Integration with Ontologies
Conference Paper WM2003, April 2003, Luzern

author: Andreas Maier[1] (maier@ontoprise.de)
co-authors: J. Aguado[2] (jessica@miramon.net)
A. Bernaras[2] (amaia@miramon.es)
I. Laresgoiti[3] (lares@labein.es)
C. Pedinaci[2] (carlos@miramon.net)
N. Peña[3] (npena@labein.es)
T. Smithers[2] (tim@miramon.net)

Abstract: One of today's hottest IT topics is integration, as bringing together information from different sources and structures is not completely solved. The approach outlined here wants to illustrate how *ontologies* [Gr93] could help to support the integration process.
The main benefits for an ontology-based approach are

- the ability to picture all occurring data structures, for ontologies can be seen as nowadays most advanced knowledge representation model
- the combination of deduction and relational database systems, which extends the mapping and business logic capabilities
- a higher degree of abstraction, as the model is separated from the data storage
- its extendibility and reusability

First we will give a motivation for our approach and build the requirement specification (1.). After describing the foundations (2.) we will introduce our ideas about new software tools supporting the ontology-based integration (3.) and present a case study (4.), where the ontology-based integration is going to be realized - both from the view of the software producer[1] and of the teams[2] [3]. Closing remarks to limitations of our approach and to related publications will complete this work.

1. Motivation for an Ontology-based Approach

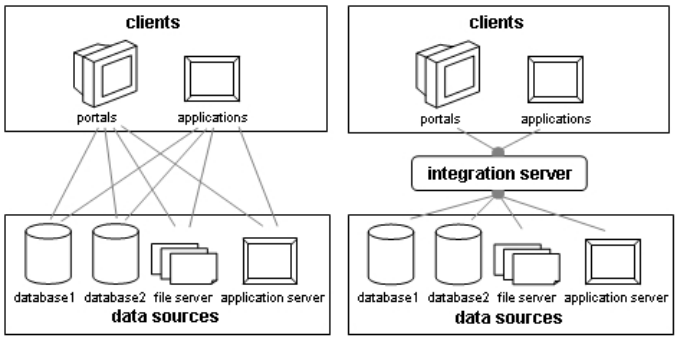
Today's users and IT professionals have high expectations towards software applications:

- they want to access the content they need
- this content must be accurate and free of redundancy
- the application must be intuitive and easy to use
- the application must be reusable and extendable
- the application must be implemented in a short and inexpensive way and within the current IT legacy environment

To meet these expectations, the content has to be identified from the different sources (i.e. databases, applications, XML-Files, unstructured text files …), and then to be integrated. But this means not building only connectors [Kr99] between applications, because

[1] ontoprise GmbH
[2] Parque Tecnolgico de San Sebastian, Spain
[3] LABEIN, Zamudio, Spain

syntactical incompatibilities could be reduced by approaches like SQL [KK01] or XML [An03]; nor it's only tying up diverse data sources and displaying them on a common interface (Picture 1 left).

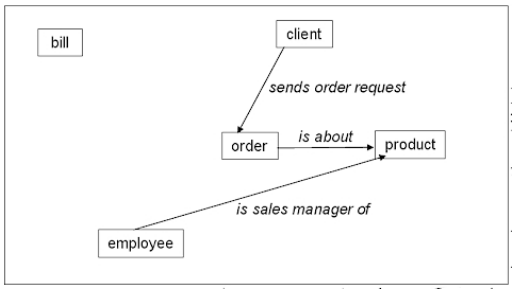


Picture 1: point-to-point-connections vs. a "real" integration solution

The goal of integration is to consolidate distributed information intelligently, free of redundancy, processed and operated by the right business logic to deliver the appropriate and condensed answer and offer the end user a simple access to it, without him needing knowledge about the underlying data structures (Picture 1 right). We believe that with ontologies there's now a model at hand to fit for this goal.

1.1. Introducing Example

A typical integration scenario found in large organisations is the management of product data and contacts (clients, suppliers, employees), which proves to be very difficult. These information lie widespread in different departures and there in different sources like ERP/PPS[4]-systems, CRM[5]-applications [Sc00] [UB03], databases, email programs, documents, organizers, etc., - often redundantly.



Picture 2: an "integration" ontology

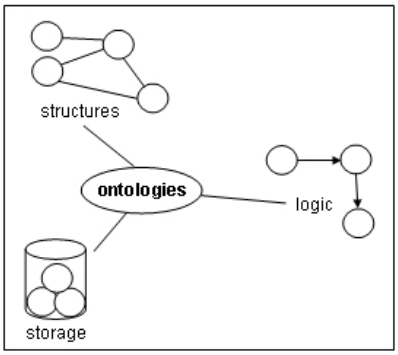
[4] ERP=Enterprise Resource Planning, PPS= Production Planning and Scheduling
[5] CRM=Customer Relationship Management

An ontology (Picture 2) could now catch up the different concepts that we want to integrate: **clients** (i.e. from a CRM-application), **orders**, **products** (i.e. from a PPS-System) and **employees** (i.e. from an intern "telephone and birthdays" list). A fifth concept, **bill**, has not been taken into consideration yet by the existing systems; we will generate its instances automatically by a rule (2.4).

1.2. Defining the Requirements

In our integration process we then have

- to picture all existing data **structures** (*requirement 1*), which can be simple table structures up to complex hierarchical structured data with deep inheritance,
- to **map and merge** these schemas (*requirement 2*),
- to define the **logic** for the whole new application (*requirement 3*) (hereby we will be supported by deductive inference mechanisms) and
- to provide a performant data **storage** for the information (*requirement 4*).



Picture 3: **ontologies meet the requirements**

In our view ontologies are the best representation model to meet these requirements (Picture 3). In the next chapter (2.) we want to prove this statement.

2. Foundations: Enabling the Ontology-based Integration

The foundations talked about next – knowledge representation, mapping, deductive logic and databases – are not new for themselves. It's their interaction what makes it necessary to shortly describe them. Thereby we will especially go into their impact on the ontology-based integration.

2.1. Requirement 1: Picture all Data Structures

In [Ma01] we compared several knowledge representation models and discussed the advantages and weaknesses of them. As a conclusion we found that ontologies are the most advanced model of all of them, summing up most of the qualities of the others:

- Like **Taxonomies** [Pe89], ontologies are able to picture *hierarchies*.

- Like **Thesauri** [Me95], **Semantic Nets** [Ho86] and **Topic Maps**[6] [PH02], ontologies contain *relations*. With them, complex contexts can be modelled and visualized in nets. *Linguistic contexts* (i.e. multilingualism or synonym relations), terminologies and classifications can be described, through which the semantic of the integration solution is increased [An01].
- Like the **EntityRelationship-Model** (ER) [BCB91] and unlike the others mentioned above, ontologies have a *data model* distinguishing schema information from facts. This is essential for storing the facts (Requirement 4: Provide a Data Storage). As relational databases do not provide an object model, they have some difficulties in picturing taxonomies and must define primary keys themselves. Every ER-model can be transformed in an ontology and, with some expense and limitations, vice versa.
- As an **object based model**, ontologies support *inheritance* and *multiple inheritance* of attributes.

2.2. Logic Models

With the ability to define mapping and deductive rules respectively axioms[7], logic brings in an important impact for the integration. Logic models (i.e. **Prolog** [Sp96], **Datalog** [DL91], **F-Logic** [KL90] [An02], **Description Logic** [Ba02]) have to be seen complementary to ontologies.
Especially F-Logic[8] [AL03] acts as a bridge between the model and logic, because it covers the ontological information as well as the rules. It can also be used to query the system similarly to SQL.
Logic models help us

- to connect the different data sources by **mapping** or **merging** rules
- to easily build a deductive "**business logic"** upon our integrated information base[9]
- to check the consistence of the knowledge base [AS92a].

2.3. Requirement 2: Mapping and Merging

If only integrating databases, an ontology-based mapping would not seem to be necessary at first. But in comparison to pure database mappings, which could be used instead, the ontological approach leads to a higher degree of abstraction, connecting not only primary keys and table rows, but working on a more conceptual level.
Before starting the mapping procedure the structures respectively the schemas have to be imported into the ontology. For thus, an ontology modelling tool must provide various

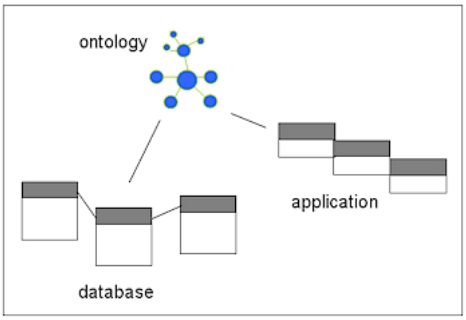
---

[6] As the dates of the references (6.) show, Taxonomies, Thesauri, Semantic Nets and the ER-model are comparatively old models. Topic Maps are the newest of them and may merge with ontologies.
[7] As a special type of axioms, constraints help us to define restrictions within our system.
[8] Comparing the mentioned logic models, we see F-Logic as the most capable of them; Prolog is rather a programming language and contains artefacts, Description Logic misses instance reasoning and Datalog isn't expressive enough.
[9] Deduction can also be very effective in a stand-alone application, but much more in an integration scenario. By consolidating applications and their underlying processes often new contexts are created and require a new business logic on top.

schema import filters for different formats (i.e. for all relevant commercial databases). Such a tool must also support the fundamental mapping types

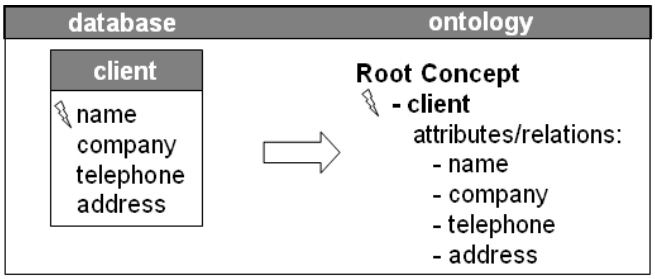
- concept-to-concept mapping
- attribute-to-attribute mapping
- attribute-to-concept mapping as well as
- conditions and constraints[10] on the mapping rules (which is not explained further here)

2.3.1. Schema Import



Picture 4: Importing diverse schemas

The equivalent to the schema import described next is the schema export (2.5), that we will need for *requirement 3* below. Beneath other schema imports (i.e. for formats like RDF[11] or DAML[12]), the SQL import plays a very important role. As the commercial databases[13] usually a different syntax of SQL, different imports for each database are needed[14].



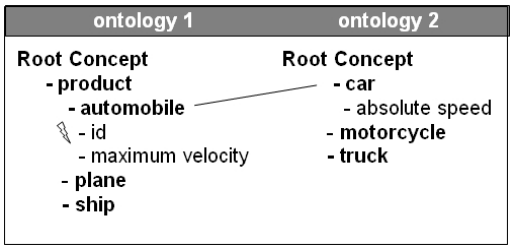
[10] i.e. unit conversions
[11] http://www.w3.org/RDF/
[12] http://www.daml.org/
[13] i.e.: MSSQL Server (Microsoft), DB2 (IBM), Oracle, mySQL
[14] An SQL import has already been realized in *OntoEdit*, ontoprise's ontology modelling tool. For detailed information and screenshots please read the product tutorial at http://www.ontoprise.de/documents/tutorial_ontoedit.pdf.

Picture 5: Importing a database table

After an import of a database table, this one is embedded as a concept into the concept taxonomy. The former primary key from the attribute "name" has moved to the object id of the concept "client". Tables are interpreted as objects or concepts, as they usually contain information about a distinct entity. Rows typically describe attributes of that entity and are coherently interpreted as attributes of concepts.

2.3.2. Concept-to-Concept Mapping

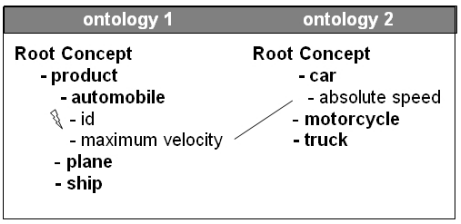
An ontology mapping process[15] is very similar to pure database respectively XML-mapping [Bo01]. In each case the two schemas, which are going to be mapped, are displayed in vertical rows parallel to each other (picture 6: concept-to-concept-mapping). If two concepts of two different sources contain the same type of information, a concept-to-concept mapping can be drawn (i.e. "automobile" and "car").



picture 6: concept-to-concept-mapping

2.3.3. Attribute-to-Attribute Mapping

An attribute-to-attribute mapping connects two attributes, stating that these contain the same information (i.e. "maximum velocity" and "absolute speed") (picture 7: attribute-to-attribute-mapping). A previous concept-to-concept mapping is prerequisite for that.

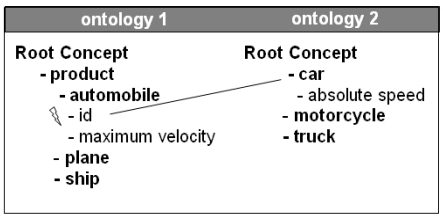


picture 7: attribute-to-attribute-mapping

[15] *OntoEdit* contains OntoMap, a mapping tool for ontologies. More information can be found at http://www.ontoprise.de/documents/tutorial_ontoedit.pdf.

2.3.4. Attribute-to-Concept Mapping

Another important aspect is the mapping of table rows (attributes) that are represented as concepts in other formats. The attribute-to-concept mapping shown below states that the primary key "id" of "automobile" in connected to the object id of "car" (picture 8: attribute-to-attribute-mapping).



picture 8: attribute-to-attribute-mapping

2.4. Requirement 3: Deductive Logic

An often asked question is: "Why using logic? Didn't databases solve all problems decades ago?"
On the one hand applications with lots of logical dependencies (i.e. configuration or variant management systems, solutions representing extensive knowledge domains, expert systems) can be realized much better with rule-based systems.
On the other hand deductive logic reduces complexity. It's a difference, if you ask

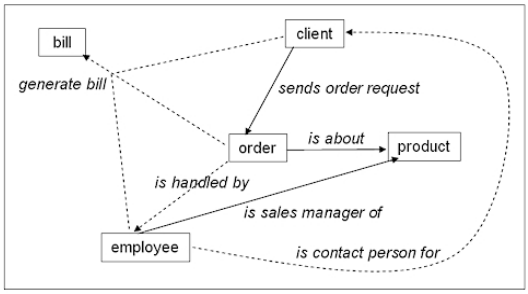
- "Who is the contact person of client 'Smith'?" or
- "Who is the employee that handles the orders of the product, that client 'Smith' has ordered?" (
- Picture 9: **using deductive rules**).

As this is just a small example, in really complex contexts with many relations between the concepts of the ontology the effort and complexity to realize in SQL quickly gets too high.
As a third reason, the user doesn't need to know the underlying data structures. For example he only knows, that he can ask for "contact persons of clients", and not the whole conceptual structure that lies behind this question.

2.4.1. Rule Modelling

It's an often underestimated fact for today's rule-based-systems in commercial environments that users do not want to encode these rules by hand, just as they dislike to write SQL queries. Therefore we will propose our idea of a visual rule editor in (3.1).

After having tied together the different data sources (2.3.1-2.3.3), we want to illustrate the extendibility that deductive logic rules offer to applications. Our basic ontology (Picture 2: an "integration" ontology) has been enriched by three simple rules to expand the knowledge base (
Picture 9: **using deductive rules**):

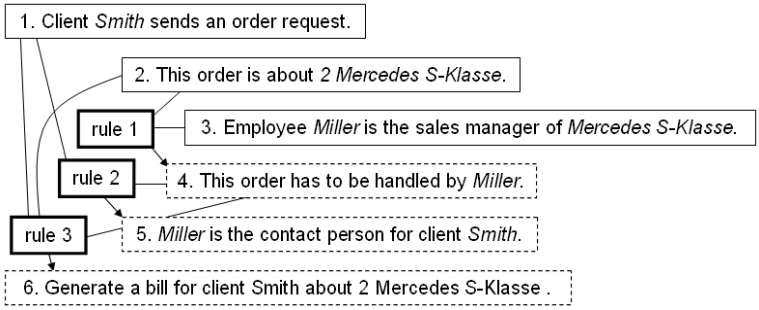


Picture 9: using deductive rules

- If an order is about a product and an employee is the sales manager of this product, this order **is handled by** the employee. (*rule 1*)
- If a client sends an order request, which is handled by an employee, this employee is the **contact person for** the client. (*rule 2*) As we see, *rule 2* is based on *rule 1*.
- If a client sends an order request, which is handled by an employee, then **generate a bill** with all available information: client, name and quantity of the product, the responsible employee … (*rule 3*)

2.4.2. Inferencing

By an inferencing process[16] the rules are applied to the given facts (1.,2.,3.) and extend the knowledge base by the newly created facts (4.,5.,6.). picture 10 visualizes this process.



picture 10: inferencing graph

In analogy to the graphical rule editor (3.1), there has to be a visual rule debugger for the modeller to show the outcome of his rules. In (3.2), we propose our idea of a visual rule debugger.

[16] Inferencing has been realized in *OntoBroker*^{TM}, ontoprise's core ontology server. Please read the product tutorial at http://www.ontoprise.de/documents/tutorial_ontobroker.pdf to get detailed information.

After the inferencing you can decide whether you want to materialize the attained new facts into the data storage (2.5.), or to keep them only virtually in the inference server.

2.5. Requirement 4: Provide a Data Storage

As a name for a mainly unchanging pool of information the terms *Knowledge Base* or *Repository* are often used. We prefer to say *data storage* instead, emphasizing that in integration solutions the content is permanently changing.
For maintenance reasons, the data itself should be kept only once, preferably in the origin application. If this application isn't able to query, we propose migrating it to a database. Although the ER-Model has its weaknesses (2.1), we suggest using relational databases as storage because of its widely spread and mature solutions. In comparison to other repositories there's no alternative concerning performance and compatibility. Therefore we need an SQL-Export for creating the database schema out of the ontology.

3. Introducing a Toolbox for the Ontology-based Integration

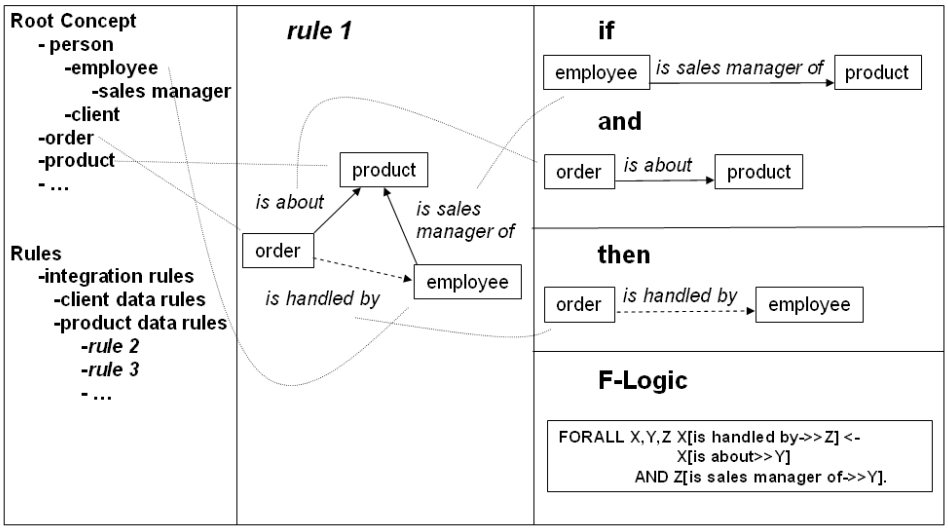
As we found in (2.) and (3.), the following components are needed within an ontology modelling environment [AS2002], for meeting the requirements for an integration solution:

- a core modelling component for concepts, attributes, relations, instances, multilingual representations and domain entries
- a schema import and export (2.3.1, 2.5) supporting various formats, particularly SQL
- a mapping tool (2.3.2-2.3.4)
- a rule editor (2.4.1)
- a rule debugger (2.4.2)

The first three points have been realized in recent modelling tools more or less. For the last two points we will introduce a rough idea of visualization, not covering all functionalities coming up.

3.1. Visual Rule Editor

In our example, *rule 1* (2.4.1) is a composition relation rule, connecting the three concepts *product*, *order* and *employee*. In our proposal (
picture 11: a visual rule **editor**), a user would select them by drag&drop from a left window, where all concepts are listed in a "is-a"-hierarchy, and move them to the center window.

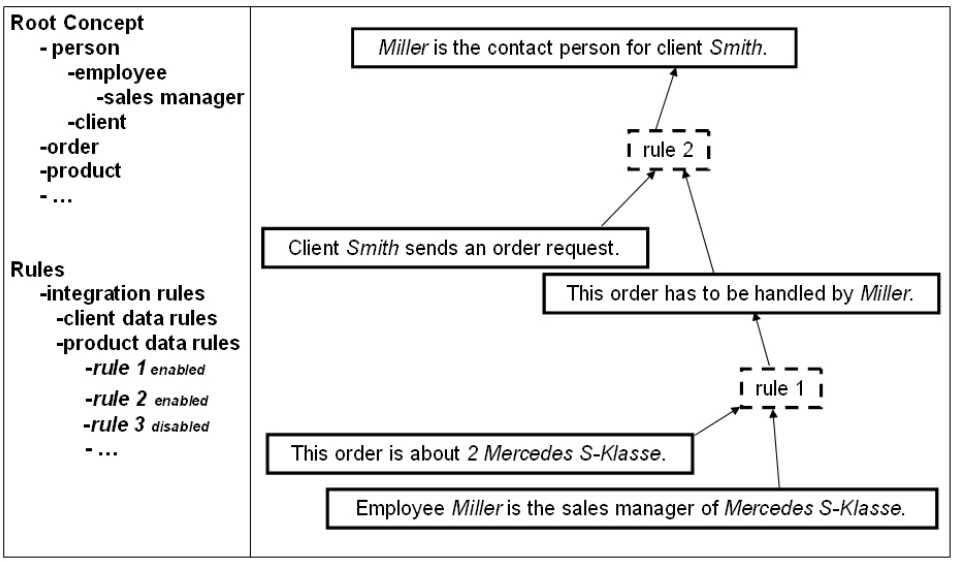


picture 11: a visual rule editor

There the modelled relations (*is about*, *is sales manager of*, *is handled by*) will appear. By moving them by drag&drop into the fields on the right side (*if*, *then*), you would create the rule shown in F-Logic code below. There you could change the rule also by hand.
In picture 11 the ability to define attribute conditions (i.e. employee.name=”Miller”) or operators (+, -, *, /, NOT, EXISTS, …; i.e. price=quantity*[price per unit]*discount) is missing and has to be added to the draft yet.

3.2. Visual Rule Debugger

The visual rule debugger is an important tool for the IT professional. It’s supposed to support him during the rule modelling phase, showing him the outcome of the rules. Thereby it visualizes the inference process for one selected new fact (picture 12: **a visual rule debugger**). If the concerning rules (*rule 1*, *rule 2*) for this new fact (i.e.: “*Miller is the contact person for client Smith*”) have been enabled in the window on the right side, a graph would appear showing the course of conclusion.

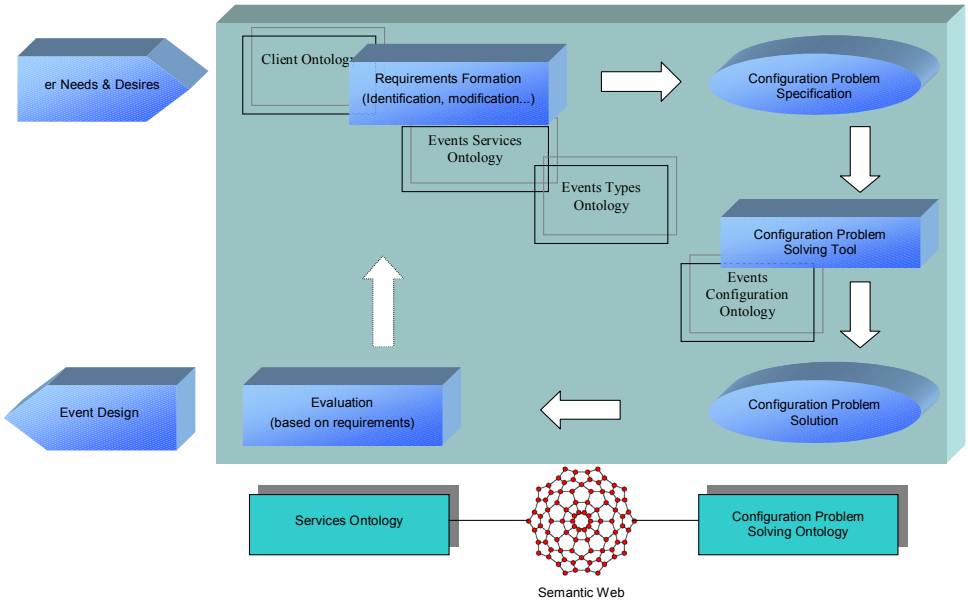


picture 12: a visual rule debugger

4. Case Study: Online Design of Events

In this section we introduce an event design support application in which complex knowledge integration is required. This application, developed within the **OBELIX project (IST-2001-33144),** aims to support event organisers and suppliers in the process of designing events such as symposiums, conferences, exhibitions, workshops or meetings, and where the service provided, i.e., a running event, should be configured and customised to the needs of the type of event and client.

The aim of this application is to develop a Web-based system that will support the collaborative process of designing customised events, thus greatly improving this type of service.



Picture 13: **System Architecture**

(Picture 13: **System Architecture**) presents a view of the system architecture. It shows how the different ontologies needed are integrated so as to provide the desired support of event designers and clients. This system architecture is derived from a Knowledge Level theory of designing, known as KLDE (see [Sm96, Sm98, Sm02a, Sm02b]).

As a Knowledge Level theory, KLDE defines the necessary and sufficient kinds of knowledge needed and generated in any kind of designing, together with the roles each kind of knowledge plays, and the relationships between them. Applying this theory to the design of the event design support system described here, made it possible to quickly and easily identify the kinds of knowledge-based support needed, and the way the different kinds of knowledge needs to be integrated.

Starting with the needs and desires of a client, the event design support system first supports the identification of a set of requirements that, when satisfied, would result in an event design acceptable to the client. It further supports the development of well formed problem specifications that are essentially attempts to operationalise some or all of the requirements. In this event design application, the problems are formed as configuration problems, which are then passed to a configuration problem solver being developed by LABEIN.

The solutions that are returned are then evaluated with respect to the current requirements. And the outcome of this evaluation then leads either to the identification of further or modified requirements, a different operationalisation, or a final design.

To support this process, the event design support system thus integrates a "Client Ontology", to support the identification and construction of the requirements set, an "Event Services Ontology" and "Event Type Ontology" to support the formulation of well formed problems that operationalise effectively some or all of the current requirements, an

"Event Configuration Problem Solving Toolbox", which in turn uses the "Events Configuration Ontology."
Each of these ontologies model different kinds of knowledge. The "Events Types Ontology", contains the different types of events that can be configured by the application. These are defined in terms of resources used, such as networks, computers, rooms, layouts, etc. The "Client Ontology" contains information about type of clients and their needs, and the "Events Services Ontology" refers to considering the design of events as an event service. In the "Configuration Problem Solving Ontology" different types of configuration problems are reflected. In the "Service Ontology" the different kinds of services are reflected. The "Events Configuration Ontology" represents different types of event configurations problems, such as the configuration of layouts in a room assigned to an event or the different resources requested by the client.
These kinds of knowledge are integrated within the system. At the same time, (and again informed by KLDE), relationships can be identified between them. For example, the "Event Service Ontology" is related to both the "Events Types Ontology" and the "Service Ontology". In the same way, the "Events Configuration Ontology" is related to both the "Events Types Ontology" and the "Configuration Problem Solving Ontology".
Thus, in this event design support application we can see clearly the need for effective integration of different ontologies, and how what ontologies are needed and how they should be integrated has been effectively established by applying the KLDE theory of designing.

5. Closing Remarks

Looking at today's available software for ontologies, [NM02] compare different tools distinguishing editors from mappers. As shown above (2.), this must be no opposite[17].

Another mapping mechanism between distributed ontologies[18] is introduced in MAFRA [MMSV02], an interactive, incremental and dynamic framework for ontology mapping.

Of course, integration across different sources depends on the compatibility of their data structures. Just as you can't map Newton's 3-dimensional world by 100% to Einstein's 4-dimensional one, the result will always be an approximation. In the worst case, when ontologies are totally orthogonal to each other, a mapping can be impossible. But should a mapping be feasible, the manual approach described here (2.3) might be supplemented by semi- or full automatic approaches. [MDH02] deliver new findings on this subject.

6. References

[AL03] J. Angele, G. Lausen: Handbook on Ontologies in Information Systems, Springer (editors: S. Staab, R. Studer), 2003/2004

[An01] J. Angele: Semantik und Wissensmodelle, Proceedings Deutscher Internet Kongress, Karlsruhe, 2001

[17] In OntoEdit, there's a modelling component as well as a mapping plug-in.
[18] i.e. in the vision of the Semantic Web

[An02] J. Angele: F-Logic Tutorial, http://www.ontoprise.de/documents/tutorial_flogic.pdf, Karlsruhe, 2002

[An03] J. Angele: XML reicht nicht aus, Springer, 2003

[AS92a] J. Angele, R. Studer: Konsistenzprüfung in wissensbasierten Systemen, Universität Karlsruhe, 1992

[AS92b] J. Angele, R. Studer: Inferenzmechanismen in wissensbasierten Systemen, Universität Karlsruhe, 1992

[AS2002] J. Angele, Y. Sure: Evaluation of Ontology based Tools, Siguenza, 2002

[Ba02] F. Baader: The description logic handbook: theory, implementation and applications Cambridge University Press, 2002

[BCB91] C. Batini, S. Ceri, C. Batini: Conceptual Database Design: An Entity-Relationship Approach, Addison Wesley Publishing Company, 1991

[Br92] K. Brinker: Linguistische Textanalyse. Eine Einführung in Grundbegriffe und Methoden, E. Schmidt, Berlin

[Bo01] R. Bourret: Mapping DTDs to Databases, 2001, at http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html

[DL91] M. Dahr, K. Lautenbach: Towards a formal theory of datalog nets, Koblenz, 1991

[Gr93] T. R. Gruber: A translation approach to portable ontology specifications; Knowledge Acquisition; 1993;

[Ho86] H. Hoffmann: On the visualization of design notions, of notion instantiations, and of structural relationships in a design data base realized as a semantic net, Darmstadt, 1986

[KL90] M. Kifer, G. Lausen: F-Logic - A Higher-Order Language for Reasoning about Objects, Inheritance, and Scheme, Freiburg, 1990

[KK01] K. Kline, D. Kline: SQL in a Nutshell, O'Reilly, 2001

[Kr99] D. Kreuz: Formale Semantik von Konnektoren, Techn. Univ. Hamburg, Dissertation, 1999

[Ma01] A. Maier, Vergleich von Wissensmodellen, December 2001

[MDH02] A. Doan, J. Madhavan, P. Domingos, A. Halevy: Learning to Map between Ontologies on the Semantic Web, Honolulu, 2002

[Me95] N. Meder, Konstruktion und Retrieval von Wissen: "Thesauri als Terminologische Lexika", Weilburg, 1995

[MMSV02] A. Mädche, B. Motik, N. Silva, R. Volz: MAFRA — A Mapping FRAmework for Distributed Ontologies in the Semantic Web, FZI - University of Karlsruhe, 2002

[MS00] A. Maier, H.-P. Schnurr: Wissensmanagement bei einer Investmentbank, 2000

[NM02] N. Noy, M. Musen: Evaluating Ontology Mapping Tools - Requirements and Experience, Stanford University, 2002

[Pe89] Christof Peltason: Wissensrepräsentation für Entwurfssysteme : d. Behandlung von Klassifikation und Taxonomie, Berlin, Techn. Univ., Diss., 1989.

[PH02] J. Park, S. Hunting: XML Topic Maps, Addison-Wesley Professional, 2002

[Sc00] W. Schwetz: Customer Relationship Management, Gabler, 2000

[SM02] Y. Sure, E. Moench: Semantic Miner: Smarter Knowledge Retrieval. Poster Session at: First International Semantic Web Conference 2002 (ISWC 2002), June 9-12 2002, Sardinia, Italia.

[Sm96] T. Smithers, 1986: On Knowledge level theories of design process, in J S Gero an F Sudweeks (Eds.) Artificial Intelligence in Design '96, Dordrecht: Kluwer Academic Publishers.

[Sm98] T. Smithers, 1988: Towards a knowledge level theory of design process, in J S Gero an F Sudweeks (Eds.) Artificial Intelligence in Design '98, Dordrecht: Kluwer Academic Publishers.

[Sm02a] T. Smithers, 2002: On Knowledge Level Theories and the Knowledge Management of Designing. International design conference - Design 2002, Dubrovnik, May 2002.

[Sm02b] T Smithers 2002: Synthesis in Designing, in J S Gero (ed), Artificial Intelligence in Design '02, Kluwer Academic Publishers, Dordrecht, pp 3--26.

[Sp96] J. M. Spivey: An introduction to logic programming through Prolog, Prentice Hall London, 1996

[SSN01] R. Studer, H.-P. Schnurr, A. Nierlich: Semantisches Knowledge Retrieval, ontoprise Whitepaper Series, 2001

[SSS02] S. Staab, R. Studer, Y. Sure: Knowledge Processes and Meta Processes in Ontology-based Knowledge Management. In: Handbook on Knowledge Management. C. W. Holsapple (ed.), Springer, 2002.

[UB03] M. Ullrich, S. v.d.Bergh: Sales Knowledge Manager - Kombination von statistischen und semantischen Ansätzen zur Verkaufsberatung, Karlsruhe, 2003