

Do not use this gear with a switching lever!

Automotive industry experience with semantic guides

Jürgen Angele¹, Hans-Peter Schnurr¹

¹ ontoprise GmbH, Amalienbadstr. 36, 76227 Karlsruhe, Germany
{schnurr, angele}@ontoprise.de
<http://www.ontoprise.de>

Abstract: One major trend may be observed in the automotive industry: built-to-order. This means reducing the mass production of cars to a limited-lot-production. Emphasis for optimization issues moves then from the production step to earlier steps as the collaboration of suppliers and manufacturer in development and delivering. Thus knowledge has to be shared between different organizations and departments in early development processes. In this paper we describe a project in the automotive industry where ontologies have two main purposes: (i) representing and sharing knowledge to optimize business processes for the testing of cars and (ii) integration of life data into this optimization process. A test car configuration assistant (semantic guide) is built on top of an inference engine equipped with an ontology containing information about parts and configuration rules. The ontology is attached to the legacy systems of the manufacturer and thus accesses and integrates up-to-date information. This semantic guide accelerates the configuration of test cars and thus reduces time to market.

1 Introduction

The scenario for this process was given by the business processes in the automotive industry around the testing of cars in the technical development department. To ensure quality improvement and innovation, a car manufacturer develops test cars. These test cars are continuously reconfigured and then tested with this new configuration. Reconfiguration means changing the engine, changing the gear, changing the electric, i.e. changing all kinds of parts. For changing parts a lot of dependencies between these parts have to be taken into account. In many cases these dependencies are only known by certain human experts and thus require a lot of communication effort between different departments of the manufacturer and between suppliers. Very often test cars have been configured which did not work. So making such dependencies exploitable by computers allows for reducing the error rate in configuring test cars with a lower communication effort. The information about car parts is stored in a considerable amount of databases. These databases have to be accessed during query-time to ensure actuality of data. During this project, an ontology has been developed which serves as a communication medium between engineers, serves as a knowledge model representing these complex dependencies and serves as an integration model for different data sources. In the following the last aspect is described in detail.

2 The Ontology

The base ontology very strongly relies on parts which are arranged in a part-of hierarchy and their properties. The instances, i.e. concrete values are most often gained from parts list in the legacy systems. Our ontologies are represented in F-Logic [KL95]. Basic concepts in the specific are represented as concepts in F-Logic and are arranged in an is-a-hierarchy. Concepts may be described by attributes and relationships to other concepts.

```
//schema
    component::DEFAULT_ROOT_CONCEPT.
    component[has_part=>>component; is_part=>> component; horsepower=>INTEGER].
// instances and relations
    tdi_engine:component. valve2:component. pump3:component.
    tdi_engine[has_part->>valve2; has_part->>pump3; horsepower->340].
//rules
    FORALL X,Y Y[is_part->>X] <- X [has_part->>Y].
//queries
    FORALL X <- X:component.
```

In this example we have defined a *component* as a basic concept. A component has a relationship *has_part* to another component and an attribute *horsepower*. Then we create an instantiation of a component *tdi_engine* being a specific engine. Concrete instances *valve2*, *pump3* are given for concept *component*. A rule is used to describe the inversivity of *has_part* and *is_part*. With a query we ask for all components in the model.



Fig. 1: An excerpt of the automotive ontology represented in OntoStudio

OntoBroker, our reasoning system, provides means for efficient reasoning in F-Logic [De99]. OntoBroker performs a mixture of forward and backward chaining based on the dynamic filtering algorithm [KL86] to compute (the smallest possible) subset of the model for answering the query. During forward chaining not only single tuples of variable instantiations but sets of such tuples are processed. It is well-known that set-oriented evaluation strategies are much more efficient than tuple oriented ones. The semantics for a set of F-Logic statements is then defined by a transformation process of F-Logic into normal logic (Horn logic with negation) and well-founded semantics [GRS91] for the resulting set of facts and rules and axioms in normal logic. In figure 1, an excerpt of that ontology is shown in a part-of view in our tool OntoStudio. It shows that e.g. a *gear* is part of a *car* and the *switching lever* is a part of the *gear*. For *motor* some attributes like *maximum power*, *type* etc. are shown.

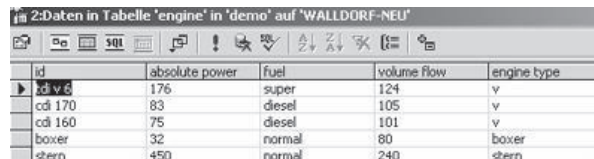
An ontology without rules describes only simple relationships between concepts like a part is a part of another part, a part is connected to another part etc. More complex relationships have to be described by rules and constraints.

3 Data Source Integration

Besides serving as a common communication language and representing expert knowledge in our scenario ontologies serve as an integration means of different legacy systems. The ontology is used to reinterpret given information sources in a common language and thus to provide a single view to different data sources. In our scenario the components data and the configuration data is already handled widespread in different departments and in different information sources like CAD-, CAE- or CAT-systems or ERP/PPS-applications, databases etc. All these IT systems accompany the whole PLM-process [Be90], beginning with the product design and ending with the product release. Our test configuration system, and thus our ontology system must access this live information to be up-to-date, to avoid inconsistent data. The ontology could now catch up these different sources and integrate them in a common logical model. This goes much beyond building just connectors [Kr99] between applications. The goal of integration is to consolidate distributed information intelligently, free of redundancy and providing users and applications a simple access to information without considering the underlying data structure or system. In our case we already have such a commonly accepted logical model: the automotive ontology. This ontology describes schema information and is not yet populated by instances. This means e.g. that there exists a concept *motor* with attributes *name*, *cylinders*, *type* etc. But there is no information about concrete *motors* like *TDI V6*, *6 cylinders*, *fuel type super* etc. available. This is achieved by attaching the ontology to one or more of the existing information sources. In the following we exemplify the mapping to a relational database.

3.1 Database schema import

The first step to connect an ontology to a database is importing the database schema and visualize it in our ontology management environment OntoStudio, the successor version of the ontology engineering environment OntoEdit [Su02]. Beneath relational database schemas, OntoEdit has also import filters for other schemas like RDF [St01] or OWL. In our example we will show the attachment of a database table *motor* to our ontology. The database table is given in figure 2. It contains information about motors like *fuel type*, *power* etc.



id	absolute power	fuel	volume flow	engine type
tdi v 6	176	super	124	v
cdi 170	83	diesel	105	v
cdi 160	75	diesel	101	v
boxer	32	normal	80	boxer
stern	450	normal	240	stern

Fig. 2: Database table engine

3.2 Database mappings

After having imported the database schema, the ontology and the schema have to be connected appropriately. OntoMap – a mapping tool included in OntoStudio – supports the fundamental mapping types (i) table-to-concept mapping, (ii) attribute-to-attribute mapping and (iii) attribute-to-concept mapping. In fig. 3 a table-to-concept mapping connects the table engine to the concept motor and additionally an attribute-to-attribute mapping from id in the database to name in the ontology. This means that every row in the database corresponds to one object in the ontology. OntoStudio automatically creates a connection to the database by the *dbaccessuserid*-connector (there are various connectors to information sources available). This built-in automatically creates a unique object ID. It is used in a rule which defines the access and the mapping to our ontology:

```
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]
<-
  dbaccessuserid ("engine",
                 X, F ("id", NAME, "absolute power", MAXIMUM_POWER,
                     "volume_flow", VOLUME_FLOW, "fuel", FUEL_TYPE),
                 "mssqlserver2000","database_motor","server_motordata:1433").
```

Another important mapping type is the mapping of attributes to concepts. This has the consequence that the attribute value is used as unique ID for an ontology instance. E.g. mapping the ID of *engine* to the concept *motor* creates an object for every different ID in the database. By that way information about one and the same object which is spread in different rows and is always identified by the same ID can be linked together. This was the case in our project for part lists.

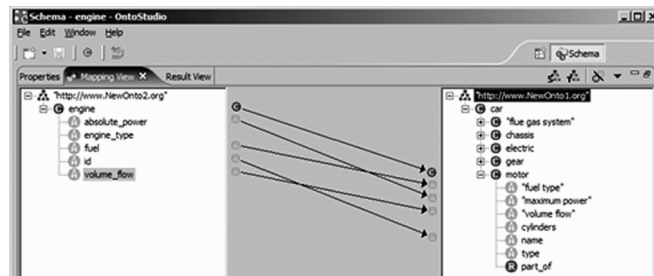


Fig. 3: Database mapping

3.3 Querying the integration ontology

Mappings as described in section 3.2 can be defined to different RDBMS and additionally to web services at the same time. A query to the integration ontology is thus at real-time translated (via the mapping rules) into calls for appropriate access builtins which in turn access the data sources (in case of an RDBMS via SQL queries) and translate the answers back into F-Logic. Thus a user or an application on top of the

ontology needs only this single ontology view and with it single vocabulary to retrieve all necessary information. In our scenario different information sources contribute to the same ontology. E.g. information about electronic parts is stored in other databases than information about mechanical parts. Thus, an engineer configuring a new test car is supported by the semantic guide in retrieving suitable and valid parts and configurations via a single and common view to distributed data sources.

4 Conclusion

In a real-life industrial project, viz. in the automotive industry at a car manufacturer, we have shown that ontologies may very well be used to enhance business processes and to integrate different information sources. In our case the ontology represents knowledge about relationships between different parts which may automatically be exploited in configuring test cars. This reduces the communication effort between the mechanical engineers, and reduces the error rate in configuring test cars. For this task the ontology is attached to the legacy systems of the manufacturer and thus accesses up-to-date information about parts and configurations. We have shown that our ontology engineering environment OntoStudio supports not only the comfortable development of ontologies but with the integrated mapping tool OntoMap also an easy to learn tool to attach ontologies to different information sources. Our semantic guide is based on our ontology run-time environment and inference engine OntoBroker. This semantic guide accelerates the configuration of test cars at our customer and thus accelerates the development of new cars as well. This reduces time-to-market in the end.

Literaturverzeichnis

- [KLW95] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and framebased languages. *Journal of the ACM*, 42; (1995) 741–843.
- [GRS91] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3); July (1991) 620–650
- [De99] S. Decker, M. Erdmann, D. Fensel, and R. Studer. OntoBroker™: Ontology based access to distributed and semi-structured information. In R. Meersman et al., editor, *Database Semantics: Semantic Issues in Multimedia Systems*. Kluwer Academic, (1999)
- [KL86] M. Kifer and E. Lozinskii. A framework for an efficient implementation of deductive databases. In *Proceedings of the 6th Advanced Database Symp.*, Tokyo, (1986) 109–116
- [MUS03] A. Maier, M. Ullrich, and H.-P. Schnurr. *Ontology-based Information Integration in the Automotive Industry*. Technical report, ontoprise whitepaper series, (2003)
- [Su02] 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]*, (2002) 221-235
- [Be90] 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*, Elsevier, Zürich, (1990)
- [Kr99] D. Kreuz. *Formale Semantik von Konnektoren*. PhD thesis, Technische Universität Hamburg (1999)
- [St01] S. Staab, M. Erdmann, A. Mädche, S. Decker. An extensible approach for Modeling Ontologies in RDF(S). In *Knowledge Media in Healthcare: Opportunities and Challenges*. Rolf Grütter (ed.). Idea Group Publishing, Hershey USA / UK. (2001)