of the advanced generation data warehouse architecture itself.

An extended information schema metamodel for the DW architecture was developed in ORM, with the accompanying Audit and Control Framework and Data Quality Firewall becoming a by-product of the business rule conformance set that was required for the realization of the ORM advanced generation data warehouse data model.

What would the world be without ORM……

References

1. Gartner Group Report. Gartner Press Release, Gartner Website – Media relations (2005), `http://www.gartner.com/press_releases/pr2005.html`
2. ISO: ISO/IEC TR9007: Technical Report on Concepts and Terminology for the Conceptual Schema and the Information Base. Editor: Joost van Greutheuysen: International Standards Organization, Geneva (1987)
3. ISO: ISO/IEC 10746-1, 2, 3, 4: Reference Model of Open Distributed Processing (RMODP), International Standards Organization, Geneva
4. ISO: ISO/IEC 9075-1, 2, 3, 4, 9, 10, 11, 13, 14 Database Language SQL (2003)
5. TC Express: a Transport Canada publication referencing Government of Canada Technology Week (GTEC) bronze medal awarded towards the implementation of a strategic information management application e-Directory at Transport Canada (2005), `http://www.tc.gc.ca/TCExpress/20021112/en/fa05_e.htm`
6. ISO: ISO/IEC 11179-3: Information Technology – Metadata Registries (MDR) – Part 3: Registry Metamodel and basic attributes (2003)
7. Lessons From History, `http://lessons-from-history.com/Level%202/Project%20Success%20or%20Failure.html`
8. Standish Group International, Inc.: Chaos Chronicles and Standish Group Report (2003), `http://www.standishgroup.com/sample_research/index.php`
9. ITtoolbox Blogs, Madsen, Mark: A 50% Data Warehouse Failure Rate is Nothing New, `http://blogs.ittoolbox.com/eai/rationality/archives/a-50-data-warehouse-failure-rate-is-nothing-4669`
10. Hufford, D.: Data Warehouse Quality: Special Feature from January 1996, DMReview.com (1996), `http://www.dmreview.com/article_sub.cfm?articleID=1311`
11. Standish Group International, Inc.: The Standish Group Chaos Report (1995), `http://www.projectsmart.co.uk/docs/chaos-report.pdf`

12. Halpin, T.: Information Modeling and Relational Databases. Morgan Kaufmann, San Francisco (2001)
13. Nijssen, G.M., Halpin, T.A.: Conceptual Schema and Relational Database Design. Prentice Hall, Victoria (1989)
14. Halpin, T.: Fact-Oriented Modeling: Past, Present and Future. In: Krogstie, J., Opdahl, A., Brinkkemper, S. (eds.) Conceptual Modelling in Information Systems Engineering, pp. 19–38. Springer, Berlin (2007)

Discovering Groups of Sibling Terms from Web Documents with XTREEM-SG

Marko Brunzel[1,2] and Myra Spiliopoulou[2]

[1] DFKI GmbH - German Research Center for Artificial Intelligence
[2] Otto-von-Guericke Universität Magdeburg, Germany
marko.brunzel@dfki.de, myra@iti.cs.uni-magdeburg.de

Abstract. The acquisition of explicit semantics is still a research challenge. Approaches for the extraction of semantics focus mostly on learning subordination relations. The extraction of coordination relations, also called "sibling relations" is studied much less, though they are not less important in ontology engineering.

We describe and evaluate the XTREEM-SG approach on finding sibling semantics from semi-structured Web documents. XTREEM-SG stands for "Xhtml TREE Mining - for Sibling Groups". It uses the XHTML-markup that is available in Web content to group together terms that are in a sibling relation to each other. Our approach has the advantage that it is domain and language independent; it does not rely on background knowledge, NLP software nor training.

We evaluate XTREEM-SG towards two gold standard ontologies. We investigate how variations on input, parameters and gold standard influence the obtained results on structuring a closed vocabulary into semantic sibling groups. Earlier methods that evaluate sibling relations against a gold standard report a 14.18% F-measure on average sibling overlap. Our method improves this number into 22.93%.

1 Introduction

Ontologies are a necessary resource for any application that requires an shared understanding of concepts, relations and axioms on a universe of discourse. This includes all applications that involve more than rudimentary information exchange among individuals, among else in the Semantic Web. The semi-automatic extraction of ontological knowledge from information sources such as annotated corpora or plain document archives is studied in the field of *ontology learning.* We concentrate on the ontology learning task of extracting semantic relations from Web document collections. In particular, we study *sibling relations*, a topic often overlooked in the field.

Concepts stand in a sibling relation to each other if they are the children of a common parent concept. Then, they constitute a *sibling group.* Identifying sibling concepts and building sibling groups is a natural way of observing the world. It reflects the well-known tendency of humans to build categories of things. Essentially, the assignment of a name/label to a sibling group corresponds to

© Springer-Verlag Berlin Heidelberg 2008

the definiton of the category, which becomes their parent concept. This is a very important step in the process of ontology *learning* or ontology *building*.

As a rudimentary example, the group of concepts {`penguin, ostrich, emu`} constitute a new concept, which the human may define as "`flightless bird`". This new concept would refine the large concept `bird`, which would otherwise consist of too inhomogeneous types of bird. In our retrospective knowledge of birds, the flightless birds are likely to appear as a concept in an ontology of birds, but its very definition is the aftermath of the discovery of a sibling group that is worth a label.

The discovery of sibling groups in an early step of the ontology learning process may also indicate whether the anticipated ontology should be built as a strict hierarchy or rather a directed graph. In our example above, a penguin is also an "`aquatic bird`"). The discovery of a sibling group with terms like `penguin, duck, swan`, will allow the ontology engineer to decide whether both aquatic birds and flightless birds should appear in the ontology. In this case, the ontology should be built with instruments that allow and support multiple inheritance. If an ontology is a priori designed to be a strict hierarchy, the discovery of one of the sibling groups below a predefined concept will prevent the discovery of the other sibling group.

Ontology learning can be observed as a *layered* process, according to the model of Cimiano [13], as depicted in Figure 1. Our work belongs to the layer "Concept Hierarchies". Research on this layer includes many methods for the discovery of *direct hierarchical relations* where a *subordination* can be observed. Less attention is paid to the discovery of of *co-concepts*, i.e. concepts which stand in a *sibling relation* to each other.

The distinction between *subordination relations* and *coordination relations* can be seen in Figure 2. Specific types of *coordination* relations in linguistics are *co-hyponymy* relations and *co-meronymy* relations. Co-hyponymy refers to hyponyms – they have a certain direct hypernym in common. Co-meronymy refers to the meronyms (parts) of a common direct holonym (whole).

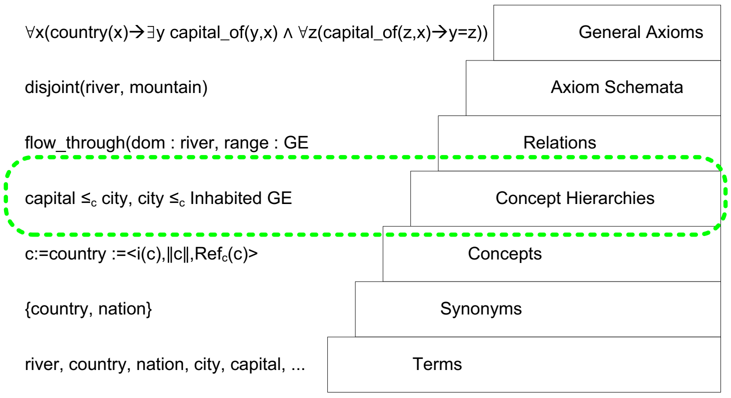

∀x(country(x)→∃y capital_of(y,x) ∧ ∀z(capital_of(z,x)→y=z))
General Axioms
disjoint(river, mountain)
Axiom Schemata
flow_through(dom : river, range : GE
Relations
capital ≤c city, city ≤c Inhabited GE
Concept Hierarchies
c:=country :=<i(c),‖c‖,Refc(c)>
Concepts
{country, nation}
Synonyms
river, country, nation, city, capital, ...
Terms

Fig. 1. Ontology Learning Layer Cake [13]

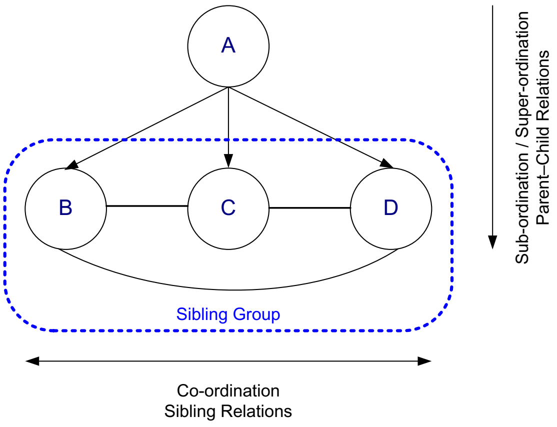


Fig. 2. Subordination and coordination relations

In this paper, we present the method XTREEM-SG for the discovery of sibling groups of terms. XTREEM-SG stands for “Xhtml TREE Mining for Sibling Groups” because it takes as input Web documents that adhere to the (X)HTML format. In the core of our approach is the *Group-By-Path* (GBP) algorithm, which exploits the structure of Web documents to deduce semantic relations [5, 6, 7]. The work presented here extends and enhances our previous work on sibling group discovery with clustering and cluster labeling [6].

The main contribution of our approach is the discovery of sibling groups without need of annotated corpora, linguistic resources or language-specific rules, although any of these may be exploited in a preprocessing step. XTREEM-SG takes as input an arbitrary, heterogeneous and possibly noisy collection of Web documents on some given subject, and a vocabulary of terms that are relevant to this subject. It identifies and returns groups of these terms that stand in a sibling relation to each other.

The paper is organized as follows: In the next section, we discuss related work. XTREEM-SG is presented in Section 3; it is based on our previous work [6]. Section 4 contains our evaluation methodology for the task of sibling groups' discovery from an arbitrary collection. According to this methodology, we evaluate XTREEM-SG with two gold standard ontologies. The experiments are presented in section 5. The last section concludes the paper with a summary of findings and an outlook.

2 Related Work

Document structure has been as indicator of similarity among documents in research for document clustering [12, 16, 35]. Metrics for structural similarity can be found in [10, 36]. In our study, we use a specific form of structural similarity, the path structure of XHTML documents. However, we do not use it to infer similarity among documents but rather to identify sibling terms within structured documents.

Our work belongs to the domain of ontology learning. Surveys on this subject include [9, 13, 20, 31]. Comprehensive overviews with special emphasis on *ontology learning from text* have been published recently [9, 13]. There are also methods on *ontology learning from structure* [24, 34]. They use database schemata and other well-structured resources to infer ontological components. However, such structures are rarely available for arbitrary topics and domains.

Methods on *ontology learning from Web documents* include [1, 19]. In the KnowItAll system [18], Web document structure is used to establish a knowledge base of extracted entities. The system proposed by Pasca [29] finds instances of WordNet concepts within big Web document collections with a rule-based mechanism. However, Pasca treats the Web documents as plain texts, ignoring the markup.

Some dedicated approaches for the discovery of semantics from Web documents are based on the so-called *Hearst patterns* [22] to discover relations among terms. Among else, they are used in the Caméléon system of Aussenac-Gilles and Jacques [2] for the discovery of relations. However, patterns of this type are rare and have low coverage, even for large document collections. Cimiano et al. discover (co-)hyponymy relations by finding and analyzing examples of Hearst patterns in the Web [14, 15]. Heyer et al discover co-hyponymy relations from plain texts with use of association measures [23]. Kruschwitz [25, 26] uses markup sections of Web documents to learn a domain model. As in our XTREEM-SG approach, Kruschwitz exploits the fact that similar concepts inside Web documents are often adorned with the same markup. HTML tags of documents are also used by Shinzato et al to discover hyponymy relations [32]. However, XTREEM-SG is considering all tags of the (X)HTML tree structure, not only specific structures.

In our earlier approach XTREEM-SP [5] we have exploited Web document markup and applied statistical measures to find Sibling Pairs of terms. Compared to XTREEM-SG, XTREEM-SP has the disadvantage of returning only binary relations. In a followup study [8], we used frequent association rules' discovery to identify sibling groups of arbitrary cardinality. However, we have obtained results of lower quality than those achieved with XTREEM-SG.

3 Clustering with the XTREEM-SG Approach

We present XTREEM-SG [6], our algorithm for the discovery of groups of sibling terms on the basis of document vector clustering. XTREEM-SG stands for Xhtml TREE Mining for Sibling Groups and is depicted in the data flow diagram of Figure 3.

We first describe the generic "XTREEM approach", which refers to (a) the exploitation of documents crawled from the Web as opposed to the processing of a well-prepared document corpus and (b) the exploitation of document markup for the vectorization. Then, we explain the XTREEM-SG subprocess that encompasses vectorization, vector clustering, cluster labeling and the identification of sibling groups of terms among the cluster labels.

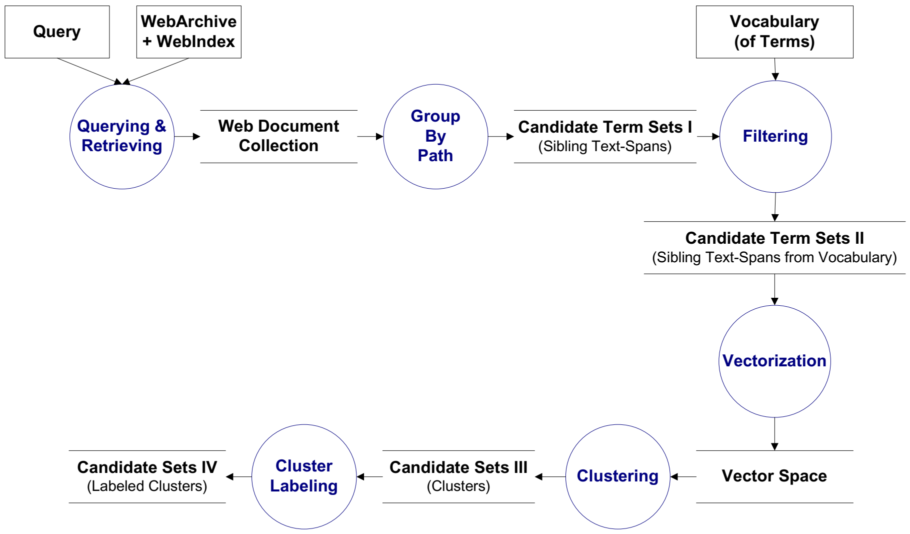

Query
WebArchive + WebIndex
Vocabulary (of Terms)
Querying & Retrieving
Web Document Collection
Group By Path
Candidate Term Sets I (Sibling Text-Spans)
Filtering
Candidate Term Sets II (Sibling Text-Spans from Vocabulary)
Vectorization
Vector Space
Clustering
Candidate Sets III (Clusters)
Cluster Labeling
Candidate Sets IV (Labeled Clusters)

Fig. 3. Dataflow diagram of the XTREEM-SG procedure

3.1 The XTREEM Approach

"XTREEM" stands for Xhtml TREE Mining. It encompasses the first three tasks of the process in Figure 3. It takes as input a set of application-specific queries, which are launched to the Web (via a search engine or crawler), and builds from them a Web document collection for the application's domain. From these documents, it extracts the candidate terms for sibling group discovery. In this context, a "term" may contain one or more words, i.e. be a multi-word term.

To choose the candidate terms, we process each Web document in the collection. We do not consider document content but rather study the markup conventions, which can be found in almost all Web documents: Different authors use different nested tags to structure pieces of information in Web documents, but each author tends to use the same structure for similar contents. The overall number of structures in use is also limited.

XTREEM exploits this observation to find terms appearing within the same syntactic structure of an XHTML (or HTML) document. The structure considered in XTREEM is the *tagpath*, a sequence of markup tags starting at the root element of the document and leading to a piece of text. A document is mapped to a set of tagpaths. The target string of a tagpath is a *textspan*. Intuitively, textspans can be of arbitrary length, ranging from single words to whole paragraphs. Textspans that appear at the end of identical tagpaths, as are e.g. members of the same list or headings of the same document chapter, are extracted by the algorithm Group-By-Path [6], which identifies the tagpaths and their associated textspans for all documents.

A textspan may contain one or more words. If there is an application-specific vocabulary, we juxtapose the textspans with the terms found in it and consider only those vocabulary terms that appear frequently as textspans. If no vocabulary is available, we use the textspans directly as terms. In this work, we concentrate on the discovery of sibling groups for *given* terms, so we assume the existence of a vocabulary. We next describe each of those tasks in more detail.

Task I: Querying and Retrieving a Web Document Collection. Traditionally, the discovery of semantics for an application domain is performed on a dedicated document corpus. This corpus consists of carefully selected documents that are representative of the domain. The coverage of the whole domain might be low, but the precision is high, because irrelevant documents, terms and ontological components are unlikely. The advantages of using a dedicated corpus for ontology learning are apparent. The disadvantage is that such corpora are not available for arbitrary applications.

In XTREEM, we *build* a collection of Web documents rather than assuming the existence of a dedicated corpus. This collection consists of the documents that satisfy a query (or set of queries) on the application domain. For example, a document collection on tourism may be acquired by issuing a query on `touris⋆`. Hence, XTREEM takes as input a set of queries that reflect the domain. Those queries should be designed with emphasis on coverage rather than precision. High coverage is necessary, to ensure that all terms of potential relevance are included. Precision is less important, because irrelevant terms and irrelevant documents are unlikely to be of statistical significance in the subsequent processing steps.

XTREEM launches the queries towards a search engine or inputs them to a crawler. Dedicated crawling tools that can make use of ontological components can also be used to this purpose (cf. Ehrig and Maedche [17]).

Task II: Building a Set of Candidate Terms with Group-By-Path. In this task, we transform each Web document into an XHTML tree structure, map it to a set of tagpaths with associated textspans with Group-By-Path [6]. We describe Group-By-Path in sequel, after providing some basic definitions.

Definition 1 (Web Document). *A "Web document" or "Web page" d is a semi-structured document following the W3C XHTML standard*[1].

Traditional HTML documents can be trivially converted to XHTML documents, so Definition 1 is not restrictive towards legacy pages in the Web. An XHTML document has a tree structure, where content/text is in the leaf nodes, while the intermediate nodes are markup elements. We use the expression *textspan* to refer to the content of a leaf node and the expressions *markup element* and *tag* for the content of an intermediate node. Figure 4 depicts an excerpt of an example Web document in XHTML format.

Definition 2 (Tagpath and Textspan). *Let M be the set of XHTML tags and let d be a Web document according to Definition 1. A "tagpath" p in d is a*

[1] http://www.w3.org/TR/xhtml1/

```
<html>
      <head>
            <title>About Dangerous Sharks</title>
      </head>
      <body>
            <h1>Dangerous Sharks</h1>
            <p>There are some shark species …</p>
            <h2>Great White Shark</h2>
            <p>The Great White Shark, also known as …</p>
            <h2>Hammerhead Shark</h2>
            <p>The head shaped …</p>
            <h2>Tiger Shark</h2>
            …
      </body>
</html>
```

Fig. 4. The XHTML tree structure of a Web document

```
<html>
<html><head>
<html><head><title>About Dangerous Sharks</title>
<html></head>
<html><body>
<html><body><h1>Dangerous Sharks<h1>
<html><body><p>There are some shark species …</p>
<html><body><h2>Great White Shark</h2>
<html><body><p>The Great White Shark, also known as …</p>
<html><body><h2>Hammerhead Shark</h2>
<html><body><p>The head shaped …</p>
<html><body><h2>Tiger Shark</h2>
<html><body>…
<html></body>
</html>
```

Fig. 5. A Web document with its tagpaths and textspans

sequence of tags leading from the root tag in d to a "textspan" e in d. Hence, p has the form $p =< m_1, m_2, \ldots, m_v >$, where $m_i \in M, i = 1, \ldots, v$.

We use the notation (p, e) to indicate that e is the textspan to which p leads.

By this definition, p is a branch of an XHTML tree. For each $m_i (i = 1, \ldots, v-1)$, it holds that tag m_i surrounds tag m_{i+1}. A tagpath is therefore a special kind of XPath [2] expression. Moreover, a document d constitutes a collection of pairs of the form (p, e), where p is a tagpath and e is the textspan at its end.

Example 1. The tagpaths and textspans of the Web document excerpt in Figure 4 are shown in Figure 5. In line 8, the tagpath `<html><body><h2>` leads to the textspan `Great White Shark`. In the next line, the tagpath `<html><body><p>` leads to a textspan that is one paragraph long. ∇

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

The Group-By-Path algorithm [6], depicted below as Algorithm 1, takes as input a Web document d in XHTML format. Essentially, d is a set of tagpaths with associated textspans, which we denote as Y. For the document d, or equivalently for the set Y, Group-By-Path identifies and groups identical tagpaths: All distinct tagpaths in d constitute a set Z (line 3). For each tagpath $p \in Z$, Group-By-Path finds all occurrences of p. For each such occurrence, it adds into a set B_p the textspan e, to which it leads (line 5). This set B_p is the *textspan-set* of the tagpath p. So, for the document d, Group-By-Path forms and returns the union of all textspan-sets $A = \cup_{p \in Z}\{(p, B_p)\}$ (lines 6 and 8).

Algorithm 1. The Group-By-Path Algorithm [6]

Input: Web document d
Output: Set of tagpaths Z and set $A \equiv A$, containing the textspan-set of each tagpath $p \in Z$

```
1: $A = \emptyset$
2: map $d$ to the set $Y$ of $(p, e)$-pairs, where $p$ is a tagpath according to Definition 2
   and $e$ is its target textspan
3: let $Z$ be the set of tagpaths in $d$, i.e. $Z := \{p|(p, e) \in Y\}$
4: for all $p \in Z$ do
5:    $B_p = \{e|(p, e) \in Y\}$
6:    $A = A \cup \{(p, B_p)\}$
7: end for
8: return $A, Z$
```

Example 2. When we apply Group-By-Path on the tagpaths of the Web document in Figure 5, we acquire the textspan-set {`Great White Shark, Hammerhead Shark, Tiger Shark`} for the tagpath `<html><body><h2>`. The textspans appearing after the occurrences of the tagpath `<html><body><p>` build another textspan-set. ∇

Task III: Filtering Textspans. By applying Group-By-Path on all documents in the Web document collection D, we obtain all candidates for sibling groups' discovery. They constitute the set of textspan-sets $\mathcal{A} = \cup_{d \in D} A_d$ and are accompanied by the set of tagpaths $\mathcal{Z} = \cup_{d \in D} Z_d$.

Before proceeding with data mining for sibling groups discovery, we perform two filtering steps upon the set of candidates $\mathcal{A}$. First, if there is an a priori known domain-specific vocabulary V, we use it to eliminate textspans from $\mathcal{A}$, which do not appear in V. This allows us to remove textspans that may be irrelevant to the domain. However, this filtering operation should be treated with caution: If V does not cover the whole domain, using it may lead to the elimination of terms which are of potential relevance. In this study, we assume that the vocabulary exists; it contains the terms for which we want to find sibling relations, as part of the ontology engineering process.

Furthermore, we remove textspan-sets that have only one member, since such sets do not contribute to the discovery of sibling relations. We use the retained textspans as (single-word or multi-word) terms, constituting a set $F \subseteq \mathcal{A}$, accompanied by the set of associated tagpaths $I \subseteq \mathcal{Z}$.

3.2 XTREEM-SG for Discovering Sibling Groups

XTREEM-SG stands for Sibling Groups discovery with the XTREEM approach. It corresponds to the second subprocess of Figure 3, starting with the vectorization upon the output of XTREEM.

Task IV: Vectorization for Clustering. In conventional document mining, analysis is performed upon *document vectors* comprised of the terms occurring in each document *or* upon *term vectors* comprised of the documents where each term occurs. For the first option, the terms constitute the feature space, upon which a document is described as a vector. For the second option, the feature space consists of the documents and a term is a vector. These two options are important for subsequent steps, so we explain them here with an example.

Example 3. Assume the one-sentence documents d_1 `''This is my cat.''` and d_2 `''The cat ate the canary.''`. The set of terms for this tiny document collection is T ={`this, is, my, cat, the, ate, canary`}. To build document vectors, we first turn T into a feature space by providing an ordering scheme. For example, we order the terms alphabetically into the list `(ate, canary, cat, is, the, this)`. In this feature space, $d_1 =< 0, 0, 1, 1, 0, 1 >$ and $d_2 =< 1, 1, 1, 0, 0, 1 >$, where 1 stands for the apperance of a term and 0 for its absence. Document mining usually involves term weighting in the vectors, using e.g. the classical TF-IDF weighting scheme of Salton and Buckley [30].

To build term vectors according to the second case above, we use the documents as dimensions/features constituting the feature space $\{\{d_1, d_2\}\}$. Then, each term in T is a vector. For example, `cat`$=< 1, 1 >$, `canary`$=< 0, 1 >$ etc. Here, 1 means that the document contains the term and 0 otherwise. ∇

For XTREEM-SG, a document has been mapped into its tagpaths. So, the equivalents of the two options are (a) a vectorization of tagpaths in the feature space of terms and (b) a vectorization of terms over the feature space of tagpaths. In the example of Figure 6, they correspond to (a) using the columns as feature space and the rows as vectors vs (b) transposing the matrix and vectorizing the columns over the rows. We use both vectorizations for the subsequent clustering task and use the terms *tagpath vectorization*, resp. *tagpath clustering*, and *term vectorization*, resp. *term clustering* for them. For term weighting in the tagpath vectorization, we have enhanced the TF-IDF weighting scheme into a function that reflects the relevance of the term for the application domain [4].

Tasks V and VI: Clustering and Cluster Labeling. XTREEM-SG performs clustering for sibling groups discovery, offering the option of either term clustering or tagpath clustering.

A cluster of term vectors consists of terms that co-occur in many tagpaths. Hence, the members of such a cluster constitute a sibling group. We obviously assume that there are no clusters with only one member. If the clustering algorithm allows for such clusters, then one-member clusters are ignored.

	great white shark	hammerhead shark	tiger shark	batoidea	pinniped	orca	...
DocumentA<html><body><h2>	**1**	**1**	**1**	**0**	**0**	**0**	...
DocumentB<html><body><table><h1>	1	0	1	0	0	0	...
DocumentC<html><body><p>...	0	0	0	1	1	0	...
...	...	...	...	...	...	...	...

Fig. 6. Matrix of tagpaths and terms for the example of Figure 5

The semantics of tagpath clustering are different. A cluster of tagpath vectors consists of tagpaths that have many terms in common. These terms are not members of the cluster, they are rather the dimensions/features, which *characterize* the cluster. These terms constitute the *label* of the cluster, which is built in the last task of XTREEM-SG.

More formally, the *label* of a tagpath cluster C is the group of terms that appear frequently in the members of the cluster, subject to a threshold τ. In particular, we define the *in-cluster-support* function $ics()$, which returns for each term/feature f and each cluster C the portion of tagpaths in C that contain f. Then, the *label* of C is the set of features, whose in-cluster-support in C exceeds a predefined lower boundary value τ, i.e. $L(C) = \{f \in F | ics(f, C) \geq \tau\}$. Each label that contains at least two terms constitutes a sibling group. One-term labels are ignored.

Tagpath clustering seems less straightforward with respect to sibling group discovery, because it demands cluster labeling. Term clustering, although more intuitive, has a serious disadvantage: A term, being a vector, can belong to only one cluster[3] and so can appear in at most one sibling group. Since a term may have multiple meanings or participate in multiple sibling relations in different contexts, forcing the term to belong to exactly one cluster is undesirable. Tagpath clustering alleviates this problem by allowing the same term to appear in multiple labels. In Section 5, we elaborate on the performance of XTREEM-SG with each type of clustering.

Details of the implementation: For both tagpath clustering and term clustering we have chosen the K-Means algorithm [21, 27]. K-Means is a widespread, easy to implement algorithm. As pointed out by Steinbach et al. [33], it has many limitations, but it is still appropriate as a first choice for experiments. We have also experimented with the more robust Bi-secting K-Means variant described by Steinbach et al. [33], but have obtained results of lower quality [3].

[3] We assume crisp clustering.

Determining the number of clusters K is a major challenge. In our experiments, we have varied the value of K and studied the influence of large values upon the algorithm's performance. Selecting a threshold for the in-cluster-support function is not straightforward either. We have performed experiments with different threshold values in subsection 5.4. For future work, we intend to model the gradual relevance of terms for the domain (as proposed among else by Novacek and Smrz [28]) and use it to choose the terms that may appear in the cluster label.

4 Evaluation Methodology

We have evaluated XTREEM-SG with respect to its ability to organize the terms of a given vocabulary V into sibling groups. Such an evaluation is challenging by nature: The objective of an unsupervised method like XTREEM-SG is to discover patterns that are not known a priori. For an evaluation of such a method, one usually considers a gold standard; in our case, this should be the sibling groups of a given ontology. However, no ontology can be assumed to express the ground truth in the sense of completely covering the application domain. Despite this challenge, we have designed our evaluation on the basis of reference ontologies. In particular, we use reference ontologies that provide the vocabulary of terms *and* the relations among them. We deliver the vocabulary to XTREEM-SG and expect to identify those sibling groups among them that are recorded in the ontology – notwithstanding that the ontology may be incomplete with respect to sibling groups.

4.1 Gold Standards for Evaluation

We use two gold standard ontologies (GSOs), both from the application domain of tourism. They are the "Tourism GSO" [4], which contains 293 concepts grouped into 45 sibling groups and the "Getess annotation GSO" [5], which contains 693 concepts and 90 sibling groups. We denote them as `GSO1` and `GSO2` respectively. We treat their concepts as "terms", so that each ontology also provides a vocabulary input to the process of sibling group discovery (cf. Task IV in subsection 3.1).

We stress here that the purpose of the experiments is not the reconstruction of the concept hierarchies in the reference ontologies but only the discovery of the sibling groups in them. Hence, although sibling groups are discovered, the naming of the concept describing them and the placement of the sibling groups in the concept hierarchy are beyond our scope.

4.2 Comparative Evaluation

We compare XTREEM-SG with two approaches. First, we extract sibling groups by applying the traditional Bag-of-Words document modeling, vectorization and

[4] http://www.aifb.uni-karlsruhe.de/WBS/pci/TourismGoldStandard.isa
[5] http://www.aifb.uni-karlsruhe.de/WBS/pci/getess_tourism_annotation.daml

labeling of document clusters; here, we consider individual tagpaths instead of documents. We refer to this baseline as the "BOW" model. Moreover, we use a variation of the method of Kruschwitz [25]. This approach uses markup of the Web documents and derives textspans. However, the textspans are not grouped. Moreover, both frequent and infrequent textspans are retained. We refer to this approach as "MU" for "M"ark-"U"p.

For the evaluation of XTREEM-SG, we consider the impact of different parameters. In particular, we vary the size of the document collection, the number of clusters to be discovered (and thus the number of anticipated sibling groups), the threshold value on term frequency for cluster labeling and the minimum frequency of the terms being considered.

4.3 Evaluation Criteria

Each of the gold standard ontologies contains a set of reference sibling groups. XTREEM-SG, BOW and MU deliver their own sets of candidate sibling groups. Intuitively, one would compare each candidate sibling group against each reference sibling group, select the best match and then count the number of common members between the candidate group and the reference group. Candidate sibling groups without match would be regarded as false positives. Reference sibling groups without match would also contribute to the error. However, selecting a *single* "best match" is neither straightforward nor is it always appropriate.

Example 4. In an ontology on tourism (or geography), all towns of the world are siblings under the concept "town of the world". Within this enormous reference sibling group, there are still many subsets of siblings that are conceptually closer to each other. Among them, one may consider (a) all towns in the same country, (b) all towns along the same river, (c) all towns having an airport, (d) all towns close to the *same* airport, (e) all towns with more than 1 million inhabitants, (f) all capital cities etc. It is quite likely to discover (from some supportive documents) that two towns are siblings according to one or more of the specific relations above. It is much less likely to discover that 3, 5 or 10 towns are siblings as "towns of the world". At the same time, finding that London and Tokyo are siblings for the relations (c), (e) and (f) is perhaps of more interest than finding out that Amsterdam, Cerbere, Hammerfest, Heraklion and and Kyoto are all towns of the world and thus siblings. ∇

This extreme example highlights a situation that is not uncommon in handcrafted ontologies, namely that not all concepts are refined in the same level of detail. Hence, it may happen that some concepts are very abstract and have a lot of children that are not really very related to each other (e.g. the towns of the world in Example 4), while other concepts are refined in more detail.

For our evaluation we need therefore a measure of the *contribution* of each candidate sibling group to each reference sibling group. We use the "F-Measure on Average Sibling Overlap" (FMASO) proposed by Cimiano and Staab [14].

Definition 3 (FMASO). *Let A and B be two sets of sibling groups. Typically, one of them, say A, will be the set of reference sibling groups, while the other,*

B, will contain the candidate sibling groups. For a reference sibling group $x \in A$ and a candidate sibling group $y \in B$, we compute the "relative overlap" between x and y as the number of common terms in the two groups divided by the number of distinct terms in the groups: $\frac{|x \cap y|}{|x \cup y|}$.

For each reference sibling group $x \in A$ we select the candidate sibling group $x' \in B$ that has the maximal relative overlap with x. This is the "sibling overlap" for x towards B: $SO(x,B) = \max_{y \in B} \frac{|x \cap y|}{|x \cup y|}$. Then, we compute the average of these values over the sibling groups in A as the "average sibling overlap" of A towards B:

$$ASO(A,B) = \frac{1}{|A|} \sum_{x \in A} \max_{y \in B} \frac{|x \cap y|}{|x \cup y|} \tag{1}$$

The average sibling overlap of B towards A is computed similarly as $ASO(B,A)$. Then, the "F-Measure on the average sibling overlap" FMASO combines the values of both functions as:

$$FMASO = \frac{2 \cdot ASO(A,B) \cdot ASO(B,A)}{ASO(A,B) + ASO(B,A)}$$

The FMASO measure partially deals with the problem highlighted in Example 4 by considering also partial matches between reference sibling groups and discovered sibling groups. Hence, the FMASO values for the mining methods will be more than zero, even if the ontology contains large groups of loosely related siblings, none of which can be found in the document collection as a whole. In fact, XTREEM-SG, BOW and MU are designed to finding sibling groups of higher support, so that the original large group can be split to groups that are semantically more compact.

The problem is not completely alleviated, though. If the reference ontology contains large sibling groups that cannot be reconstructed, then they still influence the values of the average sibling overlap, as shown in the example below.

Example 5. Assume an ontology that contains one sibling group x with 100 elements and 4 further sibling groups that contain two elements each. They constitute a set A. Assume also a method for sibling groups' discovery. It discovers all 4 sibling pairs, as well as 6 further pairs from group x, which may be overlapping. These pairs constitute a set B. According to the definition of sibling overlap in Def. 3, the sibling overlap for x is $SO(x,B) = \frac{2}{100}$, since 2 is the size of the largest group that matches x. For the other 4 sibling groups in A, the sibling overlap is 1. From Eq. 1, the average sibling overlap of A towards B is:

$$ASO(A,B) = \frac{1}{5} \cdot (4 \times 1 + \frac{2}{100}) = 0.804$$

For the average sibling overlap of B towards A, we have a contribution of the large group x to 6 pairs. For any pair among them, say $y \in B$, the sibling overlap is $SO(y,A) = \frac{2}{100}$ as before, since $x \cup y = x$. We have:

$$ASO(B,A) = \frac{1}{10}(4 \cdot 1 + 6 \cdot \frac{2}{100}) = 0.412$$

Then, the value of FMASO is:

$$FMASO = \frac{2 \cdot ASO(A, B) \cdot ASO(B, A)}{ASO(A, B) + ASO(B, A)} = 0.545$$

We now assume a method that returns a set B' containing all groups in B and 10 further sibling pairs from group x. This means that this method has found 16 sibling pairs from the large reference group. The value of $ASO(A, B')$ is equal to $ASO(A, B)$, while:

$$ASO(B', A) = \frac{1}{20}(4 \cdot 1 + 16 \cdot \frac{2}{100}) = 0.216$$

so that:

$$FMASO = \frac{2 \cdot ASO(A, B') \cdot ASO(B', A)}{ASO(A, B') + ASO(B', A)} = 0.34$$

We can see that an increase in the number of matches for the large group has a negative influence on the value of FMASO. Essentially, the large difference between the size of the reference group x and the size of the sibling groups matching is not completely alleviated. ∇

A further unresolved issue in our evaluation concerns the treatment of terms that participate in multiple sibling groups. First, a term may have more than one meanings (in our Example 4 above, there is one town Paris in France and one in Texas). Second, there may be sibling groups of different semantics; in Example 4, the terms/towns London and Tokyo are siblings under concepts (c), (e) and (f). Our reference ontologies do not support multiple inheritance, so terms may co-occur in only one group. This means that some of the false positives are not really false; rather, the ontologies are too restrictive with respect to reality. We point to this issue, but we cannot provide a remedy for it. Concerning the first case, in which a term may have multiple meanings, this is of little practical relevance for the small vocabularies of our reference ontologies.

5 Experiments

We have compared XTREEM-SG with the simplistic Bag-of-Words method (BOW) and with a Mark-Up extraction method (MU) that identifies textspans using the markup but does not group them nor does it eliminate infrequent ones. For the experiments, we have first built a large archive on the application domain "tourism" and then extracted subarchives from it by issuing more specific queries. We first describe how the Web collections were built. Then, we describe each experiment in turn.

5.1 Experimental Environment

Building the Web document collections. We have first performed a focused Web crawl and collected approximately 9.5 million Web documents on

Table 1. Queries and corresponding Web document collections

Query Name	Query Phrase	Number of Documents
Query1	"touris*"	1,468,279
Query2	"accommodation"	1,612,108
Query3	"*"	9,437,703

subjects associated to tourism. We have converted the documents to XHTML. Then, we applied an n-gram based language recognizer to eliminate non-English documents [6], because our reference ontologies are on English. Then, we indexed the documents so that the domain-specific queries could run against the archive instead of the Web. This guaranteed that all our experiments were run on exactly the same documents; this could not have been guaranteed in the live and rapidly evolving Web.

The queries issued against the initial Web document collection are depicted in Table 1. *Query*1 with the keyword+wildcard "touris*" returned documents on tourism, tourists, touristic attractions etc. *Query*2 returned documents on "accommodation". Finally, *Query*3 corresponds to the whole initial collection; we denote its keyword as "*". The number of documents returned by each query are shown in the Table.

Task III of the basic XTREEM approach (cf. subsection 3.1) juxtaposes the textspans to a vocabulary of terms. For our experiments, this vocabulary was provided by the gold standard ontologies described in subsection 4.1. Both the "Tourism GSO" (GSO1) and the "Getess annotation GSO" (GSO2) are lexical ontologies, so we used their concepts as terms.

We have studied different aspects of XTREEM-SG, comparing it to BOW and MU, varying parameters like the number of clusters and the cluster labeling threshold, and juxtaposing term clustering to tagpath clustering. The experiments are summarized in Table 2 and discussed in sequel.

Table 2. Outline of Experiments

	Method	Clustering	Queries	Varied parameter
1	GBP, BOW, MU	none (XTREEM tasks only)	Query1	
2	XTREEM-SG, BOW, MU	Tagpath clustering	Query1	Number of clusters K
3	XTREEM-SG	Tagpath clustering	Query1	K, Labeling threshold
4	XTREEM-SG	Tagpath clustering	Query1	K
5	XTREEM-SG	Tagpath clustering	all	K
6	XTREEM-SG	Tagpath clustering	Query1	K, Term frequency threshold
7	XTREEM-SG		Query1	Tagpath vs. term clustering

5.2 Experiment 1: Evaluating the Preprocessing Tasks

In our first experiment we want to investigate the extent to which sibling groups are captured by the Group-By-Path (GBP) method in contrast to the traditional

[6] http://wiki.apache.org/nutch/LanguageIdentifier

Table 3. Number of sibling groups and achieved FMASO value for each method

Ontology	Method	Number of sibling groups			FMASO
		no match	one term	multiple terms	
GSO1	BOW	18,012	29,104	1,421,163	0.206
	XTREEM-SG	12,589,016	817,289	222,037	0.247
	MU	794,325	343,891	323,428	0.235
GSO2	BOW	19,399	18,494	1,430,386	0.160
	XTREEM-SG	12,478,364	831,969	318,009	0.208
	MU	753,657	332,973	375,014	0.199

Bag-Of-Words (BOW) vector space model and to the usage of markup only (MU). To do so, we run only XTREEM (Tasks I, II, III) and tested the GBP step with BOW, respectively MU. We evaluated the collections of sibling sets for $Query1$ agains the reference sibling sets (GSO1 and GSO2) of two gold standard ontologies. Since the two ontologies have different numbers of terms, XTREEM generates a different number of candidate sibling groups for each one.

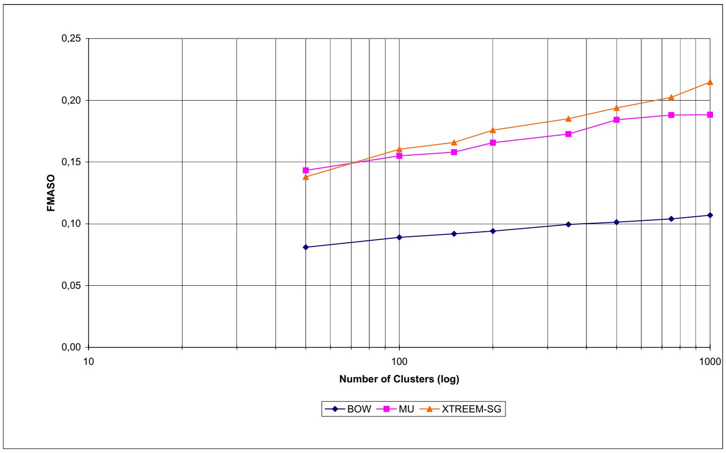
In Table 3, we show the number of candidate sibling groups returned by each method for each reference ontology. We distinguish among sibling groups that have no match in the ontology (column "no match"), those that have only one common member (column "one term") and those that have multiple common terms. Candidate sibling groups with less than two terms in the reference ontology are ignored, since they cannot contribute to sibling relations. These groups are ignored when computing the FMASO value in the last column.

The results in Table 3 show that XTREEM-SG achieves the highest FMASO values, although it generates less sibling groups with two or more terms than the other methods. Indeed, BOW builds relatively few degenerate sibling groups (groups without a match or groups with only one matching term) and many more candidate sibling groups than XTREEM-SG and MU. Nonetheless, the contribution of those groups to sibling relations according to FMASO is low. The methods MU and XTREEM-SG build comparable numbers of non-degenerate candidate sibling groups, but XTREEM-SG performs better for both ontologies.

Conclusion of Experiment 1: XTREEM performs better than the two alternative methods on the discovery of sibling groups. Admittedly, BOW is a naive method. However, the comparative superiority of XTREEM with GBP indicates that the discovery of sibling groups on the sole basis of markup is not coincidencial but a viable alternative.

5.3 Experiment 2: Sibling Groups Discovery - Tagpath Clustering

In this experiment, we have compared the performance of XTREEM-SG with GBP, BOW and MU on $Query1$, using tagpath clustering. We have set the cluster labeling threshold to $\tau = 0.2$ and have varied the number of clusters from 50 to 1000: $K = 50, 100, 150, 200, 250, 500, 750, 1000$.



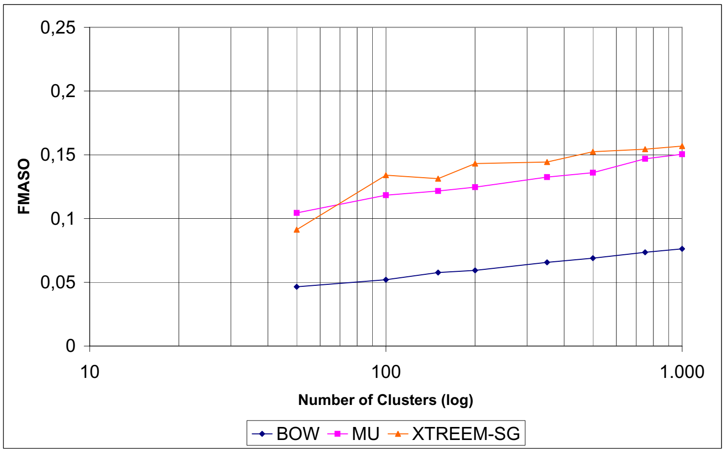
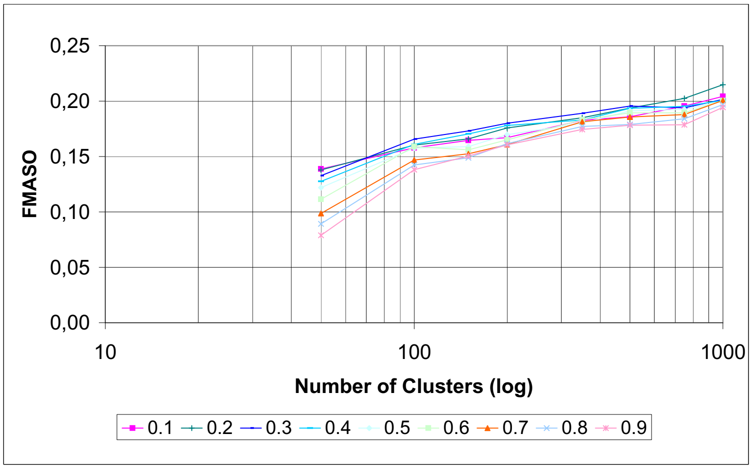


Fig. 7. Tagpath clustering with XTREEM-SG, BOW and MU

The results are in Figure 7, where the upper subfigure refers to GSO1 and the lower subfigure to GSO2. We use this convention in the following experiments as well.

We can see in the Figure 7 that FMASO improves for all methods as the number of clusters increases. Since the scale is logarithmic, the curves indicate an exponential upward trend. The influence of the number of clusters on the performance is studied further later (Experiment 4).

We observe from Figure 7 that XTREEM-SG have comparable performance, while BOW is inferior. A possible explanation is that sibling groups of MU and of XTREEM-SG are overlapping to a large extent, because they are both based on the rather small vocabularies of GSO1 and GSO2. Their coverage towards the real world of the Web is low. Hence, FMASO counts as false positives all sibling groups that are not found in the ontology, although some of them may be very reasonable.



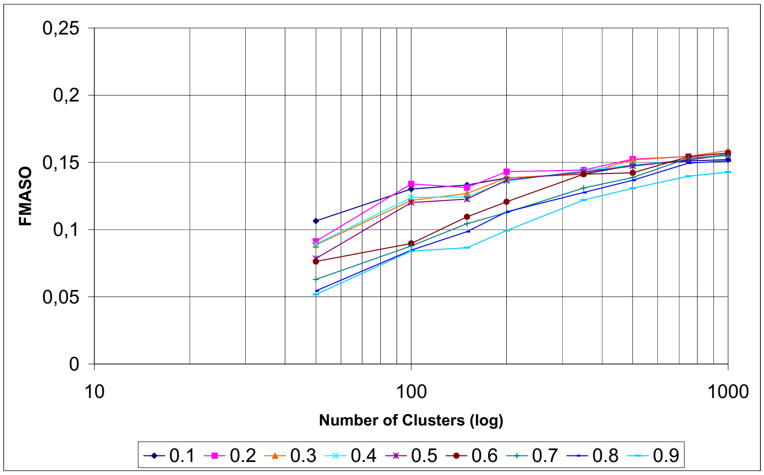


Fig. 8. Performance of XTREEM-SG for various cluster labeling thresholds

Conclusion of Experiment 2: The results of Experiment 2 are in accordance with those of Experiment 1. XTREEM-SG outperforms BOW and the sibling groups it delivers are semantically meaningful, according to FMASO.

5.4 Experiment 3: Impact of the Cluster Labeling Threshold

In this and the following experiments, we concentrated on studying the behaviour of XTREEM-SG. We performed tagpath clustering on the Web document collection of *Query*1 and varied the in-cluster-support threshold (cf. Subsection 3.2, Task VI) from $\tau = 0.1$ to $\tau = 0.9$ in steps of 0.1. The results are in Figure 8.

Similarly to Experiment 2, we also varied the number of clusters K from 50 to 1000. In Figure 8, K is replaced by its logarithm. The best results are acquired for the largest value of K and for relatively low support thresholds. For GSO1, the value of FMASO reaches 21.47% when $\tau = 0.2$ (Figure 8, upper subfigure), while the highest value for GSO2 is only 15.88% ($\tau = 0.3$). A possible explanation is again the low coverage of the ontologies with respect to the real world: Some of the detected sibling groups may be semantically correct, but they are not in the corresponding ontology. This forces FMASO to consider them as false positives.

Conclusion of Experiment 3: The contribution of discovered sibling groups to the value of FMASO increases with the number of candidate groups (reflected in the value of K) and in the size of the groups (reflected in the labeling threshold). While it is expected that a larger quantity may have a positive contribution to quality, it is pointed out that the large number of candidate groups returned by BOW did not result in high FMASO values (cf. Table 3). Nonetheless, we intend to study methods for reducing the number of returned candidate sibling groups or for ranking them.

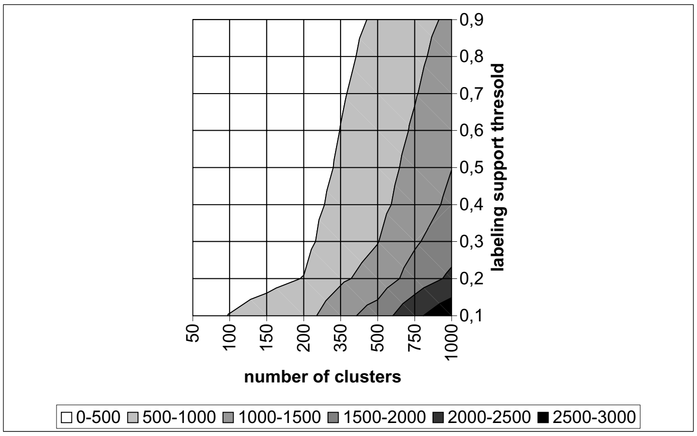
5.5 Experiment 4: Varying the Number of Clusters

In Experiments 2 and 3, we have observed that the value of FMASO increases with the number of clusters. This may indicate a bias of FMASO towards large numbers of sibling groups. In any case, the generation of many candidate sibling groups is problematic for a real-world scenario, because these groups must be inspected by the human expert. So, in this experiment, we count the number of terms that must be inspected by the ontology engineer, who decides whether a sibling relation is worth inserting in the ontology or not.

To this purpose, we enumerate the terms/features appearing in the labels of all clusters in one clustering (for given K). We denote as NODFICL (*Number of Distinct Features in Cluster Labels*) the number of terms from the vocabulary that are involved in sibling groups.

Furthermore, we sum all appearances of each feature over all labels and compute the “Sum of appearances of Features in Cluster Labels” (SOFICL). SOFICL reflects the participation of terms in multiple sibling groups – a desirable phenomenon. However, if SOFICL is very large and NODFICL is very low, this indicates that most of the sibling groups are based on a few number of terms (possibly terms with multiple meanings), while the rest of the vocabulary is not exploited.

In the upper part of Figure 9, we capture the values of SOFICL for different numbers of clusters and labeling thresholds. We concentrate on GSO1 and on the Web document collection of $Query1$. In the lower part of the Figure, we show the corresponding values for NODFICL. Despite the different computation basis, the correlation between NODFICL and SOFICL can be seen. However, NODFICL assumes low values (no more than 190), indicating that the sibling groups are built from a relatively small number of features. An explanation is that many features are very rare and cannot appear in labels, even if the in-cluster-support is set to very low values.



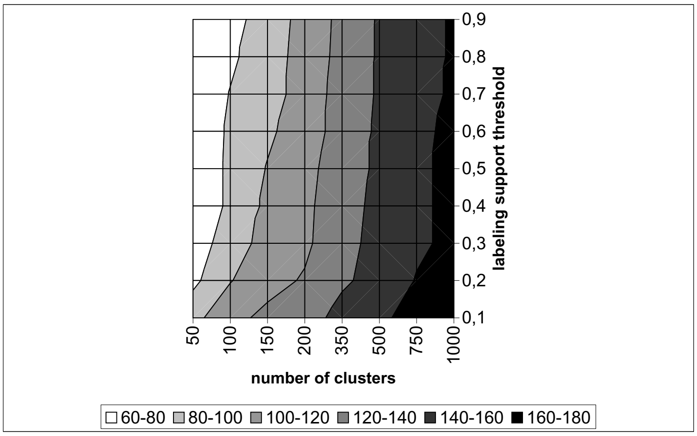
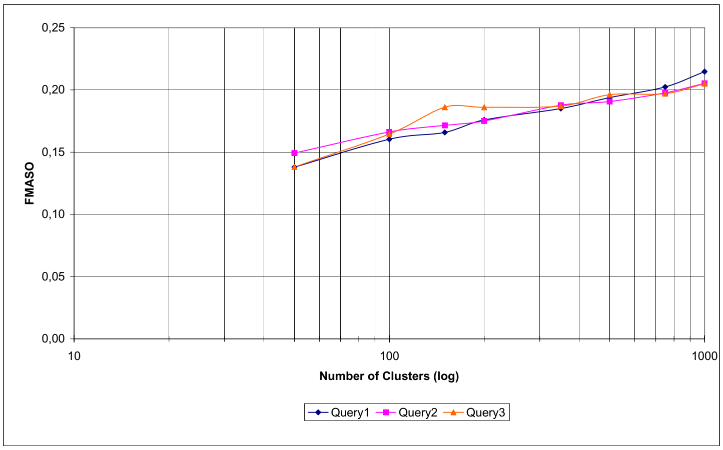


Fig. 9. Appearances of GSO1 terms in sibling groups (upper part) and number of distinct GSO1 terms in sibling groups (lower part)

5.6 Experiment 5: Impact of the Size of the Document Collection

In the previous experiments, we have concentrated on the Web document collection returned by $Query1$. We now study the behaviour of XTREEM-SG for all three queries in Table 1. We have used again tagpath clustering, varying the number of clusters K from 50 to 1000. The cluster labeling threshold was set to $\tau = 0.2$. The values of FMASO for different values of K (in logarithmic scale) are depicted in Figure 10.

The curves in Figure 10 show the same trend for all three queries. For the small ontology GSO1 (upper subfigure), the FMASO curves are very close to each other. Since the Web collection of $Query3$ is a superset of the other two, this indicates that a big collection does not necessarily improve the results. For the large ontology GSO2 (lower subfigure), the picture is slightly different: As



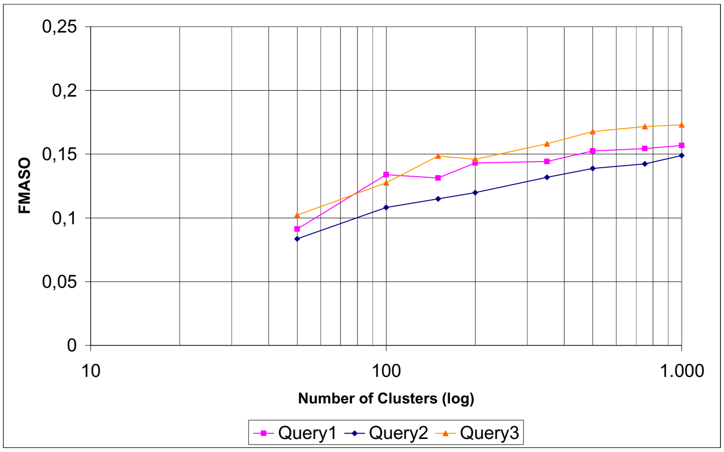
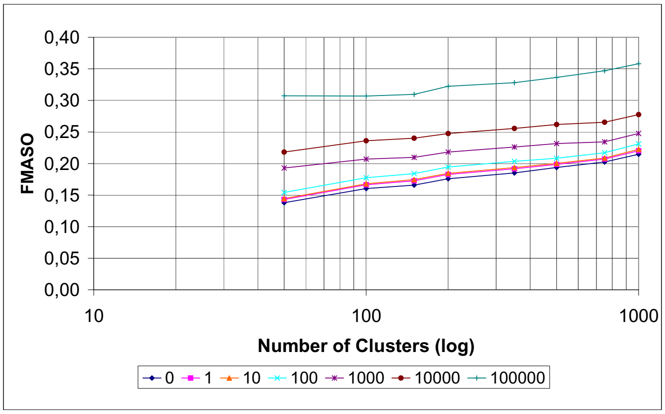


Fig. 10. Performance of XTREEM-SG with tagpath clustering for different queries

the number of clusters K increases, the performance of XTREEM-SG is lowest for the documents of *Query*2 (the smallest and most specialized Web document collection) and best for *Query*3 (the largest collection). One possible explanation is the difference in the sizes of the two ontologies: A large collection is likely to deliver more false positives towards the smaller ontology. The "turning-point" for this phenomenon seems to be K=150.

5.7 Experiment 6: Frequency Thresholds for the Vocabulary Terms

In a preliminary experiment on the coverage of our Web document collections, we discovered that some of the terms in our reference ontologies appear only rarely or not at all in our collections. Given the size of the collections even for the smallest query *Query*2 (cf. Table 1), this is unexpected. This may have



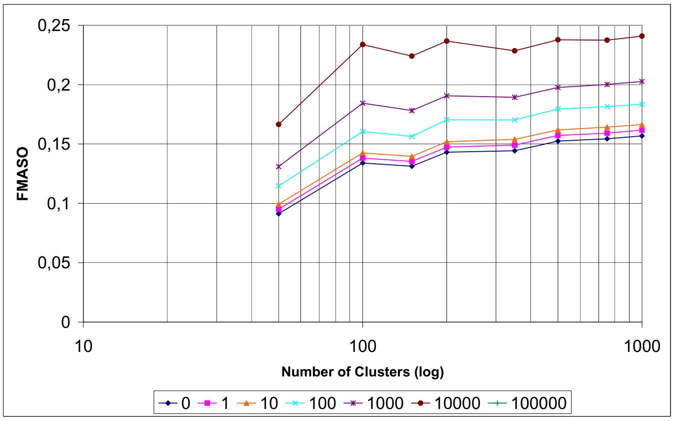


Fig. 11. Varying the frequency thresholds for the vocabulary terms

different causes: First, ontology terms imported to the English language from other languages may be misspelled (`Kindergarden` instead of `Kindergarten`). Second, some concepts of the ontology may have no counterpart term, for example `female_male` as superconcept of `female` and `male`. Obviously, such a concept may participate in a sibling group; however, the sibling group *of terms* will only contain terms, no abstract concepts.

To check the existence and impact of abstract concepts and made-up words, we have introduced a term frequency threshold $\tau_{frequency}$ over the collection and removed all terms of the vocabulary that were less frequent. In Figure 11, we show the performance of XTREEM-SG with tagpath clustering (for various K), when $\tau_{frequency}$ is 0, 1, 10, 100, 1000, 10000, 100000 appearances. We see that the performance (FMASO) improve consistently as $\tau_{frequency}$ increases.

A word of caution is due here: A frequency threshold is appropriate for the *evaluation* of an algorithm, since it may alleviate some inherent problems of the reference ontologies. However, it is less appropriate for a real-world scenario, where the expert is interested in sibling groups involving both frequent and infrequent terms.

5.8 Experiment 7: Tagpath Clustering vs. Term Clustering

In Experiment 1, we have studied the behaviour of XTREEM-SG with term clustering. In Experiments 2-6, we have used tagpath clustering. We now look closer at the influence of the type of clustering upon the performance of XTREEM-SG. In particular, we vary the number of clusters for both types of clustering and study (a) the number of sibling groups returned for each value of K and (b) the corresponding value of the FMASO quality measure. Since a clustering result also depends on the initial centroids, we run each type of clustering with 10 seeds for each value of K. We worked with the Web document collection of $Query1$.

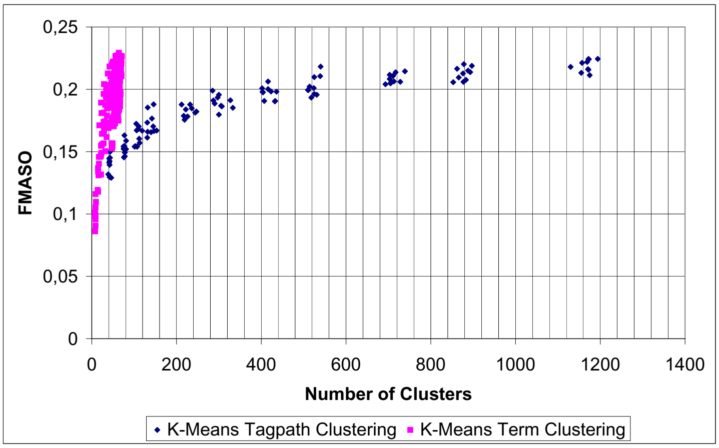
It should be stressed here that the number of sibling groups returned for a given value of K is not necessarily equal to K. First, K-Means may generate empty clusters, which are obviously ignored in our evaluation. Degenerate clusters with only one member are also ignored; in term clustering, they would correspond to sibling groups with only one term. Second, under tagpath clustering we ignore clusters without label and clusters with only one term in the label. Furthermore, if two clusters happen to have the same label, we count only one sibling group for them.

For tagpath clustering, we vary the value of K from 50 to 1400. This is in accordance with our previous findings that indicate a performance improvement when many clusters are built. The in-cluster-support threshold for cluster labeling τ is set to 0.2.

For term clustering, we start with K=10 and increase the value in steps of 10, until no additional clusters can be built. The reason is that the vectors are the terms themselves, so it makes no sense to build more clusters than there are terms. For the small ontology GSO1, which contains 293 terms, K-Means has build no more than 80 candidate sibling groups. For the larger ontology GSO2, up to 150 candidate sibling groups have been derived.

Figure 12 shows how the FMASO values vary over the number of clusters for tagpath clustering vs. term clustering. We see that term clustering returns sibling groups of higher quality than tagpath clustering, while building a smaller number of clusters. For small numbers of clusters, the quality of the two approaches is similar. As the number of clusters K increases, the quality under term clustering increases faster, while the quality under tagpath clustering saturates at a lower FMASO value – even for very large values of K.

In Table 4, we show the number of candidate sibling groups computed for the largest FMASO values. We further show the number of term appearances in the candidate sibling groups, whereby we should keep in mind that SOFICL and NODFICL are equal for term clustering. It is clear that under term clustering, the human expert needs to inspect much less candidate sibling groups (and terms) than under tagpath clustering, while acquiring results of higher quality.



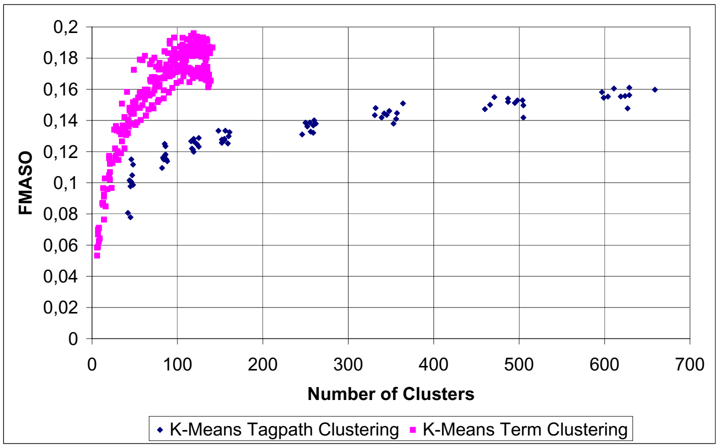


Fig. 12. Impact of K under tagpath clustering vs. term clustering

Table 4. Number of sibling groups and of term appearances in tagpath clustering and in term clustering

	Clustering type	FMASO	Candidate sibling groups	SOFICL
GSO1	term clustering	22.93%	64	253
	tagpath clustering	21.47%	545	2093
GSO2	term clustering	19.59%	119	627
	tagpath clustering	15.88%	627	2249

Conclusion of Experiment 7: XTREEM-SG with term clustering returns a smaller number of candidate sibling groups with a higher overall quality than those with tagpath clustering. Term clustering does not require a label threshold for the in-cluster-support of terms; even rare terms may appear in a candidate sibling group. Since a term may appear in at most one sibling group, we do not have the phenomenon of multiple, largely overlapping candidate sibling groups.

By this, we can conclude that term clustering is the superior option. However, we should keep in mind that under the term clustering option, a term can belong to at most one candidate sibling group, while in a real-world scenario a term may belong to many sibling groups. Hence, the superiority of term clustering may be (at least partially) due to the limited coverage of the reference ontologies towards the real world: Term clustering delivers less combinations of terms and thus less candidates for false positives.

5.9 A Set of Candidate Sibling Groups

We close the section of experiments with a set of candidate sibling groups for GSO2, depicted in Figure 13. We have applied XTREEM-SG with term clustering, setting K=200. The algorithm returned 128 sibling groups with at least two terms. In this Figure, we show 117 of these sibling groups, built upon 463 of the 693 terms. As can be seen, there are many candidate sibling groups, where the semantics of the sibling relation can be easily recognized. However, there are also groups with unclear semantics – including the 11 skipped groups.

5.10 Summary of Evaluation

We have evaluated XTREEM-SG with tagpath clustering and term clustering on two reference ontologies for different parameter settings. We have also compared our approach with the Bag-of-Words baseline (BOW) and with an algorithm that discovers sibling groups using HTML markup but does not consider the tagpaths. For our evaluation, we have used the FMASO quality measure on average sibling overlap proposed by Cimiano and Staab [14]. Our experiments show that XTREEM-SG is superior to the other methods and reaches a quality of more than 20%. The performance increases as the number of generated candidate sibling groups increases, but this leads to an undesirable overhead for the human engineer, who has to inspect the groups manually. We have shown though that XTREEM-SG with term clustering achieves higher quality than tagpath clustering, while generating substantially less candidate sibling groups for human inspection.

The absolute values for the average sibling overlap measure are low, never exceeding 30%. This can be attributed to the low coverage of the human-crafted ontologies towards the huge Web document collections we use. Nonetheless, we are still faced with the questions of whether we can indeed term the performance quality of XTREEM-SG as satisfactory. To this end, we have studied the quality performance of comparable approaches in the literature.

Cimiano and Staab have proposed an approach for sibling group discovery from Web documents [14] and have compared it with the earlier method of Caraballo [11] on the Tourism reference ontology. For the evaluation, they have used the F-measure on average sibling overlap FMASO. They have shown that Caraballo's method achieves a value of 8.96%, while their approach reaches values between 2.40% and 14.18% for different parameter settings [14].

The counterpart of the experiments of Cimiano and Staab are our experiments with the same ontology, i.e. the GSO1. Our Web document collection is

apartment	badminton	monastery	backwater	banquet	city	bargain
castle	bowling	old_town	bicycle_tour	conference	country	relaxing
guest_house	cleaning	city_museum	horse_tour	congress	state	sight
hunting_castle	fitness_studio	culture	journey	exhibition	time_interval	time
motel	handball	night_life	tour	guided_tour	town	midnight
pension	ice_hockey	shopping	tour_guide	night	back_massage	snack
sea_side	squash	sightseeing	booking	park	body_massage	thing
sightseeing_flight	table_tennis	talk	reservation	seminar	face_massage	winter_garden
sightseeing_tour	baby	menu	summer_holidays	sleep	water_gymnastics	crane
table_tennis_racket	club	mouse	eating	workshop	agent	stork
continent	group	dog	harbour_city	youth_hostel	camp	climbing
street	shore	dog_care	panorama_view	aunt	cross_country_ski_run	hang_gliding
actor	camping	horse	spare_time	bat	ski_run	skiing
face_mask	downhill_ski	ox	cliff	daughter	cinema	airport
forest	hiking	picnic	hike	grandchild	theatre	sports_holiday
mountain	kiosk	race	hiker	grandfather	hairdresser	train_station
peninsula	open_air_theater	rat	racket	regimen	secretary_service	suburb
river	tennis_lessons	afternoon	soloist	sibling	sport_shop	adventure_holiday
valley	wellness_offer	day	table_tennis_table	son	early_season	family_holiday
desert	autumn	morning	tennis_ball	uncle	main_season	holiday_home
lake	spring	week	tennis_racket	mono_ski	off_season	holiday_special
adventure	summer	beer_garden	aerobic	vista_point	season	billiard_room
pilgrimage	winter	skittle_alley	billiard	water_ski	easter_holiday	sanatorium
dinner	matinee	spit_of_land	billiard_equipment	ballet	holiday	sledge
farewell_dinner	moor	cross_country_skiing	cycler	dolphin	ski	therm
lunch	invoice	ice_skating	exchange_office	elephant	excursion	bowling_alley
breakfast	standardization	snowboard	fitness_training	giraffe	wedding	library
brunch	basketball	snowboarding	concert	mall	east_shore	beach_promenade
buffet	cycling	area	dancing	monkey	east_side	riding
fast_food	diving	floor	event	ball	north_side	car_rental
sport_equipment	football	bird	festival	boy	south_side	hotel_garden
beauty_day	golf	human_activity	musical	female	west_side	shopping_tourism
make_up	sailing	saltwater_fish	business_people	girl	cultural_activity	camping_ground
permanent_make_up	soccer	tree	dog_service	kin	guest	cruise
cafe	swimming	vesper	sports_facilities	male	village	hotel
hair_dresser	tennis	agency	visiting	person	driver	pub
harbour_area	volleyball	drive	working	relax_weekend	musician	winter_holiday
bank	barbecue	family	day_time	teenager	art_exhibition	beach
embankment	cleaning_service	tourist	island	accommodation	pageant	basilica
fortress	garden	organic_food	region	culture_tourism	panorama	castle_complex
organizer	sea_view	table_tennis_ball	traveling	shop	day_trip	cathedral
holiday_village	tennis_court	vegetarian_food	service	tourist_information	digestive	city_wall
relaxing_holiday	town_centre	cosmetic_care	hill	traveling_by_air	heat	national_art_gallery
side	climbing_wall	foot_care	midday	baby_sitter_service	cabaret	oratory
top_hotel	horse_riding_lessons	manicure	national_park	billiard_table	mini_golf	gym
yacht_port	horse_riding_school	hair_cut	nature_reserve	disco	playground	health_club
yacht_rental	cultural_institution	nail_design	art_gallery	sports_hall	thermal_bath	swimming_pool
bay	gallery	fango	christmas_special	air	action	circus
cape	market_place	fitness_room	golf_course	care	animal	zoo
adult	beach_view	indoor_swimming_pool	museum	cat	donkey	provider
child	north_shore	mini_golf_area	weekend_special	church	mammal	snow
pensioner	pedestrian_area	sauna	whirl_pool	communication	plant	district
cycling_tour	promenade	solarium	breakfast_buffet	farm	formula_one	walking_trail
weekend_trip	south_shore	steam_bath	dinner_buffet	father	overnight_stay	contract
city_harbour	avenue	surf_school	surfboard_rental	frontier	short_trip	employee
harbour	ball_game	ground_floor	badminton_court	level	art	gala_dinner
view	ball_room	sport_offer	football_pitch	mother	performance	organization
advent	billiard_ball	upper_floor	squash_court	offer	boat_rental	program
christmas	coastal_resort	bungee_jumping	instruction	opera	catering	registration
easter	fish	city_trip	water	pigeon	intangible	parachuting
holiday_place	fishing_equipment	fishing	facial_therapy	place	ruin	water_sport
dance	fishing_rod	hunting	massage	produce	canyon	hand_care
dancing_night	nature	professional_sportsman	shooting_range	walk	temple	pork
music	attic	casino	fresh_water	act	brother	puppet_theatre
sport	diving_station	dune	saltwater	gourmet	cultural_event	symposium
	metropolis	sand	surfboard		gala	

Fig. 13. A selection of candidate sibling groups found by XTREEM-SG with term clustering

not identical to their text document collection. However, our Experiment 5 (cf. subsection 5.6) shows that XTREEM-SG shows the same trend independently of the collection size. Our results from the same experiment on GSO1 show that the values of the FMASO measure are always above 10% (upper subfigure of Figure 10). The best FMASO value for XTREEM-SG for GSO1 is 22.53%, as shown in Table 4. When tagpath clustering is used instead of term clustering, the performance is lower but still above 20% (same Table, second row). Hence, XTREEM-SG achieves substantially higher quality towards the reference ontology than the method of Cimiano and Staab, as reported in [14].

All methods evaluated towards a small reference ontology suffer from its coverage limitations towards the Web. Thus, we suspect that quality results close to 100% are more likely for large ontologies, which reflect the richness of the whole Web in semantics. Our work is a contribution to the establishment of such ontologies.

6 Conclusions and Future Work

We have presented XTREEM-SG, a method that discovers semantic sibling groups of terms by exploiting the structural conventions expressed in the markup of Web documents. XTREEM-SG takes as input a set of domain-specific keywords and a vocabulary of terms, for which sibling relations should be found. Sibling group discovery is then performed upon a Web document collection that satisfies the keywords, i.e. a collection of high coverage and low precision.

By working on Web documents, without demands for linguistic preprocessing or NLP resources, XTREEM-SG is independent of language and domain. Of course, NLP advances and resources can be exploited, if available, for example to perform term disambiguation. However, the main advantage of XTREEM-SG is that it contributes to semi-automated ontology engineering even for applications where no resources nor corpora are available.

We have evaluated XTREEM-SG with two reference ontologies, studying the impact of various parameters on its performance. The experiments have shown that XTREEM-SG builds sibling groups of good quality, producing substantially better results than other methods in the literature. We have seen that the quality improves as more sibling groups are generated and we expect that this holds in a real-world scenario much more than towards a reference ontology of low coverage.

XTREEM-SG has returned best results when applying term clustering. However, term clustering has the disadvantage that a term may be assigned to at most one sibling group. In a real-world scenario, a term is likely to belong to several different sibling groups, either due to homonymy or to the existence of several contexts. Therefore, we are interested in studying soft clustering approaches, thus allowing a term to be assigned to multiple clusters/sibling groups.

Tagpath clustering allows the assignment of a term to multiple sibling groups. However, the performance of XTREEM-SG with tagpath clustering is best when the number of candidate sibling groups is very high. In a real-world scenario, these groups must be manually inspected. So, it is desirable to provide methods that either reduce the number of candidates or rank them on quality. A further approach worth investigating is agglomerative clustering: Such a method would organize the candidate sibling groups automatically into a hierarchy. However, it would have the disadvantage of assigning a term to exactly one branch. The pros and cons of dendrograms must be therefore evaluated carefully.

In the preprocessing tasks of XTREEM, we have considered all XHTML markup that can be found in documents. In future work, we want to investigate the impact of individual tags and check whether some tags may contribute more to sibling group discovery than others.

References

[1] Agirre, E., Ansa, O., Hovy, E.H., Martínez, D.: Enriching very large ontologies using the WWW. In: Staab, S., Maedche, A., Nedellec, C., Wiemer-Hastings, P. (eds.) Proceedings of ECAI Workshop on Ontology Learning, Berlin, Germany. CEUR Workshop Proceedings, vol. 31, CEUR-WS.org (August 2000)

[2] Aussenac-Gilles, N., Jacques, M.-P.: Designing and evaluating patterns for ontology enrichment from texts. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS, vol. 4248, pp. 158–165. Springer, Heidelberg (2006)

[3] Brunzel, M.: Learning of semantic sibling group hierarchies - k-means vs. bisecting-k-means. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 365–374. Springer, Heidelberg (2007)

[4] Brunzel, M., Spiliopoulou, M.: Domain relevance on term weighting. In: Kedad, Z., Lammari, N., Métais, E., Meziane, F., Rezgui, Y. (eds.) NLDB 2007. LNCS, vol. 4592, pp. 427–432. Springer, Heidelberg (2007)

[5] Brunzel, M., Spiliopoulou, M.: Discovering semantic sibling associations from Web documents with XTREEM-SP. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 469–480. Springer, Heidelberg (2006)

[6] Brunzel, M., Spiliopoulou, M.: Discovering semantic sibling groups from Web documents with XTREEM-SG. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS, vol. 4248, pp. 141–157. Springer, Heidelberg (2006)

[7] Brunzel, M., Spiliopoulou, M.: Discovering multi terms and co-hyponymy from XHTML documents with XTREEM. In: Nayak, R., Zaki, M.J. (eds.) KDXD 2006. LNCS, vol. 3915, pp. 22–32. Springer, Heidelberg (2006)

[8] Brunzel, M., Spiliopoulou, M.: Acquiring semantic sibling associations from Web documents. International Journal of Data Warehousing and Mining (IJDWM) 3(4), 83–98 (2007)

[9] Buitelaar, P., Cimiano, P., Magnini, B.: Ontology Learning from Text: Methods, Evaluation and Applications. Frontiers in Artificial Intelligence and Applications Series, vol. 7. IOS Press, Amsterdam (2005)

[10] Buttler, D.: A short survey of document structure similarity algorithms. In: Arabnia, H.R., Droegehorn, O. (eds.) Proceedings of the International Conference on Internet Computing (IC 2004), Las Vegas, Nevada, USA, pp. 3–9. CSREA Press (June 2004)

[11] Caraballo, S.A.: Automatic construction of a hypernym-labeled noun hierarchy from text. In: Proceedings of the 37th annual meeting of the Association for Computational Linguistics (ACL 1999), Morristown, NJ, USA, pp. 120–126. Association for Computational Linguistics (1999)

[12] Choi, I., Moon, B., Kim, H.-J.: A clustering method based on path similarities of XML data. Data & Knowledge Engineering (DKE) 60(2), 361–376 (2007)

[13] Cimiano, P.: Ontology Learning and Population. PhD thesis, Universität Karlsruhe, Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB) (2006)

[14] Cimiano, P., Staab, S.: Learning concept hierarchies from text with a guided agglomerative clustering algorithm. In: Biemann, C., Paas, G. (eds.) Proceedings of the ICML 2005 Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods, Bonn, Germany (August 2005)
[15] Cimiano, P., Staab, S.: Learning by googling. SIGKDD Explorations 6(2), 24–33 (2004)
[16] Dalamagas, T., Cheng, T., Winkel, K.-J., Sellis, T.: A methodology for clustering XML documents by structure. Information Systems 31(3), 187–228 (2006)
[17] Ehrig, M., Maedche, A.: Ontology-focused crawling of Web documents. In: Proceedings of the 2003 ACM symposium on Applied computing, pp. 1174–1178. ACM Press, New York (2003)
[18] Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Webscale information extraction in knowitall (preliminary results). In: Proceedings of the 13th International Conference on World Wide Web (WWW 2004), pp. 100–110. ACM Press, New York (2004)
[19] Faatz, A., Steinmetz, R.: Ontology enrichment with texts from the WWW. In: Proceedings of the First International Workshop on Semantic Web Mining at the ECML 2002, Helsinki, Finland (2002)
[20] Gómez-Pérez, A., Manzano-Macho, D.: A survey of ontology learning methods and techniques. deliverable 1.5, OntoWeb project, universidad polytecnica de madrid. Technical Report OntoWeb Deliverable D1.5, Universidad Polytecnica de Madrid (May 2003)
[21] Hartigan, J.A., Wong, M.A.: A K-means clustering algorithm. Applied Statistics 28, 100–108 (1979)
[22] Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th conference on Computational linguistics, Morristown, NJ, USA, pp. 539–545, Association for Computational Linguistics (1992)
[23] Heyer, G., Läuter, M., Quasthoff, U., Wittig, T., Wolff, C.: Learning relations using collocations. In: Proceedings of IJCAI 2001 Workshop on Ontology Learning (OL 2001), Seattle, USA. CEUR Workshop Proceedings, vol. 38, CEUR-WS.org (August 2001)
[24] Kashyap, V.: Design and creation of ontologies for environmental information retrieval. In: Proceedings of 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW 1999), Banff, Alberta, Canada (October 1999)
[25] Kruschwitz, U.: Exploiting structure for intelligent Web search. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS 2001), Washington, DC, USA. IEEE Computer Society, Los Alamitos (2001)
[26] Kruschwitz, U.: A rapidly acquired domain model derived from markup structure. In: Proceedings of the ESSLLI 2001 Workshop on Semantic Knowledge Acquisition and Categorization, Helsinki, Finland (2001)
[27] MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkley Symposium on Mathematics and Probability, pp. 281–297 (1967)
[28] Novacek, V., Smrz, P.: Empirical merging of ontologies a proposal of universal uncertainty representation framework. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 65–79. Springer, Heidelberg (2006)
[29] Pasca, M.: Finding instance names and alternative glosses on the Web: Wordnet reloaded. In: Gelbukh, A. (ed.) CICLing 2005. LNCS, vol. 3406, pp. 280–292. Springer, Heidelberg (2005)
[30] Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Technical report, Cornell University, Ithaca, NY, USA (1987)

[31] Shamsfard, M., Barforoush, A.A.: The state of the art in ontology learning: a framework for comparison. The Knowledge Engineering Review 18(4), 293–316 (2003)
[32] Shinzato, K., Torisawa, K.: Acquiring hyponymy relations from Web documents. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004), Boston, Massachusetts, pp. 73–80 (2004)
[33] Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proceedings of the KDD International Workshop on Text Mining, Boston, MA, USA (August 2000)
[34] Stojanovic, L., Stojanovic, N., Volz, R.: Migrating data-intensive Web sites into the semantic Web. In: Proceedings of the 2002 ACM symposium on Applied computing (SAC 2002), Madrid, Spain, pp. 1100–1107. ACM Press, New York (2002)
[35] Tagarelli, A., Greco, S.: Toward semantic XML clustering. In: Ghosh, J., Lambert, D., Skillicorn, D.B., Srivastava, J. (eds.) Proceedings of the Sixth SIAM International Conference on Data Mining (SIAM 2006), Bethesda, MD, USA, pp. 188–199. SIAM, Philadelphia (2006)
[36] Zhang, Z., Li, R., Cao, S., Zhu, Y.: Similarity metric for XML documents. In: Ralph, M.S. (ed.) Proceedings of Workshop on Knowledge Experience and Management (FGWM 2003), Karlsruhe, Germany, pp. 255–261. AIFB Karlsruhe, GI (2003)

Exploring the Semantic Web as Background Knowledge for Ontology Matching

Marta Sabou, Mathieu d'Aquin, and Enrico Motta

Knowledge Media Institute (KMi)
The Open University, Milton Keynes, United Kingdom
{r.m.sabou,m.daquin,e.motta}@open.ac.uk

Abstract. In this paper we propose an ontology matching paradigm based on the idea of harvesting the Semantic Web, i.e., *automatically* finding and exploring *multiple* and *heterogeneous* online knowledge sources to derive mappings. We adopt an experimental approach in the context of matching two real life, large-scale ontologies to investigate the potential of this paradigm, its limitations, and its relation to other techniques. Our experiments yielded a promising baseline precision of 70% and identified a set of critical issues that need to be considered to achieve the full potential of the paradigm. Besides providing a good performance as a stand-alone matcher, our paradigm is complementary to existing techniques and therefore could be used in hybrid tools that would further advance the state of the art in the ontology matching field.

1 Introduction

1.1 The Matching Problem in Ontologies and Databases

The issue of finding correspondences between heterogeneous conceptual structures is inherent to all systems that combine multiple information sources. The database community has identified schema matching as a core task in many application domains, such as integrating different databases (i.e., establishing mappings between their schemas), data warehousing and E-commerce (matching between different message schema) [39]. Matching also plays a major role in approaches that rely on ontologies to solve the semantic heterogeneity problem between information systems [25,38,52]. While both database schemas and ontologies provide a vocabulary of terms to describe a domain of interest, database schemas do not make explicit the semantics of their elements while ontologies, by definition (*"a formal, explicit, specification of a domain conceptualization"* [19]), do [44]. A direct implication is that matchers can try and exploit the explicit semantics of ontologies to improve their performance.

In this context, the appearance and growth of the Semantic Web, *"an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation"* [4], marks an important stage in the evolution of the matching problem. Technologies such as RDF(S) and OWL, which allow to represent ontologies and information in a formal, machine understandable way, have led to a rapid increase in the amount of online

© Springer-Verlag Berlin Heidelberg 2008

ontologies and semantic documents [26]. This online knowledge can be explored through novel infrastructures such as the Swoogle [11] semantic search engine or the Watson Semantic Web Gateway [10], which collect and index Semantic Web documents. These changes have important consequences for the design of Semantic Web applications. While early tools, resembling ontology-based information systems [12,23,29,36], relied on a small number of ontologies selected and configured at design time, we are now witnessing the emergence of a *new generation of Semantic Web applications* [37], which aim to *dynamically* select, combine and exploit online ontologies [28]. Needless to say, matching is a key component of this new class of applications.

The approaches developed both for database schemas and ontologies follow two major paradigms depending on the types of information they use to derive mappings [25,38,39,44]. *Internal* approaches typically explore information provided by the matched ontologies such as their labels, structure or instances [44]. Indeed, all the ontology matching tools evaluated within the Ontology Alignment Evaluation Initiative (OAEI'06)[1] primarily exploit label and structure similarity to derive correspondences associated to varying confidence values [14]. A limitation of such approaches is that they depend on the richness and the similarity of the internal information of the matched ontologies. For example, Aleksovski et al. [2] used two state of the art matchers, FOAM [13] and Falcon-AO [24], to match weakly structured medical vocabularies with a low overlap in their labels and obtained precision values of only 30% and 33%.

External (or background knowledge based) techniques aim to address this limitation by exploring an external resource to *bridge the semantic gap between the matched ontologies*. Indeed, continuing the example above, Aleksovski et al. obtained a precision value of 76% on the same dataset in the medical domain by exploring the DICE ontology as background knowledge [2]. As depicted in Figure 1, matchers from this category exploit an external resource by replacing the original matching problem (between concepts A and B) with two individual matching and an inference step: the two concepts are first matched to so called *anchor terms (A', B')* in the background source, and then mappings are deduced from the semantic relations of these anchors.

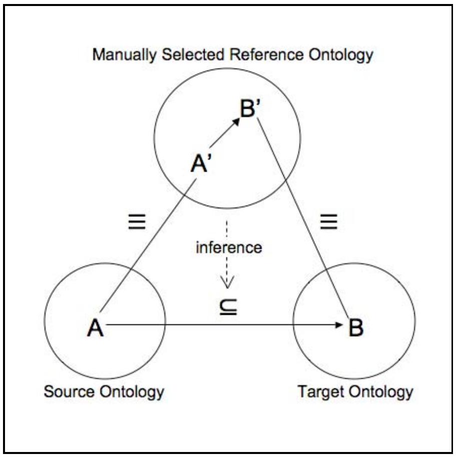


Fig. 1. Background knowledge based matching

[1] http://oaei.ontologymatching.org/2006/

A pre-requisite for the success of such matchers is the *availability* of background knowledge sources with an *appropriate coverage* of the matched ontologies. Some approaches rely on readily available, large-scale, generic resources such as WordNet or Cyc [9,15,22]. However, even if these resources cover a broad range of domains they might not cover specific domains (e.g., medicine, transportation planning) to the depth required by the matching task. In these cases, an appropriate domain ontology is either built manually (e.g., for the SIMS system [3]) or selected prior to the matching process [2]. As discussed in [3], the manual acquisition (or selection) of domain knowledge with appropriate coverage represents a considerable effort that should ideally be avoided.

1.2 Our Proposal: Exploiting the Semantic Web as Background Knowledge for Ontology Matching

We propose *a paradigm to ontology matching* based on the idea of harvesting the Semantic Web, i.e., *automatically* finding and exploring *multiple* and *heterogeneous* online knowledge sources. For example, when matching two concepts labeled *Researcher* and *AcademicStaff*, a matcher based on this paradigm would 1) identify (during matching) online ontologies that can provide information about how these two concepts inter-relate and then 2) combine this information to infer the mapping. The mapping can be either provided by a single ontology (e.g., stating that $Researcher \sqsubseteq AcademicStaff$), or by reasoning over information spread in several ontologies (e.g., that $Researcher \sqsubseteq ResearchStaff$ in one ontology and that $ResearchStaff \sqsubseteq AcademicStaff$ in another).

While this approach enjoys the advantages derived from the use of background knowledge, it provides an elegant solution to the tradeoff between the availability and the coverage of background knowledge. First, instead of relying on a single (generic or domain) ontology, we *maximize the coverage of the background knowledge* by exploring multiple online ontologies. Second, instead of selecting or building a domain ontology prior to matching, we *minimize any knowledge acquisition effort prior to matching* through the automatic selection of the background knowledge. Such an approach can be particularly helpful when a large, domain ontology does not exist but, nevertheless, the required knowledge is potentially spread over multiple different ontologies, or when the matched ontologies spread over several domains, requiring the use of a variety of ontologies.

A small-scale, preliminary evaluation of this paradigm provided encouraging results but gave little insight in its strengths and weaknesses when faced with real life situations [40]. The objective of this paper is to report on an in-depth investigation of this paradigm along the line of three main research questions:

Does it work? The main research question focuses on the feasibility of this paradigm to be successfully applied in real life matching cases. In practice this means assessing the two core assumptions that underlie this work. First, that the amount and quality of online ontologies are sufficient to be used as a basis for matching and that an alignment can be obtained in a reasonable amount of time. Second, that it is possible to build algorithms that automatically discover and combine this knowledge in an intelligent (useful) way. A proof that the use of such a paradigm is feasible and therefore worth pursuing further, would be to achieve a good performance in a real life matching experiment by using a simple, baseline implementation.

What are the limitations and how can they be avoided? Our preliminary experiments described in [40] provided limited insight 1) about the main steps needed for implementing a matcher based on this paradigm (e.g., ontology selection, knowledge combination), 2) about their relative influence on the quality of the derived mappings (e.g., are false mappings due more to the inability to select the right ontology or to the use of simple knowledge combination algorithms?), as well as 3) about typical problematic cases that need to be solved for each step (e.g., which ontology characteristics lead to false mappings and should be avoided through the selection process?). Gaining an understanding of all these issues is a pre-requisite for designing an improved technique based on the proposed paradigm.

How does it compare to other techniques? The third aim of this paper is to position the proposed paradigm in the landscape of the ontology matching field. On the one hand, we investigate the levels of performance that can be achieved with a stand-alone matcher based on this paradigm. On the other hand, since our goal is not to provide a stand-alone matcher but rather a complement to existing approaches, we analyze strengths and weaknesses with respect of other techniques in order to understand how the proposed paradigm would benefit from being combined with them, in hybrid approaches.

We rely on an experimental approach to answer these research questions. Our methodology consists of three major stages which are reflected in the structure of the paper. In the first stage, we propose two possible implementations of the paradigm and analyze the steps that are core to both (Section 3). In the second stage, we provide a baseline implementation based on the simplest solution for each of these steps (Section 4) and apply it on a real life, large-scale matching case using the experimental setup described in Section 5. The last and most important stage of our work consists in analyzing the results of the experiments (Sections 6 to 9). The performance of the implemented matcher represents a baseline that can be achieved with our paradigm and thus addresses the first question regarding its feasibility. The second research question, focusing on possible limitations, is answered by analyzing the evaluation results (Section 6 and 7). In Section 8 we assess our assumption that multiple online sources can be used for matching by providing some statistics about the number of ontologies explored to derive mappings. We address our final research question in Section 9, by comparing our results, strengths and weaknesses to those of other techniques. We conclude and point out future work in Section 10.

2 Related Work

A first body of related work consists of *approaches to matching that rely on the use of background knowledge.* We distinguish two categories of such matchers depending on the type of the explored external resource, i.e., an ontology [2,3,6,9,48] or online textual sources [50].

Several ontology based matchers rely on a large-scale generic resource such as Cyc or WordNet. The Carnot system [9,22] explores the Cyc knowledge base as a global context for achieving a semantic level integration of various information models (e.g., database schemas, knowledge bases). CTxMatch [6] (and its follow-up, SMatch [15]) translates ontology labels into logical formulae between their constituents, and maps them to the corresponding WordNet senses. A SAT solver is then used to derive mappings between the concepts. This approach has been extended to handle the problem of *missing background knowledge* [16]: if the simple techniques used to explore WordNet fail, then a second set of more complex and computationally expensive heuristics are applied to gain more knowledge.

While readily available, generic resources might fail to provide the appropriate coverage when matching is performed in a specific domain, such as medicine. In these cases, several matching approaches have opted for the use of a domain ontology. The SIMS system [3] relies on a manually built ontology about transportation planning for integrating several databases in this domain. In [2], the authors match two weakly structured vocabularies of medical terms by using the DICE ontology. Similarly, in [48] mappings between two medical ontologies (Galen and Tambis) are inferred from manually established mappings with a third medical ontology (UMLS), and by using the reasoning mechanisms permitted by the C-OWL language. Unfortunately, building (and even selecting) an appropriate domain ontology prior to matching is a considerable effort and represents a drawback of these techniques [3].

van Hage et. al [50] use the combination of two "linguistic ontology matching techniques" that exploit online texts to resolve mappings between two thesauri in the food domain. First, they rely on Google to determine subclass relations between pairs of concepts using the Hearst pattern based technique introduced by the PANKOW system [8]. Then, they exploit the regularities of an online cooking dictionary to learn hypernym relations between concepts of the matched ontologies. The strength of this approach is that, in principle, it is domain independent and therefore it does not require manual background knowledge selection. In reality, however, its precision dramatically decreases when relying on a corpus of general texts (50%), as opposed to a domain specific one (75%).

While the paradigm proposed in this paper explores ontologies as background knowledge, it differs from the above described matchers in several ways. First, we tackle the issue of *coverage* by exploring *multiple* rather than a single ontology. Second, we reduce the *knowledge acquisition effort prior to matching* by automatically selecting these ontologies. Finally, unlike some of the matchers which exploit the particularities of the background ontology [2,15], our approach is entirely domain and ontology independent.

Besides matchers based on background knowledge, our work is also related to *approaches that explore multiple (online) ontologies.* The idea of finding mappings between two ontologies by exploring other ontologies as *semantic bridges* has been discussed in [46] where a finite set of small, independently developed ontologies are interrelated by finding mappings between their concepts. These mappings are often discovered through a semantic bridge consisting of many other ontologies. Because the set of ontologies is finite, the technique can establish pairwise relations between the concepts of all ontologies (using a variety of matching techniques) and then rank and eliminate the redundant or useless ones. Our work is similar from the perspective that mappings are derived by exploring third party ontologies. However, a major difference is that we use a large set of heterogeneous ontologies where an exhaustive technique like the one of Stephens et al. cannot be applied.

The same paradigm of automatically selecting and exploring online ontologies has been proposed for solving other tasks than ontology matching. First, Alani proposes a method for ontology learning that relies on cutting and pasting ontology modules from online ontologies relevant to keywords from a user query [1]. Second, in [18] the authors describe a multi-ontology based method to disambiguate the senses of keywords that are given as a query to a search engine (e.g., *star* is used in its sense of celestial body in [*astronomy, start, planet*]). While the authors had previously relied on WordNet alone to collect possible senses for each keyword, now they exploit online ontologies to gather a larger set of senses and thus increase the quality of their method. Unfortunately, from these two methods, only the disambiguation process has been implemented and partially evaluated. Therefore, the contribution of our work to this line of research is to provide a first evaluation of automatically exploring online ontologies.

3 Proposed Paradigm

In the terminology of [44], we describe an element level matcher which relies on the use of external knowledge sources to derive mappings. In this section we investigate a set of issues that need to be considered when implementing this paradigm and conclude on a set of fine-grained research questions that should be experimentally investigated.

We describe two increasingly sophisticated strategies to discover and exploit online ontologies for matching. The first strategy derives a mapping between two concepts if this relation is defined within a single online ontology (Section 3.1). The second strategy (Section 3.2) addresses those cases when no single online ontology states the relation between the two concepts by combining relevant information which is spread over two or more ontologies. Both strategies need to address a set of tasks such as finding ontologies that contain equivalent concepts to those being matched (i.e., anchoring), selecting the appropriate ontologies, and using rules to derive mappings. We discuss all of these tasks in Sections 3.3 to 3.5. In Section 3.6 we discuss mechanisms for dealing with contradictory mappings derived from different sources.

Each strategy is presented as a procedure that takes two candidate concepts as an input and returns the discovered mapping between them. We use the letters A and B to refer to these candidate concepts. The corresponding concepts to A and B in an online ontology O_i are A'_i and B'_i ("anchor terms"). We rely on the description logic syntax for semantic relations occurring between concepts in an online ontology O_i, e.g., $A'_i \sqsubseteq B'_i$ means that A'_i is a sub-concept of B'_i in O_i. The returned mappings are expressed using C-OWL like notations [5], e.g., $A \xrightarrow{\sqsubseteq} B$. Note that we are using the C-OWL notations without relying on the formalism itself and on its semantics.

3.1 Strategy S1: Mappings within One Ontology

The first strategy consists of finding ontologies containing concepts similar with the candidate concepts (e.g., by relying on Swoogle) and then deriving mappings from their relations in the selected ontologies. Figure 2 (a) illustrates this strategy with an example where three ontologies are discovered (O_1, O_2, O_3) containing the concepts `A'` and `B'` corresponding to `A` and `B`. The first ontology contains no relation between the anchor concepts, while the other two ontologies declare a subsumption relation. The concrete steps of this strategy are:

1. Anchor A and B to corresponding concepts A' and B' in online ontologies;
2. Select ontologies containing A' and B';
3. For a given ontology (O_i) apply the following rules:
   - if $A'_i \equiv B'_i$ then derive $A \xrightarrow{\equiv} B$;
   - if $A'_i \sqsubseteq B'_i$ then derive $A \xrightarrow{\sqsubseteq} B$;
   - if $A'_i \sqsupseteq B'_i$ then derive $A \xrightarrow{\sqsupseteq} B$;
   - if $A'_i \perp B'_i$ then derive $A \xrightarrow{\perp} B$;
4. Combine all mappings derived from the considered ontologies.

For example, when matching two concepts labeled *Drinking Water* and *tap_water*, appropriate anchor terms are discovered in the TAP ontology and the following subsumption chain in the external ontology is used to deduce the mapping: *DrinkingWater* $\sqsubseteq$ *FlatDrinkingWater* $\sqsubseteq$ *TapWater*.

This strategy can be implemented in a multitude of ways depending on the type of anchoring mechanism applied in step 1, the criteria used to select the right ontologies in step 2, the complexity of the inferences employed by the derivation rules in step 3 or the strategy for integrating mappings originating from different sources in step 4. We discuss all these issues in the upcoming sections (Sections 3.3 to 3.6).

3.2 Strategy S2: Cross-Ontology Mapping Discovery

The previous strategy assumes that a relation between the candidate concepts can be discovered in a single ontology. However, some relations could be distributed over several ontologies. Therefore, if no ontology is found that relates

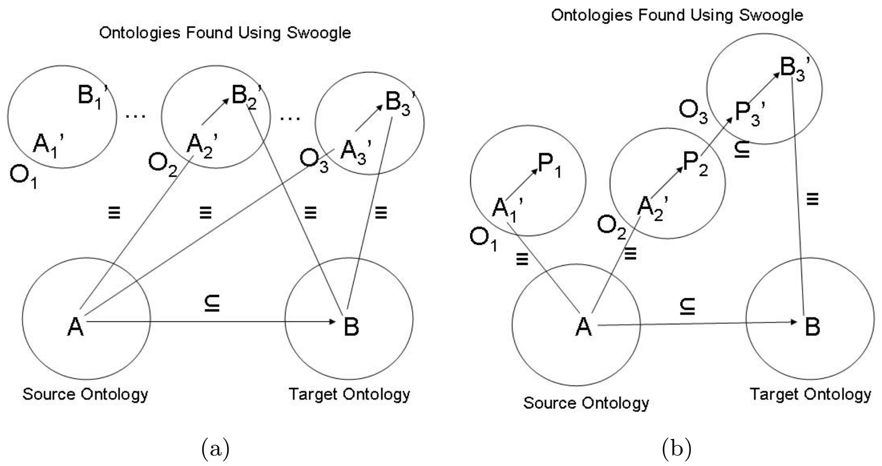


Fig. 2. Ontology matching (a) within one ontology (S1) and (b) across ontologies (S2)

both candidate concepts, then the mapping should be derived from two (or more) ontologies. In this strategy, matching is a recursive task where two concepts can be matched because the concepts they relate to in some ontologies are themselves matched. Figure 2 (b) illustrates this strategy where no ontology is available that contains anchor terms for both A and B, but where one of the parents (P_2) of the anchor term A'_2 can be matched to B in the context of a third ontology (O_3). For example, a mapping between *Cabbage* and *Meat* can be derived by taking into account that *Cabbage* $\sqsubseteq$ *Vegetable*[2] and then discovering that *Vegetable* $\perp$ *Meat*[3] through another matching step. The concrete steps are:

1. Anchor A and B to corresponding concepts A' and B' in online ontologies;
2. If no ontologies are found that contain both A' and B' then select all ontologies containing A';
3. For a given ontology O_i apply the following rules:
   (a) for each P_i such that $A'_i \sqsubseteq P_i$, search for mappings between P_i and B;
   (b) for each C_i such that $A'_i \sqsupseteq C_i$, search for mappings between C_i and B;
   (c) derive mappings using the following rules:

      – (r1) if $A'_i \sqsubseteq P_i$ and $P_i \xrightarrow{\sqsubseteq} B$ then $A \xrightarrow{\sqsubseteq} B$
      – (r2) if $A'_i \sqsubseteq P_i$ and $P_i \xrightarrow{\equiv} B$ then $A \xrightarrow{\sqsubseteq} B$
      – (r3) if $A'_i \sqsubseteq P_i$ and $P_i \xrightarrow{\perp} B$ then $A \xrightarrow{\perp} B$
      – (r4) if $A'_i \sqsupseteq C_i$ and $C_i \xrightarrow{\sqsupseteq} B$ then $A \xrightarrow{\sqsupseteq} B$
      – (r5) if $A'_i \sqsupseteq C_i$ and $C_i \xrightarrow{\equiv} B$ then $A \xrightarrow{\sqsupseteq} B$

4. Combine all mappings derived from the considered ontologies.

[2] http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf
[3] http://www.co-ode.org/resources/ontologies/Pizzademostep1.owl

The matching processes in steps 3(a) and 3(b) can be realized using either strategy S1 or S2. These two steps correspond to the recursive part of the algorithm and therefore a concrete implementation will need to avoid the exhaustive search of the semantic space. For example, mappings could be established only with the direct parents/children of A'_i (instead of *all*), the matching could stop as soon as a mapping is found or when a given amount of time has elapsed. As in the case of S1, strategy S2 can also be implemented differently depending on the chosen anchoring mechanism, ontology selection, the types of rules used and the way the final mappings are derived, as we discuss in the next sections.

3.3 Step1: Anchoring

Anchoring is a core part of all background knowledge based techniques: its role is to identify the appropriate part of the background knowledge that should be used for the matching (i.e., the part that refers to the two concepts being matched). Several anchoring mechanisms are reported in the literature.

The anchoring described in [2] is based on partial lexical matches between concept labels (i.e., it is sufficient that they share a subset of tokens) following the intuition that additional words added to a label denote a more specialized concept by constraining its meaning. For example, *"long brain tumor"* is anchored (as *narrower-than*) to *"long tumor"* because they share two tokens. Unfortunately, this strategy also introduces incorrect matches such as *"long brain tumor"* being anchored (i.e., as *narrower-than*) to *"brain"* [2].

The authors of [50] impose a strict string matching between concept labels and tokens in online texts (i.e., web pages) to establish equivalences between them. This stricter matching is likely to be more precise than the one in [2] but it still falls short of correctly anchoring polysemous words (e.g., *Squash* can be equally matched to words referring to a vegetable or a sport).

Unlike the previous two approaches which only exploit labels, S-Match [15] goes one step further and also relies on the structural information of a concept (i.e., its place in the concept hierarchy) when anchoring it into WordNet. First, the approach identifies all the WordNet senses relevant for the concept label. Then, the right sense is filtered out depending on the senses of the surrounding concepts in the hierarchy (using an algorithm presented in [30]). This approach ensures that concepts are anchored to concepts with the same sense in WordNet.

In the case of our technique, the anchoring is special because a concept is anchored to (possibly) many online ontologies with varying semantic richness. While anchoring should identify semantically (and not just syntactically) equivalent concepts (thus taking into account the semantic context of the concepts similarly to S-Match), it also needs to be light-weight enough to be usable during matching (in the case of S-Match, because a single background ontology is used, anchoring can be performed a priori). Before implementing a precise and optimal anchoring, we wished to find out:

RQ1: How well do simple anchoring techniques work? In our first implementation we use an anchoring technique similar to that of van Hage and we wish to assess the quality of the obtained results. If the results are reasonable, implementing a more complex and time consuming anchoring might not be worthwhile.

3.4 Step2: Ontology Selection

Anchoring identifies a set of ontologies that can lead to a mapping (e.g., in the example for S1, Figure 2, three such ontologies are identified). The choice of the ontologies that are used to derive mappings depends on two main design decisions: (1) the number of ontologies to be used and (2) the way they are selected. For the first design decision, we distinguish two situations:

Using a single ontology is the easiest way to deal with the multiple returned ontologies but it assumes that the discovered relation can be trusted and there is no need to inspect the other ontologies as well. In the example for S1, this would mean deriving the mapping from one of the three ontologies.

Using a subset (or all) of the returned ontologies is computationally more expensive but it has a higher accuracy by taking into account all the information that can be possibly derived from the returned ontologies. In this cases a mapping relation is derived from each ontology and then these are combined into a final mapping (see Section 3.6 for strategies about combining multiple, possibly inconsistent, mappings).

In both cases, whether using one or more ontologies, it is important to decide on some selection criteria. We distinguish two approaches to this issue:

Use the ranking mechanism of the underlying ontology search engine as the implicit selection mechanism. For example, in strategy S1 the mapping can be derived from the first ontology returned by Swoogle. Note that this ontology does not necessarily contain a relation between the candidate concepts (e.g., O_1 in Figure 2 (a)). In such cases, it could be considered that if an ontology covers the candidate concepts without relating them, then no mapping should be derived. Or, the algorithm could explore the remaining ontologies until a relation is provided by one of them (this will be the final mapping returned by the algorithm).

The selection criteria used by the search engine might not be appropriate for matching. For example, similarly to Web search engines such as Google, Swoogle ranks ontologies based on their popularity computed with a modified version of the PageRank algorithm which takes into account how many times an ontology is referenced by others [11]. Popularity, however, is not always a good indicator of an ontology's suitability for matching. Indeed, because it is frequently imported by other ontologies, FOAF is often ranked as the "best" ontology, even if this weakly structured vocabulary is of little help for deriving mappings.

Use predefined selection criteria to select the ontology (when using one ontology) or the ontologies (when using a subset of ontologies) that have the highest quality and can potentially lead to the best mapping. A pre-requisite to build a good selection mechanism that would identify the "good" ontologies is a better understanding of the ontology characteristics that typically result in good/false mappings. These could range from structural characteristics such as depth or width (i.e., deeper ontologies have a richer structure thus they would lead to more mappings than shallow ones), to domain similarity with the matched ontologies or to qualitative characteristics such as the absence of certain modeling errors.

The need to better understand what constitutes a good ontology for matching leads us to the second research question:

RQ2: Which ontology characteristics lead to false mappings? One of the goals of our experiments is to determine some of these characteristics so that they can be used to build an appropriate selection mechanism.

3.5 Step 3: Derivation Rules

The derivation rules defined for both strategies can be implemented by considering different levels of inferences. In the simplest implementation, we can rely on *direct* and *declared* relations between A' and B' in the selected ontology. But, for better results, *indirect* and *inferred* relations should also be exploited (e.g., if $A' \sqsubseteq C$ and $C \perp B'$, then $A' \perp B'$). Different levels of inferences can be considered (no inference, basic transitivity, description logic reasoning), each of them representing a particular compromise between the performance of the matching process and the completeness of the obtained alignment.

3.6 Step 4: Combining Mappings

Unlike previous techniques where mappings were based on a single ontology ([2,15]), our approach derives mappings from a variety of sources. However, mappings resulting from different sources can contradict each other.

At a simple level, different ontologies can lead to different and incoherent relations between the same pair of concepts. For example, $Seafood$ is subsumed by $Meat$ in one ontology[4], and disjoint with it in another[5], leading to two directly contradictory mappings. If the final mapping between a pair of concepts is derived from several ontologies (Section 3.4), situations when such contradictory relations are returned need to be considered. For example:

Keep all mappings. In the simplest case, all derived mappings can be returned, thus allowing the user to select the right mapping (favoring recall).

Keep mappings without contradiction. To favor precision, the algorithm could return a mapping between two concepts iff all the inferred intermediary mappings were the same (i.e., there was no contradiction).

[4] `http://reliant.teknowledge.com/DAML/Economy.owl`

[5] `http://www.w3.org/2001/sw/WebOnt/guide-src/food`

Keep the most frequent mapping. Given a set of mappings, return the most frequent mapping (i.e., the mapping that was derived from most sources).

Keep the trusted mappings only. Return mappings derived from sources that satisfy certain trust criteria.

At a more complex level, the combination of several mappings in the alignment can lead to intricate contradictions. Figure 3 provides an example of such a situation, where the concepts of $Tomatoes$ and $Vegetable$ can be related, directly or indirectly, on the basis of four different mappings, potentially derived from four different ontologies. The contradiction appears because $Tomatoes$ can be inferred to be at the same time disjoint with $Vegetable$, and a sub-class of it. This situation can be described as an incoherence in the sense that the class $Tomatoes$ is *unsatisfiable*: there cannot be any instance of $Tomatoes$, since such an object would have to be an instance of two disjoint classes: $Vegetable$ and $Fruit\_(Food)$.

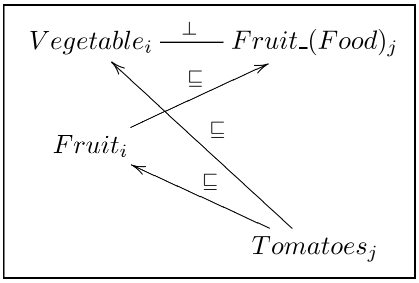


Fig. 3. Contradictory mappings

Generating such contradictions is a particularity of our technique, which combines information from different, heterogeneous knowledge sources. Incoherences are complex to detect as they require the use of reasoning mechanisms upon the source ontologies and the alignment. Handling these contradictions is a difficult task, requiring to select the appropriate strategy for removing the contradictory mappings. Therefore, an important research question is:

RQ3: How often do contradictions appear? The problem of dealing with contradictory mappings (both simple and complex) only needs to be addressed if such situations arise at all. We wish to get an insight in the scale of this phenomenon through experimental investigation.

4 Implementation Details

As described in Section 1, our methodology for exploring the proposed ontology matching paradigm consists in building and evaluating a baseline implementation. In this section we present the details of such a prototype which was built by using the simplest approach to implement all the tasks described in Section 3. We rely on Swoogle'05[6] which crawls and indexes a large amount of semantic metadata thus allowing access to a considerable part of the Semantic Web. We experiment with three implementations of the paradigm (these correspond to different configurations of the prototype):

[6] At the time of the experiments, Swoogle'06 was too unstable to allow extensive experiments.

Strategy S1, first variant stops as soon as one of the examined ontologies contains a relation between the matched concepts. We use this variant to evaluate the baseline performance of the paradigm for S1 (Section 6.1) and to understand the influence of anchoring and ontology selection (Section 6.2).

Strategy S1, second variant inspects *all* the ontologies that contain information about the two concepts to be matched and returns all the obtained mappings. We use this implementation to investigate how often simple contradictory mappings are derived between two given concepts (Section 6.3).

Strategy S2 derives a mapping between two concepts by combining information spread over several ontologies. Given the recursive nature of S2 which can lead to long execution times, we took the following design decisions to limit the search space of the matcher. First, we avoid infinite recursion by using the non-recursive S1 strategy in steps 3(a) and 3(b). Indeed, this strategy will always investigate a restricted number of ontologies (those in which the concepts to match appear). Second, we restrict the recursive part (steps 3(a) and 3(b)) to find matches only between the direct parents and children (P and C) of the anchor terms corresponding to the source concept and the target concept (B). Finally, the matcher stops as soon as a mapping is found between A and B. We evaluate the baseline performance of the paradigm for S2 in Section 7.1 and investigate typical errors in Section 7.2.

We now discuss the details common to all these individual implementations.

4.1 Anchoring Mechanism

The anchoring mechanism (i.e., finding A', B') in the case of all implementations is based on strict string matching between concept labels, similar to that of van Hage [50]. We allow for variations in naming conventions and lexical form. For simple labels (made up of one word) we find anchors that match this word as well as it's lemma (i.e., base form): a label *Persons* will be anchored to concepts labeled either *Persons* or *person*. This is achieved by performing an exact search for each lexical form of the label. For compound labels (containing multiple words) we anchor to concept labels containing the same words, in the same order, but possibly written according to different naming conventions and having different lemmas: *TeaCups* $\simeq$ *Tea_Cup* $\simeq$ *tea cup*. Concretely, for each word in the label and its lexical variants we query Swoogle for the number of labels that contain the search string as a substring (fuzzy match). For the word that has the fewest appearances, we compare all its appearances to the compound label.

4.2 Ontology Selection

The first variant of S1 as well as S2 rely on the implicit ranking mechanism of Swoogle (based on popularity) to select the ontologies from which the mapping is derived. Both implementations inspect the first ontology returned by Swoogle and if no mapping can be derived from it, then the next ontology is considered until an ontology is found from which a mapping can be derived. The second variant of S1 simply inspects all ontologies returned by Swoogle and derives a mapping from each of them when possible. Note that the selected ontologies are *not* downloaded, parsed and interpreted locally (unlike envisioned in [1] and done in [18]). Instead, their inspection is performed through the Swoogle API by using a range of functionalities such as requesting the direct parent or the disjuncts of a given concept.

We adopt a broad notion of an ontology which is not limited to the physical file in which the content is stored, nor to a given namespace but which also considers imported knowledge. The side effect of this view is that our search for a mapping is also conducted in the ontologies imported (reused) by a given ontology. It is therefore possible that A' is identified in ontology O_i while B' is defined in an ontology O_j which is imported by O_i. For example, a mapping is derived between $Dredger$ and $Vehicle$ by identifying a subsumption chain that spans three ontologies importing each other, (O_1[7], O_2[8], O_3[9]):
$Dredger_1 \sqsubseteq Ship_2 \sqsubseteq DisplacementHullWatercraft_2 \sqsubseteq Watercraft_2 \sqsubseteq Vehicle_3$

Even if in such cases a mapping is derived by combining information from several ontologies, there is still a fundamental difference with respect to mappings derived using S2. Namely, the relations between concepts from different ontologies used in S1 have been declared by the ontology creator. On the contrary, when using S2, the correspondence between concepts in different ontologies is established automatically, by using the anchoring mechanism.

Technically, the ontology selection mechanisms provided by Swoogle do not suffice for implementing our broad view on ontologies (i.e., they cannot filter ontologies based on the content of the knowledge that they import). As a result, we used a technical artefact to implement the selection step: we select all ontologies that contain an anchor for A and then inspect its hierarchy until one of the concepts equals (or is disjoint with) B. We take advantage of the fact that the Swoogle function for inspecting the hierarchical context of a concept takes into account imported content. Compared to a previous implementation where we inspected ontologies containing anchors for both A and B, this implementation discovered more mappings without being noticeably slower than its predecessor.

4.3 Derivation Rules

We use the rules described in Section 3. In both strategies we have relied on the transitivity of the subsumption relations to take advantage of indirect relations between concepts. While Swoogle's API allows for retrieving direct subsumptions, indirect relations are explored by asking several queries about direct relations (i.e., asking for the parent of the parent). To reduce the time of our experiments, we implemented S2 in such a way that only the first direct parent of the discovered A' concept is considered (instead of exploring *all* parents).

[7] `http://reliant.teknowledge.com/DAML/Transportation.daml`
[8] `http://reliant.teknowledge.com/DAML/Mid-level-ontology.daml`
[9] `http://reliant.teknowledge.com/DAML/SUMO.daml`

4.4 Detecting Contradictions

As explained in Section 3.6, because it combines heterogeneous knowledge sources, our technique may result in contradictory mappings, leading to incoherences within the generated alignment. Simple contradictions, involving only two mappings between the same two terms, are easy to detect. However, as shown in Figure 3, contradictions and incoherences may appear because of the intricate combination of more than two mappings, and therefore, the use of reasoning mechanisms for detecting such situations is required.

Reasoning on mappings has received considerable attention lately with several papers reporting on the use of inferences on mappings in order to improve the quality of alignments [33,34,47,49]. Among the diagnosis tasks defined in the literature, the detection of contradictions (called debugging in [34] and consistency checking in [47]) is recognized as being of particular importance. These studies rely on a rigorous formal framework, based on distributed description logics (DDLs). DDL is a formalism considering multiple ontologies, each of them with its own interpretation, interrelated through mappings (roughly sub-concept relations and equivalences), allowing distributed interpretations upon the ontologies globally, and upon the mappings [49]. However, relying on the DDL semantics introduces important constraints. In particular, the current implementation of DDL does not allow the use of disjoint relations in mappings. Moreover, mappings in DDLs are not transitive and are directional (e.g., $A \xrightarrow{\sqsubseteq} B$ is not equivalent to $B \xrightarrow{\sqsupseteq} A$ in DDLs), making this formalism inappropriate in our approach. Therefore, inspired by the previously mentioned work, we devised a simpler mechanism (not relying on DDLs) for detecting contradictions, using an ad-hoc reasoner (based on simple inference rules) for mappings, coupled with a classical DL reasoner (Pellet[10]) for reasoning upon the source ontologies.

We consider that the alignment contains a contradiction (incoherence) when it can be inferred, from the content of the alignment and from the source ontologies, that a concept is at the same time a sub-concept of and disjoint with another concept (e.g., $A \xrightarrow{\perp} B$ and $A \xrightarrow{\sqsubseteq} B$, or $A \xrightarrow{\perp} B$ and $A \sqsubseteq B$). According to this definition, the procedure for detecting incoherences is straightforward. For all the disjoint mappings that can be inferred from the alignment, we verify whether the involved concepts are sub-concepts of each other. Simple heuristics are used to avoid the detection of redundant contradictions. For example, if $A \xrightarrow{\perp} B$, $A \xrightarrow{\sqsubseteq} B$, and $C \xrightarrow{\sqsubseteq} A$, we only count one contradiction, even if it can be inferred that $C \xrightarrow{\sqsubseteq} B$ and that $C \xrightarrow{\perp} B$. This second contradiction is considered to be derived from the first one.

Note that our goal is not to provide a novel mechanism for incoherence detection in mappings. Indeed, the employed ad-hoc reasoner is only sufficient for detecting incoherences in an alignment. Handling these contradictions will require more advanced (and more complete) reasoning procedures (Section 6.3).

[10] http://www.mindswap.org/2003/pellet/

5 Experimental Setup

In this section we describe the experimental data sets and the real life scenario from where they originate, we provide an overview about how the experiments reported in the rest of the paper relate to the research questions identified in Section 3 and detail the methodology used for evaluating the alignments.

5.1 Experimental Scenario and Data

Our experimental data[11] is derived from a real life scenario, where two organizations wish to align their ontologies. These organizations are the UN's Food and Agriculture Organization (FAO) and the US's National Agricultural Library (NAL). Both organizations maintain large agricultural thesauri which they use for indexing their data. FAO's **AGROVOC** thesaurus, version May 2006, consists of 28.174 descriptor terms (i.e., preferred terms) and 10.028 non-descriptor terms (i.e., alternative terms). NAL's Agricultural Thesaurus **NALT**, version 2006, consists of 41.577 descriptor terms and 24.525 non-descriptor terms. Given their use to index data containing a vast amount of knowledge, these thesauri describe a broad range of domains, from animal species to chemical substances and information technology. Also, they use several technical terms (e.g., from chemistry) and a considerable amount of Latin terms (e.g., to describe animal species). There are several reasons for performing an alignment between these thesauri. First, such an alignment would facilitate data exchange between the two organizations. Second, the alignment process could identify concepts that are missing from one thesaurus but are covered by the other. Finally, an immediate benefit would be the enrichment of NALT, which currently contains only English and Spanish terms, with multilingual information contained in AGROVOC.

In our experiments we relied on both descriptor and non-descriptor terms, since the latter often describe synonyms of the preferred terms. There are several reasons behind choosing this data set as a basis for our experiments:

Large-Scale. Our hypothesis is that this large-scale, real life data set will allow us to evaluate the scalability of the proposed technique. Further, the large amount of data should provide a good test bed for all the research questions stated in Section 3.

Multi Domain. Because these thesauri contain information from a wide range of domains (and also because they are so large), it is virtually impossible to find a single ontology that could be used as a source of background knowledge to derive mappings (i.e., as current techniques do [2,15]). Therefore this is an illustrative case where it is necessary to combine knowledge from multiple background ontologies, possibly selected automatically.

Useful benchmark. A further advantage is that five state of the art ontology matching tools have been already used to derive mappings between these ontologies. These results are important for understanding how the proposed technique can complement existing technology.

[11] This data was also used in the OAEI 2006 *food* Thesaurus Mapping Task.

Table 1. Overview of the relation between research questions and experiments

Research Question	**Experiment**
Does it work?	Evaluate strategies S1 and S2 (Sect. 6.1 & 7.1) Number of explored ontologies (Sect. 8)
What are the limitations?	
Anchoring (RQ1)	Analyze results of S1 and S2 (Sect. 6.2 & 7.2)
Selection (RQ2)	Analyze results of S1 and S2 (Sect. 6.2 & 7.2)
Contradictions (RQ3)	Derive mappings from all ontologies (Sect. 6.3) Use incoherence detection mechanisms (Sect. 6.3)
How does it compare to other techniques?	Comparison with internal and external techniques (Sect. 9)

5.2 Overview of the Experiments

Table 1 summarizes the experiments reported in the rest of the paper and the corresponding research questions. To investigate the feasibility of the proposed paradigm i) we evaluate a baseline performance for the first variant of strategy S1 (Section 6) and for strategy S2 (Section 7) using the methodology described in Section 5.3 and ii) we assess the assumption that multiple online sources can be used to derive mappings by providing statistics about the number of ontologies used during the alignment process (Section 8).

Another goal of this work is to understand in what ways the baseline performance can be improved. In Section 3 we stated a set of research questions about issues that might hamper performance. Some of these questions can be answered by analyzing the results of the performance evaluation (Sections 6.2 and 7.2). Indeed, by inspecting the causes of false mappings, we can get an insight into the influence of the anchoring (RQ1) and ontology selection methods (RQ2). In Section 6.3 we assess how often contradictory mappings appear: we use the second variant of S1 to identify simple contradictions and apply incoherence detection mechanisms to detect alignment level (i.e., complex) contradictions (RQ3).

In Section 9 we compare our paradigm, based on the obtained results, to both techniques relying on information internal to ontologies (by analyzing the outcome of the OAEI'06 contest) as well as to other background knowledge based approaches (by exploring results reported in the literature). Besides the simple performance based comparison, we also discuss the potential contribution of our approach when integrated with existing techniques.

5.3 Methodology for Evaluating Alignments

One of the expected benefits of working with the AGROVOC-NALT dataset was the reuse of the Gold Standards employed to evaluate the OAEI'06 contest results. However, because the participant tools only returned equivalences, the Gold Standards have been geared towards evaluating those and thus were unusable for our results, containing subclass, superclass and disjoint relations. Given the high number of the discovered mappings, as well as the lack of Gold Standards, we performed a manual assessment of a significant subset of the results (1000 mappings in the case of both strategies).

As evaluators, we relied on nine members of our lab working in the area of the Semantic Web, and thus familiar with ontologies and ontology modeling. We performed two parallel evaluations of the sample mappings (i.e., each mapping has been evaluated by two different evaluators). The participants were asked to evaluate each mapping as *Correct*, *False* or *"I don't know"* for cases where they could not judge the correctness of the statement. They were allowed to use any kind of material (e.g., (web-)dictionaries, Google) in cases where they were not familiar with the domain and needed some more information for evaluating a given mapping (e.g., when judging that $Leukemia \sqsubseteq Neoplasm$). A specialized graphical interface has been developed to facilitate the task of the evaluators by displaying the mappings together with the context in which the mapped concepts appeared in the source ontologies (i.e., their neighborhood). We compute the precision of the alignment as the ratio of *Correct* mappings over all the evaluated mappings (i.e., those evaluated either as *Correct* or *False*). Formally:

$$Precision = \frac{Correct}{Correct + False}$$

6 Deriving Mappings from One Ontology (Strategy S1)

The matching process performed by using the first variant of S1 resulted in a total of 6687 mappings containing 2330 subclass, 3710 superclass and 647 disjoint relations. These mappings were derived during about two days by using an average laptop. Table 2 provides some examples of the derived mappings. For each mapping we present the source (AGROVOC) and target (NALT) concepts and their labels. Under each mapping we provide the URL(s) of the ontology(ies) from which the mapping was extracted, as well as the relations on which the mapping is based in these ontologies (i.e., it's explanation). For example, the first mapping was established between the AGROVOC concept *c_6617* labeled with "Rivers, Streams, Brooks, Tributaries" and the NALT concept identified as *waterways* and labeled "waterways". The mapping was derived from ontology O_1[12] which declares that *river* $\sqsubseteq$ *waterway*. O_1 has been used because the anchoring identified a correspondence between the "Rivers" label of *c_6617* and O_1's *river* concept, as well as between *waterways* in NALT and *waterway* in O_1. This example illustrates how the anchoring mechanism is flexible with respect to different naming conventions (here, it matches capitalized vs. non-capitalized words) and lexical forms (here, a match is established between the plural and the base form, or lemma, of both anchored labels).

It is interesting to observe that the second mapping spans two ontologies, the first one (*Economy.owl*) importing the second (*Mid-level-ontology.owl*). As

[12] http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/data/russia1a.rdf

Table 2. Example mappings discovered between AGROVOC and NALT using S1

Mappings	**Nr.**	**Examples**			
		AGROVOC Concept	**Labels**	**NALT Concept**	**Labels**
Subclass $(\xrightarrow{\sqsubseteq})$	2330	*c_6617*	Rivers, Streams, Brooks, Tributaries	*waterways*	waterways
		O_1:river $\sqsubseteq$ O_1:waterway			
		O_1 = http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/data/russia1a.rdf			
		c_25469	Cocaine	*narcotics*	narcotics, opioids
		O_1:Cocaine $\sqsubseteq$ O_2:Narcotic			
		O_1 = http://reliant.teknowledge.com/DAML/Economy.owl O_2 = http://reliant.teknowledge.com/DAML/Mid-level-ontology.owl			
SuperClass $(\xrightarrow{\sqsupseteq})$	3710	*c_10463*	Drinking water, Potable water	*tap_water*	tap_water, tap water
		O_1:DrinkingWater $\sqsupseteq$ O_1:FlatDrinkingWater $\sqsupseteq$ O_1:TapWater			
		O_1 = http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf			
		c_1142	Buildings, Houses, Building structures	*supermarkets*	supermarkets
		O_1:Building $\sqsupseteq$ O_1:Public_Building $\sqsupseteq$ O_1:Shop $\sqsupseteq$ O_1:Supermarket			
		O_1 = http://frot.org/space/0.1/index.rdf			
Disjoint $(\xrightarrow{\perp})$	647	*c_2761*	Exports	*imports*	imports
		O_1 = http://edge.mcs.drexel.edu/assemblies/ontologies/woolly/2003/02/functions.daml			
		c_8309	Water	*solids*	solids
		O_1:Water $\sqsubseteq$ O_1:Fluid $\perp$ O_1:Solid			
		O_1 = http://www.lri.jur.uva.nl/~rinke/aargh.owl			
Total	6687				

explained in Section 4.2, our implementation is capable of identifying such declared, cross-ontology relations and derive the corresponding mapping.

Another observation to be made is that, using an additional equivalence mapping, the mapping between *c_10463* and *tap_water* could have been inferred, thanks to the structure of the matched ontologies. Indeed, in NALT it is declared that *tap_water*$\sqsubseteq$*drinking_water*. Therefore, by establishing an equivalence relation between *c_10463* (having the label "Drinking water") and the NALT *drinking_water* concept, the mapping *c_10463* $\xrightarrow{\sqsupseteq}$ *tap_water* could be inferred. The fact that we have obtained the same result *without* relying on the structural information of the matched ontologies demonstrates the potential of our

Table 3. Evaluation of strategy S1 by both groups

	Group 1	**Group 2**	**Agreed by All**
Correct	586	666	525
False	346	299	217
Don't know	68	35	10
Precision	**63%**	**69%**	**70%**

technique to derive rich mappings even in cases when a rich structure would not be provided by the source ontologies. Indeed, this shows that internal information can be replaced by external information drawn from online ontologies. Having said that, structural information should not be purposefully ignored for the sake of using online ontologies (we have only done so to get an insight in the functioning of our technique when employed stand-alone).

6.1 Evaluation Results

In order to evaluate the precision of the alignment obtained with the first variant of S1, we randomly selected 1000 mappings (i.e., 15% of the alignment) containing an appropriate proportion of different mapping relations, namely: 100 disjuncts, 350 subclass, 550 superclass relations. These mappings were then evaluated by two groups of evaluators as described in Section 5.3. Table 3 summarizes the number of *Correct*, *False* and unevaluated (*Don't know*) mappings for each group, as well as the number of these mappings agreed by both groups. The two groups agree on 742 mappings (we exclude the "Don't know" answers because there are no real agreements on those), and therefore have an agreement coefficient of 74%. Note that a similar agreement (72%) was observed between the two groups that evaluated equivalence mappings on this dataset during OAEI'06 [14].

We obtained precision values of 63% and 69% for the two groups. The gap between these values is due to the variation in the way evaluators performed their task: some investigated each mapping thoroughly, while others simply provided no evaluation for the mappings they were not sure about. To level out these differences, we also computed the precision of the part of the alignment on which both groups agreed, as we think this better reflects the typical performance that can be achieved with our paradigm. In this case, the precision was equal to 70%. We consider this value as indicative for the baseline performance that can be obtained by harvesting online information. We compare it to typical performances of other matching approaches in Section 9.

6.2 Error Analysis

Besides getting an indication of the baseline precision that can be obtained with the proposed paradigm, we also wish to understand in which ways the performance can be improved, i.e., what are the major causes for errors and how could they be eliminated. To answer this question, we manually inspected the 217 false mappings on which both groups agreed. We observed two major causes for errors. On the one hand, 114 errors (i.e., 53%) are caused by the inherent limitations of the simplistic anchoring. On the other hand, 91 false mappings (i.e., 42%) are due to qualitatively inappropriate online ontologies. The rest of 12 (5%) false mappings are due to various smaller causes that are not significant in this analysis. Table 4 provides an overview of the type and number of identified errors as well as some illustrative examples.

Anchoring errors are a side-effect of the basic, string matching based anchoring and appear when a concept is related to an incorrect sense in online ontologies. For example, in the first entry in Table 4, concept *c_3179* describing "Game" in the sense of a hunted animal is incorrectly anchored to the *Game* concept in SUMO which represents a physical activity. In the second example, both concepts are anchored incorrectly. First, *c_6443* labeled with "Rams" and referring to an *"uncastrated adult male sheep"*[13] is put in correspondence with a similarly labeled concept ("ram"), but which refers to *Random Access Memory* in the context of the online ontology. In the same way, the *memory* concept of NALT refers to the term used as in psychology and thus has been incorrectly anchored to the identically labeled concept which refers to computer memory.

Because concept labels are ambiguous, anchoring errors are frequent and account for more than half of the false mappings (53%). Therefore, the current anchoring needs to be modified to take into account the context of the anchored concepts. Indeed, an anchoring mechanism that could prevent deriving these false mappings (thus reducing their number to 103) could potentially lead to an increase in precision from 70% to 87%.

We identified the following types of errors introduced by exploring low quality online ontologies:

Subsumption used to model generic relations. One of the most common errors in online ontologies was the use of subsumption as a way to model the fact that there exists some type of relation between two concepts, e.g., $Survey \sqsubseteq Marketing$, $Irrigation \sqsubseteq Agriculture$, $Biographies \sqsubseteq People$. This case leads to 40 false mappings (i.e., 18%).

Subsumption used to model part-whole relations. Subsumption is used in several ontologies to model part-whole relations. This resulted in incorrect mappings such as $Branch \sqsubseteq Tree$, $Leaf \sqsubseteq Plant$.

Subsumption used to model roles. Roles are often modeled as subclass relations, for example, that $Aubergine, Leek \sqsubseteq Ingredient$ ($Leek$ is a $Vegetable$ but in some contexts it plays the role of an ingredient).

Inaccurate labeling. We also found cases of correct subclass relations which introduced errors due to the inaccurate labeling of their concepts. For example, O_1[14] states that $coal \sqsubseteq industry$, where $coal$ refers to $coal\ industry$

[13] Definition from WordNet2.1.

[14] http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/data/russia1a.rdf

Table 4. Examples of several types of false mappings

Error Type	**Nr./ %**	**Examples**				
		AGROVOC Concept	**Labels**	**Rel.**	**NALT Concept**	**Labels**
Anchor	114, 53%	*c_3179*	Game, Hunted Animals	$\xrightarrow{\sqsupseteq}$	*sports*	sports, ball games, athletics
		O_1:Game $\sqsupseteq$ O_1:Sport				
		O_1 = `http://lists.w3.org/Archives/Public/www-rdf-logic/2003Apr/att-0009/SUMO.daml`				
		c_6443	Rams, Tups	$\xrightarrow{\sqsubseteq}$	*memory*	memory
		O_1:ram $\sqsubseteq$ O_1:memory				
		O_1 = `http://www.arches.uga.edu/~gonen/qos_bilal.owl`				
Subsumption as generic relation	40, 18%	*c_3954*	Irrigation	$\xrightarrow{\sqsubseteq}$	*agriculture*	agriculture, agriculture (general)
		O_1:Irrigation $\sqsubseteq$ O_1:SoilCultivation $\sqsubseteq$ O_1:Agriculture				
		O_1 = `http://sweet.jpl.nasa.gov/ontology/human_activities.owl`				
Subsumption as part-whole	16, 7%	*c_23995*	Branches	$\xrightarrow{\sqsubseteq}$	*trees*	trees
		O_1:Branch $\sqsubseteq$ O_1:Tree				
		O_1 = `http://www.site.uottawa.ca/~mkhedr/NewFuzzy.owl`				
Subsumption as role	11, 5%	*c_6211*	Products, Produce	$\xrightarrow{\sqsupseteq}$	*wool*	wool
		O_2:Product $\sqsupseteq$ O_1:ManufacturedProduct $\sqsupseteq$ O_1:TextileProduct $\sqsupseteq$ O_2:Fabric $\sqsupseteq$ O_3:Wool				
		O_1 = `http://reliant.teknowledge.com/DAML/Economy.owl` O_2 = `http://reliant.teknowledge.com/DAML/SUMO.owl` O_3 = `http://reliant.teknowledge.com/DAML/Mid-level-ontology.owl`				
Inaccurate labeling	12, 5%	*c_1693*	Coal	$\xrightarrow{\sqsubseteq}$	*industry*	industry
		O_1:coal $\sqsubseteq$ O_1:industry				
		O_1 = `http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/data/russia1a.rdf`				
		c_24833	Databases, Data bases, Databanks	$\xrightarrow{\sqsupseteq}$	*enzymes*	enzymes
		O_1:Database $\sqsupseteq$ O_1:Enzyme				
		O_1 = `http://mensa.sl.iupui.edu/ontology/Database.owl`				
Different View	12, 5%	*c_2943*	Fishes	$\xrightarrow{\sqsupseteq}$	*lobsters*	lobsters
		O_1:Fish $\sqsupseteq$ O_1:MarineInvertebrate $\sqsupseteq$ O_1:Crustacean $\sqsupseteq$ O_1:Lobster				
		O_1 = `http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf`				

rather than the concept of *Coal* itself. Similarly, for *Database* $\sqsupseteq$ *Enzime* in O_1[15], *Enzyme* refers to an *enzyme database* rather than describing the class of all enzymes. Note that this type of errors could be avoided by a semantic, context aware anchoring mechanism.

[15] `http://mensa.sl.iupui.edu/ontology/Database.owl`

Different Views. Finally, some of the explored ontologies adopted views that were not in concordance with the context of the mapping and/or the perspective of the evaluators. For example, TAP considers *lobsters* kinds of *Fishes*, a perspective with which none of the evaluators agreed.

Because a high number of errors (42%) were caused by incorrectly designed ontologies, our implementation would benefit from a selection mechanism based on the *quality* rather than the popularity of ontologies. While some approaches exist to automatically assess the quality of the ontology modeling [51], this task remains an important and difficult research question to consider as future work.

6.3 Contradictory Mappings

Research question RQ3 refers to whether contradictory mappings can be derived from different ontologies. To assess if different ontologies can contradict each other concerning the relation between a single pair of concepts (simple contradiction), we ran the second variant of S1: for every pair of concepts we derive mappings from *all* the online ontologies that mention them. As it can be expected considering the relative simplicity of the detection method, the number of such contradictions is very low and accounts to only eight pairs of concept labels (Table 5). Three of the eight pairs also appear inverted because their labels exist both in AGROVOC and NALT. For the purposes of this analysis, we can regard them as redundant thus further reducing the number of problematic pairs to five.

This first experiment shows that direct contradictions on the relation derived between a single pair of concepts are rare. However, as shown in Section 3.6, detecting these simple cases is insufficient, since contradictions can appear because of the combination of several mappings, derived from more than two ontologies. We used the implementation of the incoherence detection process described in Section 4 on the 6687 mappings generated between AGROVOC and NALT with the first variant of S1 and obtained 306 base incoherences. This result shows that contradictions actually appear in an alignment derived from online ontologies and that it is important to define strategies to deal with them.

Analyzing these incoherences can help us to better understand some limitations of our matching technique, and can hint ways of improving it. For example,

Table 5. All the label level contradictions

AGROVOC label	**NALT label**	**Nr. Subclass relations** ($\xrightarrow{\sqsubseteq}$)	**Nr. Superclass relations** ($\xrightarrow{\sqsupseteq}$)	**Nr. Disjunct relations**($\xrightarrow{\perp}$)
fruit	tomato	0	3	1
sea	river	0	1	2
energy	light	0	1	1
meat	seafood	0	2	12
seafood	meat	2	0	12
mushroom	pizza	1	0	1
tomato	fruit	3	0	1
light	energy	1	0	1

Table 6. The 10 mappings that are most involved in incoherences

Mapping	Nr. of incoherences
$People \xrightarrow{\sqsubseteq} Agents$	115
$Products \xrightarrow{\perp} Environment$	82
$Products \xrightarrow{\perp} People$	80
$Environment \xrightarrow{\sqsubseteq} Agents$	69
$Foods \xrightarrow{\sqsubseteq} Products$	66
$Organizations \xrightarrow{\sqsubseteq} Agents$	58
$Organisms \xrightarrow{\sqsubseteq} Individual$	50
$Industry \xrightarrow{\sqsubseteq} Heaters$	50
$Heaters \xrightarrow{\sqsubseteq} Organizations$	49
$Technology \xrightarrow{\sqsubseteq} Science$	38

Table 6 lists the top ten mappings that are most frequently involved in incoherences as well as the number of incoherences that they cause. This data suggests that incoherences are caused by a restricted sub-set of the alignment, and that a small sub-set of these mappings are actually involved in a large proportion of the incoherences. In other terms, incoherences are *localized* in the mappings, and detecting them helps in pointing out particular "areas" of the alignment that have to be considered as problematic.

Most concepts in Table 6 correspond to rather generic concepts (e.g., *Agents*, *Products*) likely to have lots of subclasses, which would become incoherent through inheritance. Indeed, almost 50 000 incoherences can be derived from the set of 306 base incoherences that are detected through our mechanism. This shows that this small number of incoherences (306) and the small number of mappings associated to them (454) corrupt almost the entire alignment.

In conclusion, it appears that online ontologies actually contradict each other and that this has an important influence on the formal quality of the alignments generated using our technique. Ultimately, contradictory mappings should be removed. However, automatically identifying the mappings to be remove is not trivial. Indeed, as shown in Table 6 (and already observed in [35]), the mappings that are often involved in incoherences are not necessarily wrong. On a more positive tone, several studies have been targeted towards the management or the removal of incoherences in ontologies [20,42,43]. These techniques provide solutions to facilitate the detection of the problematic sub-part of the alignment and to resolve contradictions, thus improving the quality of the entire alignment.

7 Deriving Mappings across Ontologies (Strategy S2)

The more complex S2 strategy lead to 6772 new mappings with respect to those derived with S1 (1966 subclass, 1568 superclass and 3238 disjoint relations) each obtained by combining information across ontologies. Interestingly, despite

Table 7. Some of the mappings discovered with strategy S2

Mappings	**Nr.**	**Examples**			
		AGROVOC Concept	**Labels**	**NALT Concept**	**Labels**
Subclass ($\xrightarrow{\sqsubseteq}$)	1966	*c_1014*	Boreal forests, Taiga	*habitats*	habitats
		O_1:BorealForest $\sqsubseteq$ O_2:Forest $\cong$ O_3:Forest $\sqsubseteq$ O_3:Habitat			
		O_1=`http://reliant.teknowledge.com/DAML/Geography.daml` O_2=`http://reliant.teknowledge.com/DAML/Mid-level-ontology.daml` O_3=`http://protege.stanford.edu/plugins/owl/owl-library/koala.owl`			
		c_1584	Cholesterol	*organic_compounds*	organic_compounds, organic compounds, organic chemicals
		O_1:Cholesterol $\sqsubseteq$ O_2:Steroid $\cong$ O_3:Steroid $\sqsubseteq$ O_3:Lipid $\sqsubseteq$ O_3:Organic_Chemical			
		O_1=`http://reliant.teknowledge.com/DAML/Mid-level-ontology.daml` O_2=`http://reliant.teknowledge.com/DAML/SUMO.daml` O_3=`http://onto.cs.yale.edu:8080/umls/UMLSinDAML/NET/SRSTR.daml`			
SuperClass ($\xrightarrow{\sqsupseteq}$)	1568	*c_28628*	Age groups	*elderly*	elderly, aged (people), old age (humans), geriatric people, senior citizens, older people
		O_1:AgeGroup $\sqsupseteq$ O_1:Adult $\cong$ O_2:adult $\sqsupseteq$ O_2:elderly			
		O_1=`http://sweet.jpl.nasa.gov/ontology/human_activities.owl` O_2=`http://owl.man.ac.uk/2003/why/latest/ontology.rdf`			
		c_6170	Prepared foods, Convenience foods, Ready meals, Ready to cook foods	*iced_tea*	iced_tea, iced tea, tea, iced
		O_1:PreparedFood $\sqsupseteq$ O_1:Tea $\cong$ O_2:Tea $\sqsupseteq$ O_3:IcedTea			
		O_1=`http://reliant.teknowledge.com/DAML/Mid-level-ontology.daml` O_2=`http://www.ksl.stanford.edu/projects/DAML/UNSPSC.daml` O_3=`http://www.wam.umd.edu/~katyn/CMSC828y/hw1/hw1.daml`			
Disjoint ($\xrightarrow{\perp}$)	3238	*c_1173*	Cabbages	*meat*	meat
		O_1:Cabbage $\sqsubseteq$ O_1:Vegetable $\cong$ O_2:Vegetable $\perp$ O_2:Meat			
		O_1=`http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf` O_2=`http://www.co-ode.org/resources/ontologies/Pizza_demo_step_1.owl`			
		c_935	Birds, Aves	*plants*	plants
		O_1:Bird $\sqsubseteq$ O_1:Animal $\cong$ O_2:Animal $\perp$ O_2:Plant			
		O_1=`http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf` O_2=`http://dannyayers.com/2003/08/udef.rdfs`			
Total	6772				

the fact that this strategy is more complex then S1 as it combines information from more ontologies, the time for deriving an alignment was roughly the same as for S1, i.e., around two days. In the case of the first mapping in Table 7, no ontology contains a relation between *BorealForest* and *Habitat*. However, because *BorealForest* $\sqsubseteq$ *Forest* in O_1[16] and *Forest* $\sqsubseteq$ *Habitat* in O_3,[17] the matcher derived that *BorealForest* $\xrightarrow{\sqsubseteq}$ *Habitat*.

[16] `http://reliant.teknowledge.com/DAML/Geography.daml`

[17] `http://protege.stanford.edu/plugins/owl/owl-library/koala.owl`

One interesting observation to make is that almost half of the derived mappings are disjoint relations. These are obtained by combining relations between a concept and its generic type (e.g., *Cabbage* $\sqsubseteq$ *Vegetable*) with a relation between the generic concept and one of its disjoints (e.g., *Vegetable* $\perp$ *Meat*). This results in an explosion of new mappings since all the sub-concepts of the generic concept (here, *Vegetable*) are considered disjoint with all its disjoints (here, all the subclasses of *Vegetable*, like *Cabbage*, are disjoint with *Meat*). While these additional mappings are correct, their usefulness is questionable since they establish a relation between concepts at different level of abstraction and are redundant with respect to the original "top-level" disjoint relations.

7.1 Evaluation Results

For evaluating the precision of the alignment obtained with S2, we followed the methodology described in Section 5.3. Our evaluation sample of 1000 mappings (i.e., around 15% of the alignment) contained 478 disjunct, 290 subclass, 232 superclass relations and it was assessed by two groups of evaluators. Table 8 summarizes the results of the evaluation. The agreement coefficient between groups reached 79% (they agreed on 798 evaluations), a value which is close the one obtained for strategy S1 (i.e., 74%). The precision values obtained were 66% for the first group, 63% for the second, and 70% when taking into account only the evaluations on which both groups agreed. Note that despite the increased complexity of this strategy, these values are similar to those obtained for S1: 63% and 69% per group, and 70% for the agreed mappings.

Table 8. Evaluation of strategy S2 by both groups

	Group 1	**Group 2**	**Agreed by All**
Correct	606	645	552
False	305	330	246
Don't know	89	25	7
Precision	**66%**	**63%**	**70%**

7.2 Error Analysis

To understand the major causes for false mappings, we manually investigated all the 246 mappings that were rated as *False* by both groups of evaluators. While the same types of errors as in S1 were identified in S2 as well, false mappings obtained by S2 are sometimes caused by more than one error. Indeed, we found 285 causes for the 246 false mappings. This phenomenon is a direct consequence of the fact that S2 exploits more ontologies than S1 and relies on one extra anchoring step. Table 9 displays some examples of typical errors encountered when deriving mappings across ontologies.

Anchoring Errors. We identified 167 anchoring errors. In the case of S1 anchoring errors appear when the source concepts are anchored to semantically

Table 9. Examples of typical errors in compound mappings obtained with S2

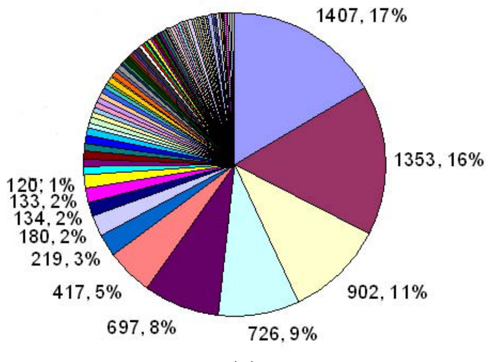
Error Type (Nr.)	**Examples**				
	AGROVOC Concept	**Labels**	**Rel.**	**NALT Concept**	**Labels**
Anchoring (167)	*c_5253*	Nucleic acids	$\xrightarrow{\sqsubseteq}$	*people*	people, persons, mankind
	O_1:NucleicAcid $\sqsubseteq$ O_1:Agent $\cong$ O_2:Agent $\sqsubseteq$ O_3:Person				
	O_1=http://mensa.sl.iupui.edu/ontology/BiologicalOntology.owl O_2=http://www.mindswap.org/2003/owl/swint/terrorism O_3=http://www.mindswap.org/2003/owl/swint/person				
	c_802	Bamboos	$\xrightarrow{\perp}$	*enterprises*	enterprises, businesses
	O_1:Bamboo $\sqsubseteq$ O_1:Plant $\cong$ O_2:Plant $\perp$ O_2:Enterprise				
	O_1=http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf O_2=http://www.dannyayers.com/2003/08/udef.rdfs				
Ontology Errors (118)	*c_139*	Adults	$\xrightarrow{\sqsubseteq}$	*universities*	universities, colleges
	O_1:Adult $\sqsubseteq$ O_1:Student $\cong$ O_2:Student $\sqsubseteq$ O_2:Department $\sqsubseteq$ O_2:Faculty $\sqsubseteq$ O_2:University				
	O_1=http://www710.univ-lyon1.fr/~s-suwa02/MSch/sc.owl O_2=http://www.srdc.metu.edu.tr/~yildiray/HW3.OWL				
	c_5326	Ohio	$\xrightarrow{\sqsubseteq}$	*North_America*	North_America, North America, America, North
	O_1:Ohio $\sqsubseteq$ O_1:USA $\cong$ O_2:USA $\sqsubseteq$ O_2:NorthAmerica				
	O_1=http://www.cwi.nl/~media/ns/IWA/VideoGen.rdfs O_2=http://islab.hanyang.ac.kr/damls/Country.daml				
	c_13735	Radishes	$\xrightarrow{\sqsubseteq}$	*ingredients*	ingredients
	O_1:Radish $\sqsubseteq$ O_1:Vegetable $\cong$ O_2:vegetable $\sqsubseteq$ O_2:ingredient				
	O_1=http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf O_2=http://cvs.sourceforge.net/viewcvs.py/instancestore/instancestore/ontologies/Attic/pizza9.daml?rev=1.2				

different concepts in online ontologies. In the case of S2, an additional anchoring process takes place for the intermediary concept that links the two concepts to be matched. This anchoring process is also prone to errors. For example, in the first mapping from Table 9, the intermediary concept is *Agent*. However, the senses of the concepts with this label in ontologies O_1[18] and O_2[19] are different: a participant in a chemical reaction in O_1 and a role played by a person in O_2.

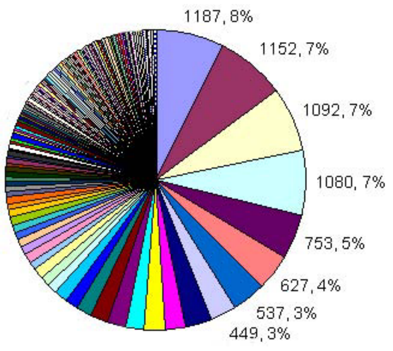
Ontology Errors. S2 was also influenced by 118 errors specific to low quality online ontologies where subsumption is used incorrectly to model general relations (73 cases, e.g., between *Student* and *University*), part-whole relations

[18] http://mensa.sl.iupui.edu/ontology/BiologicalOntology.owl

[19] http://www.mindswap.org/2003/owl/swint/terrorism



(a)



(b)

Fig. 4. Contribution to the alignment by ontologies used in (a) S1 and (b) S2

(5 cases, e.g., between *Ohio* and *USA*) or roles (6 cases, e.g., between *Vegetable* and *Ingredient*). Some mappings were also derived due to inaccurate labeling (27 cases) or to incorrect views of the world modeled in the used ontologies (7 cases).

8 Harvesting the Semantic Web

A core assumption of our work is that matching can be performed by *harvesting the Semantic Web*, i.e., by combining appropriate background knowledge from *multiple*, automatically identified online ontologies. In this section we verify this assumption by investigating the number of ontologies that were employed during the matching process.

Strategy S1 explored 226 ontologies to derive 6687 mappings.[20] Figure 4 (a) shows the contribution of each ontology to the alignment in terms of the number of mappings to which it contributed and the percentage that this number represents. An analysis of this chart reveals that there is a high variation in the contribution of different ontologies: a few ontologies provide the majority of the mappings, while most ontologies lead to a small amount of mappings. Indeed, the 11 ontologies (Table 10) for which the percentages are shown in the chart (and which account to about 5% of all used ontologies) lead to deriving approximately 76% of the alignment.

Strategy S2 used 306 ontologies to obtain 6772 mappings (Figure 4 (b)). Given the nature of this technique, i.e., that of combining multiple ontologies, a higher number of ontologies (about 80 more) than in S1 were used to derive approximately the same number of mappings. As in S1, there are a few large ontologies that contribute most mappings, however, their level of contribution is more balanced. Indeed, instead of having a single ontology contributing 17% of the alignment as in S1, in S2, the top four ontologies provide about an equal percentage

[20] Note that these statistics were computed by considering an ontology to be equivalent to a namespace, independently of the actual, physical location of the concepts in files.

Table 10. The top contributing ontologies to the alignments obtained with S1 and S2

Ontologies/Ontology Namespaces	Contribution to (%)	
	S1	S2
`http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf`	17%	8%
`http://reliant.teknowledge.com/DAML/SUMO.daml`	16%	7%
`http://reliant.teknowledge.com/DAML/Mid-level-ontology.daml`	11%	4%
`http://reliant.teknowledge.com/DAML/Economy.daml`	9%	3%
`http://gate.ac.uk/projects/htechsight/Technologies.daml`	8%	5%
`http://a.com/ontology`	5%	7%
`http://gate.ac.uk/projects/htechsight/Employment.daml`	3%	-
`http://reliant.teknowledge.com/DAML/WMD.daml`	2%	-
`http://sweet.jpl.nasa.gov/ontology/biosphere.owl`	2%	3%
`http://139.91.183.30:9090/RDF/VRP/Examples`	2%	-
`http://reliant.teknowledge.com/DAML/Geography.daml`	1%	-
`http://www.dannyayers.com/2003/08/udef.rdfs`	-	7%

of the alignment (7%). This is a direct consequence of the fact that mappings are based on multiple rather than on a single ontology.

We observe a large overlap between the top contributor ontologies to S1 and S2 (Table 10). The same seven ontologies are used (although with different levels of contribution), with TAP and SUMO being the main contributors in both strategies. Such an overlap is not surprising since these large ontologies have a good coverage of the various topic domains of AGROVOC and NALT.

These statistics strengthen our hypothesis that harvesting the Semantic Web is feasible. Our findings suggest that the strength of the Semantic Web is not only in the use of single, isolated ontologies but also in reusing, combining and making sense of knowledge spread across a variety of different ontologies. Indeed, in such a scenario where large, multi-domain ontologies are matched, it would have been difficult and time-consuming (if not impossible) to manually identify appropriate ontologies in order to derive the same amount of mappings as our technique has done without requiring any a priori knowledge selection.

9 Comparison with Other Techniques

In this section we investigate how the proposed paradigm compares against and complements existing techniques. We describe our findings both for techniques relying on internal information (Section 9.1) and for background knowledge based techniques (Section 9.2).

9.1 Comparison with Techniques Relying on Internal Information

The five matching systems applied on this dataset during the OAEI'06 contest primarily exploit information that is internal to the two matched ontologies [7,21,27,31,32]. As such, their results could be used to investigate how our

Table 11. Identifying non redundant mappings

Nr.	Alignment	Nr. Of Mappings
1	COMA++ [32]	7626
2	FALCON-AO [21]	12900
3	PRIOR [31]	11504
4	HMATCH [7]	19924
5	RiMOM [27]	13966
	Union	25224
6	Using the SW	4464
	Union	29688
	Non-redundant	27083
	Non-redundant from 6	**1915**

paradigm relates to techniques from this category. Unfortunately, because all the tools provided exact matches, their evaluation was focused on such mappings: precision was assessed manually, while recall was approximated on a rather small set of 200 mappings containing only 30% of subsumption relations [14]. In addition, while these five systems are complete, self contained tools, our paradigm is intended to be used as a complement to other existing techniques. Indeed, the current implementation does not extract any mappings that lexical and structural techniques can discover (e.g., using basic string comparison on the labels). As a consequence, comparing the recall of this implementation to the one of complete, stand-alone tools would be misleading.

We can nevertheless draw a set of conclusions which suggest that the proposed paradigm *complements* techniques exploring solely information internal to the matched ontologies. First, since our technique produces other relations than equivalences, a syntactic comparison with the alignments produced by the OAEI'06 tools yields that they are complementary (i.e., there is no overlap between them). Second, in order to *semantically* compare the matching techniques, we applied a redundancy detection mechanism on the union of their alignments[21]. We identified 1915 mappings discovered by our technique which were not redundant with the equivalence mappings identified by the OAEI'06 tools (Table 11). These mappings were obtained by exploring external sources and represent a net contribution to the alignments derived by exploring only information internal to the matched ontologies. Note that even if our technique performs anchoring using techniques similar to those employed by the OAEI'06 tools (i.e., string based comparison), it can identify additional mappings by exploring external sources. Similarly, Aleksovski et al. have shown that using syntactic technique for anchoring and then performing a deduction step using background knowledge leads to better performance than when these syntactic techniques are applied directly between the two source ontologies [2].

[21] We assume that the relations extracted by the OAEI'06 tools correspond to equivalences and consider only mappings with a confidence value greater than 50%.

9.2 Comparison with Background Knowledge Based Techniques

The performances of background knowledge based techniques described in the literature were reported on different data sets, therefore we consider them only as indicative. Unfortunately, S-Match only reports on recall values [16]. The technique of Aleksovski et al. was evaluated on a Gold Standard of mappings for 200 concepts and produced a precision of 76% (compared to 30% and 33% achieved by two traditional techniques on the same dataset) [2]. The matching techniques proposed by van Hage et al. yield a range of precision values for a manually constructed Gold Standard: 17% - 30% when relying only on Google, 38% - 50% when taking into account the context given by the Google snippets, 53% - 75% when exploring a domain specific textual resource, and finally 94% when validating the results of the domain specific extraction with the Google based techniques [50]. We conclude that the 70% precision of our technique, which could eventually be improved through better anchoring to reach 87%, correlates with the performance of the other two techniques (75% - 76%). It is important to note, that the techniques in [2] and [50] reached a high precision when exploring a *single, high-quality, domain specific resource* (i.e., DICE [2], CooksRecipes.coms Cooking Dictionary [50]) while our technique achieves comparable results when *automatically combining multiple, heterogeneous and generic ontologies*. Indeed, we have shown in Section 8 that a high number of ontologies (200 to 300) are automatically discovered and combined.

This comparison indicates that the use of online ontologies leads to comparable performance as when exploring carefully selected, domain specific background knowledge. In addition, our hypothesis is that exploring multiple and dynamically selected ontologies gathers necessary knowlegde that cannot be found in a single, generic resource, even as broad as WordNet or Cyc. Indeed, Figure 4 in Section 8 supports this intuition by depicting that our alignments have been obtained by exploring a few large resources (namely, TAP and SUMO), complemented with a large number of smaller and more specific ontologies. In this line of idea, a set of experiments have been performed to assess the additional knowledge that online ontologies provide with respect to WordNet. We found that only 33% of the alignment obtained with S1[22] (2233 mappings) could have been obtained with WordNet. These findings illustrate that our method maximizes the coverage of background knowledge by exploring complementary, online sources ranging from large, generic resources to small, domain specific ontologies.

10 Conclusions and Future Work

In this paper we describe and experimentally investigate an ontology matching paradigm based on the idea of harvesting the Semantic Web. Hereby we summarize our major conclusions and point out future work.

[22] This alignment does not contain any WordNet based mappings because we could not explore this ontology through Swoogle due to parsing errors.

Two of our main findings suggest that *the proposed paradigm is feasible* (re the first research question in Section 1). First, a baseline implementation of the technique applied on a large-scale, real life data set has led to a precision value of 70% for both strategies (Sections 6.1 and 7.1) which correlates with the performance of other background knowledge based matchers (Section 9). Each alignment has been obtained within two days by using average equipment. Given the large size of the data set we consider this time performance reasonable and appropriate for the scenario in which the alignment process took place. Second, our core hypothesis that an alignment can be generated by exploring *multiple* ontologies has been verified since our prototype has automatically selected and reused between 200 and 300 online ontologies (Section 8). In a broader context, these encouraging results indicate the potential of the Semantic Web for solving real life problems [41].

We have experimentally assessed the core limitations of the current implementation (the second research question in Section 1) by investigating the fine-grained research questions stated in Section 3. *A first, major limitation of our prototype is its simple, string comparison based anchoring (RQ1)* which generated more than half of the false mappings for S1 (53%) and also had a significant negative influence on the precision of S2 (Sections 6.2 and 7.2). Indeed, if these mappings could be avoided the precision of S1 would increase from 70% to 87%. Therefore, a high priority task is the design and implementation of an anchoring mechanism that takes into account ontological context. Ongoing experiments with an adaptation of the semantic similarity technique employed in [18] have already lead to promising results [17].

Besides anchoring errors, another major source of false mappings (42% in the case of S1) is the exploitation of online ontologies that contain modeling errors, mostly related to an inaccurate use of subsumption relations to model generic relations, roles and part-whole relations (Sections 6.2 and 7.2). These findings indicate that *the ontology selection mechanism should focus on the quality of the selected ontologies (RQ2)* rather than on their popularity as in the case of Swoogle. Although already considered in the literature [51], the automatic evaluation of such qualitative features remains a challenging area of future work. When investigating the frequency of contradictory mappings (RQ3) we found *a low number of simple contradictions* (affecting only 8 out of 6425 pairs of labels – Section 6.3). This suggests that the implementation of a mechanism for combining mappings from different ontologies would not significantly improve results. At the same time, *complex (alignment level) contradictions are more frequent than expected*, with our automatic incoherence detection mechanism identifying 306 base incoherences that corrupted the entire alignment (since incoherences were inherited by several subclasses of generic concepts). Fortunately, these mappings can be isolated and disposed of automatically, thus leading to the improvement of the alignment (Section 6.3). We plan to integrate an incoherence detection step into the matcher so that problematic mappings can be identified and excluded already *during* matching.

The third main research question stated in Section 1 refers to the relation of the proposed paradigm with other ontology matching approaches. Already when used as a stand alone matcher our prototype obtained precision values of 70% (and potentially even 87% given a more sophisticated implementation) comparable with the performance of state of the art matching tools (Section 9). *Besides a remarkable performance, the matcher is also complementary to existing techniques and could be more beneficial when used in a hybrid matcher.* Indeed, the obtained alignment is complementary with the results of existing tools (those used during the OAEI'06) i) by providing other relations than equivalences and ii) by identifying a set of mappings that are semantically non redundant with the union of all equivalence mappings obtained by the other tools. A hybrid matcher combining these two types of techniques would derive as many mappings as possible with traditional techniques and then it would explore external background knowledge for those entities about which not enough information exists to derive a mapping. Such a hybrid method has the potential to considerably advance the state of the art in ontology matching by exploring the Semantic Web.

Acknowledgements

We thank S. Angeletou, C. Baldassarre, G. Gracia, L. Gridinoc, D. Guidi, A. Llorente, V. Lopez, V. Uren and F. Zablith for their help with the work reported in this paper. This work was funded by the AKT GR/N15764/01, Open Knowledge IST-FF6-027253 and NeOn IST-FF6-027595 projects.

References

1. Alani, H.: Position Paper: Ontology Construction from Online Ontologies. In: Proc. of WWW (2006)
2. Aleksovski, Z., Klein, M., ten Katen, W., van Harmelen, F.: Matching Unstructured Vocabularies using a Background Ontology. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS, vol. 4248. Springer, Heidelberg (2006)
3. Arens, Y., Chee, C.Y., Hsu, C.N., Knoblock, C.A.: Retrieving And Integrating Data From Multiple Information Sources. International Journal of Cooperative Information Systems 2(2), 127–158 (1993)
4. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34–43 (2001)
5. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: Contextualizing ontologies. Journal of Web Semantics 1(4), 24 (2004)
6. Bouquet, P., Serafini, L., Zanobini, S.: Peer-to-Peer Semantic Coordination. Journal of Web Semantics 2(1) (2005)
7. Castano, S., Ferrara, A., Messa, G.: Results of the HMatch Ontology Matchmaker in OAEI 2006. In: Shvaiko, et al. [45]
8. Cimiano, P., Handschuh, S., Staab, S.: Towards the Self-Annotating Web. In: Proc. of WWW (2004)
9. Collet, C., Huhns, M.N., Shen, W.: Resource Integration Using a Large Knowledge Base in Carnot. IEEE Computer 24(12), 55–62 (1991)
10. d'Aquin, M., Sabou, M., Dzbor, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Motta, E.: WATSON: A Gateway for the Semantic Web. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519. Springer, Heidelberg (2007)

11. Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., Kolari, P.: Finding and Ranking Knowledge on the Semantic Web. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729. Springer, Heidelberg (2005)
12. Dzbor, M., Domingue, J., Motta, E.: Magpie - towards a semantic web browser. In: Fensel, D., Sycara, K.P., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870. Springer, Heidelberg (2003)
13. Ehrig, M., Sure, Y.: FOAM - Framework for Ontology Alignment and Mapping; Results of the Ontology Alignment Initiative. In: Proc. of the Workshop on Integrating Ontologies (2005)
14. Euzenat, J., Mochol, M., Shvaiko, P., Stuckenschmidt, H., Svab, O., Svatek, V., van Hage, W.R., Yatskevich, M.: Results of the Ontology Alignment Evaluation Initiative 2006. In: Shvaiko, et al. [45]
15. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: Semantic Schema Matching. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 347–365. Springer, Heidelberg (2005)
16. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: Discovering Missing Background Knowledge in Ontology Matching. In: Proc. of ECAI (2006)
17. Gracia, J., Lopez, V., d'Aquin, M., Sabou, M., Motta, E., Mena, E.: Solving Semantic Ambiguity to Improve Semantic Web based Ontology Matching. In: ISWC Workshop on Ontology Matching (2007) (to appear)
18. Gracia, J., Trillo, R., Espinoza, M., Mena, E.: Querying the Web: A Multiontology Disambiguation Method. In: Proc. of ICWE (2006)
19. Gruber, T.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5(2), 199–220 (1993)
20. Haase, P., Qi, G.: An analysis of approaches to resolving inconsistencies in DL-based ontologies. In: Proc. of Int. WS on Ontology Dynamics, IWOD (2007)
21. Hu, W., Cheng, G., Zheng, D., Zhong, X., Qu, Y.: The Results of Falcon-AO in the OAEI 2006, Campaign. In: Shvaiko, et al. [45]
22. Huhns, M.N., Jacobs, N., Ksiezyk, T., Shen, W., Singh, M.P., Cannata, P.E.: Integrating enterprise information models in Carnot. In: Proc. of Intelligent and Cooperative Information Systems (1993)
23. Hyvonen, E., Makela, E., Salminen, M., Valo, A., Viljanen, K., Saarela, S., Junnila, M., Kettula, S.: MuseumFinland – Finnish Museums on the Semantic Web. Journal of Web Semantics 3(2) (2005)
24. Jian, N., Hu, W., Cheng, C., Qu, Y.: Falcon-AO: Aligning Ontologies with Falcon. In: Proc. of the Workshop on Integrating Ontologies (2005)
25. Kalfoglou, Y., Schorlemmer, M.: Ontology Mapping: The State of the Art. The Knowledge Engineering Review 18(1), 1–31 (2003)
26. Lee, J., Goodwin, R.: The Semantic Webscape: a View of the Semantic Web. IBM Research Report (2004)
27. Li, Y., Li, J., Zhang, D., Tang, J.: Result of Ontology Alignment with RiMOM at OAEI 2006. In: Shvaiko, et al. [45]
28. Lopez, V., Motta, E., Uren, V.: PowerAqua: Fishing the Semantic Web. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011. Springer, Heidelberg (2006)
29. Lopez, V., Pasin, M., Motta, E.: AquaLog: An Ontology-portable Question Answering System for the Semantic Web. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532. Springer, Heidelberg (2005)
30. Magnini, B., Serafini, L., Speranza, M.: Making explicit the semantics hiden in schema models. In: Proc. of the Human Language Technology for the Semantic Web WS (2003)

31. Mao, M., Peng, Y.: PRIOR System: Results for OAEI 2006. In: Shvaiko, et al. [45]
32. Massmann, S., Engmann, D., Rahm, E.: COMA++: Results for the Ontology Alignment Contest OAEI 2006. In: Shvaiko, et al. [45]
33. Meilicke, C.: Reasoning about Mappings in Distributed Description Logics. Bachelor Thesis, University of Mannheim (August 2006)
34. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Improving Automatically Created Mappings using Logical Reasoning. In: Shvaiko, et al. [45]
35. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Repairing Ontology Mappings. In: Proc. of AAAI (2007)
36. Mika, P.: Flink: Semantic Web Technology for the Extraction and Analysis of Social Networks. Journal of Web Semantics 3(2) (2005)
37. Motta, E., Sabou, M.: Next Generation Semantic Web Applications. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185. Springer, Heidelberg (2006)
38. Noy, N.F.: Semantic Integration: a Survey Of Ontology-Based Approaches. SIGMOD Record 33(4), 65–70 (2004)
39. Rahm, E., Bernstein, P.: A Survey of Approaches to Automatic Schema Matching. The VLDB Journal 10, 334–350 (2001)
40. Sabou, M., d'Aquin, M., Motta, E.: Using the Semantic Web as Background Knowledge for Ontology Mapping. In: Shvaiko, et al. [45]
41. Sabou, M., Gracia, J., Angeletou, S., d'Aquin, M., Motta, E.: Evaluating the Semantic Web: A Task-based Approach. In: Proc. of ASWC/ISWC (2007)
42. Schlobach, S.: Diagnosing Terminologies. In: Proc. of AAAI (2005)
43. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging Incoherent Terminologies. Journal of Automated Reasoning (2007)
44. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg (2005)
45. Shvaiko, P., Euzenat, J., Noy, N., Stuckenschmidt, H., Benjamins, R., Uschold, M. (eds.) Proc. of the ISWC Ontology Matching WS (November 2006)
46. Stephens, L.M., Gangam, A.K., Huhns, M.N.: Constructing Consensus Ontologies for the Semantic Web: A Conceptual Approach. World Wide Web Journal 7(4), 421–442 (2004)
47. Stuckenschmidt, H., Serafini, L., Wache, H.: Reasoning about Ontology Mappings. In: Proc. of the ECAI WS on Context Representation and Reasoning (2006)
48. Stuckenschmidt, H., van Harmelen, F., Serafini, L., Bouquet, P., Giunchiglia, F.: Using C-OWL for the Alignment and Merging of Medical Ontologies. In: Proc. of the First Int. WS. on Formal Biomedical K.R. (KRMed) (2004)
49. Tamilin, A.: Distributed Ontological Reasoning: Theory, Algorithms and Applications. PhD Thesis, University of Trento (February 2007)
50. van Hage, W., Katrenko, S., Schreiber, G.: A Method to Combine Linguistic Ontology-Mapping Techniques. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729. Springer, Heidelberg (2005)
51. Völker, J., Vrandecic, D., Sure, Y.: Automatic Evaluation of Ontologies (AEON). In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729. Springer, Heidelberg (2005)
52. Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Huebner, S.: Ontology-Based Integration of Information—A Survey of Existing Approaches. In: Proc. of the WS on Ontologies and Information Sharing (2001)

Time-Aggregated Graphs for Modeling Spatio-temporal Networks[⋆]

Betsy George[⋆⋆] and Shashi Shekhar

Department of Computer Science and Engineering,
University of Minnesota,
200 Union St SE, Minneapolis, MN 55455, USA
{bgeorge,shekhar}@cs.umn.edu
http://www.spatial.cs.umn.edu/

Abstract. Given applications such as location based services and the spatio-temporal queries they may pose on a spatial network (e.g., road networks), the goal is to develop a simple and expressive model that honors the time dependence of the road network. The model must support the design of efficient algorithms for computing the frequent queries on the network. This problem is challenging due to potentially conflicting requirements of model simplicity and support for efficient algorithms. Time expanded networks, which have been used to model dynamic networks employ replication of the networks across time instants, resulting in high storage overhead and algorithms that are computationally expensive. In contrast, the proposed time-aggregated graphs do not replicate nodes and edges across time; rather they allow the properties of edges and nodes to be modeled as a time series. Since the model does not replicate the entire graph for every instant of time, it uses less memory and the algorithms for common operations are computationally more efficient than for time expanded networks. One important query on spatio-temporal networks is the computation of shortest paths. Shortest paths can be computed either for a given start time or to find the start time and the path that lead to least travel time journeys (best start time journeys). Developing efficient algorithms for computing shortest paths in a time variant spatial network is challenging because these journeys do not always display optimal prefix property, making techniques like dynamic programming inapplicable. In this paper, we propose algorithms for shortest path computation for a fixed start time. We present the analytical cost model for the algorithm and compare with the performance of existing algorithms.

Keywords: time-aggregated graphs, shortest paths, spatio-temporal data- bases, location based services.

[⋆] This work was supported by NSF/SEI grant 0431141. The content does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

[⋆⋆] Corresponding author.

© Springer-Verlag Berlin Heidelberg 2008

1 Introduction

The growing importance of application domains such as transportation networks, emergency planning and location based services highlights the need for efficient modeling of spatio-temporal networks (e.g. road networks) that takes into account changes to the network over time. The model should provide the necessary framework for developing efficient algorithms that implement the frequent operations posed on such networks. Frequent queries on such networks might include finding the shortest route from one place to another or a search for the nearest neighbor. The shortest route would depend on the time dependent properties of the network such as congestion on certain road segments, which would increase the travel time on that segment. The result of a nearest neighbor search could also be time sensitive if it is based on a road network.

Modeling such a network poses many challenges. Not only should the model be able to accommodate changes and compute the results consistent with the existing conditions, it should do so accurately and simply. In addition, the need to answer frequent queries quickly means fast algorithms are required for computing the query results. The model should thus provide sufficient support for the design of correct and efficient algorithms for frequent computations.

Related work in the field of databases fall into three broad categories (1)spatial network databases, (2) graph Databases, and (3) spatio-temporal databases. The recent release of Oracle (version 10g) includes a network data model to store and maintain the connectivity of link-node networks and supports basic features such as shortest path computation [15]. The Network Analyst extension of ArcMap from ESRI supports a network geodatabase and provides basic algorithms (e.g., shortest path, service area, closest facility, etc.) [7]. However, these products do not address the time variance of spatial networks, which is crucial in applications such as route computations and emergency planning. Although the need for live traffic information is increasing, there has been little work on the modeling and algorithms for spatio-temporal network databases. Chorochronos [13], studied various aspects of spatio-temporal databases including ontology, modeling, and implementation. However, researchers have yet to study spatio-temporal networks in this framework.

Graph databases [5,6,7,19,23,24] also primarily deal with spatial networks that do not vary with time. Research in graph databases that accounts for temporal variations perform computations over a snapshot of the network [4,10,18], and do not consider the interplay between the edge travel times and the existence of edges. Ding [4] proposed a model that addresses time-dependency by associating a temporal attribute to every edge and node of the network so that its state at any instant of time can be retrieved. This model performs path computations over a snapshot of the network. Since the network can change over the time taken to traverse these paths, this computation might not give realistic solutions. It does not propose an algorithm for the least travel time paths.

Research in Operations Research is based on the time expanded network [11,12,14,16,17,21]. This model duplicates the original network for each discrete time unit $t = 0, 1, \ldots, T$ where T represents the extent of the time horizon. The expanded network has edges connecting a node and its copy at the next instant in addition to the edges in the original network, replicated for every time instant. The approach significantly increases the network size and is very expensive with respect to memory. Because of the increased problem size due to replication of the network, the computations also become quite expensive.

As the first step towards the study of spatio-temporal network databases, we proposed a spatio-temporal network model named time aggregated graph [8]. The proposed model, a time-aggregated graph, models the changes in a spatio-temporal network by collecting the node/edge attributes into a set of time series. The model can also account for the changes in the topology of the network. The edges and nodes can disappear from the network during certain instants of time and new nodes and edges can be added. The time-aggregated graph keeps track of these changes through a time series attached to each node and edge that indicates their presence at various instants of time. Our analysis shows that this model is less memory expensive and leads to algorithms that are computationally more efficient than those for the time expanded networks. Here, we build on this work by presenting a case study of this model using a routing algorithm (SP-TAG) that computes the shortest path in the given network for a given start time.

1.1 An Illustrative Application Domain

An important application domain for spatio-temporal network databases is transportation science [9], a multi-disciplinary field that requires expertise from different domains. The difficulty, but also fascination, of this professional practice derives from the intrinsic complexity of transportation systems, which have both physical and behavioral elements.The physical elements in the systems (e.g., vehicles, infrastructure, etc.) are governed by the laws of physics. On the other hand, the mechanisms underlying the functionality and performance of these physical elements are often connected to travelers' behavioral choices. Traditionally the center of behavioral choice modeling [22] has been user equilibrium [25], the idea that all travelers use the least inconvenient routes and no individual can unilaterally improve his/her travel. A key assumption of user user equilibrium is that travelers have perfect information about road conditions, and indeed this is generally true for commuters, who learn recurrent congestion patterns from their day-to-day travels. However, the assumption does not hold when the congestion is non-recurrent, in particular, when an extreme event occurs, and transportation network conditions become dynamic and uncertain. Thus one of the greatest challenges in transportation science is how to manage traffic in time-varying transportation networks, especially in disaster situations. This challenge cannot be met without the development of spatio-temporal databases. Currently, transportation management generates tremendous volumes of data and a large semantic gap exists between transportation science concepts and the concepts supported by current database systems. Emergency traffic management requires research in computer science to develop appropriate spatio-temporal database representations and query processing algorithms to make decisions in a timely

Table 1. Example Queries with Time-variance and Flow Networks

	Static	Time-Variant
Graph (No capacity constraints)	Which is the shortest travel time path from Minneapolis downtown to airport?	Which is the shortest travel time path from Minneapolis downtown to airport at different times of a work day?
Flow Network	What is the capacity of Twin-Cities freeway network to evacuate Minneapolis downtown?	What is the capacity of Twin-Cities freeway network to evacuate Minneapolis downtown at different times in a work day?

manner. Popular models of emergency traffic use time-variant flow-network [1] operations like min-cut and max-flow [2]. The queries typically encountered in emergency traffic management would involve time-variant properties, as illustrated by Table 1.

In addition to emergency management, many important applications, including travelers' trip planning, and consumer business logistics need to be built upon spatio-temporal network databases. Commuters often try to find a suitable time to start their commute so that they spend the least time in the traffic. There are many factors affecting the start time and the shortest route such as congestion levels, incident location, and construction zone. This is illustrated by the simple time-variant network shown in Figure 1. It can be seen that the travel time from node N1 to node N2 changes with the start time. If the travel starts at $t = 1$, the commute time would be 6 units; travel on the same route would take 4 units if the start time is moved to $t = 3$. This shows that the shortest paths in a time-dependent network vary with time which adds a new dimension to shortest path computation which cannot be ignored. Figure 2 illustrates traffic sensor networks on urban highways which measure congestion levels at two different times (e.g. 5:07pm and 9:37pm). With the increasing use of sensor networks that monitor traffic data on spatial networks and the consequent availability of time-varying traffic data, it becomes important to incorporate this data into the models and algorithms related to transportation networks. However, existing spatio-temporal databases do not provide adequate support for spatio-temporal networks. The problem of time variant networks finds similar applications in

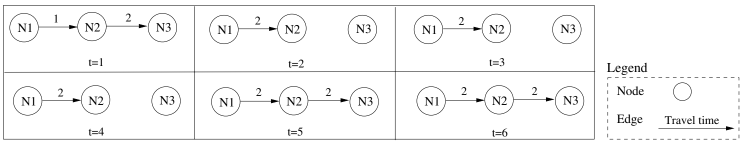


Fig. 1. Network at various instants

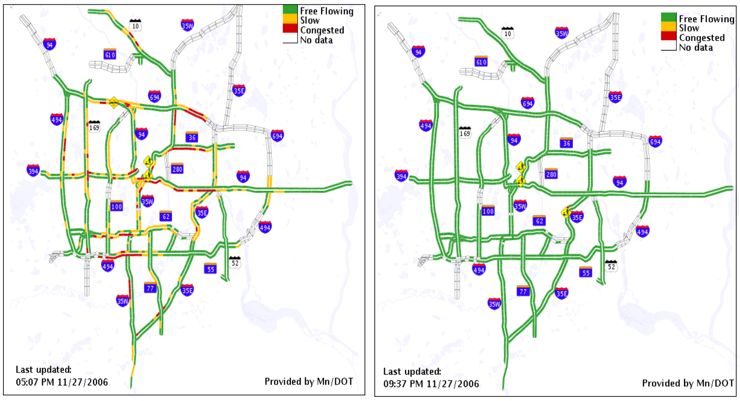


Fig. 2. Sensor networks periodically report time-variant traffic volumes on Twin Cities highways (Best viewed in color, Source: Mn/DOT)

business operations such as freight delivery services, one of whose main concerns is to reduce logistic costs such as fuel consumption, which is influenced by road congestion levels that vary over time.

1.2 Broad Computer Science Challenges

A time-variant graph is a graph whose edge and node properties and topological structure are time dependent. For example, traffic volume on urban highways varies over the time of day which leads to variation in travel time. In addition to network parameter values, the network topology can also change with time due to the unavailability of certain road segments during some periods of time due to repair or natural calamities. There are also be cases where the road segments are unavailable periodically due to traffic management strategies such as using all lanes of a street in the same direction to handle peak time congestion. Conventional graph algorithms cannot easily be applied to the snapshots at discrete time instants to evaluate frequent queries without accounting for relationships among snapshots.

Time-variant graphs raise many challenges for database research. Due to their potentially large and evergrowing sizes, a storage-efficient representation is critical to reduce and possibly eliminate redundant information across different time-points. Second, new data model concepts need to be investigated to represent and classify potentially new alternative semantics for common graph operations such as shortest-path and connectivity. For example, a shortest path between a given pair of nodes may have at least two interpretations, one for a given start time-point and the other for the shortest travel-time for any start time in a given time interval. A third challenge is the design of efficient and correct query processing strategies and algorithms since some of the commonly assumed graph-properties may not hold for spatio-temporal graphs. For example, consider the optimal prefix property (a requirement for the greedy approaches [2]) for shortest paths in a graph. While each prefix path (path from a source node to an intermediate node in an optimal path) is optimal in a static graph, it may not be optimal in a spatio-temporal graph due to the potential wait at the intermediate node. In the network shown in Figure 1, the best time to start a journey from node N1 to node N3 is $t = 4$, which takes 4 time units. The optimal path from N1 to N3 that starts at $t = 4$ is not optimal for the intermediate node N2. The best start time for a path from N1 to N2 is $t = 1$, which proves to be sub-optimal for a journey from N1 to N3. The lack of optimal prefix property in best start time shortest paths rules out the possibility of using a greedy strategy in algorithm design.

Our Contributions: Our approach to spatio-temporal databases has the following components:
*Graph Aggregation:*The temporal variation in the topology and parameter values can be represented using aggregates as edge/node attributes in the graph used to represent the spatial network. The edges and nodes can disappear from the network during certain instants of time and new nodes and edges can be added. The time-aggregated graph keeps track of these changes through a time series attached to each node and edge that indicates their presence at various instants of time.

Query Language: A query language needs to represent common queries.A key challenge is to define a complete set of logical operators for the time-aggregated graph.

Query Processing: The time aggregated graph with the proposed query operators will be used to process queries pertaining to the domain applications. A frequent query that arises in spatio-temporal networks is the shortest path computation. The algorithm needs to consider the availability of the required edges and nodes at the appropriate time instants. If the shortest route and the shortest route travel time are time-dependent, shortest path computation can be performed for a given start or it can find the least travel time path over the entire time period of interest.

In this paper we describe a model for spatio-temporal networks called the time aggregated graph based on graph aggregation.. The time-aggregated graph keeps track of the time-dependence of a graph through a time series attached to each node and edge that indicates their presence at various instants of time. We show that this model has less storage requirements than time expanded networks since it does not rely on replication of the entire network across time instants. We define a set of logical operators based on the time aggregated graph. We also propose an algorithm for computing the shortest route from one node to another based on this model.

1.3 Scope and Outline of the Paper

The paper presents a model for spatio-temporal networks called time aggregated graphs.

The rest of the paper is organized as follows. Section 2 discusses the basic concepts of the proposed model. This section provides an explanation of the model based on graph aggregation and the logical data model. It also explores design choices for the physical representation of the model and provides a comparison of the choices in the context of various logical operations. Section 3 proposes an algorithm for the shortest path computation based on this model. It also proposes the cost model for this algorithm. In section 4, we conclude and describe the direction of future work.

2 Basic Concepts

Spatial networks that show time-dependence serve as the underlying networks for most location based services. Traditionally graphs have been extensively used to model spatial networks (e.g. road networks) [19]; weights assigned to nodes and edges are used to encode additional information. In a real world scenario, it is not uncommon for these network parameters to be time-dependent. Formulation of computationally efficient and correct algorithms for the shortest path computation that takes into account the dynamic nature of the networks is important. Models of these networks need to capture the possible changes in topology and values of network parameters with time and provide the basis for the formulation of computationally efficient and correct algorithms for the frequent computations like shortest paths. Given the set of frequent queries posed by an application on a spatial network and the patterns of variations of the spatial network with time, we need to find a model that supports efficient and correct algorithms for computating the query results, while trying to minimize the storage and cost of computation. In this section we discuss the basics of the model used to represent spatial networks called "time aggregated networks" [8]. The algorithm presented in this paper is formulated based on this model. Time aggregated graphs can not only capture the time-dependence of network parameters, but also account for the possibility of edges and nodes being absent during certain instants of time.

2.1 The Conceptual Model

A graph $G = (N, E)$ consists of a finite set of nodes N and edges E between the nodes in N. If the pair of nodes that determine the edge is ordered, the graph is directed; if it is not, the graph is undirected. In most cases, additional information is attached to the nodes and the edges. In this section, we discuss how the time dependence of these edge/node parameters are handled in the proposed model, the time-aggregated graph. We define the time-aggregated graph as follows.

$taG = (N, E, TF, f_1 \ldots f_k, g_1 \ldots g_l, w_1 \ldots w_p | f_i : N \rightarrow \mathbb{R}^{TF}; g_i : E \rightarrow \mathbb{R}^{TF}; w_i : E \rightarrow \mathbb{R}^{TF})$ where
N is the set of nodes,
E is the set of edges,

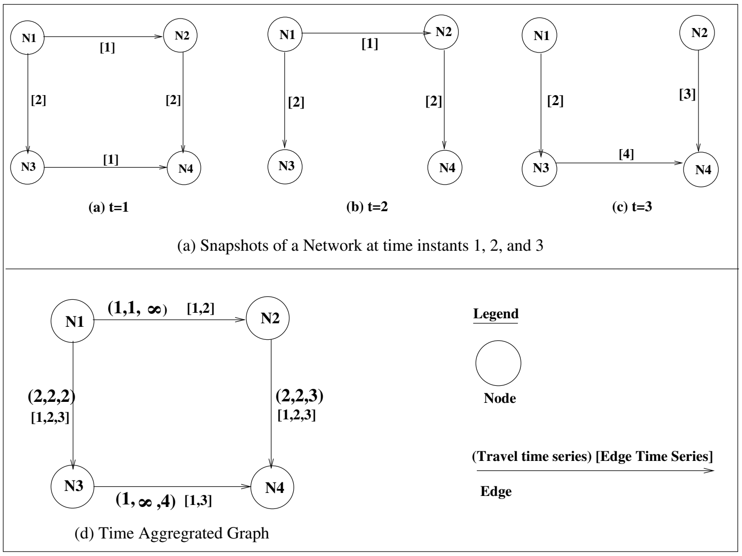


Fig. 3. Network At Various Time Instants and the Time Aggregated Graph

TF is the length of the entire time interval,
$f_1 \dots f_k$ are the mappings from nodes to the time-series associated with the nodes,
$g_1 \dots g_l$ are mappings from edges to the time series associated with the edges, and
$w_1 \dots w_p$ indicate the time dependent weights (eg. travel times) on the edges.
Each edge has an attribute, called an edge time series that represents the time instants for which the edge is present. This enables the time aggregated graph to model the topological changes of the network with time. We assume that each edge travel time has a positive minimum and the presence of an edge at time instant t is valid for the closed interval $[t, t + \sigma]$.

Figure 3(a,b,c) shows a network at three time instants. The network topology and parameters change over time. For example, edge N3-N4 is present at time instants $t = 1, 3$, and disappears at $t = 2$ and its weight changes from 1 at $t = 1$ to 4 at $t = 3$. The time aggregated graph that represents this dynamic network is shown in Figure 3(d). In this figure, edge N3-N4 has two attributes, both being a series. The attribute $(1, 3)$ represents the time instants at which the edge is present and $[1, \infty, 4]$ is the weight time series, indicating the weights at various instants of time. Figure 4(a) shows the time aggregated graph (corresponding to Figure 3(a),(b),(c)) and the time expanded graph that represent the same scenario. Edge weights in a time expanded graph are not explicitly shown as edge attributes; instead they are represented by edges that connect the copies of the nodes at various time instants. For example, the weight 1 of edge N1-N2 at $t = 1$ is represented by connecting the copy of node N1 at $t = 1$ to the copy of node N2 at time $t = 2$. The time expansion for the example network needs

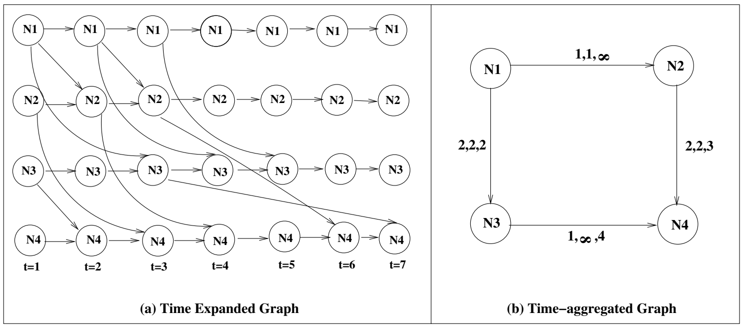


Fig. 4. Time-aggregated Graph vs. Time Expanded Graph

to go through 7 steps since the latest time instant would end in the network is at $t = 7$. For example, the traversal of edge N3-N4 that starts at $t = 3$ ends at $t = 7$, the travel time of the edge being 4 units. The number of nodes is larger by a factor of T, where T is the number of time instants and the number of edges is also larger in number compared to the time-aggregated graph. If the value of T is very large in a spatial network, it would result in enormously large time expanded networks and consequently slow computations.

2.2 A Logical Data Model

Basic Graph Operations

We extend the logical data model described in [19] to incorporate the time dependence of the graph model. The framework of the model consists of two dimensions (1) graph elements, namely node, edge, route and graph and (2) operator categories that consist of accessors, modifiers and predicates. A representative set of operators for each operator category is provided in Tables 2, 3 and 4. Table 2 lists a representative set of 'access' operators. For example, the operator *getEdge(node1,node2,time)* returns the edge properties of the edge from node *node*1 to node *node*2, such as the edge identifier (if any) and associated parameters at the specified time instant. For example operator *getEdge(N1,N2,1)* on

Table 2. Examples of Operators in the Accessor Category

	at_time	**at_all_time**	**at_earliest**
Node	get(node,time)	get_node(node)	get_node_earliest_Presence (node,time)
Edge	getEdge(node1,node2,time)	get_edge(node1,node2)	get_edge_earliest_Presence (node1,node2,time)
Route	getRoute(node1,node2,time)	get_route(node1,node2)	get_route_earliest_Presence (node1,node2,time)
Graph	get_Graph(time)	get_Graph()	—

Table 3. Examples of Operators in the Modifier Catefory

	insert		**delete**		**modify**	
	at_time	**at_all_time**	**at_time**	**at_all_time**	**at_time**	**at_all_time**
Node	insert(node, time,value)	insert(node, valueseries)	delete(node, time)	delete(node) delete(node)	update(node, time,value)	update(node, valueseries)
Edge	insert(node1, node2, time,value)	insert(node1, node2, valueseries)	delete(node1, node2, ,time)	delete(node1, ,node2) ,node2)	update(node1, node2,time value)	update(edge, valueseries)
Route	insert(node1, node2,time)	insert(node1, ,node2)	delete(node1 ,node2,time)	delete(node1, node2)		
Graph	insert(graph time)	insert(graph) insert(graph)	delete(graph, time)	delete(graph) delete(graph)	update(graph, ,time)	update(graph)

the time-aggregated graph shown in Figure 3 would return the travel time of the edge N1-N2 at $t = 1$, that is 1. Similarly, *get_edge(node1,node2)* returns the edge properties for the entire time interval. In Figure 3, the operator *get_edge(N1,N2)* would result in $(1, 1, \infty)$. *get_edge_earliest(N3,N4,2)* returns the earliest time instant at which the edge N3-N4 is present after $t = 2$ (that is $t = 3$). Table 3 shows a set of modifier operators that can be applied to the time aggregated graphs. For example, Figure 5(a) and (b) show a time aggregated graph before and after the *insert(N1,N4,3,4)* operation. this operation inserts edge N1-N4 at time instant $t = 3$ and the edge cost is 4. We also define two predicates on the time-aggregated graph.

exists_at_time_t: This predicate checks whether the entity exists at the start time instant t.

exists_after_time_t: This predicate checks whether the entity exists at a time instant after t.

Table 4 illustrates these operators. For example, node v is adjacent to node u at any time t if and only if the edge (u, v) exists at time t as shown in the table. *exists(N1,N2,1)* on the time aggregated graph in Figure 3 returns a "true" since the edge N1-N2 exists at $t = 1$.

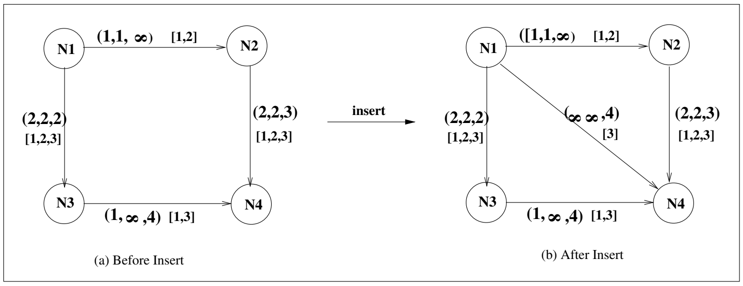


Fig. 5. A Time Aggregated Graph before and after an Insert Operation

Table 4. Predicate Operators in Time-aggregated Graphs

	exists_at_time_t	**exists_after_time_t**
Node	exists(node u,at_time_t)	exists(node u,after_time_t)
Edge	exists(node u,node v, at_time_t)	exists(node u,node v, after_time_t)
Route	exists(node u,node v,a_route r at_time_t)	exists(node u,node v,a_route r, after_time_t)

We list below, the fundamental entities in graphs, namely, *Graph*, *Node*, and *Edge* and a series of common operations that are associated with each class.

```
public class Graph   {
  public void add(Object label, timestamp t);
  // node with the given label is added at the time instant t.

  public void addEdge(Object n1, Object n2, Object label,
                          timestamp t, timestamp t_time)
  // an edge is added with start node n1 and end node n2 at
  // time instant t and travel time, t_time.

  public Object delete(Object label, timestamp t)
  // removes a node at time t and returns its label.

  public Object deleteEdge(Object n1, Object n2, timestamp t)
  // deletes the edge from node n1 to node n2 at t.

  public Object get(Object label, timestamp t)
  // returns the label of the node if it exists at time t.

public Iterator get_node_Presence_Series(Object n1)
// the  presence series of  node n1 is returned.

public Object getEdge(Object n1, Object n2, timestamp t)
// returns the edge from node n1 to node 2 at time instant t.

public Iterator get_edge_Presence_Series(Object n1, Object n2)
// the  presence series of edge from node n1 to node n2
// is returned.

public Object get_a_Successor_node(Object label, timestamp t)
//  an adjacent node of the vertex is returned if an edge exists
//  to this node at a time instant at or after t.

public Iterator get_all_Successor_nodes(Object label, timestamp t)
//  all adjacent nodes are returned if edges exist to them
//  at time instants at or after t.

public Object get_an_earliest_Successor_node(Object label,timestamp t)
//  the adjacent node which is connected to the given node with
//  the earliest time stamp after t is returned.

public timestamp get_node_earliest_Presence(Object n1,
                                            timestamp t)
//  the earliest time stamp after t at which the node n1
//  is available is returned.

public timestamp get_node_Presence_after_t(Object n1,
                                           timestamp t)
//  Part of the presence time series of node n1 after time t
//  is returned.


public timestamp get_edge_earliest_Presence(Object n1, Object n2,
                                                       timestamp t)
//  the earliest time stamp after t at which the edge from
// node n1 to node n2 is available is returned.

public timestamp get_edge_Presence_after_t(Object n1, Object n2,
                                                      timestamp t)
//  Part of the presence time series of edge(n1-n2) after time t
//  is returned.

}
```

A few important operations associated with the classes **Nodes** and **Edges** are p rovided below.

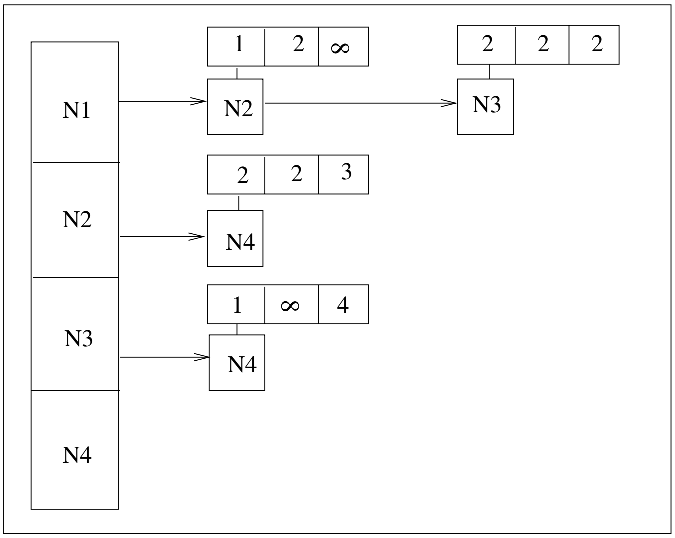
```
public class Node  {
        public Node(Object label, timestamp t)
        // the constructor for the class. A node with the appropriate
        // label is created at the time t.

        public Object label()
        // returns the label associated with the node if it exists at t.
}

public class Edge  {
        public Edge(Object n1, Object n2, Object label,
                        timestamp t_inst, timestamp t)
        // the constructor for the class. an edge is added with start
        // node n1 and end node n2 at time instant t and

        // travel time, t_time.
        public Object start()
        // returns the start node of the edge.

        public Object end()
        // returns the end node of the edge.
}
```

2.3 Physical Data Model

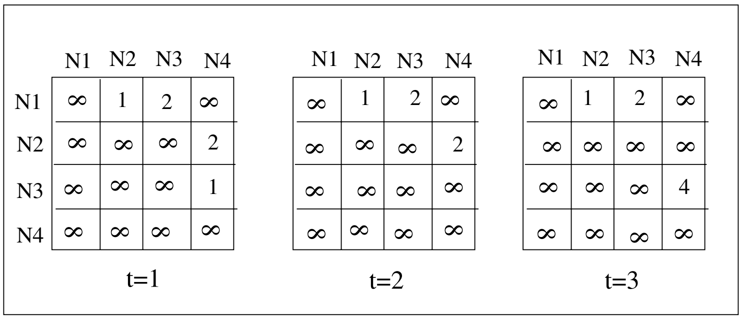
A static graph $G = (V, E)$ can be represented using an adjacency matrix. This is a $|V| \times |V|$ matrix, A such that the element a_{ij} is defined as
$a_{ij} = w_{ij}$ if $ij \in E$ and w_{ij} is the weight of the edge ij and
$a_{ij} = 0$, otherwise. This representation requires $O(N^2)$ memory. It can be seen that the storage required for this representation is independent of the number of edges in the graph, in relation to the number of nodes. In other words, there is no saving in memory even when the graphs are sparse. One representation that can exploit such sparsity is the adjacency list representation. The adjacency list representation of a graph $G = (V, E)$ consists of an array of lists, one for each vertex $v \in V$. The list corresponding to a vertex v contains all vertices that are adjacent to v in G. For a directed graph, the space requirement for the lists is $O(m)$ where $m = |E|$. The total memory requirement is $O(n+m)$ where $n = |V|$. The weight of each edge uv is stored with the vertex v in u's adjacency list. This representation is especially suitable for sparse graphs.

Time aggregated graphs can be represented by either one of the representation, with the necessary modifications. These representations need to be extended to include the time series representations on edges (corresponding to time dependent edge costs) and nodes. Adjacency list representation is extended by adding a list to each vertex in the adjacency list. Adjacency list representation uses an array of pointers one pointer for each node. The pointer for each node points to a list of immediate neighbors. Stored at each neighbor node are the edge presence series and travel times for the edge starting from the first node to this neighbor. Since the length of the time series is T, where T is the length of the time period, the adjacency list representation would require $O((m + n)T)$, where n is the number of nodes and m is the number of edges.

To extend the adjacency matrix to represent the time aggregated graph, a third dimension can be added. The new matrix A would be $n \times n \times T$, requiring $O(n^2T)$ memory. Figure 6 (a) and (b) show the adjacency list and adjacency matrix representations for the time aggregated graph shown in Figure 3. For example, the edge N1-N2 in the graph at $t = 1$ is represented by the pointer from N1 to N2 in the adjacency list. The array $(1, 2, \infty)$ is stored at N2 to represent the travel times at $t = 1, 2, 3$ for the edge N1N2. In the adjacency matrix the presence of edge N1N2 at a time instant $t = 1$ is represented by $A[1, 2, 1] = 1$, since the travel time for the edge is 1 unit at $t = 1$. Since the edge is absent at an instant $t = 3$, $A[1, 2, 3] = \infty$ which indicates an infinite edge cost at time instant $t = 3$. Note that the start node, the end node and the time instant are represented by the first, second and the third dimension of the matrix. Though the adjacency matrix has been



(a) Adjacency List Representation



(b) Adjacency Matrix Representation

Fig. 6. Storage Structures for Time Aggregated Graph

illustrated as three snapshots in Figure 6(b) for the sake of clarity, they are represented in one, three-dimensional matrix.

Logical operations on a time-aggregated graph can be classified as

1. Topology first operators (graph dominated operations). Examples include get_route(n1,n2) and get_edge(n1,n2).
2. Time-first operators (Time dominated queries).
   Some examples are get_Graph(time t) and get_edge_at_t(n1,n2,t).

Both representations are equally capable of handling graph dominated queries. To compute time first operations (snapshot queries such as to find the graph at a given time instant), adjacency matrix representation is more suitable. In this

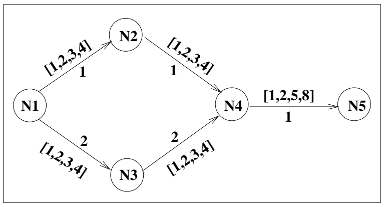


Fig. 7. Illustration of Shortest Paths

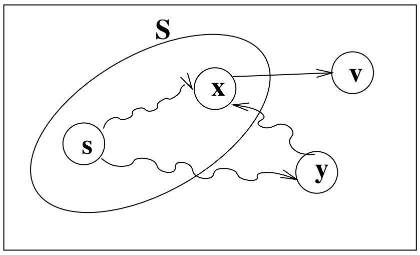


Fig. 8. Correctness of SP-TAG Algorithm

representation, these queries represent the time slices of the matrix at the given time instants.

Graphs representing transportation networks are generally sparse and hence adjacency list representation is more likely to be storage efficient compared to adjacency matrix representations. The choice is hence a tradeoff between the storage cost and the frequency of time dominated queries. We expect route queries (which are topology first queries) to be more frequent and since adjacency list representation is capable of handling these, based on storage costs, we used adjacency lists in our implementations. Moreover, most databases use adjacency list representation.

Comparison of Storage Costs with Time Expanded Networks
According to the analysis in [20], the memory requirement for time expanded network is $O(nT) + O((n+m)T)$, where n is the number of nodes and m is the number of edges in the original graph. The framework of a time aggregated graph would require a memory of $O(n+m)$, where n is the number of nodes and m is the number of edges. Edges and nodes with time-varying attributes have attribute time series associated with them. If the average length of the time series is $\alpha(\leq T)$, the memory required is $O(\alpha m + \alpha n)$, assuming an adjacency list representation. The total memory requirement for a time aggregated graph is $O(n + m + \alpha m + \alpha n)$. This comparison shows that the memory usage of time-aggregated graphs is less than that of time expanded graphs if $\alpha < T$.

3 Shortest Path Computation for Time Aggregated Graphs (SP-TAG Algorithm)

In a time dependent network, the shortest path and its traversal time are dependent on the start time at the source node. Though it is common to apply greedy strategies in optimization problems such as shortest path computation, this approach presents a challenge in time aggregated graphs, where not all shortest paths display an optimal sub-structure. Figure 7 gives an example. For the sake of simplicity, the travel times are assumed to be constant in this example. It can be seen that a shortest path (N1-N3-N4-N5) from N1 to N5 for the start time $t = 1$, which takes 5 time units does not display optimal substructure. The path from N1 to N4 following the above path is not optimal (shortest path being, N1-N2-N4). Although such paths that do not display optimal sub-structure could exist, it can be proved that there is at least one optimal path which satisfies the optimal sub-structure property [8].

Lemma 1. There is at least one optimal path which satisfies the optimal sub-structure property.

Proof. As Figure 7 illustrates, a shortest path fails to display optimal structure of due to a potential wait at intermediate node (u), after reaching this node traversing the optimal path from s to u. Consider the optimal path from s to u. Append this path to the path $u - d$ (allowing a wait at the intermediate node u) from the optimal path. This would still be the shortest path from s to d. Otherwise, it will contradict the optimality of the original shortest path.

This result enables us to use a greedy approach to compute the shortest path.

The algorithm, called SP-TAG uses greedy strategy to find the shortest path for a fixed start time. Every node has a cost associated with it which represents the travel time to reach the node from the source node. The algorithm picks the node with the least cost and updates the costs of its adjacent nodes. While finding the adjacent nodes, each edge is selected at its earliest available time instant (*get_edge_earliest_Presence* operation in the algorithm description). A trace of the algorithm is given in table 5. The table entries are the costs associated with each node (representing the arrival times at the node) at each iteration. The node marked as "closed" is the node with minimum cost selected for expansion. The travel times are assumed to follow the FIFO property.

Lemma 2. The SP-TAG algorithm is correct.

Proof. The proof of correctness of the algorithm which follows a greedy strategy follows the proof of correctness for Dijkstra's algorithm to find the shortest path from a source node to a destination. The key difference in time aggregated graph is that each edge has a presence series. SP-TAG employs a greedy approach where it selects the earliest available time instant as the traversal time of the edge. Since waits are permitted at intermediate nodes, this admissible approach does not violate the optimality of the shortest path even while considering the time-dependence of edge presence.

Table 5. Trace of the SP-TAG Algorithm for the Network shown in Figure 7

Iteration	**N1**	**N2**	**N3**	**N4**	**N5**
1	1 (closed)	∞	∞	∞	∞
2	1	2 (closed)	3	∞	∞
3	1	2	3 (closed)	3	∞
4	1	2	3	3 (closed)	6
5	1	2	3	3	6 (closed)

Algorithm 1. Shortest Path (SP-TAG) Algorithm

```
Input:
  1) $G(N,E)$: a graph $G$ with a set of nodes $N$
           and a set of edges $E$;
     define type $nn$ positive integer
     Each node $n \in N$ has one property:
        $NodePresenceTimeSeries$ : series of $nn$
     Each edge $e \in E$ has two properties:
        $Edge\ Presence\ TimeSeries$,
        $Travel\ time\ series$ : series of $nn$
     $\sigma_{u,v}(t)$ - travel time of edge $uv$ at time $t$.
  2) $s$: Source node, $s \subseteq N_G$;
  3) $d$: Destination node, $d \subseteq N_G$;
  4) $t_{start}$: Start Time;
Output: Shortest Route from $s$ to $d$ for $t_{start}$
Method:
        $c_s = t_{start}; \forall v \neq s, c_v = \infty$;
        // $c_u$ is the cost at the node $u$.
        insert($Q, s$);
        //$Q$ is a min-heap.
        while $Q$ not empty {
           $u = extract\_min(Q)$;
           for each node $v$ adjacent to $u$ do {
              $t = get\_edge\_earliest\_Presence(u, v, c_u)$;
              $\sigma = get\_edge(u, v, t)$;
              $relax(u, v, \sigma)$;
              insert($Q, v$) if $v$ is relaxed;
              }
           update($Q$);
        }
     }
  }
Output the route from $s$ to $d$.
```

To prove the correctness of the algorithm, we need to show that the cost of a node, when it is closed, is the shortest path distance to the node. This can be proved by induction on the set of closed nodes (S in Figure 8). Let v be the next node to be closed. Suppose the cost of node v was last updated when node x was added to S and v is adjacent to x. When x was added to S, a shorter path to v through x was discovered. Assume that the cost of v is not the shortest path cost. This would be due to the existence of a path $s \cdots y \cdots xv$ as shown in Figure 8. Since x was closed before y, the shortest path to x is inside S by inductive hypothesis. Therefore, the length of the path from s to v through y cannot be shorter that the path $s \cdots xv$. The cost of v cannot be further reduced by forming a path through nodes outside S. hence, the cost of the node when it is closed is the shortest path distance to the node.

Lemma 3. The time complexity of the SP-TAG algorithm is $O(m(\log T + \log n))$ where T is the number of time instants, n is the number of nodes and m is the number of edges in the time aggregated graph.

Proof. The cost model analysis assumes an adjacency list representation of the graph with two significant modifications. The edge time series is stored in the sorted order. Attached to every adjacent node in the linked list are the edge time series and the travel time series.

For every node extracted from the priority queue Q, there is one edge time series look up and a priority queue update for each of its adjacent nodes. The time complexity of this step is $O(\log T + \log n)$. The asymptotic complexity of the algorithm would be
$O(\Sigma_{v \in N}[degree(v).(\log T + \log n)]) = O(m(\log T + \log n))$.
The time complexity of the SP-TAG shortest path algorithm based on a time expanded network is $O(nT \log T + mT)$ [3]. Assuming a sparse graph where m is $O(n)$, $nT \log T < m \log T$. The SP-TAG algorithm is faster than the algorithm based on time expanded graph if $m \log n < mT$. In other words, the SP-TAG algorithm is faster if $\log n < T$.

3.1 Summary of Experimental Evaluation

An experimental analysis of the SP-TAG algorithm was performed to compare its run-time with an algorithm based on a time-expanded graph. Time expanded graphs make copies of the original network for every time instant under consideration. In our experiments the following were selected as the independent parameters: 1) network size represented by the number of nodes; and 2)the length of the time interval in terms of number of time instants. The networks chosen were road maps from the Minneapolis downtown area in USA with radii of .5 mile, 1 mile, 2 miles and 3miles. This was appended with travel time series of various lengths. The travel time series were synthetically generated. This data was fed to both a time expanded graph generator, which generates an expanded graph encoding of the travel time information. An algorithm for computing the shortest path for a given start time was run on this graph. The SP-TAG algorithm was run on the same dataset and the results were compared. The experiments were conducted on a SUN Solaris workstation with 1.77GHz CPU, 1GB RAM and UNIX operating system. The experimental results reported are the average over 5 experiment runs with networks generated using the same input parameters, but with different destination nodes.

Experimental Results and Anlaysis
We wanted to answer two questions: (1) How does the network size (number of nodes, number of edges) affect the performance of the algorithms? (2) How does the length of the time series affect the performance of the algorithms?

In the experiment to evaluate the effect of network size on the performance of the algorithm, we fixed the length of the travel time series and varied the network size to observe the run times of the SP-TAG algorithm and time expanded graph

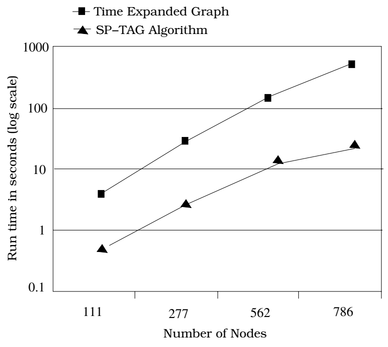


Fig. 9. SP-TAG Algorithm: Run-time With Respect to Network Size

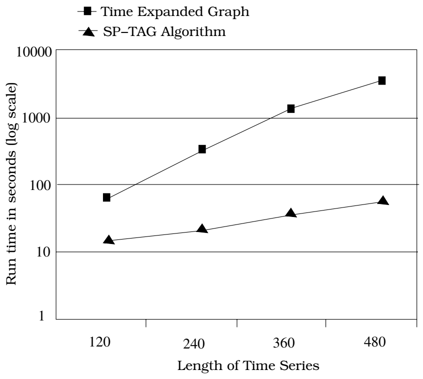


Fig. 10. SP-TAG Algorithm: Run-time With Respect to Length of Time Series

based algorithm. Figure 9 shows the run-time of the fixed start time algorithm based on the time aggregated graph and the performance of the algorithm based on the time expanded graph. As can be seen, the SP-TAG algorithm runs faster than the time-expanded graph based algorithm in all cases; further, its run-time seems to increase at a slower rate.

In the second experiment, we evaluated how the number of time instants affects the performance of the algorithms. We fixed the network size, and varied the length of the time series to observe the run-time. The number of time instants was varied and the network size parameters were fixed. As seen in Figure 10, the SP-TAG algorithm performs better.

Discussion: Due to the interplay between the travel times and the availability of the edges, the shortest path displays some interesting properties. Consider the time aggregated graph shown in Figure 11. Assume that edges are present at all time instants. The travel times are time dependent as shown in the cost time series. For example, the travel time of edge N2-N3 is 5 units at $t = 2$ and 1 unit at $t = 3$. Consider a journey that starts at $t = 1$ at node N1. It reaches node N2 at $t = 2$. Since edge N2-N3 is available at $t = 2$ (this being the earliest availability), the greedy algorithm would traverse the edge at $t = 2$ reaching node N3 at $t = 7$, travel time being 5 time units at $t = 2$. If the algorithm had made a decision to wait at N2 until $t = 3$, N3 would be reached at $t = 4$. This indicates that the travel times should follow the FIFO property for the

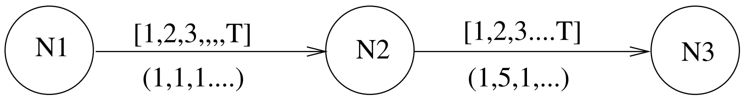


Fig. 11. Illustration of Effect of non-FIFO travel times

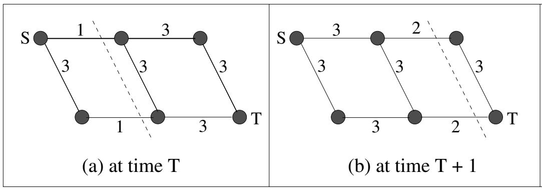
greedy algorithm to compute the optimal path. Though the FIFO property is satisfied by travel times in most situations, strategies can be explored to handle non-FIFO travel times also.

4 Conclusions and Future Work

Spatio-temporal networks form a key part of critical applications such as emergency planning and there is a great need for database support in this area. The paper describes a model based on time aggregation to represent a spatio-temporal network and proposes an algorithm for shortest path computation. It also defines a set of logical operators on the time aggregated graphs. We present an analytical evalution of the shortest path algorithm and storage cost requirements of the proposed time aggregated graph. Evaluation shows that the algorithms based on time aggregated graphs significantly reduce the computational cost compared to similar algorithms based on time expanded networks and the model is less memory expensive than the existing models.

We plan to evaluate the performance of the algorithms using real-traffic datasets shortly. We expect this evaluation to give new insights into the average case run time of the algorithms, which we expect to be significantly better than the worst case complexity. We believe that time aggregated graphs can accomodate the time-varying capacities of the road networks. The proposed algorithms need to be extended to give optimal solutions subject to the constraints of time-varying capacities. This would extend the use of the algorithms to domains such as evacuation planning in emergency management, where capacity constraints in the network pose significant challenges. Flow networks [1] have been extensively used in evacuation planning. We plan to use time aggregated graphs to represent time-variance in flow networks. Time-variance poses novel challenges for flow network operations by introducing alternative interpretations of traditional operations. Consider a query to identify bottleneck capacity of a transportation network (modeled as a minimum cut) shown in Figure 12 at two time instants T and T+1. The numbers associated with various edges represent their capacities. At time T, the bottleneck (i.e., minimum cut) of this network is 2 for flows starting from node S towards destination node T as shown in Figure 12(a). At time T+1, the bottleneck changes to 4 as shown in Figure 12(b). Thus, the minimum-cut of this time-variant flow-network may be a function of time. A database may allow aggregate queries over time-variant network-flow properties like min-cut. Figure 12(c) shows an example of a query to find an average among time variant min-cuts with temporal range. We also plan to incorporate the algorithms as building blocks that finds the shortest paths in the CCRP evacuation planner [14]. We will also explore other graph problems in the context of time aggregated graphs. We would explore ways to include spatial attributes at nodes and edges and incorporate necessary changes in the algorithms.

Spatial properties need to be represented in the time aggregated graph, which might add to the effectiveness of the model and may lead to the formulation of efficient algorithms. For example, the spatial location of a node can be



```
SELECT AVERAGE(MINCUT(node, edge(t)))
FROM node, edge
WHERE t BETWEEN T AND T+1;
```

(c) Query Example

Fig. 12. Two Min-cut graphs with time variant capacity and a query example

represented as a node attribute. We plan to explore effective ways to incorporate spatial properties of nodes and edges in the model. We also plan to include operators that handle time intervals as parameters.

Acknowledgment

We are thankful to the reviewers for their helpful comments, especially in relation to the computational complexity of the shortest path algorithm. We wish to thank Sangho Kim from the Spatial Database Research Group at the University of Minnesota for his valuable input on time-variant flow graphs. We are grateful to the members of the Spatial Database Research Group at the University of Minnesota for their helpful comments and to Kim Koffolt for her help in improving the readability of this paper.

This work was supported by NSF/SEI grant 0431141, Oak Ridge National Laboratory grant and US Army Corps of Engineers (Topographic Engineering Center) grant. The content does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

References

1. Ahuja, R., Magnanti, T., Orlin, J.: Network Flows - Theory, Algorithms, and Applications. Prentice-Hall, Englewood Cliffs (1993)
2. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, ch. 26, Flow Networks. MIT Press, Cambridge (2002)
3. Dean, B.C.: Algorithms for minimum-cost paths in time-dependent networks. Networks 44(1), 41–46 (2004)
4. Ding, Z., Guting, R.H.: Modeling temporally variable transportation networks. In: Proc. 16th Intl. Conf. on Database Systems for Advanced Applications, pp. 154–168 (2004)
5. Erwig, M.: Graphs in Spatial Databases. PhD thesis, Fern Universität Hagen (1994)
6. Erwig, M., Guting, R.H.: Explicit graphs in a functional model for spatial databases. IEEE Transactions on Knowledge and Data Engineering 6(5), 787–804 (1994)
7. ESRI. ArcGIS Network Analyst (2006), `http://www.esri.com/software/arcgis/extensions/`

8. George, B., Shekhar, S.: Time-aggregated Graphs for Modeling Spatio-Temporal Networks - An Extended Abstract. In: Proceedings of Workshops at International Conference on Conceptual Modeling (November 2006)
9. Hall, R.W. (ed.): Handbook of Transportation Science. Kluwer Academic Publishers, Dordrecht (2003)
10. Hamre, T.: Development of Semantic Spatio-temporal Data Models for Integration of Remote Sensing and in situ Data in Marine Information System. PhD thesis, University of Bergen, Norway (1995)
11. Kaufman, D.E., Smith, R.L.: Fastest paths in time-dependent networks for intelligent vehicle highway systems applications. IVHS Journal 1(1), 1–11 (1993)
12. Kohler, E., Langtau, K., Skutella M.: Time-expanded graphs for flow-dependent transit times. In: Proc. 10th Annual European Symposium on Algorithms, pp. 599–611 (2002)
13. Koubarakis, M., Sellis, T.K., Frank, A.U., Grumbach, S., Güting, R.H., Jensen, C.S., Lorentzos, N.A., Manolopoulos, Y., Nardelli, E., Pernici, B., Schek, H.-J., Scholl, M., Theodoulidis, B., Tryfona, N.(eds.): Spatio-Temporal Databases. LNCS, vol. 2520. Springer, Heidelberg (2003)
14. Lu, Q., George, B., Shekhar, S.: Capacity Constrained Routing Algorithms for Evacuation Planning: A Summary of Results. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633. Springer, Heidelberg (2005)
15. Oracle. Oracle Spatial 10g, An Oracle White Paper (August 2005), http://www.oracle.com/technology/products/spatial/
16. Orda, A., Rom, R.: Minimum weight paths in time-dependent networks. Networks 21, 295–319 (1991)
17. Pallottino, S., Scuttella, M.G.: Shortest path algorithms in tranportation models: Classical and innovative aspects. In: Equilibrium and Advanced transportation Modelling (Kluwer), pp. 245–281 (1998)
18. Rasinmäki, J.: Modelling spatio-temporal environmental data. In: 5th AGILE Conference on Geographic Information Science, Palma, Balearic Islands, Spain (April 2002)
19. Shekhar, S., Chawla, S.: Spatial Databases: Tour. Prentice-Hall, Englewood Cliffs (2003)
20. Sawitzki, D.: Implicit Maximization of Flows over Time. Technical report, University of Dortmund (2004)
21. Dreyfus, S.E.: An appraisal of some shortest path algorithms. Operations Research 17, 395–412 (1969)
22. Sheffi, Y.: Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Method. Prentice-Hall, Englewood Cliffs (1985)
23. Shekhar, S., Liu, D.: CCAM: A Connectivity-Clustered Access Method for Networks and Networks Computations. IEEE Transactions on Knowledge and Data Engineering, vol. 9 (January 1997)
24. Stephens, S., Rung, J., Lopez, X.: Graph data representation in oracle databese 10g: Case studies in life sciences. IEEE Data Engineering Bulletin 27(4), 61–66 (2004)
25. Wardrop, J.: Some theoretical aspects of road traffic research. Proceedings of the Institution of Civil Engineers 2(1) (1952)

An Architecture for Emergent Semantics

Sven Herschel[1], Ralf Heese[1], Jens Bleiholder[2], and Christian Czekay[1]

[1] Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin, Germany
{herschel,rheese,czekay}@informatik.hu-berlin.de
[2] Hasso-Plattner-Institut für Softwaresystemtechnik
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
jens.bleiholder@hpi.uni-potsdam.de

Abstract. Emergent Semantics is a new paradigm for inferring semantic meaning from implicit feedback by a sufficiently large number of users of an object retrieval system. In this paper, we introduce a universal architecture for emergent semantics using a central repository within a multi-user environment, based on solid linguistic theories.

Based on this architecture, we have implemented an information retrieval system supporting term queries on standard information retrieval corpora. Contrary to existing query refinement strategies, feedback on the retrieval results is incorporated directly into the actual document representations improving future retrievals.

An evaluation yields higher precision values at the standard recall levels and thus demonstrates the effectiveness of the emergent semantics approach for typical information retrieval problems.

1 Introduction

The elementary challenge in all object retrieval tasks is to find an object representation that can later effectively and efficiently be matched against a user query in order to find and rank the objects according to the user's needs.

Researchers in information retrieval (IR), to select a prominent example, have been very successful in finding document representations for an effective later retrieval. Albeit being today's state of the art, the vector space model [1] used in conjunction with Latent Semantic Indexing [2], constitutes only a syntactical approach in finding a so-called semantic representation.

Emergent semantics aims to emerge object representations by aggregating many user's opinions about the object content, therefore providing object representations that a majority of actual users of a system agree upon. We believe that finding such a representation considerably improves precision, since it has been created by the users themselves. A basic example illustrates the idea behind emergent semantics: In a park near our campus, the landscape architects decided on not paving walkways initially. Instead, they covered the entire area with lawn. After a year they came back and knew exactly where to pave walkways, since the walkers had obviously decided which pathways they will use by actually walking them: the paths were all torn and muddy. We claim that this approach can be

© Springer-Verlag Berlin Heidelberg 2008

transferred into the area of computer science and, termed emergent semantics, represents a major advancement in the way object representations are created and maintained.

Our contribution with this paper is a formal approach to the emergent semantics paradigm, an architecture for utilizing its full potential and an implementation within the area of IR that demonstrates the possibilities associated with this paradigm. Our preliminary results show higher precision values at the standard recall levels, thus demonstrating the effectiveness of the emergent semantics approach for typical IR problems.

1.1 Applications of Emergent Semantics

To illustrate its potential, we outline some of the applications we see for emergent semantics.

Human view. Emergent semantics leads to object representations created by humans and created for humans. This in turn leads to an improved user experience, because an object is represented by the terms humans would expect for it.

Data reduction. Emergent semantics might be used to identify terms with higher relevancy within an initial object representation. This can be used to reduce the size of the object representation to contain more representative terms.

Inferring synonyms & cross-language meaning. By evaluating co-occurrences between object representations emergent semantics techniques can be used to infer synonym relationships between terms or even relationships across languages.

Creation of controlled vocabularies. We expect emergent semantics to play an important role in finding controlled vocabularies within domains. After an emergent semantics system has been deployed, the most frequently used representation tokens can be used, together with additional information like co-occurrence measures, to form a controlled vocabulary for the respective domain.

Retrieval of non-text objects. While many approaches exist for (purely syntactical) meaning extraction from textual objects, only a few exist for the automatic representation of multimedia objects such as images, music or videos. Instead of leaving the creation of representations for these objects to experts, emergent semantics can help find an object representation the majority of users agrees upon.

1.2 Structure of This Paper

This paper is structured as follows: in Section 2, "Groundwork", we introduce the linguistic background of syntax, semantics, and pragmatics. To further underline the semiotic concepts introduced in this section, we deliberately chose our section headlines to potentially carry different meanings within this paper and outside its scope. We then adopt these cognitions for computer science by introducing a universal architecture for emergent semantics in "Model House" (Section 3). In Section 4, "Construction", we introduce our implementation of this architecture for a classic IR scenario in order to be able to present promising results of our evaluation in Section 5. Related work is outlined in "Neighborhood" (Section 6), and "Roof and windows" (Section 7) concludes the paper with an outlook on open research issues and future work.

2 Basement

In this section, we introduce the linguistic background of syntax, semantics, and pragmatics in order to motivate our architecture for emergent semantics. This is essential for understanding the process leading from the user need, expressed by a query, to the retrieval and ranking performed by the system. Syntax, semantics and pragmatics are all subfields of *semiotics*, the study of signs, their meaning, and their interpretation by humans. According to Morris [3], these subfields can be described as follows:

Syntax: the study of signs and their interrelation
Semantics: the study of signs and their relation to the objects they represent
Pragmatics: the study of signs and their relation to the user interpreting them

According to this theory, every sign is assigned a meaning within the context of a person's understanding. So the letters *t, r, e*, and *e* form the word *tree* which, in English, refers to the biological wooden structure or it refers to a mathematical figure that branches in a single root. Different people, however, may have something different in mind when being confronted with the word *tree*. Concepts may reach from a beautiful day in the park to the latest forest fire in Portugal.

Let us take a slightly different look at these concepts. The semiotic triangle according to Ogden [4] (see Figure 1 puts the three semiotic concepts *sign*, *thought* and *referent* into relation. Signs are the basis for communication. To enable a successful communication, both parties must be able to read and understand the *sign*. The corresponding *thought* evoked within the receiving party's mind shall then refer to the real-world object or concept, the so-called *referent*. Because of synonyms and homonyms it is difficult to directly put a *sign* into direct relation to a *referent*. Ogden therefore decided to relate these two using a dashed line.

Computer science algorithms explore specific transitions between *sign*, *thought* and *referent*, as depicted in Figure 1. Algorithms for generating object representations from signs are usually syntactical with the implied hope of inferring something similar to semantics by applying clever extraction algorithms.

An example for such an advanced algorithm is Latent Semantic Indexing (LSI) in the context of textual IR [2] which claims to extract document representations that are – by human judgement – considered good representations for the respective documents. LSI aims to directly walk the dashed line from a *sign*, i.e., the syntax to the *referent*, i.e., the semantics: sophisticated algorithms analyze the collection content and regroup all terms from the standard term-document

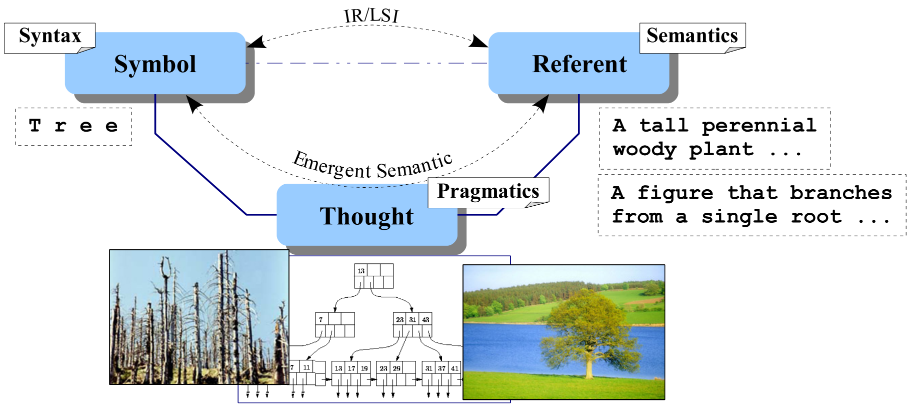

IR/LSI
Syntax
Symbol
Referent
Semantics
T r e e
A tall perennial woody plant ...
A figure that branches from a single root ...
Emergent Semantic
Pragmatics
Thought

Fig. 1. Semiotic triangle

matrix into *concepts* by using the computationally expensive Singular Value Decomposition (SVD). The resulting document-concept matrix can then be pruned for efficiency and transformed back into a corresponding term-document matrix in order to be accessible for retrieval algorithms based on the Vector Space Model (VSM). Latent Semantic Indexing algorithms therefore extract the meaning of a document by representing a document with terms found in the document collection. The obvious drawback of this purely syntactical approach is that only terms already found in the document collection itself can be used within document representations.

Emergent semantics in turn exploits the user's *thought* when walking from *sign* to *referent*. It aggregates different users' *thoughts* about the meaning of a *sign* in order to create a *referent* representation that the majority of users agrees upon. With only little loss of accuracy, we identify the *referent* with its agreed-upon semantics in this paper.

The user's *thought*, i.e. the user's pragmatic view includes the specific user background, i.e., culture, education, and social background, as well as very time-dependent influences like the user's mood, for example. It is our assumption that this diverse background will lead to a lot of noise, if applied unconditionally to the document representations. It is therefore necessary to provide means to eliminate this noise. Our architecture provides means for user-based as well as for collection-based noise elimination. Please refer to the description of our *annotation filter* in Section 3.1.

3 Model House

In this section, we introduce the basic building blocks of an emergent semantics (EmSem) architecture and explain their functionality. We arranged the components of our architecture to reflect the semiotic subfields introduced in the last chapter, so our description follows a typical query processing workflow. Please

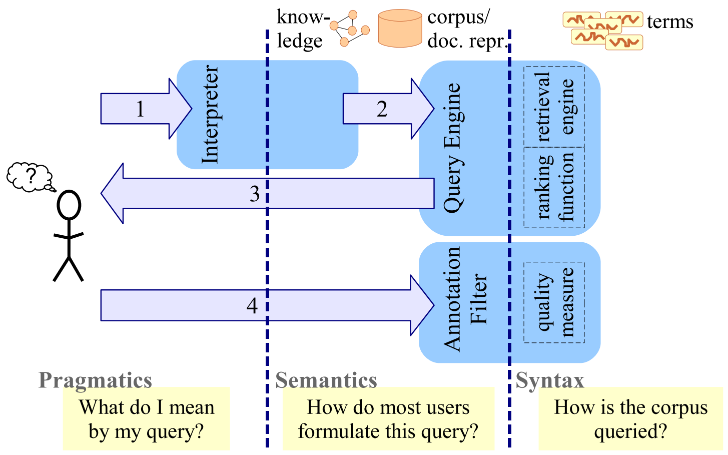


Fig. 2. An architecture for emergent semantics

refer to Figure 2 for an overview of both the architecture and the fundamental query processing steps. For now we assume a simple environment consisting of a central repository storing all data and a large number of clients querying the server. The individual components are explained in the sections below.

3.1 Ingredients

In the context of emergent semantics, we understand object retrieval as a four step process. At first, a user develops an idea of her information need – "what do I mean by my query?" She formulates a *query* to the best of her knowledge (arrow 1). This query implicitly includes her individual context, e.g., her academic or social background, or, her *pragmatics*, for short.

The query is then analyzed by a component we term the *interpreter*. The interpreter is responsible for reformulating the query in such a way that the user's pragmatic background is resolved and syntactically made explicit. This is accomplished by two means: (a) utilizing query expansion or query reduction strategies – possibly in conjunction with a user-specific profile – in order to reduce the possibilities of (mis)interpreting the query [5], (b) calibrating the query to be in accordance with the retrieval system, i.e., replacing query terms with terms from a controlled vocabulary.

The result of this interpretation step is what we term a *canonical query*, i.e., a query that does not contain any pragmatic context any more (arrow 2). The interpreter thus bridges pragmatics and semantics.

The canonical query is then fed into the retrieval system, which chooses an appropriate algorithm to query the database – it answers the question "how is the corpus queried?". Next to the query itself, the query engine uses three additional "ingredients" to retrieve and rank suitable objects: *terms*, *corpus knowledge*, and *external knowledge*. *Terms* provide the means to syntactically represent both the query and all stored objects. *Corpus knowledge* is derived from the entirety of the object collection, i.e., the term-document matrix, including weights for ranking. *External knowledge* contains additional information that is independent from the object collection. It may include for instance lexical databases such as WordNet[1]. The query engine therefore bridges semantics and syntax within our architecture.

The query engine returns a list of ranked results (arrow 3) which is returned to the user. The ranking function analyzes the object representations to determine the rank of an object. The result consists of a list of object surrogates which the user reviews to determine the objects that fulfill her information need. For example, the surrogate of a text document could be the authors, the title, and the abstract of this document.

By actually retrieving the relevant objects the user feeds her *original query*, including all its implicit pragmatic context, back to the system (arrow 4). The idea of feeding back the original query is as follows. If an object is found by some of the terms within a query and the object is marked as relevant then all terms of the query are also added to the representation of this object. This way the individual pragmatic context of a user is transferred into the object representation. If this last step is performed a sufficiently large number of times by a sufficiently large number of users, new object semantics are being created. This is our reason for calling it *emergent semantics*: the pragmatic view of many users of an information system is gradually converted into object semantics. To illustrate the process of emergent semantics we ran experiments which show the change of the document representation over time (see Section 5.2).

Finally, an *annotation filter* with a specified quality measure is responsible for altering the object representations. If a user retrieves an object from the system, a simple annotation filter adds unconditionally all query terms to the representation of this object. More sophisticated filters better eliminate the noise introduced by the diverse backgrounds of many users. With *noise* we refer to terms the do not describe a document adequately, e.g., terms with a high or low document frequency. This elimination of unwanted noise is essentially accomplished in two ways: (1) by applying user specific filters ("user profiles") and possibly aggregating these into opinion networks ("collaborative filtering") or (2) by applying syntax-based algorithms to prune unwanted or unnecessary terms from set of query terms about to be added to the object representation. For instance, a filter could ignore all terms having a high document frequency or choose terms to include or exclude with the help of external knowledge. In section 4, we present further alternatives for the annotation filter. For instance, a term of an object representation may grow old and decrease in importance to reflect the change of language usage. Alternatively, the size of an object annotation may be fixed, e.g., if a new term is added another term is removed. We experimented with syntax-based filters and found them already to be quite effective to minimize changes to the object representations (see Section 5.1).

Since changing the object representation has a direct impact on the ranking of an object, it influences subsequent queries immediately.

[1] http://wordnet.princeton.edu/

3.2 Distinctions

Three aspects of the emergent semantics paradigm should be emphasized, since they differentiate the approach from the current state of the art:

Entirely new terms. EmSem allows entirely new terms to be introduced into the system. Even advanced approaches like LSI or query expansion can only work with terms already contained in the collection. In addition, we tackle the synonym problem, since a sufficient number of users will annotate the object with relevant synonyms.

Simple architecture. Keeping things simple, EmSem directly alters object representations. No new layers in query processing, no user specific state to be held, instant gratification to all users.

Living object representations. The entire process is based on the assumptions that most users "know best". If users change their mind about the supposed meaning of an object, the meaning of the respective object will change over time (e.g., historic events that are reevaluated after some time, cars that become vintage cars, or changes in the use of language).

4 Construction

In this section, we instantiate the previously motivated architecture within the area of information retrieval. IR aims at satisfying a user information need usually expressed in natural language [6]. To accomplish this task and to effectively rank documents according to a user's needs, an IR system models the relationship between a query and the relevant object, henceforth called *documents*, to this query. Approaches include the Boolean model, the vector space model [7], and the probabilistic model [8]. In the following, we introduce basic terminology and demonstrate how emergent semantics can be exploited within an information retrieval scenario.

4.1 Terminology

We consider a collection of (text) documents, which form the *corpus* $\mathcal{C}$ on which retrieval is performed. A common way to summarize content and meaning of these documents is to represent them using terms. Therefore, each document contained in the corpus is represented by a finite set of terms taken from a *universe of terms* $\mathcal{T}$. This annotation is called the *document representation.*

Definition 1 (Document Representation). *The* document representation *of a document d is a set of terms: $r(d) = \{t_1, \ldots, t_n\}, d \in \mathcal{C}, t_i \in \mathcal{T}$.*

In full-text IR, the prevalent form of IR today, these terms correspond to the words of a (text) document, usually preprocessed and filtered, i.e., by a stopword filter or a stemmer, eliminating the most frequent words of a language and converting the remaining words into their canonical form.

The user expresses her information need by issuing a query to the system in natural language, which is immediately preprocessed and filtered, similarly to the document filtering. As a query itself is interpreted as a document, we define it in the same way as the document representation:

Definition 2 (Query). *A user* query *is a set of terms:* $q = \{t_1, \ldots, t_n\}, t_i \in \mathcal{T}$.

As a result of query evaluation, the system returns a ranked list of document surrogates to the user. Based on this list, the user selects the documents which fulfill her information need. We define the answer to a query as follows:

Definition 3 (Answer). *Let q be a query. The answer of q is defined as $D_q = \{d_1, \ldots, d_n\}, d_i \in \mathcal{C}$. We denote with $D_r \subseteq D_q$ the set of documents classified as relevant by the user.*

Please note, that set D_r is specific for each user and each query of the system. Particularly, this implies that a specific D_r does not necessarily contain *all* relevant documents but only the ones *classified* as relevant by the user.

4.2 Document Retrieval

Before effective IR can be performed, the entire collection must be indexed. This phase is called the bootstrapping phase and usually needs to be performed only once for each collection. As a result of bootstrapping, the components *terms* and *corpus knowledge* of our architecture (see Figure 2) are initialized: while *terms* represent the syntactical document representations, stemmed and stripped of stopwords, the *corpus knowledge* in our case is the term-document matrix of our collection filled with TF/IDF weights.

Example 1. As a running example, we consider a query $q = \{RDBMS, SQL, language\}$ for the terms *RDBMS, SQL* and *language*, and a document corpus D_q of a thousand documents. Among others, it contains three documents with the following document representations:

- $r(d_1) = \{SQL, relational, \ldots\}$
- $r(d_2) = \{SQL, language, relational, \ldots\}$
- $r(d_3) = \{SQL, language, \ldots\}$

Although the above document representations are not completely shown, we assume in this example, that the term *language* does not occur in the document representation of d_1 and that the term *RDBMS* does not occur in any document representation of the corpus.

To initialize the system, we calculate the TF/IDF matrix for the document corpus. Figure 3 shows the term frequency, document frequency, and the TF/IDF weights only for the documents d_1, d_2 and d_3. We calculate the weight of a term t in document d according to the following formulae:

$$weight(t, d) = tf_{t,d} \cdot idf_t \tag{1}$$

$$tf_{t,d} = \frac{n_t}{|d|} \tag{2}$$

$$idf_t = \log \frac{|D|}{|\{d : t \in d\}|} \tag{3}$$

with n_t being the number of occurrences of term t in document d, $|d|$ being the total number of terms in d, $|D|$ being the total number of documents in the corpus and $|\{d : t \in d\}|$ being the number of documents the term t appears in. □

	$r(d_i)$			$\|d_i\|$
	SQL	language	relational	
d_1	12	0	50	8000
d_2	5	60	50	7000
d_3	12	14	50	10000
df	40	80	70	

	$r(d_i)$		
	SQL	language	relational
d_1	6.97	0.00	23.98
d_2	3.32	31.23	27.40
d_3	5.57	5.10	19.18

Fig. 3. Term frequency, document frequency (left), and the TF/IDF matrix (right)

Once the bootstrapping phase is finished, the system is ready to process queries. An incoming query is first processed by the interpreter component (see Section 3.1), e.g., the user specific context is eliminated by adding or removing query terms. Considering the query $q = \{RDBMS, SQL, language\}$ as an example, the interpreter may decide on the basis of external knowledge to substitute the term *RDBMS* by the terms *relational* and *database*, because *RDBMS* does not occur in the document corpus. For clarity and simplicity, we assume in the following that the interpreter did not change the query.

The query is then forwarded to the query engine. The *retrieval engine* is based on an inverted index of all terms $\mathcal{T}$ and the *ranking function* is based on the vector space model: both documents and queries are represented as term vectors carrying term weights from the term-document matrix above. The similarity between a query and each document is calculated as the angle between these vectors: the smaller the angle, the more relevant is the document for the respective query. We set each dimension of the query vector to "1" if the term appears in the query and to "0" otherwise. We can therefore reduce the classic formula

$$score_{t,d} = \frac{v_1 \cdot v_2}{||v_1||||v_2||} \tag{4}$$

to

$$score_{t,d} = \sum_{t \in q} tf_{t,d} \cdot idf_t \tag{5}$$

Example 2. In our example, the user submits the query $q = \{RDBMS, SQL, language\}$ to the information retrieval system. Considering only the documents d_1, d_2, and d_3, the query engine ranks the documents as follows (score in parenthesis): $d_2(34.55), d_3(10.67)$ and $d_1(6.97)$. □

4.3 Document Annotation and Annotation Filtering

After the query has been processed, the user evaluates the document surrogates returned by the IR system and selects the relevant documents D_r by actually retrieving them. We assume that the document surrogates contain enough information that a user can decide on the relevance of a document. For instance, the system returns the title and the abstract of a document. The actual retrieval of a document leads to the document annotation of the relevant documents being completed with the query: $\forall d \in D_r : r(d) = r(d) \cup r(q)$. The intuition behind this is that if a document is retrieved by *some* of the terms within the query and this document is additionally marked as relevant then *all* terms of the query are also relevant for the document representation.

However, adding all query terms to the document representation is an example for a very simple kind of an annotation filter. All terms are considered to be of equal quality and importance. In the following, we outline three more sophisticated strategies to modify the document representation: Our first approach utilizes internal knowledge to select the terms of a query being added to the document representation—internal knowledge which has been computed by analyzing the document collection. The last two approaches model the intuition that terms can also be removed from a description when they have become unimportant.

Approach 1: Leaving out the unimportant. A variation of the basic method described above is to ignore query terms with a high or very low document frequency in the collection. On the one hand, terms with high document frequency are assumed not to be discriminative enough to be useful in describing documents and thus, would not improve a document representation. On the other hand, terms with low document frequency describe a very specific facet of a document, if they are not just misspelled. Since we are interested in a document representation that is common sense, we remove these terms. In Figure 4 terms are mapped to their document frequencies. Only terms in the hatched area are added to the document representation. A disadvantage of this approach is that the probability of new terms being added to the representation decreases. However, as a positive side effect of omitting such terms, a smaller region of the TF/IDF matrix has to be recalculated which results in a more desirable computational behavior. Another way of limiting the number of added terms it to restrain valid terms to the ones of a controlled vocabulary, or a domain ontology. This implies the disadvantage that terms which are completely new and not part of the vocabulary will never be added to the representation. On the other hand, we can guarantee that all added words are potentially useful in representing the documents, as they have been introduced into the controlled vocabulary by domain experts.

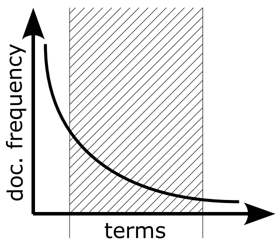


Fig. 4. Ignoring unimportant terms

Approach 2: Including decay in the ranking function. Going back to our metaphor from the beginning: as pathways emerge by usage, pathways also disappear, if they are no longer used. Lawn recovers, until the pathway completely disappears. The equivalent in our IR system is to mark every term added to a document representation with a timestamp and store it separate from the original document representation. Each time a term is added to the document, its timestamp is set to the current time. The system then calculates rankings not only on the basis of TF/IDF values but also incorporates the age of the term. When computing the score, the weight values are adjusted by multiplying it by an exponentially decreasing function according to the timestamp stored: $w_i = w_i \times exp(1/now - timestamp)$, reducing its weight with increasing age. Figure 5 shows the decaying weight of terms (hatched area) over time. Terms older than a specific age are not considered anymore (time cutoff). Introducing decay only for newly added terms allows the system to always find documents by terms of their original description and not become less effective than standard TF/IDF retrieval.

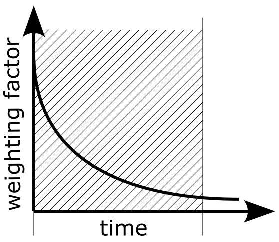


Fig. 5. Weighting factor for approach 2

Approach 3: Terms queueing up. Another way of implementing the concept of decreasing importance of terms over time is to limit the number of additionally added terms stored with a document to a fixed number of n terms and keep these terms in a FIFO queue (first-in first-out). New terms are added without restriction by just putting them in the queue, automatically removing the oldest term. However, they might not be removed forever: if they are reused and newly added, they become part of the document representation again. Terms added more than once are also kept more than once in the queue. During weight computation, recently added terms are considered more important and therefore get a slightly higher weight (see Figure 6), than terms added some time ago, according to their position in the queue. As these terms are stored in addition to the standard term-document matrix, documents can always be found by terms of their original representation and thus the system can not become less effective than standard TF/IDF retrieval. The difference to the previous technique lies in not including all terms ever related to the document with its description and in not needing to adjust the document retrieval procedure.

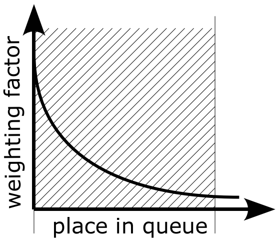


Fig. 6. Weighting factor for approach 3

4.4 Computational Issues

Since the TF/IDF-matrix depends on the document representation, some parts of the matrix have to be recalculated. The following list enumerates the possible changes to the document representation and recalculated elements of the TF/IDF matrix. Be $t \in r(q)$ and $d \in D_r$:

Existing term $(t \in r(d))$ The weight of the term t increases for the document d, all other values for all other documents remain unchanged. The timestamp of term t also needs to be adjusted and in case of alternative (3), the new term is introduced into the queue, the last term is removed from the queue and its weight is also changed.

New term/document combination $(t \notin r(d) \wedge t \in \mathcal{T}_\mathcal{C})$ Weights for all documents are recalculated, because the document frequency changes for t. In case of alternative (3), the new term is introduced into the queue, the last term is removed from the queue and its weight is also changed.

New term to corpus $(t \notin r(d) \wedge t \notin \mathcal{T}_\mathcal{C})$ Since the document frequency of term t is known in advance, it is sufficient to calculate the weight of t with regard to document d. In case of alternative (3), the new term is introduced into the queue, the last term is removed from the queue and its weight is also changed.

Example 3. To illustrate document expansion, please consider in the following how the ranking changes when terms are added to a document representation. The user evaluates the document surrogates of d_1, d_2, and d_3 and decides, for example, that d_1 and d_2 are relevant but d_3 is not. As the user retrieves the first two documents the system adds the original and unmodified query terms $\{RDBMS, SQL, language\}$ to the document representation of d_1 and d_2. Please note, that annotation filters provide means to eliminate terms being not related to a document.

Adding the query terms to the document representation has an impact on the TF/IDF matrix (see Figure 7 for the updated matrix). The matrix changes as follows.

A new column is added, because the term *RDBMS* does not occur in the document corpus at all. All values for d_1 and d_2 are recalculated according to the TF/IDF algorithm. All other cells of this column are simply set to the value zero. Adding the term *SQL* to the document representation of d_1 and d_2 changes the number of terms of only these documents and the rows of both documents have to be recalculated. The term *language* does not occur in d_1. Thus, the system has to recalculate the complete column of this term. As a result, the TF/IDF weights of the term language change for all documents. On one hand, the weight of the term *language* increases for document d_1, because the term

	$r(d_i)$				$\lvert d_i \rvert$
	SQL	language	relational	RDBMS	
d_1	13	1	50	1	8003
d_2	6	61	50	1	7003
d_3	12	14	50	0	10000
df	40	80	70	2	

	$r(d_i)$			
	SQL	language	relational	RDBMS
d_1	7.54	0.45	23.97	1.12
d_2	3.98	31.58	27.39	1.28
d_3	5.57	5.08	19.18	0.00

Fig. 7. Recalculated TF/IDF matrix after adding the query terms

was added to the document representation and is now related to this document. On the other hand, the corresponding values for the other documents decrease, because now there is one more document containing the term language. Thus, this term discriminates documents to a lesser extend.

After changing the TF/IDF matrix the weights of the queries according to the query q change to $d_2(36.84), d_3(10.65)$ and $d_1(9.12)$. If another user would run the same query and would also select d_1 and d_2 as relevant, then the ranking would change to $d_2(39.29), d_1(11.27)$ and $d_3(10.65)$ – the last two documents changed places. As time passes, the added term *RDBMS* would lose importance, lowering its weight and eventually resulting in d_1 and d_3 changing places in the ranking again. This is what we referred to as *Living document representations* in Section 3.2. □

5 Assessment

In this section we describe two experiments, which we ran to evaluate different aspects of our emergent semantics approach. Both experiments deal with the pragmatics part (users thoughts) in the retrieval process although highlighting different facets. In our first experiment, we compare the standard TF/IDF approach with our emergent semantics approach within a classical information retrieval scenario using the standard CACM corpus. We use implicit user feedback by annotating the documents rated relevant by the user with the search terms used to find the documents. We show that walking the way from semantics via pragmatics to syntax using this approach is beneficial. Here, terms used to enhance the document descriptions are not completely arbitrary, but come from the set of search terms used.

In our second experiment, we alleviate this and tackle the question what happens if the set of terms that can be used to enhance document descriptions is completely free. We study the question if in such a scenario, that is similar to a collaborative tagging scenario, user descriptions of documents can possibly agree. We also show the beneficial application of a special annotation filter. So both, using implicit (feeding back search terms) as well as explicit (user generated tags) user feedback, can be used in our setting to walk the line from semantics over pragmatics to syntax, enriching document descriptions and enhancing the retrieval process.

5.1 CACM Collection

For our first experiment, we chose the standard “Communications of the ACM” information retrieval collection (CACM) to evaluate our approach. We decided to use this collection, because it features an overlap between query terms as well as an overlap between corresponding result sets (see “Evaluation remarks” below).

Experimental setup. We used the document title and, if available, the document abstract, for indexing and retrieval. They were tokenized and indexed by the Apache Lucene inverted index [9] using the Vector Space Model with TF/IDF weighting. After initializing the system, the following steps were repeated for each query in the corpus:

Query. We chose a query from the CACM corpus and presented it to the system as a disjunctive query ("or" semantics).
Retrieval and ranking. The IR system retrieves documents from the index and ranks them using the vector space model with TF/IDF weighting of terms.
Feedback. According to the gold standard in the data set, we calculate precision and recall measures. We then tokenize the query and attach it to all relevant documents as content (see our architecture description in Section 3).

Feedback, the last step during the evaluation, is optimal in the sense that all relevant documents are selected from the set of retrieved documents. An alternative would be to select only a few documents from the result list for attaching the query to it. If the documents to be tagged were selected randomly from the result list, this approach would yield similar evaluation results. It the documents to be tagged were selected according to their position in the result list, this approach would result in a positive feedback cycle where results that are preferred once will always be preferred in the future and therefore achieve a high ranking position that is rather stable. We have not evaluated this further, but we believe that the only solution for breaking this positive feedback cycle is to include decay in the ranking function as discussed in Section 4.3 and evaluated in the experiments of Section 5.2.

Evaluation results. As performance indicators for our emergent semantics approach, we determined precision-recall measures along the eleven standard recall levels. See [6] for a discussion of individual IR performance indicators.

The results in Figure 8 demonstrate a major improvement in retrieval performance after feeding back all query terms unconditionally. These results mean that retrieval performance increases with each query if multiple users pose the same query to the system. In addition, we experimented with an *annotation filter* (see Figure 2), which only allowed feeding back terms with a document frequency smaller than 30 (1% of the documents). This prevents feedback of frequent terms. However, we were surprised to see that the results were only slightly worse than the results before; this implies that adding only very few (relevant) terms to the document description can already considerably improve future retrievals.

These results might have been anticipated: Few people would deny that feeding back the exact queries into the system will lead to improved retrieval performance for the same queries. Nonetheless it demonstrates the viability of the emergent semantics paradigm. To the best of our knowledge, there exists no data set specifically tailored to evaluate the performance of an emergent semantics system. If such a data set existed, we would gladly evaluate our approach against it.

Evaluation remarks. We chose the CACM collection after evaluating the (also standard) Medline collection. It turned out, however, that there is no overlap

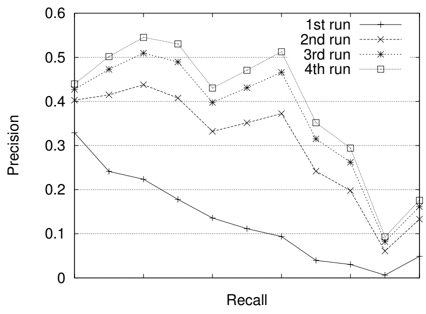


Fig. 8. Augmenting document representation with all query terms

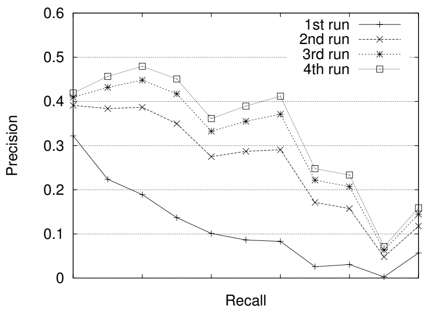


Fig. 9. Augmenting document representation only with infrequent query terms

in retrieval results between the queries defined in the Medline collection and therefore the emergent semantics approach will not work. Ideally, we look for a collection of highly correlated queries with a highly overlapping result set. In such a scenario, emergent semantics unfolds its full potential.

It should be noted that we do not compare our results to today's best precision/recall values. It is our ambition to introduce the new paradigm of actually modifying document representations instead of using (possibly user-specific) query refinement strategies. Therefore, the baseline of our comparison is standard document retrieval using the vector space model and TF/IDF weighting. We expect emergent semantics techniques to have a similar impact on more sophisticated IR algorithms.

5.2 Live Experiment

In our second experiment we tested our assumption that many users "know best" and that they actually agree on the semantics of a given document instead of having completely different opinions about the document content. Agreement means improved document retrieval, disagreement means less effective document retrieval as well as a lot of noise induced into the document representation. We furthermore implemented and evaluated an annotation filter to further improve the quality of the document annotations.

Experimental setup. For this experiment we implemented a web application and presented users with different photographic pictures and asked them to annotate the pictures with terms "they would use with a search engine for retrieving the photo". There were 215 photos available showing different motives, some of which were very simple (such as a pair of sunglasses) and some of which were difficult to describe. The first ten pictures we presented to the user came from a pool of twenty simple pictures to accustom the user to the process. The remaining photos were presented in random order, thereby eliminating any possibility that the particular order in which we showed the pictures might influence the result.

The experiment was conducted in two phases. In the first phase, the user had to annotate each picture with up to five terms which she would use in retrieving the photo with a search engine. In the second phase, we asked the user if she would consider a given annotation useful. The purpose of the second phase was to verify that the annotations given in phase one are actually valid in the user's opinion. We showed for each picture the five most frequent terms from phase one and for each term asked the user if he would use it in a search for the given picture. The user could choose between *Yes*, *No*, *Redundant* and not having any opinion on the particular term. The *Redundant* option was for cases such as "little girl" and "girl" in which one would not use the second phrase in a search not because it has the wrong semantics but because its meaning is already contained in the first one. Consequently, we counted the *Redundant* option as *Yes*.

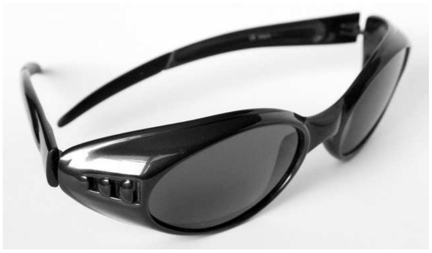
Finally, we used the collected data to simulate a simple annotation filter which restricted the number of terms per picture to ten and implemented a decay function to select the terms to be substituted for new terms (cf. Approach 2 in Section 4.3). We assigned a weight to each term which changes as follows: Before adding a new term to the document representation we decrease the weights of all terms by one. Afterwards we add the term to the document representation and changed it as follows: (1) If the picture is already annotated with the term, we increase a weighting factor for this term by a constant value. (2a) If the term is not contained in the document representation and the maximum number of terms is not then we just add the term to the document representation. (2b) If we would exceed the limit of terms by adding the term we remove the term with the lowest weight.

User participation. The participation for phase one exceeded our estimations. We had 140 users giving over 21,000 annotations. Half of our users annotated more than two dozen pictures with 20 users even annotating all 215 available pictures. This resulted in every picture getting between 58 and 261 annotations with an average of 99. Judging from the feedback we got, most of our users were university staff or students.

Phase two had similar annotation counts, with over 20,000 annotations from 53 users, resulting in 65–170 annotations per picture and an average of 93.

Experimental results. To give you some general impression of the results, we present the data we gathered on one simple and one difficult picture in detail. For this paper, we translated all the originally German terms into English.

Figures 10 and 11 show our results from phase one for the two pictures. In our experiments we used the user's input verbatim. This resulted in counting the same word contained in two phrases as two different terms. Therefore, the tail of the terms in Figure 11 contains the term "coke" and various combinations of this term with other terms, e.g., "bottle of coke", "snacks and coke" or "bottle of coke with shoes". The tables show the top five terms for each picture and the percental share they got out of all annotations the picture received (second column). Since we asked the participants to choose terms they would use to search for a picture, we interpret these terms as if they were fed back unconditionally (unfiltered) to



Term	Unfiltered	Filtered
sunglasses	52.2%	70.0%
glasses	5.9%	8.0%
black sunglasses	4.4%	6.0%
cool	4.4%	2.0%
modern	2.9%	2.0%
sum	69.8%	88.0%

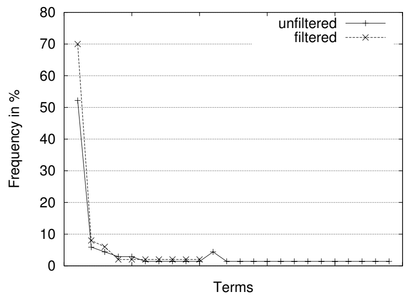
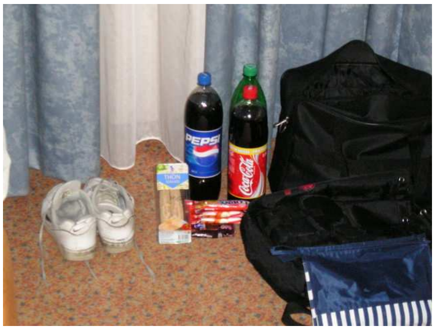


Fig. 10. Term frequencies for the simple sunglasses picture



Term	Unfiltered	Filtered
shoes	6.6%	22.0%
bag	5.5%	11.0%
coke	5.5%	7.0%
provisions	4.4%	./.
backpack	4.4%	3.0%
sum	26.4%	43.0

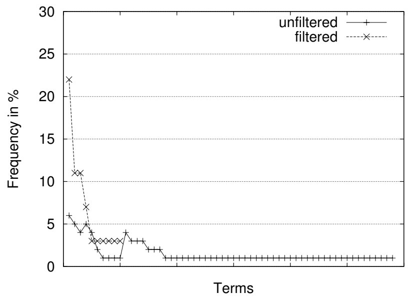


Fig. 11. Term frequencies for the difficult picture

the document representation of the pictures. The third column shows the result of using a simple annotation filter as described above.

In the diagram of Figure 10, there is a distinctive peak containing the terms that most users agree upon and a long tail containing background noise, e.g., terms that describe a specific view on the image. In this case our method worked very well, as the opinion of the majority of users about the semantics of the picture is clearly separated from the background noise. Using the described annotation filter results even in a higher peak for the top term. In Figure 11, the peak turns out to be much smaller, indicating that the consensus is much less focused compared to the simple picture. Although the top three terms are still contained in the document representation we see that some frequent terms are removed from the document representation.

Nonetheless, looking at the whole collection of pictures, we see that almost all corresponding diagrams show a distinctive peak such as the one in Figure 10.

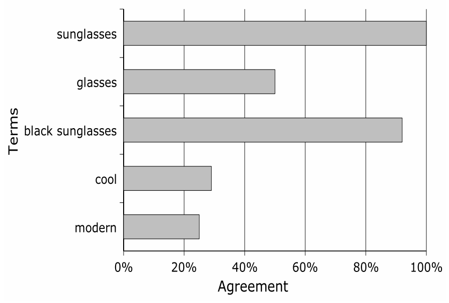


Fig. 12. User agreement with the top five terms of the simple sunglasses picture

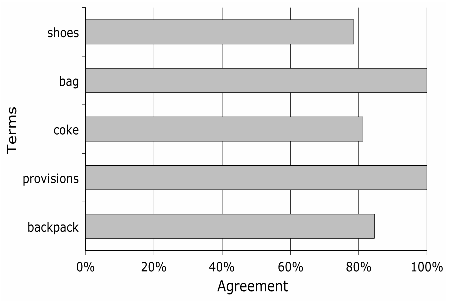


Fig. 13. User agreement with the top five terms of the difficult picture

The top term usually makes up for 15–50 percent of all annotations a picture received. In the majority of pictures, the first ten terms represent more than half of all the annotations of that picture and often even the first five terms suffice for that.

The results of phase two are shown in the Figures 12 and 13. For each term, we show the percentage of *Yes* votes it received. Figure 13 shows that although in phase one the top five terms for the difficult picture did not represent a very high percentage of the given annotations, most users agreed that all five terms would be useful in a search for the picture. The same holds true for the majority of all gathered terms, which proves that the user's opinion we extracted from phase one is generally seen as being a good semantical representation of the various pictures.

We observed another interesting effect in Figure 12. In phase one, some users tended to describe not the picture itself but rather the associations it evoked, e.g., "modern" for the sunglasses, "hot" for a picture of flames or "anchor" for a picture showing only a few chain links. Since terms of this kind usually have a very broad meaning and are therefore not useful when searching for a particular picture, users often rejected these terms in phase two. We later asked users about their reasons for providing these broad terms, and they stated that (although in no way being required to do so) they felt a need to fill out as many of the five text boxes as possible. However, after providing the obvious, i.e. "sunglasses", they resorted to broader terms to describe the picture "better".

In our opinion the collected data shows quite clearly that a large group of users can successfully agree on the semantics of a given picture. We conclude from this experiment that a simple majority vote is sufficient to extract those semantics and that one usually needs only a handful of terms to describe the picture satisfactorily for most users. Our evaluation of the annotation filter shows that smart filter algorithms are capable of even further improving the value of the best annotations.

Raw data availability. Since we believe our data might be interesting for others to study or check our assumptions, we provide a depersonalized database dump at http://emsem.informatik.hu-berlin.de. Please be advised that the experiment (and, correspondingly, all collected data) is in German.

6 Neighborhood

With emergent semantics we claim to integrate many users' opinions on objects in order to find better document representations. Our work relies on previous work in the area of information retrieval and bears some similarity with previous work in the areas of collaborative tagging, and syntax-based emergent semantics approaches.

Information retrieval. Our emergent semantics approach relies on an underlying IR infrastructure. We utilize the widely accepted vector space model [10]. In this model, each document and each query is represented by a vector and the similarity between document and query is represented by some similarity measure between the two vectors. A common similarity measure is the cosine angle between the two vectors, common approaches for calculating the weights in each vector include TF/IDF weights [1] and Latent Semantic Indexing (LSI) weights [2]. In our system we use TF/IDF weights—this approach has been proven to be very effective for most information retrieval tasks and the corresponding weights are easily calculated. In contrast, recalculation of LSI weights which is based on complex matrix operations, is quite expensive in the event of changes in the document collection.

We take advantage of these information retrieval algorithms, but advance the purely syntactical analysis of the document content by integrating users' opinions into the system, thus allowing for more representative terms or even additional terms that have not yet existed within the collection before.

Document expansion. Document expansion is the most closely related technique in information retrieval. It is a means to improve information retrieval effectiveness by incorporating additional terms into documents or their representations. Ide and Salton [11] experimented with manipulating document vectors according to relevance feedback in order to improve ranking effectiveness. However, they did not perform any real document expansion.

Actual document expansion, i.e., adding terms to the actual documents, was first performed by Singhal and Pereira [12] in the context of speech retrieval. The reasoning behind this is that the rather inaccurate speech transcripts are expanded with related terms from an external, though semantically related, corpus. Note that this work is specifically intended to improve the reduced quality of transcribed documents.

Billerbeck and Zobel [13] analytically evaluate two document expansion techniques against traditional query expansion techniques and find that query expansion yields better rankings. Document expansion, however, was performed entirely syntactical, i.e., by analyzing the existing corpus and adding *presumably* related terms from the corpus to each document. No user feedback was incorporated into document representations and therefore no new terms were added to the term universe.

Our work overcomes the restriction that a fixed vocabulary is used in document descriptions. It adds a social dimension to previous work in document expansion in a sense that it incorporates actual feedback and perspectives of users. With every new user comes new vocabulary. That way our approach extends the term universe that is used in document descriptions, and also changes over time into a term universe that is best suited in all the users opinion. Furthermore, our work is backed by a coherent architecture that includes sophisticated features like an interpreter and an annotation filter that provide some more potential improvements.

Collaborative tagging. While the user context plays an important role both in collaborative filtering approaches [14,15] as well as in contextual service adaptation [5], they both introduce a separate layer into the object retrieval process. Emergent semantics as a paradigm leaves the underlying retrieval system untouched and achieves its goal through direct modification of document representations. This is also contrary to the standard approach in using relevance feedback where the query is expanded [6].

Annotation or tagging systems ([16], [17]) bear some similarity to the emergent semantics paradigm, however, they usually require the user to explicitly determine suitable tags or annotations for the object. During system usage, object descriptions are not altered. Emergent semantics takes advantage of the user query (implicit information) to accomplish its goal. Similarly, Grosky et al. emerge the semantics of multimedia objects, i.e., web pages, by including objects along a user's browsing path into the context of the respective object [18].

Two common problems with using annotations are that 1) there are always too less annotations available and 2) one can never be sure that annotations correctly describe the content of an artefact, e.g., an image. A very innovative approach for tagging systems was proposed by Luis van Ahn [19]. He reformulates the task of tagging images into a game[2], so users voluntarily spend ours of their time "playing" the annotation game and having fun, while in the background image annotations are collected and evaluated. That way many annotations can be collected during a short period of time and users are given an incentive to provide correct annotations as they otherwise would "loose" the game. In our second experiment we showed that users can reach agreement on image descriptions, thus providing suitable annotations for retrieval. The approach of [19] shows how to create such image descriptions with high agreement among users.

Tagging is only one way of emerging classifications and structure. In [20] Voss et al. introduce collaborative thesaurus creation present in current Wikimedia projects and compare it to the strictly hierarchical Dewey Decimal Classification (DDC) and the strictly tagging-based approach of del.icio.us [3].

[2] http://www.espgame.org

[3] http://del.icio.us

Emergent semantics for autonomous agents. Considering a network of individual, self-interested agents that aim to communicate with each other, the problem of these agents interacting with each other and "understanding" each other on a semantic level is another novel view on emergent semantics. In [21] the authors regard the sum of all schema mappings between individual agents as the *semantics* of a distributed information system. These semantics are evaluated and improved by processing queries along cycles within the network and assessing the quality of their results. While being a very interesting approach for evolving schema mappings between autonomous agents, it strongly focuses on automatic schema evolution instead of on human-driven semantics generation, as in our work.

7 Roof and Windows

In this paper, we presented emergent semantics, a new paradigm for integrating many users' opinions directly into an object retrieval system. On the linguistic side, this paradigm represents the transition from many user's individual pragmatic views to a unifying semantic representation of the object. We proposed an architecture capturing all aspects of emergent semantics and detailed the expected functionality of its components.

An implementation of the emergent semantics approach within the area of IR, including promising results within a standardized evaluation on the CACM corpus, demonstrates the feasibility of our approach. We further show that user-based emergent semantics actually works and that many users agree on only a few terms as object descriptions. An evaluation of one of our annotation filter proposals demonstrates that annotation filters further improve object representations in this context.

Future work includes exploration of other applications of emergent semantics, especially its potential ability to reduce the size of object representations, while still allowing for good retrieval results. We will also further evaluate annotation filters and user-based interpreters for further improving object retrieval with less resources. Another appealing research direction is the application of the emergent semantics paradigm within a distributed environment.

Acknowledgment. This research was supported by the German Research Society (DFG grants no. NA 432 and GRK 316), and by the German Ministry of Research (InterVal Berlin Research Center for the Internet Economy).

References

1. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill Book Company, New York (1984)
2. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by Latent Semantic Analysis. Journal of the American Society of Information Science 41(6), 391–407 (1990)
3. Morris, C.W.: Foundations of the Theory of Signs. Chicago University Press, Chicago (1938)

4. Ogden, C., Richards, I.A.: The meaning of meaning: A study of the influence of language upon thought and of the science of symbolism, 8th edn. Harcourt Brace Jovanovich, New York (1923) (reprint)
5. Jacob, C., Radusch, I., Steglich, S.: Enhancing legacy services through context-enriched sensor data. In: Proceedings of the International Conference on Internet Computing (2006)
6. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press, New York (1999)
7. Salton, G., Lesk, M.E.: Computer evaluation of indexing and text processing. Journal of the ACM 15(1), 8–36 (1968)
8. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. Journal of the American Society for Information Science 27(3), 129–146 (1976)
9. The Apache Software Foundation (2006), http://lucene.apache.org
10. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18(11), 613–620 (1975)
11. Ide, E., Salton, G.: Interactive search strategies and dynamic file organization in information retrieval. In: The SMART Retrieval System - Experiments in Automatic Document Processing, pp. 373–393. Prentice-Hall, Englewood Cliffs (1971)
12. Singhal, A., Pereira, F.: Document expansion for speech retrieval. In: Gey, F., Hearst, M., Tong, R. (eds.) Proceedings of the SIGIR conference, Berkeley, CA, pp. 34–41. ACM Press, New York (1999)
13. Billerbeck, B., Zobel, J.: Document expansion versus query expansion for ad-hoc retrieval. In: Proceedings of the 10th Australasian Document Computing Symposium, Melbourne, Australia (2005)
14. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating "word of mouth". In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 210–217. ACM Press, New York (1995)
15. Oard, D.W.: The state of the art in text filtering. User Modeling and User-Adapted Interaction 7(3), 141–178 (1997)
16. Heflin, J., Hendler, J., Luke, S.: SHOE: A knowledge representation language for internet applications. Technical Report CS-TR-4078 (UMIACS TR-99-71), University of Maryland (1999)
17. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems. J. Inf. Sci. 32(2), 198–208 (2006)
18. Grosky, W.I., Sreenath, D.V., Fotouhi, F.: Emergent semantics and the multimedia semantic web. SIGMOD Rec. 31(4), 54–58 (2002)
19. von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: CHI 2004: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 319–326. ACM Press, New York (2004)
20. Voss, J.: Collaborative thesaurus tagging the wikipedia way. In: Proceedings of the Collaborative Web Tagging Workshop. Colocation with the 15th international WWW conference 2006, Edinburgh (2006)
21. Cudré-Mauroux, P., Aberer, K., Abdelmoty, A.I., Catarci, T., Damiani, E., Illarramendi, A., Jarrar, M., Meersman, R., Neuhold, E.J., Parent, C., Sattler, K.U., Scannapieco, M., Spaccapietra, S., Spyns, P., Tré, G.D.: Viewpoints on emergent semantics. In: Spaccapietra, S., Aberer, K., Cudré-Mauroux, P. (eds.) Journal on Data Semantics VI. LNCS, vol. 4090, pp. 1–27. Springer, Heidelberg (2006)

Author Index