



SIMDAT

Data Grids for Process and Product Development using Numerical Simulation
and Knowledge Discovery

Project no.: 511438

Grid-based Systems for solving complex problems – IST Call 2

Integrated project



***D.6.1.4 + D6.1.5: Finalized implementation of the Ontology
infrastructure prototype (version1) / Ontology services:
evaluation of the results of the first project phase***

Start date of project: 1 September 2004

Duration: 48 months

Due date of deliverable: 2006-04-01

Actual submission date: 2006-05-09

Lead contractor for this deliverable: ontoprise

Revision: 1.1

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination level		
PU	Public	X
PP	Restricted to other programme participant (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Copyright

Copyright © ontoprise and other members of the SIMDAT consortium, www.simdat.org, 2006

Table of Contents

1	Executive Summary	5
2	References	6
3	Introduction	9
3.1	Purpose and Scope of Document	9
3.2	Definitions, Acronyms and Abbreviations	10
4	Technology improvements description	11
4.1	PM12 Environment	11
4.1.1	Service Discovery	11
4.1.2	Semantic Information Integration	12
4.1.2.1	Architecture and Implementation of Information Integration Prototype	13
4.1.2.2	Implementation Aspects of Adaptors and Bus	14
4.2	Lessons learnt	15
4.2.1	Domain Models	15
4.2.2	Integration framework for the interoperability of distributed sources	17
4.2.2.1	Performance	17
4.2.2.2	(Data-) Grid Compliance	18
4.2.3	Usability, End-User Tools	18
4.3	PM18 Improvements	19
4.3.1	Schema-Import from Remote Ontology Servers	20
4.3.2	Semi-structured Resources	21
4.3.3	Concept for Schema-Changes	21
4.3.4	Detecting the kind of change	23
4.3.5	Visualization of Changes	24
5	State-of-the-art and the contribution of SIMDAT	25
5.1	Data Integration	25
5.1.1	Integration Strategies	26
5.1.2	Heterogeneity	27
5.1.3	Comparison of integration approaches	27
5.1.3.1	Mediators	28
5.1.3.2	Federated Databases	28
5.1.3.3	Data Warehouse	29
5.1.3.4	Semantic Integration	29
5.1.3.5	Other approaches	30
5.1.3.6	Comparison	31
5.1.4	The Contribution of SIMDAT	32
5.2	Service Discovery	32
5.2.1	Semantic Description of Web Services	33
5.2.1.1	OWL-S	34
5.2.1.2	WSMO	35
5.2.2	Applications of Service Discovery	36
5.2.3	The contribution of SIMDAT	36
6	Matrix structure of document	37
6.1	Work provided to the application activities	37
6.2	Feedback received from the application activities	37

7	Specific and modular requirements documentation.....	38
7.1	Introductory Remarks	38
7.1.1	The Supplier-Producer Role Model.....	38
7.1.2	Query Languages	38
7.2	Requirements	39
7.2.1	OGSA-DAI based Ontology-Service	39
7.2.2	SPARQL-Interface	40
7.2.3	Capabilities of Representation (for Product Data).....	40
7.2.4	Additional Connectors I.....	41
7.2.5	Additional Connectors II	41
7.2.6	Additional Connectors III.....	42
7.2.7	Knowledge-Services Support	43
7.2.8	Query Semantics.....	43
7.2.9	Semantic Integration Security.....	44
7.2.10	Management of References for Bulk Data	44
7.2.11	Management of Schema Changes.....	45
7.2.12	Thesaurus-Support for Product Discovery	46
7.2.13	Simple Online Ontology-Maintenance	46
7.2.14	Additional Connectors IV.....	47
7.3	Outlook after PM30	48
8	Implementation and test plans for PM 24 and PM 30.....	49
8.1	Major Milestones	49
8.2	Implementation Plan until PM24.....	49
8.3	Implementation after PM24.....	49
8.4	Test Plan	50
9	Conclusions	51
10	Table of Figures	52

1 Executive Summary

This document describes the status and the achievements of Workpackage 6 (“Ontologies”) at Project Month 18, including the state of the art, “lesson’s learnt”, feedback from the application areas, requirements for the coming project phase and implementation plans.

The infrastructure that was established within the first 18 months covers two basic aspects: the discovery of services and the integration of heterogeneous sources. With respect to service discovery this includes:

- a set of ontologies for the semantic description of services with a focus on the bioinformatic domain;
- a semantic service broker encapsulating the OntoBroker® inference engine;
- an annotation standard in XML-syntax to represent service metadata;
- a service for the registration of search- and analysis-services with the semantic broker based on the annotation standard;
- a graphical tool for the creation of annotation data.

Regarding semantic integration the infrastructure

- provides a basic form of a “semantic layer” through which sources are accessed;
- allows to create ontologies as representative models problem solving environments in the application areas rely on;
- supports the import of existing relational database schemas in terms of a syntactical integration;
- supports the creation and management of mappings between the different models based on declarative rules;
- supports the evaluation of ontologies and rules by a reasoning engine, allowing to access the data of the integrated sources via several semantic layers;
- provides a service that takes path expressions of a generic query language (referring to a stored ontology), performs the syntactic translation and communicates with the ontology server.

The first phase of the project revealed some performance and usability issues with the current infrastructure, which are partially reflected by the requirements for the coming phase. The latter also include a transfer of semantic technologies into additional applications for the areas that have not yet been covered.

The requirements and the implementation plans show that the focus for the coming phase is on the generalization of the integration framework by the usage of grid middleware as the base for data integration. There is no comparable infrastructure available which consequently exploits the advantages of semantic technologies while on the other hand building on data grid technology.

The position of SIMDAT in the grid-sector and related areas is much influenced by the application-oriented approach of the project. Core problems of service oriented architectures based on a large number of resources with a high degree of autonomy can be targeted with the technology extended, developed and applied in SIMDAT based on ontologies. That’s why SIMDAT will have a considerable impact on future solutions and on the grid sector as such.

2 References

- [Aman2001] AMANN, B.; FUNDULAKI, I.; BEERI, C.; VERCOUSTRE, A.; SCHOLL, M.: MappingXML Fragments to Community Web Ontologies. ., Proceedings of the FourthInternational Workshop on the Web and Databases (WebDB'01) 2001
- [Aman2002] AMANN, B.; BEERI, C.; FUNDULAKI, I.; SCHOLL, M.: Querying XML sources using an Ontology-based mediator. In: Proceedings of CoopIS/DOA/ODBASE 2002
- [Ange2004] ANGELE, J.; STAAB, S; MOENCH, E.; OPPERMAN, H.; WENKE, D. ET AL.: Project Halo: Towards a Digital Aristotle. *AI magazine*, 25 (2004) 4, 29-49
- [Bane1987] BANERJEE, J.; CHOU, H.-T.; KIM, H.J.; KORTH, H.F.: Semantics andimplementation of schema evolution in object-oriented databases. *ACM SIGMOD Record*, Vol. 16 (1987) No. 3, pp 311-322
- [Bati1986] BATINI, C.; LENZERINI, M; NAVATE, S.B.: A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, Vol. 18 (1986) No. 4
- [Bene2000] BENEVENTANO, D.; BERGAMASCHI, S.; CASTANO, S.; CORNI, A.; GUIDETTI, R.; MALVEZZI, G.; MELCHIORI, M.; VINCINI, M: Information Integration: The MOMIS project demonstration. ., Proceedings of the 26th International Conference on Very Large Data Bases (VLDB'00) 2000
- [Calv1998] CALVANESE, D.; DE GIACOMO, G.; LENZERINI, M.; NARDI, D.; ROSATI, R.: Description Logic Framework for Information Integration. In: *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR'98)* Trento, IT, 1998, pp 2-13
- [Carey1995] CAREY, M.J.; HAAS, L.M.; SCHWARZ, P.M.; ARYA, M.; CODY, W.F.; FAGIN, R.; FLICKNER, M.; LUNIEWSKI, A.; NIBLACK, W.; PETKOVIC, D.; THOMAS, J.; WILLIAMS, J.H.; WIMMERS, E.L.: Towards heterogeneous multimedia information systems: The Garlic Approach.In: *Proceedings of the 5th International Workshop on Research Issues in DataEngineering – Distributed Object Management (RIDE-DOM'95)*, , 1995, pp 124-131
- [Comp2002] COMPATANGELO, E.; MEISEL, H.: EER-ConcepTool: a 'reasonable' environment for schema and ontology sharing. In: *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02)*, 2002, pp 527-534
- [Conr1997] CONRAD, S.: Föderierte Datenbanksysteme – Konzepte der Datenintegration.: Springer, 1997
- [Conr2002] CONRAD, S.: Integration heterogener Datenbestände. Jahrbuch der Heinrich-Heine-Universität Düsseldorf., Düsseldorf 2002
- [D.11.1.2] MAYER, S.; WIRCH, W.; OTT, B.; CAZEAUX, J. P.; REICHENEDER, J.; TZANNETAKIS, N.: D.11.1.2 Documentation of Software Design and Initial Implementation for SIMDAT Automotive Demonstrators (Integrates D.9.1.2, D.10.1.1, D.11.1.3). SIMDAT deliverable, Audi AG 2005
- [D.18.1.1] AUBERT, G.: D.18.1.1 Consolidated Meteorology Requirements. ., SIMDAT deliverable 2005
- [D14.1.2] ZIMMERMANN, F.: WP14.1.2 Production of first prototype. SIMDAT Deliverable, NEC Europe Ltd. 2005
- [D20.1.3] AUBERT, G.: D20.1.3 Alpha version of the demonstrator focussing on thetransfer of real-time data under strict security. ., SIMDAT deliverable 2005

- [D3.1.2] KRÜGER, M.; HOPPE, H.-C.: D.3.1.2 Distributed Data Repository Access Infrastructure (First Version). ., SIMDAT deliverable 2005
- [D6.1.1] WEITEN, M.: D.6.1.1: Ontology Technology Requirement and Implementation Plan. SIMDAT deliverable. ontoprise GmbH 2005
- [D6.1.2] WEITEN, M.; WEINDEL, M.: D6.1.2 Description of service ontology and first implementation. SIMDAT deliverable, 2005
- [D6.1.3] WEITEN, M.; RUDLOF, S.; WIRCH, W.; DREHER, AXEL: D.6.1.3 Requirements analysis for additional ontology services. SIMDAT deliverable, 2005
- [Erdm2001] ERDMANN, M.: Ontologien zur konzeptuellen Modellierung der Semantik von XML. Dissertation. Universität Karlsruhe, 2001
- [Fens2002] FENSEL, D.; BUSSLER, C: The Web Service Modeling Framework WSM. *Electronic Commerce: Research and Applications*, 1 (2002), 113-137
- [Gall2004] GALLOPOULOS, S.; HOUSTIS, E.; RICE, J.: Computer as Thinker/Doer: Problem-Solving Environments for Computational Science. ., IEE Computational Science and Engineering 2004
- [Garcia-Molina1994] CHAWATHE, S.; GARCIA-MOLINA, H.; HAMMER, J.; IRELAND, K.; PAPAKONSTANTINOY, Y.; ULLMAN, J.; WIDOM, J.: The TSIMMIS Project: Integration of Heterogeneous Information Sources. In: *Proceedings of the 10th Meeting of the Information Processing Society of Japan (IPSJ-94)*, 1994, pp 7-18
- [Garcia-Molina1996] PAPAKONSTANTINOY, Y.; GARCIA-MOLINA, H.; ABITEBOUL, S.: Object Fusion in Mediator Systems. In: *Proceedings of the 22nd International Conference on Very Large Databases*, 1996
- [Goble2004] LORD, P.; BECHHOFFER, S.; WILKINSON, M.; SCHILTZ, G.; GESSLER, D.; GOBLE, C.; STEIN, L.; HUL, D.: Applying Semantic Web Services to bioinformatics: Experiences gained, lessons learnt. ., 3rd International Semantic Web Conference (ISWC), Hiroshima, Japan 2004
- [Goh1997] GOH, C.H.: Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems. ., Dissertation. Massachusetts Institute of Technology, 1997
- [Gome2004] GOMEZ-PEREZ, A.; FERNÁNDEZ-LÓPEZ, M.; CORCHO, O.: Ontological Engineering. London: Springer, 2004
- [Inmo1996] INMON, W.: Building the Data Warehouse. John Wiley, 1996
- [Kife1995] KIFER, M. AND LAUSEN, G.: Logical foundations of object-oriented and frame-based language. *Journal of the ACM*, 42 (1995) 4, 741-843
- [Kife1997] KIFER, M. AND LAUSEN, G.: FLogic: A higher-order language for reasoning about objects. *SIGMOD Record*, 18 (1997) 6, 134-146
- [Kirk1995] KIRK, T.; LEVY, A.Y.; SAGIV, Y.; SRIVASTAVA, D.: . In: THE INFORMATION MANIFOLD. (Eds.): *Proceedings of the American Association of Artificial Intelligence (AAAI) Spring Symposium on Information Gathering from heterogeneous, distributed Environments*, , 1995, pp 85-91
- [Mano2001] MANOLESCU, I.; FLORESCU, D.; KOSSMANN, D.: Answering XML Queries on heterogeneous data sources. In: *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, , 2001, pp 241-250
- [Motta2004] CABRAL, L.; DOMINGUE, J.; MOTTA, E.; PAYNE, T.; HAKIMPOUR, F.: Approaches to Semantic Web Services: An Overview and Comparisons. In: *In proceedings of the First European Semantic Web Symposium (ESWS2004)* Heraklion, Crete, Greece, 2004

- [Shet1990] SHET, A.; LARSON, J.A.: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, Vol. 22 (1990) No. 3
- [Terz2003] TERZIYAN, V.; KONONENKO, O.: Semantic Web Enabled Web Services: State-of-Art and Industrial Challenges. In: JECKLE, M.; ZHANG, L.-J. (Eds.): *Web Services - ICWS-Europe 2003*, 2003, pp 183-197
- [Wach2001] WACHE, H.; VÖGELE, T.; VISSER, U.; STUCKENSCHMIDT, H.; SCHUSTER, G.; NEUMANN, H.; HÜBNER, S.: Ontology-Based Integration of Information - A Survey of Existing Approaches. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01) Workshop: Ontologies and Information Sharing*, , 2001
- [Wagn2004] WAGNER, M.; PAOLUCCI, M.: Enabling Personal Mobile Applications through Semantic Web Services. 3rd International Semantic Web Conference (ISWC), Hiroshima, Japan 2004
- [Wied1997] WIEDERHOLD, G; GENESERETH, M.: The conceptual basis for mediation services. *IEEE Expert*, Vol. September/October (1997), pp 38-47

3 Introduction

3.1 Purpose and Scope of Document

This document describes the infrastructure that has been created based on semantic technologies for the integration and discovery of resources in distributed environments. Achievements of the first 18 months of the SIMDAT project with respect to ontologies and related applications are explained. The first 12 months can be seen as an important feedback loop helping to adjust the strategies for the establishment of a semantic middleware. Within the second phase of the project, the results of the first phase will be extended in terms of a generalized architecture. This includes a generalized infrastructure for supporting functionality such as data transport and the support for additional types of sources. The flexibility and scalability of the infrastructure available will be increased. Additional domain models and task specific models will be created in cooperation with the application areas (according to their needs).

This document includes the feedback received within the first phase of the project as well as requirements of the application areas for the second phase. As described in [D6.1.2], SIMDAT project followed a “vertical approach” with respect to the application of semantic technologies in the different application areas. Therefore requirements fall into two categories:

- requirements for a consequent extension of what has been established within the first phase (based on existing prototypes);
- requirements for a “horizontal transfer” of the existing approaches.

The targeted middleware covers two important aspects with respect to professional users within a distributed environment:

- “Information driven”:
Users (which includes “technical users” such as programmers) need to fulfill a certain task based on problem-solving environments. The problem for those users is the access the information they need in the form they need it due to the heterogeneity of the distributed sources and the lack of explicit semantics.
- “Process/Goal driven”:
Users have a goal that is based on a typical process such as a data analysis of a certain category of data. They do not have all technical details of the technical environment. Their approach is therefore not service-oriented (“I need the service x”) or data-oriented (“I need service x for datasource y”) but goal-oriented (“I want to perform an analysis of type x on a dataset that is characterized by ...”).

This document shows that there are generic approaches for those two aspects which can be categorized in terms of “integration” and “discovery”, while the first can be the base for the second one. Within the first phase of the project has emphasized those basic approaches. The current phase (after PM12) takes into account the first feedback and experiences in the application areas, emphasizing aspects like scalability and flexibility, which are crucial for an orientation towards industrial settings. It also implies an increased importance of the combination of different technologies.

There are two basic categories into which the activities in the area of ontologies fall:

- conceptual work;
- technology take-up, architectural design and implementation.

Conceptual work includes the creation of domain models but also work on standards. This document as well as other deliverables shows that the focus in SIMDAT with respect to the conceptual level is on domain models rather than on generic models or the semantic enrichment of existing specifications or standards.

3.2 Definitions, Acronyms and Abbreviations

F-logic	Frame Logic http://portal.acm.org/citation.cfm?id=210335
OWL	Web Ontology Language Recommendation by the W3C http://www.w3.org/TR/owl-features/
OWL-DLP	Web Ontology Language - Description Logic Programs, a.k. OWL Easy, Subset of OWL, containing all OWL Syntax could be explained in Horn Logic http://logic.aifb.uni-karlsruhe.de/
OWL-S	Web service ontology http://www.daml.org/services/owl-s/1.0/
PSE	<u>P</u> roblem <u>S</u> olving <u>E</u> nvironment
RDF(S)	Resource Description Framework Schema http://www.w3.org/RDF/ and http://www.w3.org/TR/rdf-schema/
SPARQL	Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/
WSMO	Webservice Modelling Ontology
XML	Extensible Markup Language Recommendation by the W3C http://www.w3.org/XML/

4 Technology improvements description

4.1 *PM12 Environment*

4.1.1 Service Discovery

One of the main advantages of ontologies is the possibility to efficiently handle the complexity of a particular domain and provide a common language. In this case ontologies provide a common view of how the services available can be characterized for their users. Users can refer to this model in order to specify the service they need based on their goal. The model describes the scientific domain rather than the technical details of relevant processes and data sources (e.g. release details). The latter information is thus encapsulated. The benefit of this concept increases with the complexity of the domain and the technical environment as well as the number of services, making it difficult to specify the concrete service instance needed for a particular task.

As described in [D6.1.3] the infrastructure for service discovery consists of basic components and parts that have been customized for the Pharma application area. It allows to encapsulate the technical details of search- and analysis services and use domain-specific descriptions for their retrieval and description.

The goal within SIMDAT pharma (and other application areas relying on semantic services) is to:

- provide an infrastructure for the description, annotation and discovery of services;
- provide the models required to describe the characteristics of the relevant services and all related concepts (domain ontologies in combination with ontologies for data sources);
- allow users to specify abstract service profiles on different levels of granularity in order to retrieve appropriate services;
- separate the service description, the definition of the abstract profile and the actual retrieval (or matchmaking) process;
- support users with domain knowledge but a limited knowledge about the details of available services (and associated data sources).

The Infrastructure that has been established within the first phase of the project includes the following basic components:

- a set of ontologies for the semantic description of services with a focus on the bioinformatic domain;
- a semantic service broker encapsulating the OntoBroker® inference engine;
- an annotation standard in XML-syntax to represent service metadata;
- a service for the registration of search- and analysis-services with the semantic broker based on the annotation standard;
- a graphical tool for the creation of annotation data.

This set of components provides a runtime-environment for the retrieval of services as well as capabilities for the administrative tasks necessary for the management of semantic metadata.

[D14.1.2] describes the pharma prototype which contains all building blocks for service discovery. [D6.1.3] and [D14.1.2] contain the conceptual as well as the technical base for the infrastructure with respect to service discovery.

In addition [D6.1.3] describes typical use cases as well as possible rule-based extensions.

4.1.2 Semantic Information Integration

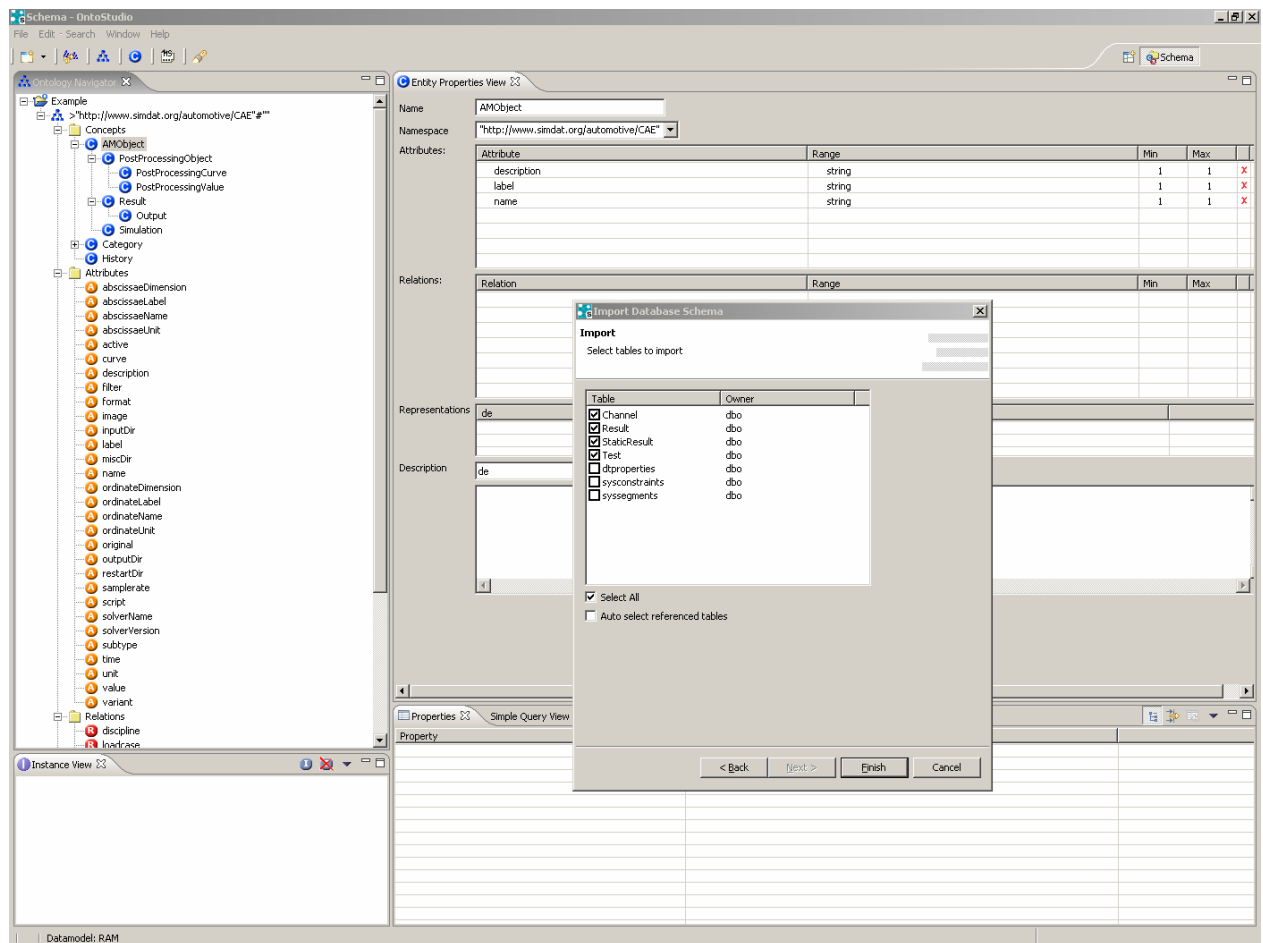


Fig. 1: OntoStudio schema import

At the current stage of the project, an environment for semantic integration has been provided, that:

- provides a basic form of a “semantic layer” through which sources are accessed;
- allows to create ontologies as representative models problem solving environments in the application areas rely on;
- supports the import of existing relational database schemas in terms of a syntactical integration (providing a live access to the source database via its representation as “flat” ontology);
- supports the creation and management of mappings between the different models based on declarative rules (in F-Logic);
- supports the evaluation of ontologies and rules by a reasoning engine, allowing to access the data of the integrated sources via several semantic layers (e.g. using articulating ontologies);
- provides a service that takes path expressions of a generic query language (referring to a stored ontology), performs the syntactic translation and communicates with the ontology server (see **Fehler! Verweisquelle konnte nicht gefunden werden.** for the specialized Automotive solution: client PSE is SimManager, query language ‘Expression Language’ serialized in xml there).

Integration of external sources via rules:

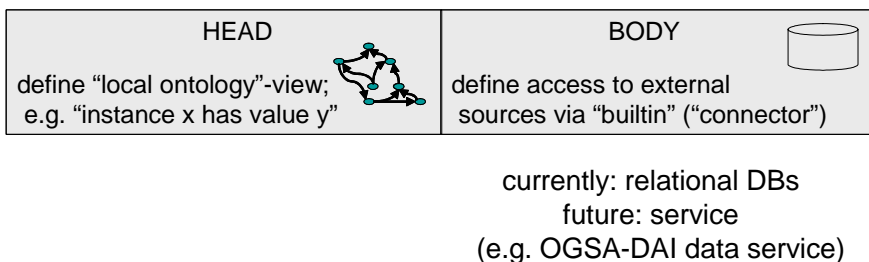


Fig. 2: **Rule based integration**

Figure 2 above illustrates the basic principle of the rule-based integration schema. For every integrated data source a set of rules is generated that maps the "ontological view" onto the schema of the source system. Based on those rules the integration can be realized in a single step, mapping the data schema of the source system directly onto a domain model, or in two steps, where the first step is a "syntactical" or "technical" integration resulting in a flat model.

4.1.2.1 Architecture and Implementation of Information Integration Prototype

Next sections give some information about design and implementation aspects of the information integration prototype for the integration of different data models from different PSEs.

Architecture of Implemented Ontology Prototype

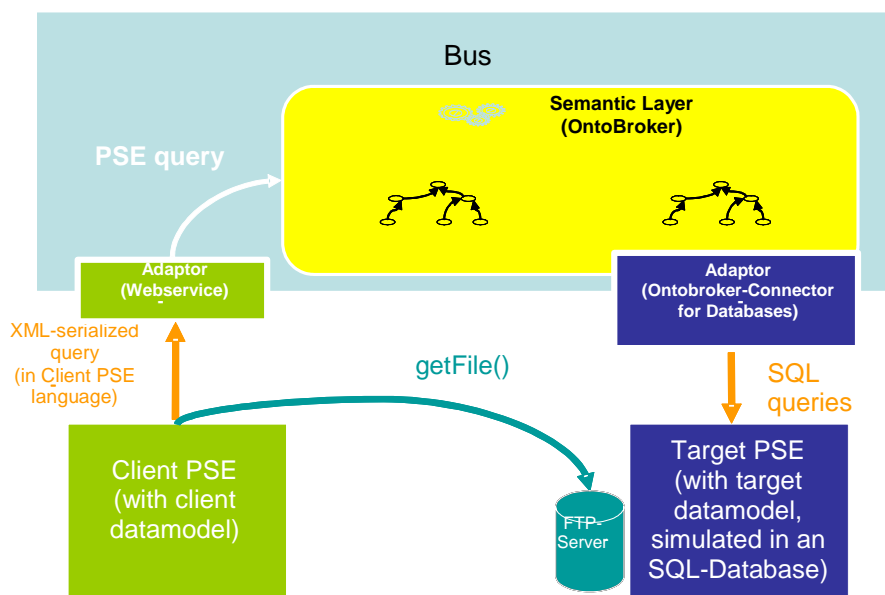


Fig. 3: **Architecture of Ontology Prototype**

Components

The underlying architecture of the prototype for the project month 12 (Auto-1 Ontology Prototype PM12) is illustrated in figure 3. The main components are

- a client PSE,
- a target PSE, and

- the infrastructure based on the OntoBroker.

The last one is composed of a web-service serving as an adaptor to the OntoBroker's reasoning machine, of the reasoner of OntoBroker and of the built-in-connector of OntoBroker serving as an adaptor to the database where the target PSE's data is stored.

Communication

The communication is initiated by the client side through the sending of a query to the web service from the Client PSE side. The web service translates the query to the query language which can be understood by OntoBroker (F-Logic) and passes the transformed query to the OntoBroker. OntoBroker evaluates the query sent to it by deploying the ontologies published on it before and by applying the mapping rules to build the response to the query. During this process it connects to the Client PSE database over a built-in-connector to pull the data qualified for answering the query. The retrieved data are handed over to the web service, which sends them to the querying client. If the answer contains a reference to a remote vault data, the mechanism for retrieving such data, for instance binary large objects (blobs), has to start.

Queries and Results

The important point in the query evaluation is that the querying system (client PSE) does not need to know about the data model of the other system (target PSE). It creates the query based on its own vocabulary. The responsibility for making a reasonable semantic translation of the query to the semantics of the queried system lies with the semantic layer, represented by OntoBroker.

The same is applicable for the query result: the client PSE expects special kind of data as response to its query. In other words, the client PSE wants to handle the results of the query sent to the target system in the same way as it would handle results of queries to the local system.

While answering the query no object streaming is deployed to build a result object. Instead, only object identifiers and attribute values are transmitted as result. That means that if the client queries for an object, it gets an object identifier from the target system which is then used to retrieve the corresponding attributes of that object. This means that the objects can be built while querying for their attributes. However, the representation of such objects corresponds to the vocabulary (data model) of the querying system, not of the queried system, i.e. if a client PSE queries for some instances of a concept named *C_client* on both the client and the target system side, it will not know anything about the concept name *C_target* used for the classification of these objects in the target system. Instead, it will deal with the objects from the target system as if they were described with concepts from the client system.

Publishing of Ontologies

The data models for client and target PSE are published on the OntoBroker through the creation of the appropriate ontology for the client PSE data model and automatic SQL schema import functionality of OntoStudio¹ for the target PSE data model.

4.1.2.2 Implementation Aspects of Adaptors and Bus

The second part of the architecture implemented for project month 12 is the web service which captures a prototypical implementation of an adapter for the querying system, some semantic layer functionalities of the bus and an adapter for a queried system. For the sake of simplification the queried system is emulated in a database of the prototype. This allowed us to concentrate on the main aspect of the prototype: handling of the data models with different structures. Figure 3 presents a detailed view on the adaptors and bus in the current prototype.

¹ Product of ontoprise for editing of ontologies and more.

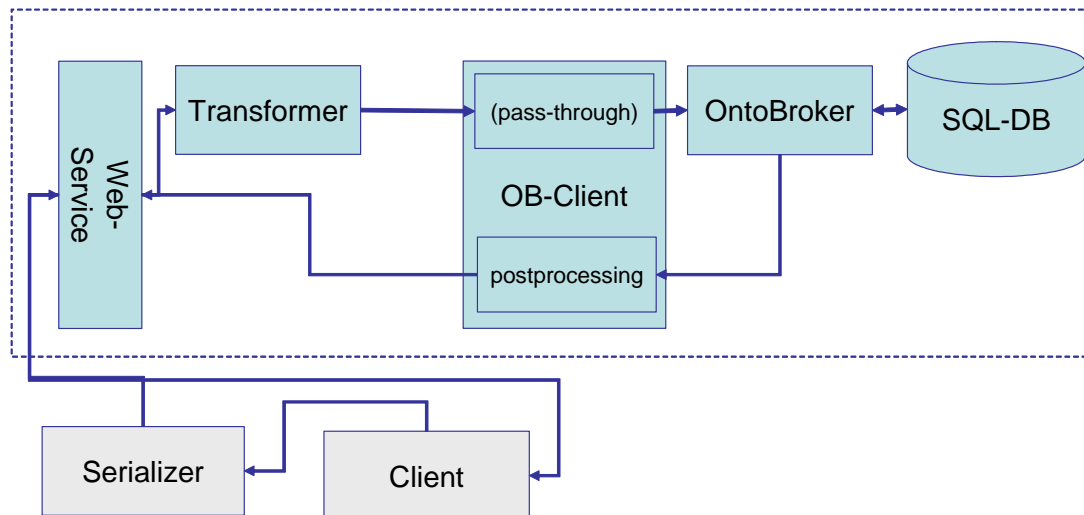


Fig. 4: Detailed view on the implementation aspects of adaptors and bus

Web Service

The web service is a clear defined interface for the clients, used for a strict decoupling of the systems. How to call it technically is defined by a WSDL document. Its query arguments are given as client queries in XML format, its results are arrays of strings.

Transformer

The Transformer translates a client query in XML format into a corresponding F-Logic query. First it parses the XML and converts it into an object graph. Then this object graph is transformed into an F-Logic query².

OB-Client

The OB-Client gets the F-Logic query from the Transformer and passes it to the OntoBroker. The result received from the OntoBroker is additionally processed to fit the expected format.

OntoBroker

The OntoBroker contains an inference machine, which holds the used ontologies and accesses data in the SQL DB containing target PSE data via a so called built-in, which is represented in the diagram as connector component.

4.2 Lessons learnt

4.2.1 Domain Models

The experiences in the first phase of the SIMDAT project have shown that the necessary level of detail and the corresponding effort for models differs between the applications. The modelling effort can be divided into:

- generic models and task-specific models (i.e. top-level models for services)

² Both languages can be quite different: e.g. client query language expressions don't need to have variables, whereas F-Logic queries always have them.

- generic models with a focus on certain domains (e.g. models for resources in the scientific area)
- domain models (e.g. models for product data or biological taxonomies).

A great part of the modelling effort in the area of service discovery went into the development and combination of domain models. This development is motivated by the question “what is the kind of queries the ontology is expected to provide the answer for”. In the pharma area, the difficulty was to align this domain-specific part with the generic part, based on the OWL-S model. In addition conceptual work was performed to develop the necessary domain models and link them to existing models. The reusable part of the whole framework is basically limited to the technical infrastructure (annotating services, registering services, retrieval of services, etc.). The resulting models provide a very good base for extensions, such as a more detailed resource model (as proposed by ...).

Much of the effort in the Pharma-area is related to the complexity/size of the existing models involved and the requirement of OWL-S compliance. The link between the generic service level and the domain level has to be established using the extension mechanisms available, as explained in D6.1.2, section 3.2.1. This would have been easier relying on OWL-DL in combination with a DL-reasoner. However, the latter solution would imply conceptual problems with respect to rule-based extensions. The current solution gives room for both approaches in the future: consequently switching to an OWL-S/OWL-DL approach or using OWL-S as top-level model exploiting the advantages of the LP-paradigm. The possibilities and the implications of rule-based extensions therefore need to be evaluated³.

In the Automotive area the situation was different due to the fact that the focus was on the overlap (and possible conflicts) of diverse existing data models. Implementing a demonstrator still required domain-specific modelling effort but under different conditions. There was no top-level model, only an articulating model. The concepts to be included in the models were clearly defined. However, the effort was limited due to the fact that the chosen models only represent a subset of the data models in use.

The domain specific models in the Meteo- and the Aerospace-area are still to be developed. The underlying scenario in the Meteo-area is related to the Pharma service discovery scenario. However, the approach is not directly transferable. The process related to the retrieval of Meteo data-service-products differs from the service in that queries going to the catalogue are less precise and less specified but the degree of flexibility is higher, since a combination with “traditional” keyword-based searches is required. In the Meteo-Area a taxonomic approach is sufficient, which should reduce the necessary effort.

³ The most prominent approach in this area is SWRL. It is mainly a proposal for a rule-based extension of the OWL-language. However, in the very general case the combination of OWL and SWRL can easily destroy the decidability of “pure” OWL. There is no general solution for the computation of OWL/SWRL models. [Motik] have proposed a restriction of SWRL-rules allowing to remain in a decidable fragment of first order logic.

4.2.2 Integration framework for the interoperability of distributed sources

4.2.2.1 Performance

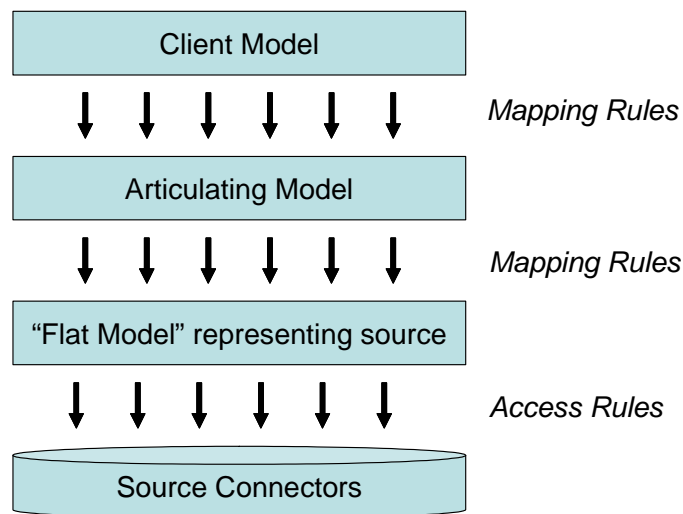


Fig. 5: “Dependency Layers” for rules

There have been general performance problems with the PM12 demonstrator. The response time of the system was often not acceptable. End-users using interactive controls of the client system (SimManager) such as tree-controls would expect a response to certain actions (such as expanding the node of a tree) within some seconds. The performance is important, since the acceptance of the approach by engineers would be crucial in an industrial setting.

Main factors influencing the performance of the system are:

- the communication overhead (network-communication, XML-serialization and –parsing, etc.);
- the reasoning process (related to the complexity of rules);
- the transformation process between the client and the OntoBroker;
- the access to the connected external source by the OntoBroker.

Regarding the PM12 prototype the last two aspects proved to be the main limiting factors for short responding times of the Ontology-Server. The communication overhead is potentially critical once several layers of webservice-based communication are involved. The future architecture introduces at least one additional layer of this type. Therefore this aspect might become more important as a main limiting factor.

Regarding the reasoning process the characteristics of rules have to be considered.

The rules resulting from the integration process introduce three “dependency layers” (as indicated in figure 5 above). This means that different models are connected via rules, which can be seen in terms of a data flow⁴. The rules include:

- Mapping rules between the different models;
- Rules providing the base for the “technical integration” (the link between the source system and the flat model representing the corresponding data model within that system).

⁴ Note that this consideration is independent from the way the rules are actually evaluated, so it does not necessarily imply backward-chaining or a top-down evaluation of rules.

It is important that there are no cycles within those dependency layers. This could be the case if additional rules would be introduced. The evaluation of the rules in the different layers leads to low-level queries to the connected sources. The evaluation process within the reasoner is not a “query translation”. One query by the client can lead to several queries to a data source even if the client-query could in principle be satisfied by just a single query to the source. This can lead to a considerable overhead, especially if queries to the source are “expensive”, which is typically the case for decoupled components.

4.2.2.2 (Data-) Grid Compliance

The current framework uses webservice-communication as interface to the ontology service and therefore allows de-coupling the query-client and the semantic layer. However it does not consequently build on grid-frameworks and uses a direct connection to the underlying data sources. The discussions around the PM12-infrastructure and possible improvements/extensions made clear that a general approach for a number of aspects around the actual semantic integration is required. This includes security and the access to external sources in form of “data containers” via references (as described in [D6.1.3]). Grid-frameworks such as the globus-toolkit provide a base for a general approach. Therefore the current architecture has to be made “grid-compliant”:

- Connectors in the semantic layer should not access integrated sources directly but via an appropriate framework (encapsulating parts of the system-specific characteristics);
- The Ontology Service itself has to be available in terms of a data grid resource based on such a framework.

This de-coupling of components has implications on the performance of the Semantic Layer as described in the previous section. Therefore performance optimizations on the internal level (of the Semantic Layer) will be even more important for a “grid compliant” version of the infrastructure.

4.2.3 Usability, End-User Tools

The main contribution of semantic technologies within SIMDAT concerns middleware for the description, annotation, mediation and discovery of different sorts of resources. This implies most work on the server side rather than on the client side. However, the first phase of SIMDAT made obvious that considerable work has to be done for components that have user interfaces in order to establish a base for a productive use of the infrastructure. Those components are not bound to the central idea of supporting distributed (and mostly service oriented) environments but provide a general support for the application of semantic technologies. Examples are:

- The annotation of resources, namely services
The annotation of services is a crucial step in the service discovery infrastructure (as described in [D6.1.2]). A productive solution will always include end user tools, such as the TUAM tool developed by FhG SCAI. Even though the further development of TUAM is not “semantic grid specific”, TUAM is an intrinsic part of the infrastructure since efficient means of annotation have to be provided.
- Mediation
Mediating between heterogeneous sources is a demanding task, since it involves much domain knowledge and know-how about formal representations, rules, etc. Appropriate tool support is therefore crucial. Fortunately SIMDAT can rely on existing work here (in the form of OntoMap). However, the feedback during demonstrations made clear that the usability of tools like OntoMap with regard to visualization and navigation features is a key aspect for the scalability of the integration approach. One important aspect of the SIMDAT approach to mediation is the fact that local schemas are under local control. Dealing with schema changes also influences the maintenance of mappings and the underlying tool support.
- Editing rules, tracing, debugging

The meetings in the Automotive section within the first phase of SIMDAT made clear that the work with rules and inference engines requires some tool support. The related requirements are documented in section 3.1.1 of [D6.1.1]. It has been decided in that phase of the project to first of all concentrate on the core aspects of mediation in a distributed environment. However, as stated for the other examples, appropriate tool support might be crucial for the scalability and the efficiency of the approach, especially in terms of maintenance.

The two latter aspects might be targeted in later phases of the project, while the work on annotation is continued in the ongoing phase.

4.3 *PM18 Improvements*

The improvements described here concern the interoperability framework. Those include the following results/features:

- First version of schema-import from and life access to a remote ontology server
- Support for semi-structured data sources (spreadsheets in a first version)
- Basic concept for the management of schema-changes
- Basic concept for (rule-)debugging

All results apart from the last one can be seen as building blocks for the future version of the integration framework.

4.3.1 Schema-Import from Remote Ontology Servers

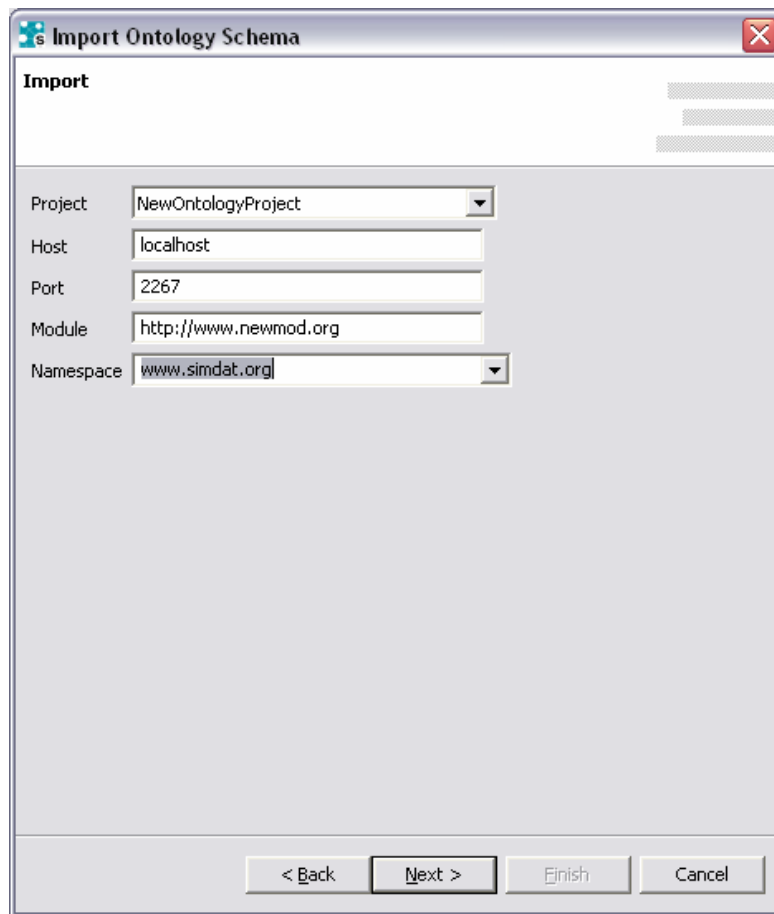


Fig. 6: Remote Ontology Import

A first prototype for the import of ontologies (on the schema level) from a remote OntoBroker instance has been created. It stores the imported concept hierarchy and property data locally and creates rules for the access to the remote instance data. It allows for example to map a local ontology (including local instance data) onto the remote ontology, represented by the redundant schema. A typical use case would be the wrapping of local resources as previously described. A local resource such as a file is represented by a schema in the local OntoBroker instance. The schema along with a set of rules and built-ins for data access form the “ontology wrapper” for the local resource and define how that resource is to be interpreted. The wrapper is then provided for import by another OntoBroker instance.

However, this first Prototype still lacks some important features:

- It has no web-service- or even wsrf-compliant interface.
Both, the schema-import and the access to instance data use a socket-based interface. This is not a general solution for distributed ontologies and/or resource wrappers, since it lacks the necessary flexibility and extensibility. Features such as a service registry or different mechanisms of data delivery would have to be built on a proprietary solution. Therefore the remote schema import has to be embedded into the new architecture.
- It provides no capabilities for the management of schema-changes
Once the local schema changes, the redundant (imported) schema has to be re-imported. This requires a notification process and means of adapting mappings, possibly semi-automatically (see also section 7.2.11).

The current version of the schema import is based on a “pull” mechanism, i.e. the direct import of the schema is triggered by the importing OntoBroker instance. The assumption is that the importing instance has the metadata for the remote instance, which means the administrator has to know which schema can be imported from which instance. An alternative approach would be a registry where instances can publish schemas and metadata for the access to instance data. This approach is not yet part of the planned architecture.

4.3.2 Semi-structured Resources

The current version of the infrastructure allows to annotate and integrate spreadsheet files. Users can create metadata for a particular spreadsheet file to define how this spreadsheet file is to be interpreted logically. A spreadsheet-connector for MS Excel® spreadsheets supports the access to spreadsheet-files as a source of instance data, similar to the database schema import used for the PM12-integration-prototype. This functionality has mostly been developed in the SEKT-project (EU IST IP 2003-506826) and is currently extended in SIMDAT.

Users can annotate spreadsheets with the help of an OntoStudio® plugin based on an existing concept hierarchy. Out of the annotation model a set of rules is created automatically. The latter defines how instance data is generated out of the spreadsheet contents. Each rule contains a call to the spreadsheet-connector.

The spreadsheet connector caches the contents of the annotated table. It reloads the data any time the file attributes have changed. However, it cannot yet handle changes of the table structure. It does only support “flat” table models.

A problem with integrated spreadsheets is that the tables contained can be changed much easier than for example database schemas, since table-layouts can be changed and created in many ways and the maintenance process cannot be controlled systematically. Users define the layout of a table rather than the structure. The concept for schema-changes (see next section) can therefore only be applied under the assumption that these changes have are comparable to those that typically occur in database schemas.

The spreadsheet connector along with the visual annotation plugin support the wrapping of resources as described in the previous section.

4.3.3 Concept for Schema-Changes

As a first step for the management of schema changes a concept has been developed that is suited especially for database schemas and therefore exploits the characteristics of mapped database schemas. This concept might be adapted or extended once PSEs as data sources are integrated into the infrastructure. The characteristics of schema changes in those PSEs therefore have to be analysed in later phases of the project.

In this document just a summary of the concept is given, leaving out the example that has been worked out as case study. A more detailed description will be given when a basic version of a change management support has been implemented.

A core component is a tool that analyses different versions of an imported schema and derives which changes have taken place (with a certain probability). It is referred to as “diff-component” in the following sections.

It should be mentioned that the interpretation of a database schema in terms of an ontology can be achieved in two ways:

- The database schema is imported as “flat schema” interpreting tables as concepts, simple columns as attributes and foreign keys as relations. The “intended” model of the database (the domain ontology) is then mapped onto the resulting flat schema. This approach therefore consists of two layers and allows to use the automatic schema-import provided by OntoStudio®. This has been done for the PM12 demonstrator.

- The domain ontology is mapped directly onto the database schema using more complex rules.

In both cases the domain ontology can be seen as the “interpreting layer” defining the view onto the database schema.

Detecting Changes

There are two approaches for this step, a manual and a semi-automatic:

- The domain expert manually re-imports the schema of the external system (either in regular intervals or following a notification process) and lets the diff-component display the change analysis.
- An external (web)service-component uses the schema-import functionality and the diff-component to check imported schemas
 - in defined regular time-intervals (“virus-scanner-like”) or
 - triggered by a third-party notification or certain exceptions.

That component would compare a previously stored version of the imported schema to an automatically re-imported schema. Whenever differences occur, the detected changes would be reported (to a registered administrator). This would require the component to always have reading access to the schema of the imported system. There could be several of such services to represent different access profiles for different departments.

4.3.4 Detecting the kind of change

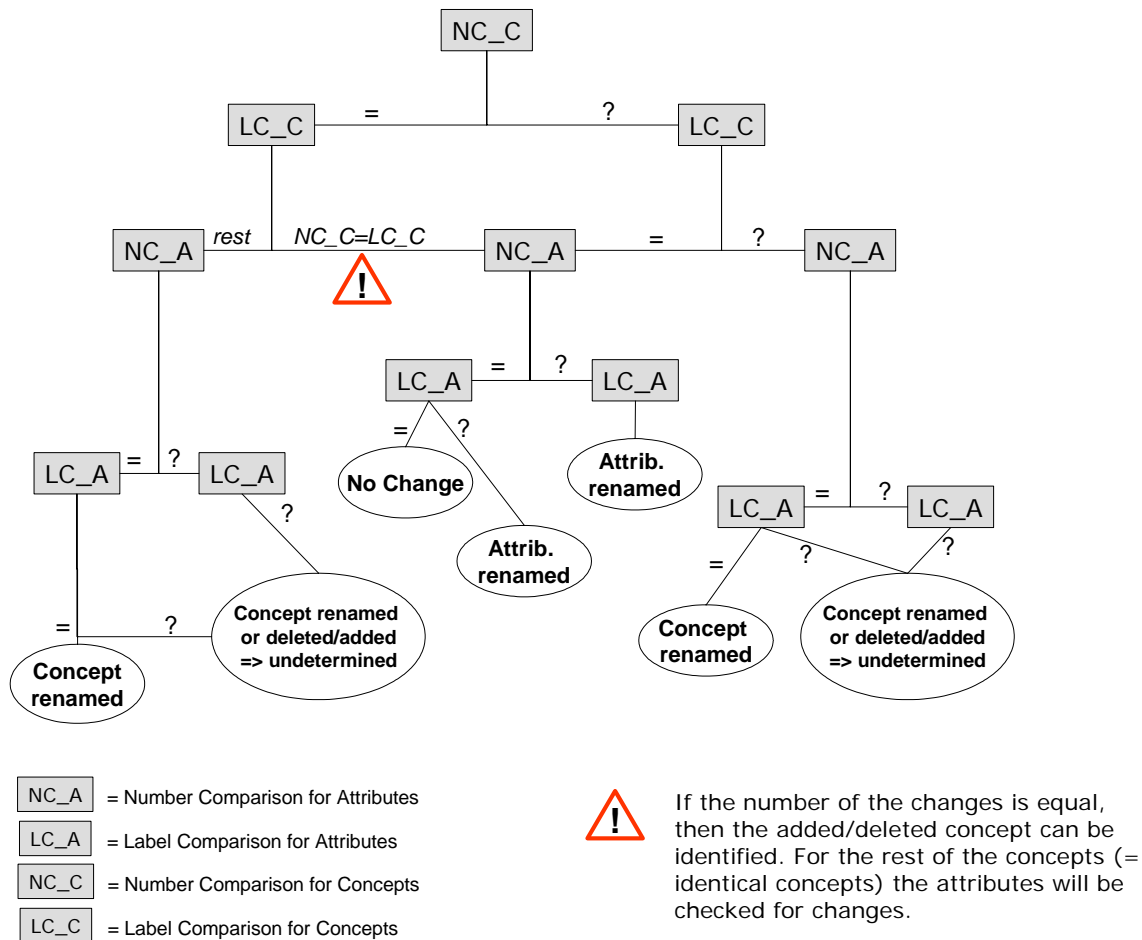


Fig. 7: **Schema of the diff-component**

As part of the maintenance process possible changes or their equivalents have to be defined and identified. The considerations in this context hold for database schemas as previously explained. Changes can be characterized based on a set of operations: add, delete and change/rename. Those operations can be applied to every basic element of a database schema (table, column, foreign key). A foreign key can be seen as a special form of attribute, but it should be treated separately as the consequences of changes differ from those of simple attributes.

The consequences of those basic operations can be different dependent on the semantics of the particular column/table or foreign key. If for example a column is renamed, the domain expert has to decide whether this affects the semantics of the attribute. If this is not the case, the name has to be adapted in the particular access rule. A special case that should be mentioned concerns the removal of tables which are mapped onto concepts that are part of a concept hierarchy. While the concept onto which the table is mapped has to be deleted, the sub-concepts have to remain in the hierarchy. This can be achieved if the super concepts of the deleted concept become the direct super-concepts of those sub-concepts (lifting the sub-concepts upwards in the hierarchy).

Some operations can be expressed as a sequence of the basic operations mentioned above, such as a change of the reference of a foreign key, which can be seen as a combination of a delete- and an add-operation.

Figure 7 contains a schema of the analysis steps taken by the diff-component. At the end of the process the kind of change is indicated to the user (administrator of the ontology server taking the role of the semantic layer).

4.3.5 Visualization of Changes

Once the diff-component has detected changes in the imported schema, they have to be visualized in an appropriate way to support the domain expert in adapting the interpreting layer. This is not limited to a visualization of the changes between the two different versions of the imported schema. All elements of the interpreting layer (model) that are affected by the changes should be visualized.

This can be realized based on the analysis of the mappings between the interpreting model, an intermediate flat schema and the data source. The OntoBroker API allows to detect all rules that contain a certain connector-call. Taking the affected connector-calls as starting point possibly affected elements of the ontology can be traced based on the dependency graph. If this analysis should not be part of the API but performed externally (by another service), an introspection feature for rules has to be made available in form of a query interface. A first prototype of that introspection feature is available.

5 State-of-the-art and the contribution of SIMDAT

5.1 Data Integration

Looking at existing approaches for data- and information management it turns out that the term “integration” is not used consistently. A middleware that has the capabilities to aggregate results from different sources might be considered to provide integration support. OGSA-DAI for example claims to be an integration platform. It supports joins over distributed databases. However, it has no means of dealing with schema conflicts, redundancies, etc.

From our perspective integration is not a purely technical task and it is not just aggregating queries over different sources. Amman for example describes data integration as process in which access to autonomous and heterogeneous sources is provided in a form which is equivalent to a single data source with a single schema. We use the following definition:

Data integration describes the process for accessing heterogeneous data via an agreed combining schema using a specific middleware component. This component is responsible for the processing of user requests / queries against the data schemas via the global schema and the transformation of results in terms of that schema.

Important aspects for data integration include distribution, autonomy and heterogeneity. The term distribution refers to the distributed data stores, which might still have a centralized management. Distribution often implies a certain autonomy with respect to design, communication and execution.

Design autonomy describes the requirement, that the schemas of the distributed sources should not be changeable by the integration middleware in order to keep the compatibility to local applications. Communication autonomy means that local systems can “decide” on their own with which other systems they communicate and whether they support the integration or not. Execution autonomy is given if a system can independently schedule operations. This includes the management of transactions, which is necessary for performance optimizations for example. This would not be possible if a central component would imply a certain order of operations.

The heterogeneity of data sources is a consequence of their distribution and autonomy. Heterogeneity refers to differences with regard to schemas, semantics, data representation and technical aspects (protocols, low-level access, etc.).

The most important approaches for information integration can be categorized as follows:

- the mediator concept [Wied1997]
- federated databases [Shet1990], [Conr1997]
- the data warehouse concept [Inmo1996]
- grid-based architectures and enterprise-application-integration (EAI) approaches

In most cases the integration is realized via an additional integration layer, transforming the original 2-layer architecture (data source layer and user level) into a 3-layer architecture. There are some similarities with the 3-layer ANSI SPARC architecture, implying that the integration takes place on the *logical level*. This makes sense taking into account that the logical level concerns the structure of data sources and data schemas are integrated rather than the data itself. Reasons for the 3-layer approach are given for example by Wiederhold:

- the intermediate layer (integration layer) includes knowledge which is out of the scope of the local systems;
- intelligent processing of data includes uncertainty, which can be managed by databases;
- an intermediate layer supports the integration of any number of data sources;

- an intermediate layer allows to provide “value added services” such as for data aggregation, filtering or abstraction all users and applications concerned.

For these (or similar) reasons Wiederhold argues that an integration on the application level is not appropriate. The analysis of the approaches mentioned above show that a global schema as a base for the integration layer is necessary. This schema combines the local schemas and provides a unique view on them for queries on the user level. There are different possibilities for this combination, out of which the most popular ones are the Global-as-View (GaV) and the Local-as-View (LaV) approach (see following section). Examples for systems based on LaV are Information Manifold [Kirk1995] and [Mano2001]. Systems implementing the GaV-approach are TSIMMIS [Garcia-Molina1994] , Garlic [Carey1995] , COIN , MOMIS and Squirrel.

5.1.1 Integration Strategies

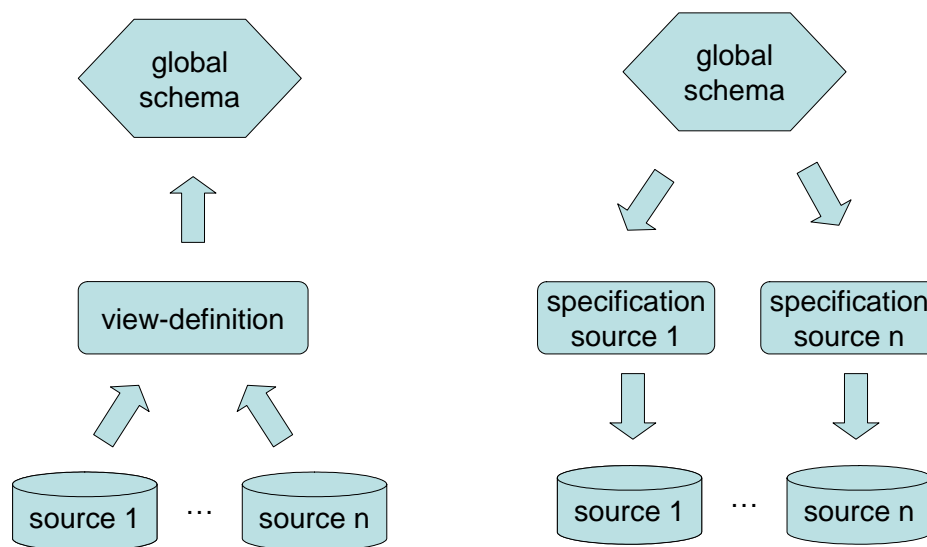


Fig. 8: **Global-as-View versus Local-as-View**

The GaV-approach defines the global schema as a view on the local schemas. This means that the global schema is a virtual, integrated view on the data sources. All relations between the data sources are considered. Queries against the local systems are used to compose this view, using the query language available (SQL or rule-based language). An advantage of this method is the efficient and simple processing of queries against the global schema. A major disadvantage is a lack of flexibility with respect to the addition of new data sources.

The LaV-approach is based on a given or predefined global schema, over which local schemas are defined as views. It can therefore be seen as the inversion of the GaV-approach. Every local schema contains a part of the global model. Every data source is regarded as an isolated system, not considering relations between the sources. The disadvantage is that the processing of queries is rather complex, since the responses of the local systems need to be combined in a meaningful way.

There is also an approach to combine both in form of a BaV (Both-as-View) based on invertible schema transformations.

Another important aspect for the classification of integration approaches is the strategy. A binary strategy combines only two schemas in one step, either in form of a ladder (combining two single schemas and the result with the next single schema) or in a balanced way (resulting in a symmetric tree). N-ary strategies can combine any number of schemas in a single step. A special case is the one-shot integration, integrating all schemas in one single step.

5.1.2 Heterogeneity

As mentioned before, the management of heterogeneity is a main aspect of integration approaches. Heterogeneity can be further distinguished depending on the kind of differences: structural (schema-) and semantic differences or differences regarding integration rules and transaction management. Heterogeneity is a consequence of the “natural development” of data sources (typical “grown systems”), the different (meta-) data models (e.g. ERM vs. object-oriented) and the differences regarding the abstraction of real world domains that are modeled by schema designers. A great variety of modeling primitives increases the probability that a particular domain is modeled in different ways while the underlying semantics doesn't change.

The heterogeneity of data sources is a main challenge of data integration. However, there is no unique classification of heterogeneity problems. Some classifications are:

The classification according to Goh [Goh1997]

Goh distinguishes between schematic, semantic and intensional heterogeneity. Schematic heterogeneity is further classified into type conflicts, label conflicts, aggregation conflicts and generalization conflicts. A typical type conflict would be for example the storage of a date either as “String” or “Date”. Label conflicts concern synonyms for attributes and tables. Aggregation- and generalization conflicts are a result of the design process with respect to clustering and hierarchical structuring of data. Semantic heterogeneity concerns the different interpretation of equal schemas.

This kind of heterogeneity can be further differentiated into label conflicts on the data level, scaling conflicts, unit conflicts and mixing conflicts. A scaling conflict would for example arise if a scale from one to ten is used on one side, while the other side uses a percentage value for the same attribute.

Mixing conflicts if data sets are integrated which seem to be equal but are actually different. This holds for example for a “current exchange rate” that is kept in two systems but with a different interval for updates.

Intentional conflicts concern conflicts with integrity constraints as well as conflicts regarding the underlying domain. Domain conflicts describe the problem that a clear statement about the equality or similarity of data sets is not possible, since the underlying schemas are very different. However, there might still be a possible semantic relation between the data sets, such as a class-subclass relation.

Other classifications

There are other classifications, such as by Kashyap or Conrad. Kashyap's classification is orthogonal to the one of Goh. However, many elements can be found in all approaches. Kashyap's classification is based on the term incompatibility, which is further differentiated into domain incompatibility, value incompatibility, incompatibility on the abstract level and schema differences. He also describes entity conflicts, which can arise if semantically equivalent data sets using incompatible keys.

Conrad uses only four categories of conflicts: semantic conflicts, label conflicts, heterogeneity conflicts and structural conflicts. Semantic conflicts are similar to the domain conflicts of Goh, which means they refer to overlaps and intersections of data sets with semantic similarities.

5.1.3 Comparison of integration approaches

In order to compare different integration approaches, the management of heterogeneity conflicts is an important aspect. Additional aspects would be, whether the integration approach is

- complete
- sound
- minimal
- comprehensible.

Complete means that the integrating schema contains all elements of each of the local schemas (no data loss). Sound means that all information contained in the global schema needs to be present in at least one of the local schemas. If there are additions in the global schema, they should not lead to contradictions. A minimal integration is given if elements that exist in several local schemas are only represented by a single element in the global schema. Finally the integrating schema should be comprehensible for administrators and domain experts.

Other aspects for the comparison of integration approaches are:

- data management (centralized vs. decentralized storage of data or schema information)
- topicality (usage of historical data vs. online access)

5.1.3.1 Mediators

The mediator-architecture goes back to Gio Wiederhold and was presented 1992 in the first place. Wiederhold's approach is based on a 3-layer architecture, where mediators are placed in between a set of data sources and a set of applications. Every mediator is responsible for a certain set of data sources and provides a number of possible operations on the data. Adding a source to such a set requires a complete re-implementation of the mediator, whereas some implementations have tried to overcome this limitation.

The basic functions of a mediator according to Wiederhold are:

- access to multiple heterogeneous sources and collection of relevant data;
- abstraction and transformation of data in a common representation;
- integration of the transformed data (based on matching keys)
- aggregation of data.

In addition to those basic functions several "value added services" might be provided, such as for the interpolation of data or the creation of an index.

Wiederhold suggests the application of "structural models" (similar to a global schema) to describe the data sources of a mediator and their relationships ("resource model") as well as the user model of the applications. In a different approach he suggests the usage of ontologies as a base for the resource model. This shows that the mediator approach is rather abstract and can be realized based on different technologies. It can be seen as a "meta approach" for a middleware which provides a global, integrated schema of the data sources as well as a number of operations on them.

A prominent example is the TSIMMIS-project [Garcia-Molina1994] which extends the mediator architecture by other components, such as extractors (for unstructured data), constraint managers and translators. Those components do not extend the mediator approach at such but further specialize it. An important aspect of TSIMMIS is that a mediator can combine translators (on top of data sources) and provide a global schema for those based on the GaV approach. However, a single global schema for all mediators is not possible. TSIMMIS uses an Object Exchange Model for data representation, which is language with some similarities to XML, allowing hierarchical representations (nested objects) and path expressions for queries.

Other examples are Information Manifold, which uses a description logic language instead of OEM or STYX [Aman2001], which is based on XPath, XML and XSLT.

5.1.3.2 Federated Databases

Among others federated databases were introduced by [Conr1997] and [Shet1990] as another possibility for the integration of heterogeneous sources. According to [Conr1997] federated databases represent the ideal case of a new system that "global users" can access in terms of a single, homogeneous source. In contrast to the mediator architecture federated databases follow a centralizing approach, which does not envisage the combination of several federated databases. Apart from this, there are many similarities with mediators.

Usually federated databases use a logical integration, which means the actual data remains in the local sources. However, some systems use materialized views to provide data.

5.1.3.3 Data Warehouse

Inmon uses the following definition of Data Warehouse [Inmo1996]:

„A data warehouse is a subject oriented, integrated, non-volatile, and time variant collection of data in support of management’s decisions.“

“Subject orientation” refers to the model for the goals regarding the application of the data warehouse. “Integration” means that data processing and -analysis is based on integrated data. “Non-volatile” means that the provided data sets are stable. Data warehouses usually have read-only access to the underlying sources. “Time variant” stresses the fact that data warehouses are often used to draw comparison over certain time intervals.

Bauer describes a data warehouse as a physical database which provides an integrated view on any kind of data for the purpose of analysis. That a data warehouse is a physical database (containing data itself) is a major difference to federated databases and mediator architectures.

The architecture of data warehouses is not described here. However, there are some aspects to be pointed out. In contrast to federated databases emphasize data cleaning. Schema transformation on the other hand does not play such an important role.

5.1.3.4 Semantic Integration

Recently ontologies have been proposed as a possible alternative to existing approaches that have been presented in the previous sections ([Calv1998], [Wach2001]). Using ontologies seems to be promising especially with respect to semantic conflicts of heterogeneous sources since they provide a rich, explicit, and formal model for the representation of real world problems. The numerous modeling constructs are only limited by the underlying representation language (description logics, frame-based logics, semantic networks) and goes beyond the possibilities of classical models such as ERM. The implicit semantics of the data model can be made explicit with the help of domain experts and therefore be exposed to end-users and their applications.

Ontology-based approaches do not only use the data models as such but also their interpretation. The first step is to take over the existing data model in terms of a “lightweight ontology”. In the following steps that model is enriched based on the provided means of modeling in order to describe the semantics of the model. Once this has been done for all sources, the relations between those sources are modeled.

Most research projects focus on this last aspect: the mediation of ontologies based on mapping-, alignment- or merging approaches.

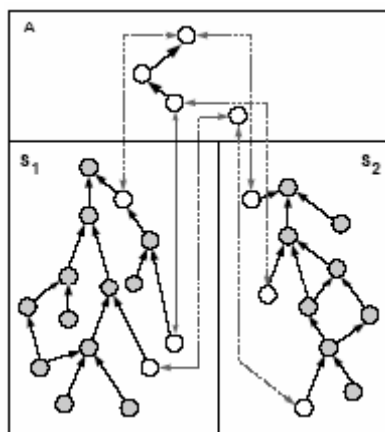


Fig. 9: **Concept of articulating ontology [Comp2002]**

Mapping approaches explicitly model semantic relations between concepts, relations and attributes of two ontologies without changing them. This is an important feature for data integration since it guarantees the autonomy of data sources. A major disadvantage is the effort needed for the creation of mappings since they can only be established pairwise.

A further enhancement of the mapping approach is the concept of articulating ontologies, which was developed to improve the scalability and maintainability of integration/mediation systems. It uses an articulating ontology onto which “local” ontologies are mapped instead of pairwise mappings (see fig. 11). The articulating ontology is strictly spoken not the integration but the interoperation of the “local ontologies”, since it “articulates” the relations between those ontologies. There can be several articulating ontologies for different user groups and applications.

The alignment of ontologies extends the mapping approach by possible changes in the ontologies involved in order to increase the intersection between them. This is in conflict with the autonomy of data sources if applied as part of a data integration process.

Merging approaches completely replaces two or more ontologies by a single ontology which provides a global model for data (analogous to a GaV approach). Such a global view for all data sources might be very difficult if not impossible to establish in practical cases where a large number of sources is involved. It also conflicts with the autonomy of local sources.

The COG-project has developed an approach that does not only focus on the mediation between heterogeneous models but considers all integration steps based on semantic technologies. The infrastructure that was established in COG supports different source formats (such as relational databases, XML data sources and spreadsheets) based on the Unicorn Workbench. However, in contrast to the infrastructure used and extended in SIMDAT, Unicorn supports only the mappings of (database- and other) schemas onto a central ontology. This significantly reduces the flexibility of the integration approach together with the fact that Unicorn does not follow a declarative approach (based on rules).

5.1.3.5 Other approaches

Other approaches like EAI, repositories or web-service-based architectures can not be covered in detail here. Repositories have some similarities to the other approaches presented (including ontologies) while EAI is not directly comparable.

The capability of web services to provide any functionality can be used to access different data sources and make them available for other web services. The combination of several of such services would in principle allow the integration of those sources. The Open Grid Service Architecture – Data Access and Integration (OGSA-DAI) provides an infrastructure for this approach. The goal of the Open Grid Services Architecture was to create standards for the support and implementation of web-service-

based Grid Services. There are several definitions of grids, out of which the data grid is the most relevant here.

While OGSA provides an architecture for Grid Services in general, OGSA-DAI focuses on the application of this architecture for the integration (in a wider sense) of data sources via Grid Services. Every data source is accessible through a separate grid service, which exposes this source over a standardized XML-interface. This service is therefore similar to the translators of the mediator approach. The purpose of those services is generalized query interface. The integration capabilities can not be compared to the approaches previously described. OGSA-DAI does not support the development of a global schema and has no means of an integration of local schemas.

However, the comparison to translators already shows that OGSA-DAI covers aspects of a complete integration infrastructure such as distributed data access and security and can therefore be used as a main building block.

5.1.3.6 Comparison

The following section draws a comparison between the different approaches. Note that a detailed comparison would have to be based on the different implementations, since there are significant differences even between those systems that belong to the same category. The mediator concept for example is more a “meta approach” (as mentioned before) and therefore difficult to compare to other approaches

Data Management, Applications, Topicality

Apart from data warehouses all integration approaches use decentralized data storage since they establish an integration layer on top of the local sources. This supports the autonomy and distribution of local sources. It also allows direct access to actual data. The latter is not so important for the kind of applications that data warehouses are meant for. This stresses the fact that comparison criteria like data management, topicality and application depend on each other.

Minimality, Soundness, Completeness, Comprehensibility

The previous sections show that minimality, soundness and comprehensibility strongly depend on the schema designer. A final comparison is therefore not possible. In general those criteria can be met with all integration approaches that have been presented in sections 5.1.3.1 to 7.2.9. However, there are some aspects of semantic integration that provide a much better base for comprehensibility:

The mediation between heterogeneous models (as used in SIMDAT) is based on a declarative approach using rules. The conceptual gap between mapping patterns (such as concept-to-concept mappings), their visualization and their “grounding” on rules can be minimized. Rules can be represented in the same representation format as the ontologies themselves. The OntoBroker inference engine has an explanation feature enabling users to retrace the inferencing process.

Soundness is fulfilled by all approaches, while ontologies support the representation of integrity and domain rules for the global schema (depending on the representation language).

Structural Heterogeneity

Schema conflicts (like column-based vs. row-based approach) and problems related to isomorphism which can only be resolved by ontologies and partially by mediators. All other structural conflicts can be resolved by the approaches presented in sections 5.1.3.1 to 7.2.9. The heterogeneity is in some cases eliminated due to the transformation into the relational model (federated databases) and OEM (mediators). Ontologies don’t need this transformation step since the expressiveness of the underlying languages directly support representation of those models.

Semantic Heterogeneity

There are significant differences between the different approaches regarding semantic heterogeneity which is not surprising considering the complexity of this type of conflict. Heterogeneity regarding value conflicts can in general only be resolved or reduced by data cleaning, which can be performed prior to the actual integration. Only data warehouses consider this aspect explicitly. Rule-based integration approaches as in SIMDAT would need procedural attachments allowing transformation operations on single values.

Domain conflicts can only be resolved after an extensive analysis of the data sets. However, ontologies define explicit statements about the domain as part of the model, which is not possible with other approaches. Rules could for example be used to eliminate the chance of mix-ups. For other approaches those statements would have to be hard-coded, creating some kind of black-box (outside of the model).

Unit conflicts can be resolved with all approaches, using the capabilities for arithmetic transformations in the integration layer (middleware). A general solution for scaling conflicts is only provided by ontologies.

Label conflicts on the schema level do not pose a major problem for any approach, while label conflicts on the level of values can only be resolved with much manual effort.

5.1.4 The Contribution of SIMDAT

SIMDAT consequently builds on the capabilities of ontologies establishing a technical infrastructure for the integration and mediation of distributed sources. It therefore exploits all the advantages mentioned in the previous sections.

Major difference to existing approaches and other research projects are:

- SIMDAT is use-case driven
The goal is not only to develop and analyse innovative approaches but establish solutions of industrial relevance. That's why the development is influenced a lot by the application areas and other technology areas such as distributed data access. This also implies that the focus is on the usage, extension and combination of existing (and well-established) technologies rather than on the creation of new generic approaches.
- SIMDAT has an implementation focus
In the area of data grids and ontologies others work on the extension of the OGSA-DAI architecture itself. This is an important contribution to the idea of the semantic grid. Currently this work is more on a conceptual level. The SIMDAT approach is somewhat orthogonal to this work since the semantic integration is set up on top of OGSA-DAI in a comparatively pragmatic way. However, this is still a complex task, especially when aspects like scheme changes and security are considered.

SIMDAT fills a major gap: Setting up a technical infrastructure that combines the capabilities of semantic technologies and grid technologies with a focus on industrial requirements.

5.2 Service Discovery

Web services have established as a standard where organizations have to access services that can be located within or outside of that organization. This well-known technology is the base for rather powerful and complex technologies such as the semantic description, automated search and discovery of services. The usage of standards and communication patterns supports the integration and re-use of services. Standards like SOAP, WSDL, UDDI and XML for the base for service oriented architectures that support complex processes, where the executing instance of a service might be retrieved at runtime. The combination of web services can be realized in different ways, depending on the type of composition (orchestration/choreography) and the time when they are coupled.

Typical applications of web services involve three roles: a service registry (such as a UDDI-registry), a service provider and a service requester. The service provider publishes a service on the registry. A service requester checks the service registry for an appropriate service covering the required functionality and finally calls the service (often based on SOAP over http, smtp or ftp). The information about how to invoke the service is contained in the registry.

The main problem with UDDI concerns the inefficient means of search and service discovery. This is related to a shortcoming of another standard, the Web Service Definition Language (WSDL). WSDL does not support semantic descriptions of web services. It provides information regarding interfaces, data types, bindings and addresses based on a standardized XML-schema. This includes information to physically locate a service (e.g. host, port) and information to call the service (parameters, return values). The WSDL-description focuses on an existing service rather than on the abstraction of a service. It's therefore not suited to describe classes of services on an abstract level.

5.2.1 Semantic Description of Web Services

Ontologies provide a shared understanding of a domain of interest to support communication among human and computer agents (such as Web Services), typically being represented in a machine-processable language. Thus, ontologies serve as formal models of certain domains, helping to structure information sources in a comprehensible way and describe complex dependencies efficiently. As mentioned before ontologies rely on explicit conceptualizations, in contrast to the internal data models of common applications such as database-schemes. They support knowledge acquisition by declarative means, while reasoning mechanisms needed to solve particular problems are provided by ontology-processing tools.

Ontologies can be used to describe the characteristics of a service, such as

- what a service does (functionality),
- how a service works,
- how it is to be invoked,
- etc.

Ontologies allow to set up complete models rather than just attaching keywords to a service reference. This makes it possible to describe complex structures and encapsulate them with different levels of abstraction, taking advantage of the features that ontology-based approaches provide, such as hierarchical structures, inverse relations, etc.

The semantic description of web services goes beyond the plain syntactical description of specific services. WSDL does for example not support the description of non-functional properties or information regarding the composition of services. The goal of semantic service descriptions on the other hand is to support the evaluation of service metadata and the discovery of services based on abstract specifications. Semantic service descriptions can be seen as the link between a service and the outside world, by using ontologies. The latter describe the inputs, outputs, preconditions and effects of a service and the resource which provides it, in a way that can be mapped to semantic descriptions of the real world.

Another characterization of semantic web services identifies the following objectives:

- Provide a comprehensive web service description framework.
- Define a web service discovery framework.
- Provide a scalable web service mediation framework.

Core aspects for scalable web service discovery, mediation and composition identified by [Fens2004] are:

- Document types, which describe the content of business documents.

- Semantics, which is introduced as semantic descriptions to be correctly by the service requesters and providers.
- Transport binding, which is an agreement between service requestor provider on the transport mechanism to be used for service requests.
- Exchange sequence definition, which is transport-level communication follow in inherently unreliable data communication networks.
- Communication process definition, a manifestation of business logic business messages exchange sequence.
- Security. Data contained in the messages between service requester provider should be private and unmodified as well as non-reputable.
- Syntax. Documents can be represented in one of syntaxes available.

[Oberle2005] point out that a semantic management of Web Services should not try to tackle full automation of all web service management tasks as its objective. They argue that a major disadvantage of a full automation is that is not clear what kind of machinery could constitute a semantic model that would allow for full automation including all aspects of web services that might matter. The authors further stress the trade-off between efforts for the management of web services and efforts for the semantic modeling of web services. The SIMDAT activities are in line with this statement.

[Motta2004] describe current efforts in developing Semantic Web Service infrastructures as follows:

- Usage Activities
Usage activities define the functional requirements, which a framework for Semantic Web Services ought to support.
- Architecture
The architecture of SWS describes the components needed for accomplishing the activities defined for Semantic Web Services.
- Service Ontology
The service ontology aggregates all concept models related to the description of a Semantic Web Service.

5.2.1.1 OWL-S

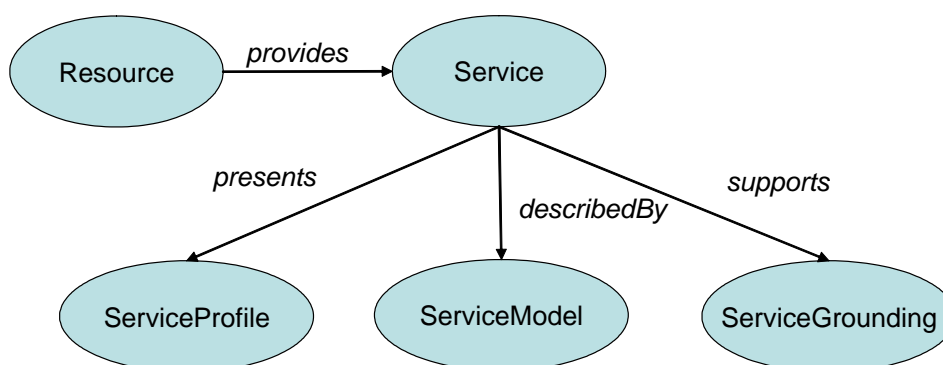


Fig. 1: OWL-S top level concepts

OWL-S goes back to the Darpa Agent Markup Language for Services (DAML-S). It provides a service model based on which an abstract description of a service can be provided.

Fig 1 shows the top-level concepts of OWL-S:

- **Service**

This concept directly corresponds to the actual service that is described semantically (every service that is described maps onto an instance of this concept);

- **Service Profile**

The service profile specifies the functionality of a service. The concept is the top-level starting point for customizations of the OWL-S model that support the retrieval of suitable services based on their semantic description. Within SIMDAT pharma the service profile has been used to describe the relations of services to data sources.

- **Service Model**

The service model specifies the how the functionality of the particular service is realized. This concept is currently not used within the semantic broker.

- **Service Grounding**

The service grounding specifies how the service has to be invoked (syntax of calls to the service; needs to be known for clients that want to use the service).

While OWL-S is a model rather than a language it has a (default) language-binding since it is defined based on the OWL language.

As [Motta2004] state, OWL-S combines the expressivity of description logics (in this case OWL) and the pragmatism found in the emerging web services Standards, to describe services that can be expressed semantically.

5.2.1.2 WSMO

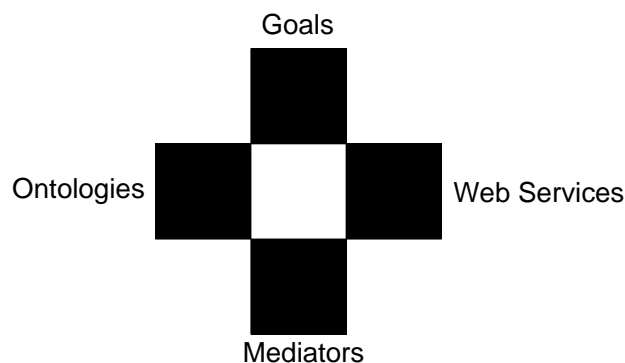


Fig. 2: **WSMO core elements according to [WSMO]**

The Web Service Modeling Ontology (WSMO) provides a framework for semantic descriptions of web services. The authors of [WSMO] describe the following core elements as building blocks of WSMO:

- Ontologies as the terminology for other WSMO elements;
- Web services as computational resources of a domain; those are described based on ontologies.
- Goals of users requesting a certain functionality of a web service; those as well are described with the help of ontologies;
- Mediators to handle interoperability problems between different WSMO elements.

WSMO acts as a meta-model for semantic web services based on the Meta Object Facility (MOF). Semantic descriptions according to the WSMO meta model can be defined on a formal language (WSML).

WSMO emphasizes the role of mediators to overcome interoperability problems. Ontology mediators handle semantic mismatches or gaps between the models used.

A major difference to OWL-S is the language binding. OWL-S is based on OWL plus additions for the description of conditions. WSMO defines a set of languages, whereas a major goal was to support different expressiveness and computational guarantees. This is especially important for the usage of rules with ontologies. Another difference is the abstraction of goals and service capabilities WSMO uses.

5.2.2 Applications of Service Discovery

There are several applications of service discovery following a similar approach as in SIMDAT. Three examples are introduced in this section.

Bioinformatics

[Goble2004] applied semantic web service technologies to the bioinformatics domain in several projects, namely myGrid and Bio-Moby. Both projects target bioinformatics and rely on web service technology as well as standards like WSDL and UDDI. The core components are service interfaces, service descriptions, domain ontologies, registry and a matchmaker. In the first development track of Bio Moby (called “MOBY-S”) the representation formalism of the Gene Ontology has been adopted for the creation and maintenance of domain ontologies in order to stay within the format familiar to the community of potential users. The second development track (“S-MOBY”) relies on OWL-DL as driving modelling paradigm (as well as OWL-DL RDF/XML as messaging layer). However, the projects mentioned do not rely on OWL-S (or a similar standard-ontology) as “grounding model” linking ontological concepts and WSDL-like information. The authors argue, that this is not required since their service interfaces are either not heterogeneous (MOBY-S) or the service interface is described by an upper ontology (other than OWL-S).

In contrast to the modelling approach taken in SIMDAT, myGrid focuses on bioinformatics rather than biologists. A main effort in SIMDAT has been the concept for abstraction from IT-specific information that biologists might find it hard to deal with.

Engineering Domain

[Terz2003] have developed a framework for industrial maintenance services organized in a peer-to-peer network. The background of their application is the maintenance of field devices. Communication within the maintenance network is based on web services. The capabilities of services provided by nodes within this network are described semantically, thus enabling service discovery.

Mobile Computing

[Wagn2004] describe a framework (“toolbox”) for mobile user-centered services based on semantic web technologies. They have created a testbed for internet radio stations as web services with varying service characteristics (as one possible application for their framework). The authors use a preference-based matching algorithm for service discovery.

5.2.3 The contribution of SIMDAT

A major goal of the approaches for service discovery within SIMDAT was the support of practical applications. As described in [D6.1.2], an important aspect for the service discovery infrastructure including the models created was the abstraction of service capabilities for non-technical users. We consequently followed a goal-based rather than a service-based approach. This should be one of the core aspects of semantic web services. The practical applicability of the idea of semantic service descriptions and service discovery as well as the advantages over purely syntactic registries can only be shown if real-world examples of industrial relevance are targeted. This requires a technical infrastructure for semantic service registries, service annotations and service discovery as well as appropriate models. Those models should allow users to specify abstract service profiles on different levels of granularity in order to retrieve appropriate services.

6 Matrix structure of document

6.1 Work provided to the application activities

	Automotive	Aerospace	Pharma	Meteo
Conceptual Work	<ul style="list-style-type: none"> ▪ Development of integration architecture in cooperation with the application area ▪ Mediation of domain models 	<ul style="list-style-type: none"> ▪ Discussions of a possible transfer of the integration technology with “representatives” (see requirements) 	<ul style="list-style-type: none"> ▪ framework of domain-specific models bringing together the upper service model and existing domain models 	<ul style="list-style-type: none"> ▪ Discussion and refinement of ideas for semantic enhancements of the existing prototype (see requirements)
Implementation	<ul style="list-style-type: none"> ▪ software specific to proprietary systems (query transformation) ▪ web-service interface for queries ▪ SQL-rewriter⁵ 	<ul style="list-style-type: none"> ▪ support for semi-structured resources (spreadsheet-connector and annotation schema)⁶ 	<ul style="list-style-type: none"> ▪ semantic service registry interface ▪ ontology server encapsulation for annotation (with TUAM) ▪ annotation schema for semantic service metadata 	

6.2 Feedback received from the application activities⁷

Automotive	Aerospace	Pharma	Meteo
<ul style="list-style-type: none"> ▪ OGSA-DAI based Ontology-Service ▪ SPARQL-Interface ▪ Additional Connectors III ▪ Additional Connectors III ▪ Semantic Integration Security ▪ Management of References for Bulk Data 	<ul style="list-style-type: none"> ▪ Capabilities of Representation (for Product Data) ▪ Knowledge-Services Support ▪ Additional Connectors III 	<ul style="list-style-type: none"> ▪ RDF interface to ontology server for TUAM mapping tool ▪ Refactoring of domain Ontologies⁸ 	<ul style="list-style-type: none"> ▪ Thesaurus-Support for Product Discovery ▪ Simple Online Ontology-Maintenance

⁵ This was a direct consequence of the performance issues with the PM12 automotive-ontology-demonstrator.

⁶ This is technology that already is a first step towards the support of simple “representatives” targeted by the Aerospace application area. However, most of the current implementation was developed in another project, as previously explained.

⁷ The requirements are the consequence of feedback received from the application areas. The “lessons learnt” section describes the background of some of the requirements.

⁸ For a description of the Pharma requirements regarding annotation refer to the SIMDAT deliverable D14.1.3

7 Specific and modular requirements documentation

7.1 *Introductory Remarks*

This section contains requirements for the semantic infrastructure identified at PM18.

7.1.1 **The Supplier-Producer Role Model**

First there is a difference between supplier and producer roles related to entities in the real world, and master and slave roles in a certain interaction. There are supplier-producer, producer-producer and supplier-supplier relationships, but by definition only master-slave interaction relations.

Second it depends on the context, which role an entity has in relation to another one: e.g. an entity being a supplier for a producer may also be a producer related to another supplier.

In the following the immediate P-to-P relations between two entities are considered: there are supplier-producer, producer-producer and supplier-supplier relationships.

Supplier-Producer Relationship

The supplier-producer relationship is asymmetric:

- A producer accesses systems located at one of its supplier, but not vice-versa. Therefore it needs access rights for the supplier system.
- The producer needs to know – and often may dictate – the data model the supplier has to expose to it, but not vice-versa.

There is one master-slave interaction relation, both entities take their correspondent roles – producer the master, supplier the slave.

Producer-Producer or Supplier-Supplier Relationship

The producer-producer as well as supplier-supplier relationship is symmetric. In this case no one has the control on its own. It can be modeled as two master-slave interaction relations in opposite directions, where each producer/supplier takes both master and slave roles dependent on the current interaction's data and control flow.

Consequences for the needed Architecture

In a real world scenario security issues have to be taken into account: the requirements for managing user credentials and access rights as well as connection security are from there.

Implications for the Prototypes

In the scenario for the CAE/CAT Integration Prototype PM24 (and its extensions made until PM30) all systems are running at Audi. This is an inhouse scenario, so there are no security issues, which had to be handled in a typical supplier-producer scenario, even if this prototype would go into production.

For other prototypes the implications have to be clarified.

7.1.2 **Query Languages**

The integration layer provides a query interface to the ontologies stored. In order to select an appropriate query language, the following aspects have to be considered:

- If a particular query language is chosen as the base for the ontology service, it has to be made sure that there is no major mismatch between the intended model of the query language and the way the ontology service can serve this model, including its formal semantics. If this is not the case (e.g. if a query language serves a relational model while the model of the ontology

service is graph-based), the model for that query language needs to be redefined, resulting in a “re-use” of an existing syntax. The latter option is somewhat critical, since it is likely to lead to misunderstandings for those familiar with the existing model of the language.

- The query language should be expressive enough to efficiently exploit the capabilities of the ontology service in terms of information integration. This is usually a trade-of, since the expressivity influences other aspects (such as optimization of the execution). A limited expressivity can force the query client to implement filtering processes and divide queries into sub-queries.
- The learning curve of the query language influences the productivity of its users as well as the acceptance. Since semantic information integration is an emerging technology, acceptance is a crucial factor.
- Other aspects explained by Haase et al. are closure, adequacy, orthogonality and safety. Apart from orthogonality, those aspects are of minor importance and/or not applicable for the targeted integration framework. The typical use case is the integration of external data by PSEs. Closure would be of limited use here, since it would still require a form of post-processing of query-results. Safety can’t be guaranteed, since the use of built-ins is unavoidable.

A current candidate is the SPARQL query language, excluding the “construct” feature (which would support closure). Its syntax is similar to SQL-syntax, which is an advantage in terms of acceptance. It has been developed for RDF, so it serves RDF semantics. The ontologies stored in the repository (of the OntoBroker) would be interpreted in terms of an RDF graph. This might require additional rules on the side of the ontology service.

7.2 Requirements

7.2.1 OGSA-DAI based Ontology-Service

Name	OGSA-DAI Ontology-Service		
Application Activity	Auto (and later Aerospace)		
Prototype(s)	CAE/CAT Integration Prototype PM24		
Date Created	15 th March 06	Priority	High
Created By	MSC	Technology component	OntoBroker + Extensions;
Description			
The Ontology-Service has to be OGSA-DAI compliant in terms of an OGSA-DAI data source.			
Relation to prototype			
The CAE/CAT Integration Prototype PM24 uses the Ontology-Service via OGSA-DAI to retrieve data from the Tec.Manager			
Requested functionality			
The service has to offer query-processing against the semantic layer embedded in the OGSA-DAI environment.			
Validation			
Query tests against predefined ontologies (based on existing unit-tests for the OntoBroker)			
Assumptions			

7.2.2 SPARQL-Interface

Name	Ontology Service SPARQL Interface		
Application Activity	Auto (and later Aerospace)		
Prototype(s)	CAE/CAT Integration Prototype PM24		
Date Created	15 th March 06	Priority	High
Created By	MSC	Technology component	OntoBroker + Extensions;
Description			
The Ontology-Service has to be accessible via SPARQL.			
Relation to prototype			
The CAE/CAT Integration Prototype PM24 uses the Ontology-Service via OGSA-DAI to retrieve data from the Tec.Manager			
Requested functionality			
Validation			
Comparison of the results of a set of predefined SPARQL and F-Logic queries			
Assumptions			

7.2.3 Capabilities of Representation (for Product Data)

Name	Modelling and Querying Capabilities for Product Data		
Application Activity	Aerospace		
Prototype(s)			
Date Created	15 th March 06	Priority	High
Created By	BAE	Technology component	OntoStudio
Description			
Ability to create appropriate and extendable ontology for scenario test case describing the product as well as the concepts in the design process			
Relation to prototype			
Requested functionality			
<ul style="list-style-type: none"> • Modeling functionality to import, create and manage ontologies related to product data • Expressivity to represent “STEP-like” models • Capabilities to query semantic product descriptions efficiently 			
Validation			
Assumptions			

7.2.4 Additional Connectors I

Name	Tec.Manager-Connector		
Application Activity	Auto		
Prototype(s)	CAE/CAT Integration Prototype PM24		
Date Created	15 th March 06	Priority	High
Created By	ESI	Technology component	OntoBroker + Extensions;
Description			
A connector for the OntoBroker to access the Tec.Manager has to be provided.			
Relation to prototype			
<ul style="list-style-type: none"> • Tec.Manager is accessed via data service as part of the CAE/CAT Integration Prototype PM24. 			
Requested functionality			
<ul style="list-style-type: none"> • OntoBroker uses the data service provided by Tec.Manager to retrieve data. • The connector can be used to encode the “syntactic integration” in form of rules manually. • The connector reflects the query-mechanism of parameterized queries that the Tec.Manager currently provides. • If technically appropriate (required Interfaces are there, etc.) the Tec.Manager Schema has to be published onto the ontology server (OntoBroker). 			
Validation			
<ul style="list-style-type: none"> • Predefined queries going to Tec.Manager via the Semantic Layer. 			
Assumptions			
<ul style="list-style-type: none"> • Expressivity of Semantic Layer is capable to describe models “hosted” by Tec.Manager. • Tec.Manager data service is sufficient interface. 			

7.2.5 Additional Connectors II

Name	SimManager-Connector		
Application Activity	Auto		
Prototype(s)	CAE/CAT Integration Prototype PM24-30 (extension of PM24 prototype until PM30)		
Date Created	15 th March 06	Priority	Medium
Created By	MSC	Technology component	OntoBroker + Extensions;
Description			
A connector for the OntoBroker to access the SimManager has to be provided.			
Relation to prototype			
<ul style="list-style-type: none"> • SimManager is accessed via data service as part of the CAE/CAT Integration Prototype PM24-30. 			
Requested functionality			

<ul style="list-style-type: none"> • OntoBroker uses the data service provided by SimManager to retrieve data. • The connector can be used to encode the “syntactic integration” in form of rules manually. • The connector reflects the query-mechanism that the SimManager provides. Its language is <ul style="list-style-type: none"> ○ EL (Expression Language), or ○ SPARQL. • If technically appropriate (required Interfaces are there, etc.) the SimManager Schema has to be published onto the ontology server (OntoBroker).
Validation
<ul style="list-style-type: none"> • Predefined queries going to SimManager via the Semantic Layer should give the same result as the same (in case of SPARQL query-mechanism) or semantically corresponding (EL query-mechanism) queries to the SimManager directly.
Assumptions
<ul style="list-style-type: none"> • Expressivity of Semantic Layer is capable to describe models “hosted” by SimManager (already proved by Auto-1 Ontology Prototype PM12). • SimManager data service is a sufficient interface.

7.2.6 Additional Connectors III

Name	“Representative”-Connector for Product Data		
Application Activity	Aero		
Prototype(s)			
Date Created	15 th March 06	Priority	High
Created By	BAE	Technology component	OntoBroker + Extensions;
Description	<p>A connector for the OntoBroker to provide access to some form of “simple data source” has to be provided. This can be either a spreadsheet-format or XML-files. The Aerospace cost estimation scenario is based on “representatives” instead of PSEs. That implies a certain degree of freedom regarding the sources. They have to be able to capture product data in a at least semi-structured way.</p>		
Relation to prototype			
Requested functionality	<ul style="list-style-type: none"> • OntoBroker uses connector to access “simple source” for instance data • The connector can be used to encode the “syntactic integration” in form of rules manually. • For accessing non-relational data some meta information about it is necessary (in a relational database this is given by the SQL schema). E.g. it is important to know, if something should be treated as a relation or an attribute ⁹. Therefore annotations related to the structure of such data (e.g. semi-structured data (XML), Excel sheets) are required. 		
Validation	<ul style="list-style-type: none"> • Implementation of non-simplistic product data schema into source and existence of corresponding ontology. 		
Assumptions			

⁹ Relation: leading to other objects; attribute: leading to data.

•

7.2.7 Knowledge-Services Support

Name	Semantic Support for Knowledge-Services		
Application Activity	Aerospace		
Prototype(s)			
Date Created	15 th March 06	Priority	Medium/Low
Created By	MSC	Technology component	OntoBroker + Extensions;
Description			
Ability of knowledge service to access ontologically described data: Knowledge services should be enabled to use a set of “semantically pre-processed data sources”.			
Relation to prototype			
Requested functionality			
To be defined. Possible scenario: Selection of “semantic source”, definition of query or queries, deployment to knowledge service			
Validation			
Assumptions			

7.2.8 Query Semantics

Name	preservation of query semantics		
Application Activity	Auto (to some degree Aerospace)		
Prototype(s)	CAE/CAT Integration Prototype PM24, CAE/CAT Integration Prototype PM24-30 (extension of the former one)		
Date Created	15 th March 06	Priority	High
Created By	MSC	Technology component	OntoBroker + Extensions;
Description			
The semantics of queries going to the Semantic Layer (with OntoBroker as core component) have to be preserved when external Systems are accessed.			
Relation to prototype			
<ul style="list-style-type: none"> • An OntologyService query interface – with OntoBroker behind – to be used by a SimManager client is part of CAE/CAT Integration Prototype PM24. • Tec.Manager is accessed via OGSA-DAI data service as part of CAE/CAT Integration Prototype PM24. • A SimManager query interface to be used by an OntoBroker connector is part of the CAE/CAT Integration Prototype PM24-30. 			

Requested functionality			
	<ul style="list-style-type: none"> The Tec.Manager data service connector and the representation of the model hosted by Tec.Manager should support a semantic access to Tec.Manager preserving the semantics of the queries. 		
Validation			
	<ul style="list-style-type: none"> Predefined queries going to Tec.Manager directly and via the Semantic Layer, predefined queries going to SimManager directly and via the Semantic Layer. 		
Assumptions			
	<ul style="list-style-type: none"> Expressivity of Semantic Layer is capable to describe models “hosted” by Tec.Manager; Tec.Manager data service is sufficient interface. Expressivity of Semantic Layer is capable to describe models “hosted” by SimManager; Sim.Manager query interface is sufficient. 		

7.2.9 Semantic Integration Security

Name	Security infrastructure for ontology service		
Application Activity	Auto (and later Aerospace)		
Prototype(s)	CAE/CAT Integration Prototype PM24-30 and later		
Date Created	15 th March 06	Priority	
Created By	MSC	Technology component	OntoBroker + Extensions;
Description	<p>The OntologyService accesses external sources such as PSEs. On the long-term run there have to be efficient capabilities to secure those sources and authenticate users. However, the strategy is currently not clear:</p> <ul style="list-style-type: none"> Authenticate on user-level as if users connect directly Use “technical user” and manage rights externally (manage rights for Ontology Service itself) 		
Relation to prototype			
Requested functionality			
	<ul style="list-style-type: none"> Authentication User-credentials Connection security: <ul style="list-style-type: none"> encryption at transport layer robustness against distorted connections 		
Validation			
Assumptions			

7.2.10 Management of References for Bulk Data

Name	Management of References for Bulk- or Mass-Data
Application Activity	Auto (and later Aerospace)

Prototype(s)	(not before PM30)		
Date Created	15 th March 06	Priority	
Created By	MSC	Technology component	OntoBroker + Extensions;
Description	<ul style="list-style-type: none"> The OntologyService is not suited for delivering mass data directly. If static references – stored as meta data – are not sufficient, there is a need for dynamically created references to mass data, together with needed data transformations and replications. This is an advanced requirement (wish list)! 		
Relation to prototype	<ul style="list-style-type: none"> none so far 		
Requested functionality	<ul style="list-style-type: none"> delivery of dynamically created references to mass data, which may be <ul style="list-style-type: none"> converted, and/or replicated caching mechanism for references 		
Validation	<ul style="list-style-type: none"> 		
Assumptions	<ul style="list-style-type: none"> OntologyService is not suited for delivering mass data directly (through the OntoBroker). 		

7.2.11 Management of Schema Changes

Name	Management of Schema-Changes		
Application Activity	Auto (and later Aerospace)		
Prototype(s)	CAE/CAT Integration Prototype PM24-30 and later		
Date Created	15 th March 06	Priority	Medium
Created By	DWD	Technology component	OntoBroker + Extensions;
Description	<p>The Semantic Layer needs to be aware of schema-changes in the integrated sources. This should be handled in a semi-automatic way, possibly proposing changes that are likely.</p>		
Relation to prototype	<ul style="list-style-type: none"> The CAE/CAT Integration Prototype PM24 works with a fixed Schema. Depending on the way the Schema is reflected (either manually or via some schema-import functionality) a simple case of schema-change might be included. 		
Requested functionality	<ul style="list-style-type: none"> Notification: the notification itself is likely to be limited to the indication that “something has changed” while the identification of the change itself is part of a separate process. “Push/Pull”: Depending on the way the source is integrated and how the schema is published 		
Validation			

<ul style="list-style-type: none"> • Handling of a set of basic (predefined) changes for integrated sources: <ul style="list-style-type: none"> ○ Concept added (“concept” referring to the equivalent on the source-side such as a database table or some other “concept-like” entity) ○ Concept removed ○ Concept renamed ○ Property added ○ Property removed ○ Property renamed
Assumptions
<ul style="list-style-type: none"> • The representation of the external schema is not generated completely manually

7.2.12 Thesaurus-Support for Product Discovery

Name	Thesaurus-Support for Product Discovery		
Application Activity	Meteo		
Prototype(s)			
Date Created	15 th March 06	Priority	
Created By	DWD	Technology component	
Description	The first level of application of semantic technologies in the Meteo-Area is terms of a thesaurus-approach.		
Relation to prototype			
Requested functionality	<ul style="list-style-type: none"> • Provide Thesaurus based on existing models (Sweet, CF, ISO 19115 to be evaluated) • Extend Thesaurus with multilanguage functionality • Enhance the discover service using the thesaurus database • Extend the metadata maintenance tools with thesaurus database 		
Validation			
Assumptions			

7.2.13 Simple Online Ontology-Maintenance

Name	Simple Online Ontology-Maintenance		
Application Activity	Meteo		
Prototype(s)			
Date Created	15 th March 06	Priority	
Created By	DWD	Technology component	
Description			

The first level of application of semantic technologies in the Meteo-Area is terms of a thesaurus-approach. It requires simple means of online-maintenance of ontologies (without sophisticated locking mechanisms etc.)			
Relation to prototype			
Requested functionality			
<ul style="list-style-type: none"> Simple maintenance functions without data-base like multi-user capabilities 			
Validation			
Assumptions			

7.2.14 Additional Connectors IV

Name	Interface of ESI Visual Composer to PDM systems		
Application Activity	Auto		
Prototype(s)	Auto-3 Prototype PM24-30		
Date Created	15 th March 06	Priority	Medium
Created By	ESI	Technology component	OntoBroker + extensions
Description	<p>As part of its functionalities, Visual Composer maintains the link between the CAD world and the CAE world, allowing a CAE engineer to load a CAD assembly coming from a PDM system into his environment and to update this assembly when new versions are available. Therefore, Visual Composer must be interfaced with all the PDM systems of the customers. The description of CAD assembly may vary from one PDM to another, which means that an adapter needs to be developed for each PDM, to be able to load CAD assemblies into Visual Composer. The Visual Composer representation of CAD assemblies (that is customizable by the user) is then the reference ontology that the PDM systems need to comply with.</p>		
Relation to prototype			
<ul style="list-style-type: none"> In most CAE processes, the first step consists in loading a CAD representation of the studied part into the Problem Solving Environment. 			
Requested functionality			
<ul style="list-style-type: none"> provide a module that would automate as much as possible the development of the adapters using semantic rules. 			
Validation			
<ul style="list-style-type: none"> Loading of CAD assemblies coming from the target PDM into Visual Composer 			
Assumptions			
<ul style="list-style-type: none"> The PDM has at least one human-readable (ASCII, xml) export format 			

7.3 Outlook after PM30

Besides this document having the focus just until PM30 a short outlook onto things possibly coming thereafter.

The Auto-1 Ontology Prototype PM12 has shown some performance problems. These problems will probably arise again with the new CAE/CAT Integration Prototype PM24-30. Performance may be increased – amongst other measures – by *query optimisation*¹⁰ at the ontology service: thereby F-Logic rules leading to multiple queries to external systems may be improved, ideally by automated rewriting of them.¹¹ Going this path could be interesting *after* PM30.

Besides this there are improvements regarding many of above requirements possible after PM30, because the chosen pre-PM30 solutions mostly will be minimal ones – with respect to fulfilling the requirements –, which in general could be improved and/or extended.

¹⁰ This could be the name of the corresponding requirement.

¹¹ There is another query oriented optimisation possibility at the *clients* of the ontology service: if navigation often goes the same paths, some contents may be queried in advance and/or cached at the client side (but this is outside the scope of this document).

8 Implementation and test plans for PM 24 and PM 30

8.1 Major Milestones

- SPARQL interface for OntoBroker (first version): end of May;
- OGSA-DAI activities at OntoBroker for SPARQL querying (first versions): mid of June:
 - for OntoBroker SPARQL query interface,
 - for accessing to be integrated data sources;
- interface to an external PSE on the OntoBroker-side (first version, in cooperation with LMS): end of June.

For a more detailed description of ontology requirements with their relation to planned demonstrators see Section

8.2 Implementation Plan until PM24

The implementation plan is a compilation of information from Section 7.2 Requirements plus the planned timelines.

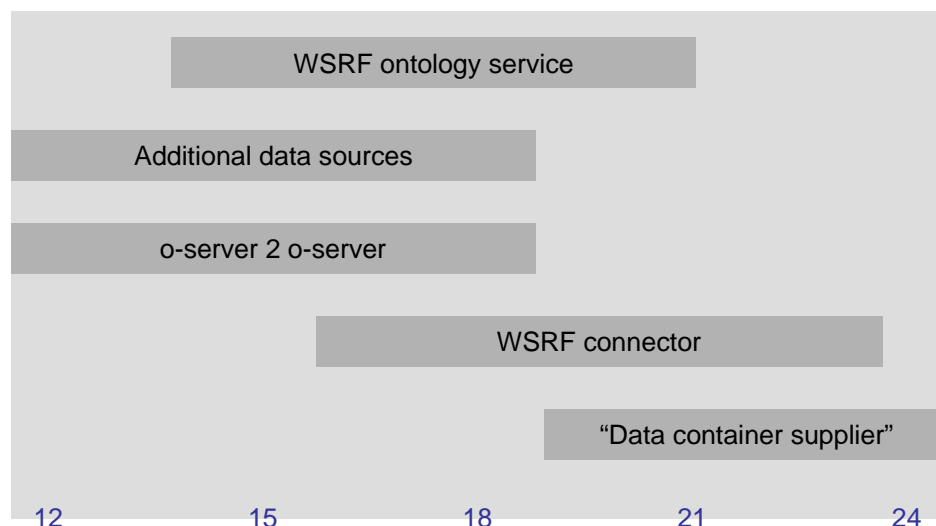


Fig. 3: Implementation plan for basic components

8.3 Implementation after PM24

For the time after PM24 it is not possible to give a concrete implementation plan now. But it is possible to list the mainly affected requirements. These are

- 7.2.9 Semantic Integration Security,
- 7.2.11 Management of Schema Changes.

We will start working at these requirements after PM24, but it is not plannable, how far we will have gone until PM30. Moreover these works surely will outreach PM30.

8.4 Test Plan

The test plan is a compilation of information from Section 7.2 Requirements.

Requirement	Validation
7.2.1 OGSA-DAI based Ontology-Service	
7.2.2 SPARQL-Interface	Comparison of the results of a set of predefined SPARQL and F-Logic queries
7.2.3 Capabilities of Representation (for Product Data)	
7.2.4 Additional Connectors I	Predefined queries going to Tec.Manager via the Semantic Layer.
7.2.5 Additional Connectors II	Predefined queries going to SimManager via the Semantic Layer should give the same result as the same (in case of SPARQL query-mechanism) or semantically corresponding (EL query-mechanism) queries to the SimManager directly.
7.2.6 Additional Connectors III	Implementation of non-simplistic product data schema into source and existence of corresponding ontology.
7.2.7 Knowledge-Services Support	
7.2.8 Query Semantics	Predefined queries going to Tec.Manager directly and via the Semantic Layer, predefined queries going to SimManager directly and via the Semantic Layer.
7.2.9 Semantic Integration Security	
7.2.10 Management of References for Bulk Data	
7.2.11 Management of Schema Changes	Handling of a set of basic (predefined) changes for integrated sources: <ul style="list-style-type: none"> ▪ Concept added (“concept” referring to the equivalent on the source-side such as a database table or some other “concept-like” entity) ▪ Concept removed ▪ Concept renamed ▪ Property added ▪ Property removed ▪ Property renamed
7.2.12 Thesaurus-Support for Product Discovery	
7.2.13 Simple Online Ontology-Maintenance	
7.2.14 Additional Connectors IV	Loading of CAD assemblies coming from the target PDM into Visual Composer

9 Conclusions

Within the first phase of SIMDAT the applicability of semantic technologies in distributed environments for the discovery and integration of resources has been proved. Along with the basic idea of SIMDAT the focus has been on practical applications and implementations driven by the requirements of the application areas. The first feedback loop has helped to generate a strategy for a systematic extension of the infrastructure and a horizontal transfer of the approaches.

The approach for the second phase of SIMDAT, combining grid-middleware with semantic technologies seems to be very promising.

Current impact on the Grid technology sector

Within the first phase of SIMDAT we have achieved:

- Models as well as a technical infrastructure for the annotation, registry and discovery of services;
- SOA with consideration of standards (WSDL, WSI) and standard tools (Tomcat, Eclipse, Axis);
- integration of different data sources with
 - different query languages,
 - different data model description languages, and different data models;
- translation of a path oriented query language into F-Logic queries;
- shown a running distributed client and target system with client and target running at different hosts

Together with the application areas we could prove the industrial relevance of the underlying approaches and the need for the kind of infrastructure we established. We could show that ontologies have the potential to play a major role for the future development of grid technologies – not just as a research topic but as a solution to industrial problems. The cooperation with the application areas and the other technology areas allows us to beyond the core concepts of semantic technologies in distributed environments and envisage the challenges of real-world applications including aspects of scalability and maintainability for example. In areas, the service discovery as well as the interoperability of data resources we emphasized a complete approach covering different aspects of relevance for professional users.

Expected impact on the Grid technology sector

In the coming phase of the project we focus on the generalization of the infrastructure provided. We'll provide important building blocks for industrial solutions supporting advanced solutions like the flexible encapsulation of local resources based on distributed ontology servers and the enhancement of established grid middleware by semantic technologies. There is no comparable infrastructure available which consequently exploits the advantages of semantic technologies while on the other building on data grid technology.

Even though we rely on prototypical developments those activities are expected to significantly future solutions in industrial projects for data and information management in distributed environments. Service oriented architectures as such have recently gained much importance. Innovative approaches ready to be transferred into real-world solutions are required in the next years rather than the next decades. Core problems of service oriented architectures based on a large number of resources with a high degree of autonomy can be targeted with the technology extended, developed and applied in SIMDAT based on ontologies. That's why SIMDAT will have a considerable impact on future solutions and on the grid sector as such.

10 Table of Figures

Fig. 1: OntoStudio schema import	12
Fig. 2: Rule based integration	13
Fig. 3: Architecture of Ontology Prototype	13
Fig. 4: Detailed view on the implementation aspects of adaptors and bus	15
Fig. 5: “Dependency Layers” for rules	17
Fig. 6: Remote Ontology Import	20
Fig. 7: Schema of the diff-component	23
Fig. 8: Global-as-View versus Local-as-View	26
Fig. 9: Concept of articulating ontology [Comp2002]	30
Fig. 1: OWL-S top level concepts	34
Fig. 2: WSMO core elements according to [WSMO]	35
Fig. 3: Implementation plan for basic components	49
