Ontology-Based Information Integration in the Automotive Industry

Andreas Maier[1], Hans-Peter Schnurr[1], and York Sure[2]

[1] Ontoprise GmbH, Karlsruhe, Germany
{maier,schnurr}@ontoprise.de
http://www.ontoprise.de
[2] Institute AIFB, University of Karlsruhe, Germany
sure@aifb.uni-karlsruhe.de
http://www.aifb.uni-karlsruhe.de/WBS/

Abstract. Information integration is still one of today's hottest IT topics. Neither merging the information from different data sources nor preparing it for the end user's access has been completely solved. The goal of this paper is to present a holistic approach to integration by using ontologies and logic. There are several reasons for an ontology-based approach: Ontologies are able to cover all occurring data structures, for ontologies can be seen as nowadays most advanced knowledge representation model. They are able to cover complexity, for the combination with deductive logic extends the mapping and business logic capabilities. As the model is separated from the data storage, we get a higher degree of abstraction, whereby the semantics of the whole system is increased. Ontologies are extendible and highly reusable and deliver the user a better access to his relevant content.
In this paper we describe how current capabilities of knowledge representation, mapping of structures and description of the business logic are extended. In an elaborated case study of a specific R&D process in the automobile industry [Ste03] we demonstrate that the complex integration process can be realized much easier, faster, more understandable and less expensive than before, without changing the existing IT legacy environment.

1 Introduction

1.1 Motivation for Integrated IT Solutions

Today's users and IT professionals have high expectations towards software applications: They want to access the content they need and this content must be accurate and free of redundancy. The application must be intuitive and easy to use, reusable and extendable. It 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 just connectors [Kre99] between applications, because

D. Fensel et al. (Eds.): ISWC 2003, LNCS 2870, pp. 897–912, 2003.

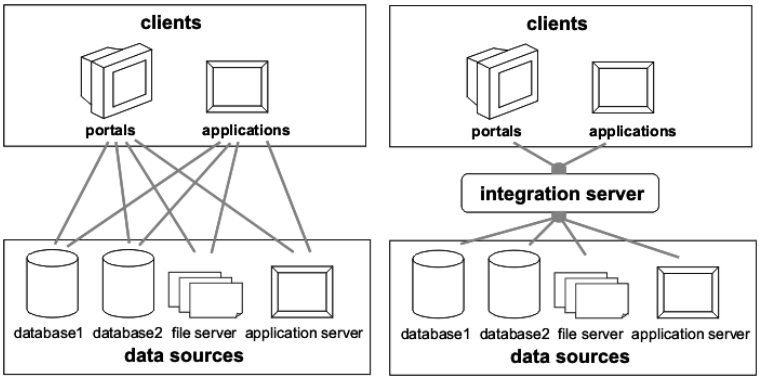

Fig. 1. Point-to-point-connections vs. a "real" integration solution

syntactical incompatibilities could be reduced by approaches like SQL [KK01] or XML; nor it's only tying up diverse data sources and displaying them on a common interface as shown in Figure 1 on the left side.

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 (figure 1 right). We believe that with ontologies there's now a model at hand to fit for this goal. From the scientific direction it's the Semantic Web Initiative [FHLW03] which focuses the demand for a better information integration, i.e. by introducing common standards like RDF [LS99] and RDF schema [BG02]. From the market side software vendors and service provider target the customer's needs by promising solutions, which often lack of integrated data, i.e. solutions for Knowledge Management, Content Management, Data Warehousing or Business Intelligence.

1.2 Foundations I: What Is an Ontology?

An ontology is a knowledge representation model. Other models for example are extended entity relationship models, thesauri or topic maps. The most common definition of "ontology" is "An ontology is an explicit specification of a conceptualization." by Tom Gruber [Gru95].

Ontologies serve as a means for establishing a conceptually concise basis for communicating knowledge for many purposes. We restrict our attention to domain ontologies that describe a particular small model of the world as relevant to applications and have shown their usefulness in these application areas. Typically, an ontology is constructed and maintained in a collaborative effort of domain experts, end-users and IT specialists.

1.3 Foundations II: Deductive Logic

Logic models (i.e. Prolog [Spi96], Datalog [DL91], F-Logic [KLW95], Description Logic [Baa02]) help us to connect the different data sources by mapping or merging rules. As a second important point they help to easily build a deductive "business logic" upon our integrated information base and to check the consistence of the knowledge base. 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.

Especially F-Logic[1] acts as a bridge between models and logic, because it covers ontological (schema and instance) data as well as rules. Every object in the ontology (concepts, relations, attributes, instances) can be addressed by F-Logic atoms and, thus, be embedded into logical rules. Furthermore, F-Logic can also be used to query the system similarly to SQL. Besides efficient querying for instance data, F-Logic allows also for querying schema data itself and even for combinations of schema and instance data. In the following we will use F-Logic as logic format for the code examples.

1.4 Structure of This Paper

Initially we gave a motivation for our approach and describe the scientific foundations of ontologies and logic in Section 1. Next, we build up the requirement specification for the integration process in Section 2 which consists of the four main requirements, viz. (1) cover all existing data structures, (2) mapping, (3) modelling the business logic, and (4) providing a data storage. Based on these requirements we define structured procedure steps which will be illustrated in detail in Sections 3–6, viz. (I) schema import, (II) creating relations, (III) create mappings, (IV) rule modelling, (V) inferencing, (VI) schema export, and (VII) materialization. Our automotive case study will be subsequently worked out step-by-step, whereby it serves as a running example. Last, but not least, we conclude in Section 7.

2 Starting the Integration Process

2.1 Introducing the Case Study: PLM in the Automotive Industry

We now describe a setting from a case study performed at the German car manufacturer Audi AG, further details can be found at [Ste03]. A typical integration scenario found in the automobile industry is the management of product components data and its permitted configurations, which proves to be very difficult. These information lie widespread in different departments and there in different sources like CAD-, CAE- or CAT-systems or ERP/PPS -applications, databases, email programs, documents, organizers, etc., – often redundantly. All these IT

[1] see `http://www.ontoprise.de/documents/tutorial_flogic.pdf`

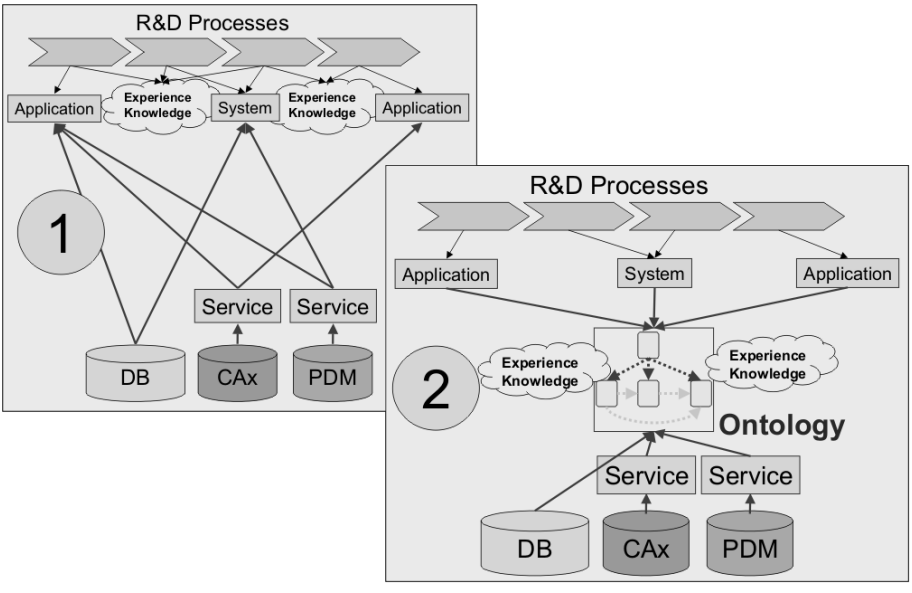

Fig. 2. A typical (1) and an integrated (2) R&D process

systems accompany the whole PLM[2]-process [Ber90], beginning with the product design and ending with the product release. To support this process, the contained information wants to be integrated for many reasons: The designer has to take constructional restrictions into account, the constructor must implement the designer's directives, the quality department wants to reduce the defect rate and the sales department has to meet the customer's request for individual pre-configured products.

An ontology could now catch up the different sources that we want to integrate. As Figure 2 shows, ontologies serve as a mediating meta model between the data sources and the user's access. Typically (see 1) different applications and systems support different process steps, the experience knowledge of engineers is crucial (i) to combine the different sources during the process, and (ii) to close the gap between different process steps and across application and system boarders. Ontologies (see 2) (i) formalize experience knowledge e.g. from engineers, and (ii) intermediate between different applications, systems and underlying services and data sources.

[2] PLM=product lifecycle management, CAD=Computer Aided Design, CAE=Computer Aided Engineering, CAT=Computer Aided Tests, ERP=Enterprise Resource Planning, PPS=Production Planning and Scheduling

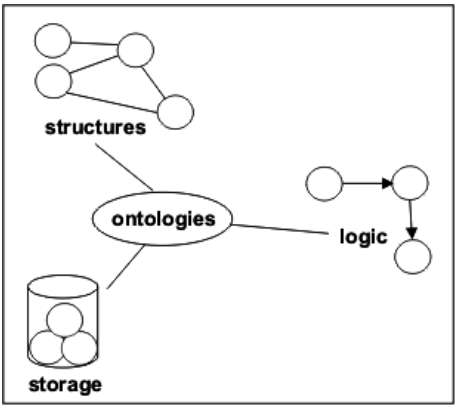

Fig. 3. Ontologies meet the requirements 1–4

2.2 Defining the Requirements

In our integration process we have to cover all existing data structures (*requirement 1*), which can be simple table structures up to complex hierarchical structured data with deep inheritance. After connecting to these structures we have to map them among each other (*requirement 2*). After this step we define the business logic for the whole new application (*requirement 3*) (hereby we will be supported by deductive inference mechanisms). If the data isn't kept in the origin sources, we have to provide an appropriate data storage for the information (*requirement 4*). As we will demonstrate in the coming sections, ontologies are a well-suited representation model to meet these requirements (see Figure 3).

2.3 Comparing Different Integration Approaches and Tools

A comparison of different approaches and tools for the information integration has one remarkable result: beside the ontology-based one, all approaches of information integration range on a pure syntactical level. They neither aim at building one common conceptual representation model (like an ontology), nor are they able to define a business logic based upon this model. Some tools (like e.g. the BizTalk Server of Microsoft) compensate this with rather focusing the transaction-oriented aspect of an integration process, than the information-oriented aspect outlined in the following Table 1.

2.4 Procedures

To the four requirements we allocate six procedure steps. Every step will be explained further below; together they form a process chain, which has to be worked out.

Table 1. Comparison of different approaches and tools

Approaches // Tools	**Covering existing data structures**	**Mapping**	**Business logic**	**Data storage**
relational databases				X
XSLT/XML	X	X		
RDF	X			
F-Logic	X	X	X	
BizTalk Server (Microsoft)		X		
Drag & Relate (SAP/Toptier)		X		
Clio (IBM)				
OntoEdit/Ontobroker (Ontoprise)	X	X	X	

1. Cover all existing data structures
   - I Schema import
   - II Creating relations
2. Mapping
   - III Create mappings
3. Modelling the business logic
   - IV Rule modelling
   - V Inferencing
4. Providing a data storage
   - VI Schema export
   - VII Materialization

We use the case study in the automotive industry as a running example to illustrate our process model by a real-world scenario. Currently we provide tool support for the steps I–V, VI and VII are future work which remains to be done.

3 Requirement 1: Cover All Data Structures

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 [Pel89], ontologies are able to cover hierarchies. Like Thesauri [Med95], Semantic Nets [Hof86] and Topic Maps [PH02], ontologies contain relations. With them, complex contexts can be modelled and visualized in nets. Linguistic contexts (i.e. multi-lingualism or synonym relations), terminologies and classifications can be described, through which the semantic of the integration solution is increased. In comparison to ontologies and databases, these models have no data model. Thus they do not contain instances and attributes, which is necessary for the management of mass data. They also miss

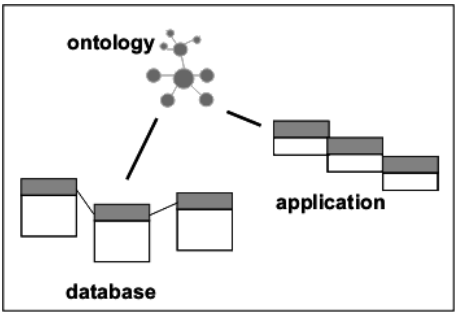

Fig. 4. Importing diverse schemas

a structured language like SQL to query the model. Taxonomies, Thesauri, Semantic Nets and the ER-model are comparatively old models. Topic Maps (an ISO standard) are the newest of them and potentially merge with the W3C efforts on ontologies. Like the Entity Relationship 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. In comparison to databases, ontologies have a higher semantic, because they name the relations instead of linking primary keys and are multilingual. As an object based model, ontologies support inheritance and multiple inheritance of attributes and relationships.

3.1 Procedure Step I: Schema Import

By a schema import we connect a former independent data source to the ontology shown in Figure 4. Beneath other schema imports (i.e. for formats like RDF or OWL), the SQL import plays a very important role.

After an import of a database table into an ontology, this one is embedded as a concept into the concept taxonomy as shown in Figure 5. The former primary key from the attribute name has moved to the object id of the concept motor. Tables are interpreted as 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.

OntoEdit is an ontology engineering environment [SEA+02,SSA02]. An SQL import has already been realized in OntoEdit[3], e.g. for Microsoft SQL Server, IBM DB2, Oracle, and mySQL.

When importing a database table, OntoEdit creates an automatical connection to the database by the `dbaccessuser`-Built-in (Built-ins include e.g. connectors to various data structures etc.). For example there's a connection

[3] see `http://www.ontoprise.de/documents/tutorial_ontoedit.pdf`

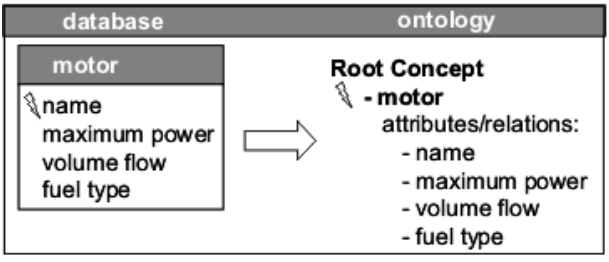

Fig. 5. Importing a database table

to the fields `name`, `maximum_power`, `volume_flow` and `fuel_type` from table `table_motor` at database `database_motor` of type `MSSQLServer` on computer `server_motordata`:

```
FORALL X, NAME, MAXIMUM_POWER, VOLUME_FLOW, FUEL_TYPE
    X:Motor[name->>NAME;
            maximum_power->>MAXIMUM_POWER;
            volume_flow->>VOLUME_FLOW;
            fuel_type->>FUEL_TYPE]
    <-
    dbaccessuser(
        "table_motor",
        F(  "name", NAME,
            "maximum_power", MAXIMUM_POWER,
            "volume_flow", VOLUME_FLOW,
            "fuel_type", FUEL_TYPE),
        "mssqlserver2000",
        "database_motor",
        "server_motordata:1433") AND
    concat("Motor", NAME, X).
```

The equivalent to the schema import is the schema export, that we will need for *requirement 3* (provide a data storage).

3.2 Procedure Step II: Creating Relations

If there have to be relations between so far unlinked concepts, i.e. because these were coming from different data sources, we can join them by logical rules. The new relations can be derived from very complex coherences. The concepts `motor` and `catalyst` for example are still unrelated, but `catalyst` has the attribute `belongs_to_motor` like shown in Figure 6 below. As the identifying numbers (121, 134, ...) are the same, a rule could be: If `cnr_motor` is the same number than the `motor name` without `CNR_`, then this `catalyst` belongs to that `motor`.

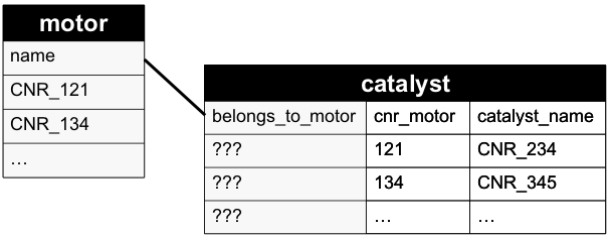

Fig. 6. Creating relations

In F-Logic, this would be expressed like:

```
FORALL X,Y,Z1,Z2
    X:catalyst[belongs_to_motor->Y:motor]
    <-
    X[cnr_motor->Z1] AND
    Y[name->Z2] AND
    concat("CNR_",Z1,Z2).
```

3.3 Continuing the Case Study: After the Schema Import

The import process should result in one or more ontologies that contain the complete model of all data sources. In our case study the created automobile ontology now contains the constituting concepts for the PLM system (see Figure 7).

4 Requirement 2: Create Mappings (Procedure Step III)

In practice there's often more than one core ontology. For this, mapping between them is needed. Also in a Semantic Web scenario, where the interaction between lots of ontologies is intended, mapping is very important. Ontology mapping connects not only primary keys and table rows, but is working on a more conceptual level, which leads to a higher degree of abstraction comparing to a simple database mapping. OntoMap, a mapping plug-in included in OntoEdit, supports the fundamental mapping types (i) concept-to-concept mapping, (ii) attribute-to-attribute mapping and (iii) attribute-to-concept mapping. Describing conditions and constraints (i.e. unit conversions) on the mapping rules is not explained further here. In the last section we will elaborate more on the difficulties found during the creation of the mappings in our case study.

4.1 Concept-to-Concept Mapping

An ontology mapping process is very similar to pure database respectively XML-mapping[4]. In each case the two schemas, which are going to be mapped, are

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

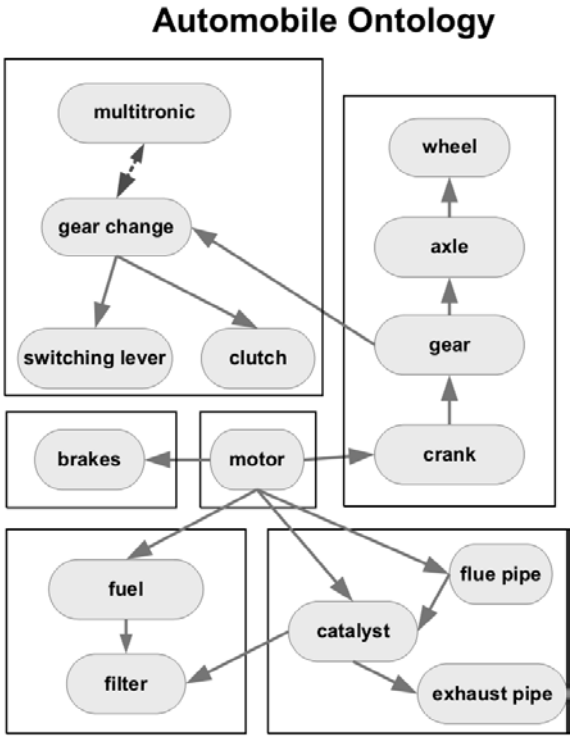

Fig. 7. Concepts of the automobile ontology

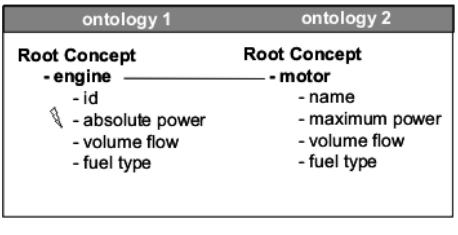

Fig. 8. Concept-to-concept-mapping

displayed in vertical rows parallel to each other (see Figure 8). If two concepts of two different sources contain the same type of information, a concept-to-concept mapping can be drawn (i.e. `engine` and `motor`).

The F-Logic syntax would be:

```
FORALL X
    X:ontology2#motor
    <-
    X:ontology1#engine.
```

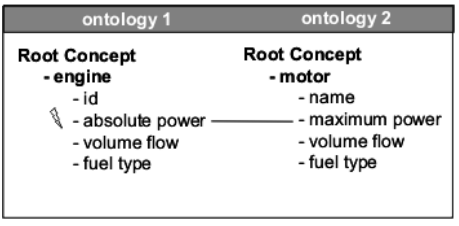

Fig. 9. Attribute-to-attribute-mapping

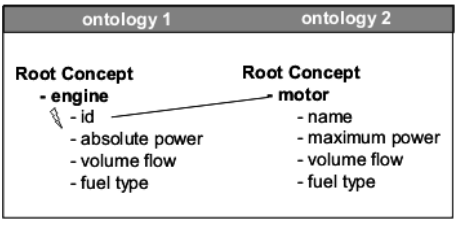

Fig. 10. Attribute-to-concept-mapping

4.2 Attribute-to-Attribute Mapping

An attribute-to-attribute mapping connects two attributes, stating that these contain the same information (i.e. `absolute power` and `maximum power`) (see Figure 9). A previous concept-to-concept mapping is prerequisite for that.

```
FORALL X,Y,Z
    Z:ontology2#motor[ontology2#maximum_power->>Y]
    <-
    X:ontology1#engine[ontology1#absolute_power->>Y].
```

4.3 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 `engine` is connected to the object id of `motor` (see Figure 10).

```
FORALL X,Y,Z
    Y:ontology2#motor
    <-
    X:ontology1#engine[ontology1#id->>Y].
```

5 Requirement 3: Enhancing the Business Logic

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. Additionally, deductive logic reduces complexity. In really complex contexts with many relations between the concepts of the ontology the effort and complexity of implementation in pure SQL quickly gets too high. As another important reason for logic, the user doesn't need to know the underlying data structures. He only knows his question and not the whole conceptual structure that lies behind it.

5.1 Procedure Step IV: Rule Modelling

Having tied together the different data sources, we now illustrate the extendibility that logic rules offer to applications. Our basic ontology (see Figure 7) is now going to be enriched by five simple rules to expand the knowledge base:

- **Rule 1:** The tolerated transmission power of a car's crank N_crank must be higher than the power of the motor N_{motor}, but must not exceed the one of the gear N_{gear}. To sum up, the condition
  $$N_{motor} < N_{crank} < N_{gear} < N_{axle} < N_{wheel}$$
  has to be valid.
- **Rule 2:** The maximum power of the motor must not exceed the one of the brakes.
  $$P_{motor} < \mid P_{brakes} \mid$$
- **Rule 3:** For the volume flow VF_x there must be:
  $$VF_{motor} < VF_{fluepipe} < VF_{catalyst} < VF_{exhaustpipe}$$
- **Rule 4:** The filter installed in a catalyst must be able to filter the motor's fuel.
- **Rule 5:** If there's a gear change, then there has to be a switching lever and a clutch. If there is a multitronic instead, then there must be no gear change.

Rule 2 for example could be encoded in F-Logic as:

```
FORALL X,Y,Z1,Z2,Z3
    message("Warning: The motor's maximum power
            exceeds the one of the brakes!")
    <-
    X:motor[maximum_power->>Z1] AND
    Y:brake[maximum_power->>Z2] AND
    abs(Z1,Z3) AND
    lessorequal(Z2,Z3).
```

These five rules are now added to the ontology to provide them for the Inferencing process.

5.2 Procedure Step V: Inferencing

Inferences are information, that are derived by means of logical conclusions. Inferencing engines like Ontobroker[5] [DEFS99] use a formal logic calculus to

[5] see `http://www.ontoprise.de/documents/tutorial_ontobroker.pdf`

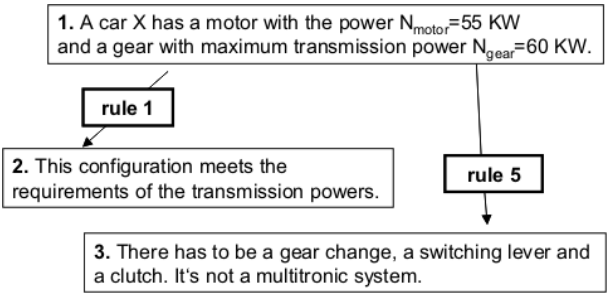

Fig. 11. An inferencing graph

generate new information out the input facts and rules. By an inferencing process the rules are applied to the given facts and extend the knowledge base by the newly created facts. Figure 11 visualizes this process.

The inference engine, running as a service on a computer, would now be able to deliver also the new facts to the user's access.

The remaining steps are at the moment of writing this paper not yet fully implemented. Therefore we will describe them in the next section as future work.

6 Future Work – Requirement 4: Provide a Data Storage

As a description 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 is not possible for technical reasons (i.e. if the application isn't able being queried), we propose migrating it to a database. Although the ER-Model has its weaknesses (see previous section), we suggest using relational databases as storage because of their widely spread and mature solutions. The recently emerging repositories like the RDFS-repository Sesame [BKv02] offer an alternative to databases. But in comparison to repositories there's at the moment no alternative concerning performance and compatibility.

Procedure Step VI: Schema Export. To export an ontology schema or a part of it into a database, we need an SQL-Export, which creates the database schema out of the ontology and writes it into a new database. An SQL export has already been realized in OntoEdit and works in analogy to the SQL import (see Section 3).

Procedure Step VII: Materialisation. Before we start an inference server, we decide whether we want to materialize all attained new facts (i.e. to explicitly store all newly derived facts from rules). If not, all inferences are computed during runtime, when a query affords it. The decision whether to materialize or not depends on the data change rate. If it is high, materialisation doesn't make sense. The materialized facts can be written into a data storage, for example into a database. Ontobroker does this when using an internal database.

7 Conclusion

We have shown in a real-life industrial case study, viz. in the automotive industry at the car manufacturer Audi AG [Ste03], that the formalization of experience knowledge and the integration of heterogeneous data sources with the help of ontologies is feasible and beneficial. Key enabler in this case study are (i) OntoEdit [SEA+02,SSA02], an ontology engineering environment and its mapping extension OntoMap and (ii) Ontobroker [DEFS99], an F-Logic based inference engine.

Looking at today's available software for ontologies, [NM02] compare different tools. In particular, they distinguish ontology editors from ontology mappers. As we have shown above, this categorisation doesn't mean that it couldn't be combined in one product. In OntoEdit for example, there's a modelling component as well as a mapping plugin[6]. Another mapping mechanism between distributed ontologies (i.e. oriented towards the vision of the Semantic Web) 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 might be supplemented by semi- or full automatic approaches. In our current point of view, these automatic approaches will bring in only a limited additional benefit. [DMDH02] delivers new findings on them.

In our practical setting, we were able to model all required mappings via F-Logic. All fundamental mappings were done (potentially by domain experts, here by ontology experts) with our graphical mapping plugin OntoMap, the more complex mappings required manual efforts in F-Logic by ontology experts.

Future development work includes the development of graphical tool support for more complex mappings. We have shown the feasibility of mappings for a practical setting in a (rather) closed domain, future work has to show the applicability for open domains such as envisioned by the Semantic Web.

References

[Baa02] F. Baader. *The description logic handbook: theory, implementation and applications*. Cambridge University Press, 2002.

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

[Ber90] T. Bernold. Product life: from design to disposal : life-cycle engineering: the key to risk management, safer products and industrial environmental strategies. In *International Conference on Industrial Risk Management*, Zürich, 1990. Elsevier.

[6] OntoEdit is based on a flexible plugin framework, further details can be found e.g. in [Han01].

[BG02] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft 12 November 2002, 2002. available at http://www.w3.org/TR/PR-rdf-schema/.

[BKv02] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF Schema. In Horrocks and Hendler [HH02], pages 54–68.

[DEFS99] S. Decker, M. Erdmann, D. Fensel, and R. Studer. *Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information*, pages 351–369. In Meersman et al. [MTS99], 1999.

[DL91] M. Dahr and K. Lautenbach. Towards a formal theory of datalog nets. Fachberichte informatik, Universitaet Koblenz-Landau, 1991.

[DMDH02] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the World-Wide Web Conference (WWW-2002), Honolulu*, 2002.

[FHLW03] D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors. *Spinning the Semantic Web*. MIT Press, 2003.

[Gru95] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928, 1995.

[Han01] S. Handschuh. OntoPlugins – a flexible component framework. Technical report, University of Karlsruhe, May 2001.

[HH02] I. Horrocks and J. A. Hendler, editors. *Proceedings of the First International Semantic Web Conference: The Semantic Web (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science (LNCS)*, Sardinia, Italy, 2002. Springer.

[Hof86] 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. In *Informatics and Psychology Workshop, Darmstadt*, 1986.

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

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, 1995.

[Kre99] D. Kreuz. *Formale Semantik von Konnektoren*. PhD thesis, Technische Universitaet Hamburg, 1999.

[LS99] O. Lassila and R. Swick. Resource Description Framework (RDF). Model and Syntax Specification. W3C Recommendation 22 February 1999, 1999. available at http://www.w3.org/TR/REC-rdf-syntax.

[Med95] N. Meder. *Konstruktion und Retrieval von Wissen: Thesauri als Terminologische Lexika*. Indeks Verlag, Weilburg, 1995.

[MMSV02] A. Maedche, B. Motik, N. Silva, and R. Volz. MAFRA – a MApping FRAmework for distributed ontologies. In A. Gómez-Pérez and V. R. Benjamins, editors, *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web (EKAW 2002)*, volume 2473 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 235–250, Siguenza, Spain, 2002. Springer.

[MT+02] R. Meersman, Z. Tari, et al., editors. *Proceedings of the Confederated International Conferences: On the Move to Meaningful Internet Systems (CoopIS, DOA, and ODBASE 2002)*, volume 2519 of *Lecture Notes in Computer Science (LNCS)*, University of California, Irvine, USA, 2002. Springer.

[MTS99] R. Meersman, Z. Tari, and S. Stevens, editors. *Database Semantics: Semantic Issues in Multimedia Systems*. Kluwer Academic Publisher, 1999.

[NM02] N. Noy and M. Musen. Evaluating ontology mapping tools – requirements and experience. Technical report, Stanford University, 2002.

[Pel89] C. Peltason. *Wissensrepraesentation fuer Entwurfssysteme: d. Behandlung von Klassifikation und Taxonomie*. PhD thesis, Technische Universitaet Berlin, 1989.

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

[SEA+02] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In Horrocks and Hendler [HH02], pages 221–235.

[Spi96] J. M. Spivey. *An introduction to logic programming through Prolog*. Prentice Hall London, 1996.

[SSA02] Y. Sure, S. Staab, and J. Angele. OntoEdit: Guiding ontology development by methodology and inferencing. In Meersman et al. [MT+02], pages 1205–1222.

[Ste03] H. Stegmueller. Audi erprobt semantische Technologien – Zeit fuers Wesentliche. *Digital Engineering Magazin*, 2003(3):44–45, 2003.