# Do Not Use This Gear with a Switching Lever! Automotive Industry Experience with Semantic Guides

Hans-Peter Schnurr and Jürgen Angele

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

**Abstract.** Besides the reduction of time to market, there may be observed another trend in the automotive industry: built-to-order. Built-to-order reduces 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

Having a look at the shares of vehicle sales in US from 1970 – 2001 (see figure 1) we observe that the big three automobile vendors (Chrysler, Ford, General Motors) considerably lost market shares in that time period. One of the reasons was that before the early nineties the poor quality of their cars compared to the competitor's cars has been responsible for this loss. Then the big three started a quality offensive which resulted in a slight market gain until 1994. But after 1994 the big three again lost market shares. The reason for the second loss which lasts until today is the slow innovation in automotive industry in US. The competitors in Asia and Europe have been able to strongly reduce the time for developing new cars. As a consequence time-to-market is one of the main optimization goals in the automotive industry.

Another very important trend in consumer oriented production industry is built-to-order. Built-to-order means that a product is immediately produced and delivered after the consumer configured the product according to his whishes. With this strategy Dell edged out a lot of its competitors on the PC market. In contrast to that in automotive industry cars are first developed and then manufactured in large amounts with a high degree of optimization. Very often the results are huge amounts of cars which cannot be sold and thus produce costs for the investment and for storing them. Finally these cars must be sold with large sales discounts which again reduce the profit of the

Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729, pp. 1029 – 1040, 2005.

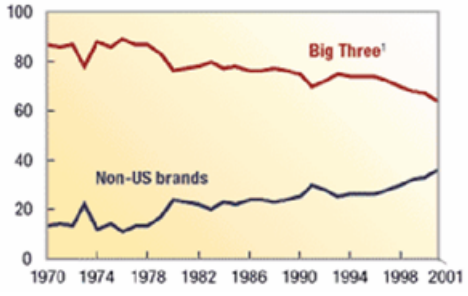


**Fig. 1.** Market shares of vehicle sales (source: Wards automotive yearbook)

manufacturer. Built-to-order avoids all these problems but has a severe change of logistic processes and business processes as consequence. Built-to-order reduces 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 between suppliers and manufacturers in development and delivering. Thus knowledge has to be shared between different organizations and departments. Therefore, the main emphasis has to be put on optimizing these business 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 testing of cars and (ii) integration of life data into this optimization process.

The scenario for this process was given by the business processes around the testing of cars. Our client has a fleet of 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, between the manufacturer and suppliers and between suppliers. Very often test cars have been configured which did not work or which hampered the measurement of the desired parameters. So making such dependencies exploitable by computers allows for reducing the error rate in configuring test cars with a lower communication effort. This in turn accelerates the development of new cars and enhances the collaboration between manufacturer and suppliers. Thus it reduces time-to-market and supports the built-to-order process.

## 2 Knowledge Representation in Automotive

### 2.1 The Representation Language F-Logic

Conceptual (or Ontology) modeling deals with the question of how to describe in a declarative and abstract way the domain information of an application, its relevant vocabulary, and how to constrain the use of the data, by understanding what can be drawn from it. Corresponding abstract representation languages support the under-

standing of such descriptions, their rapid development, their maintenance and their reuse.

Representing our domain in our representation language F-logic [1] provides a lot of advantages:

- F-Logic represents knowledge on a high level of abstraction in an explicit way. Thus knowledge is represented in well understandable independent knowledge chunks. It is not hidden in lower levels like SQL programs or even in computer programs.
- The abstraction level of F-Logic supports the rapid development of such models and the maintenance. Both aspects are very important in this domain since new dependencies arise with every new car model. This allows also to directly communicate F-Logic models between the users (in our case the engineers) and thus allow users to evaluate these models.
- The model is immediately executable. This means that as soon as the development of the model has started it could be automatically exploited by queries. Thus it immediately gives a value-add for the user. This distinguishes F-Logic also from other conceptual modeling languages like UML etc. which are not expressive enough to describe the entire system in detail to provide an executable prototype.
- F-Logic provides a clear and well-documented semantics (well-founded semantics cf. [2]). This means that the knowledge models, i.e. the ontologies are interpretable and understandable in an inambigious way independently from an explicit implementation of a reasoning system (which is often the case for other business rules systems).

Basic concepts in the specific are represented as concepts in F-Logic and are arranged in an isa-hierarchy. Concepts may be described by attributes and relationships to other concepts.

```
//schema
component:: DEFAULT_ROOT_CONCEPT.
component[  has_part=>>Component;
            is_part=>> Component].
motor::component[
            maximum_power=>INTEGER].

// instances and relations
tdi_engine: motor.
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 (below the root concept). A component has a relationship *has_part* to another component and an

attribute *maximum_horsepower*. A *motor* is a special component. Then we create an instantiation of a component *tdi_engine* being a specific motor. Concrete instances *valve2*, *pump3* are given for concept *component*. A rule is used to describe the inversity of *has_part* and *is_part*. With a query we ask for all components in the model.

OntoBroker, our reasoning system, provides means for efficient reasoning in F-Logic [3]. OntoBroker performs a mixture of forward and backward chaining based on the dynamic filtering algorithm [4] 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 the well-founded semantics [2] for the resulting set of facts and rules and axioms in normal logic.

### 2.2 Answer Justification with F-Logic

There are many reasons that users and applications need to understand the provenance of the information they get back from applications. One major motivating factor is trust. Trust and reuse of retrieval and deduction processes are facilitated when explanations are available. Ultimately, if users and/or applications are expected to trust, use and reuse application results, potentially in combination with other information or other application results, users and agents may need to understand where the derived and source information came from at varying degrees of detail. This information, sometimes called provenance, may be viewed as meta information about information told. Provenance information may include source name, date and author(s) of last update, authoritativeness of the source, degree of belief, degree of completeness, etc.

Our approach for answer justification is based on meta-inferencing. While processing a query the inference engine is producing a log-file of the proof tree for any given answer. This proof tree itself is represented in F-Logic. It contains the instantiated rules that were successfully applied to derive an answer. This file acts as input for a second inference run, where answers are produced, that are explaining the proof tree in natural language and by that how the answer to the original query was inferred.

Rules which are important for justifying results were explicitly named. For these rules certain explain rules are formulated which will be applied in the second, the meta-inference run. Frequently the named rules corresponded to important scientific laws (like load transmission), while less important rules were typically required for technical reasons, e.g. in order to translate between two alternative representations of the same content, but were not important for the human to understand the solution proposed by the system.

## 3 Business Logic Enhancements

The setting here is the configuration of test cars. These test cars are continuously reconfigured and then tested with this new configuration. Reconfiguration means that according to orders of the test engineers parts have to be replaced by other parts, parts have to be dismantled or have to be built into the test cars. These test cars are then used to test for specific characteristics and to gain series of measurements either in

house or in test drives. For changing parts a lot of dependencies between these parts have to be taken into account. In many cases this knowledge about these dependencies is available by experts only. These experts need a lot of time for communication and often enough test cars have been wrongly configured due to inefficiencies in communication.

Besides describing the knowledge about a domain, ontologies serve as mediators between data sources [5]. By this way up-to-date data about parts etc. from the legacy systems of the manufacturer are available. This integration aspect is handled in more detail in the next section.

This ontology will be used in two different ways. In a first step it will be integrated into a software assistant which helps the engineer in configuring test cars. The engineer asks the assistant for a reconfiguration and the system answers with the dependencies which have to be taken into account and the contact information for experts in this case. Additionally to these answers the assistant will provide explanations which help the engineer to understand and validate the decision of the assistant.

### 3.1 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.

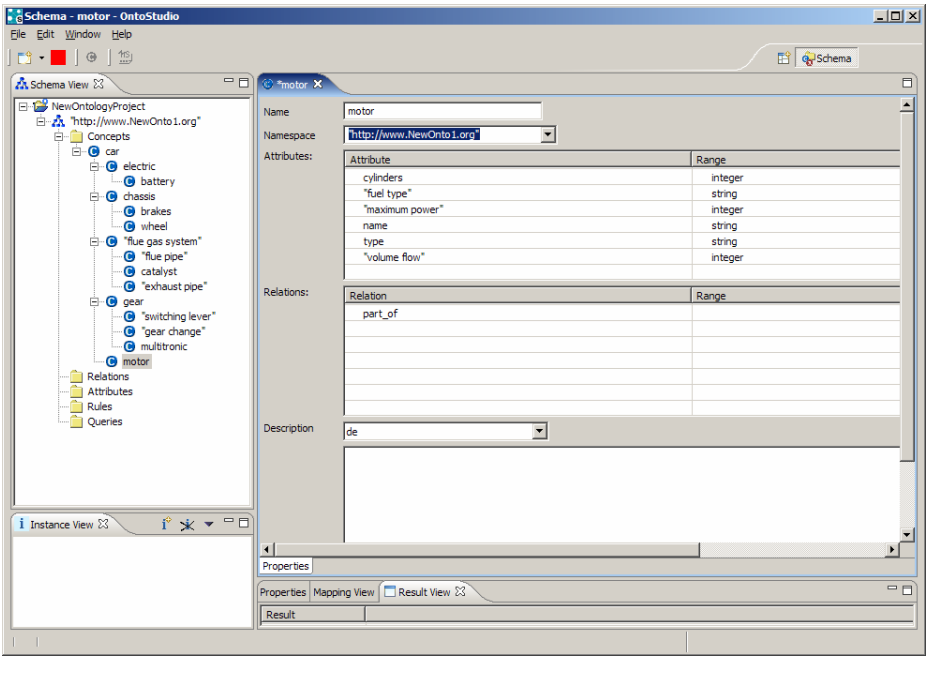


**Fig. 2.** An excerpt of the automotive ontology in OntoStudio™

In figure 2, an excerpt of that ontology is shown in a part-of view. 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. It is this more complex knowledge which has to be captured by the ontology to help configuring test cars. In the following such constraints are presented:

**Constraint 1:** The maximum power of the motor must not exceed the one of the brakes.

$$Pmotor < | Pbrakes |$$

**Constraint 2:** The filter installed in a catalyst must be able to filter the motor's fuel.

**Constraint 3:** If there is a multitronic, then there must not be a switching lever and a clutch.

These constraints are then added to the ontology by using rules:

**Rule 1:** The maximum power of the motor must not exceed the one of the brakes: Pmotor < | Pbrakes |

```
FORALL X,Y,Z,Z1,Z2,Z3
message("The motor's maximum power exceeds the one of
the brakes.")
<-
X:testcar[hasMotor->Y;hasBrake->Z] and Y[maximum_power-
>>Z1] AND Z[maximum_power->>Z2] AND abs(Z1,Z3) AND les-
sorequal(Z2,Z3).
```

**Rule 2:** The filter installed in a catalyst must be able to filter the motor's fuel.

```
FORALL X,Y,Z1,Z2
message("The installed filter uses another fuel type
than the motor")
<-
X:motor[fuel_type->>Z1] AND Y:filter[fuel_type->>Z2]
AND not equal(Z1,Z2).
```

**Rule 3:** If there is a multitronic, then there must not be a switching lever and a clutch.

```
FORALL X,Y
message("A multitronic can not be combined with a
switching lever or a clutch.")
<-
X:multitronic AND (Y:switching_lever OR Y:clutch).
```

It is clear that these single constraints look very simple. In most cases it is not the complexity of such a single relationship which creates the complexity of the task but the overwhelming amount of such rules and constraints which all interfere with a lot of others and thus make the task to configure a correct test car so complex and error prone. On the other hand the simplicity of the single rules gives us the hope that they can be created and maintained by the engineers themselves [6]. With its rules F-Logic is a very powerful language able to express arbitrary complex relationships. Other languages like OWL are much more restrictive. E.g. OWL does not allow to express conditions where variables are chained over different conditions like in rule 1.

### 3.2 Inferencing

Inferences are facts, derived by means of logical conclusions. Inferencing engines like SiLRI [7] or OntoBroker [3] use a formal logic calculus to generate new facts out of input facts and rules. By that way our ontology with the rules is immediately executable after being loaded into OntoBroker. This means that queries could be posed to OntoBroker which in turn draws logical conclusions by evaluating the rules and produces answers to the queries.

In an inferencing process the rules are applied to the given facts and extend the knowledge base by the newly created facts. Figure 3 visualizes this process.

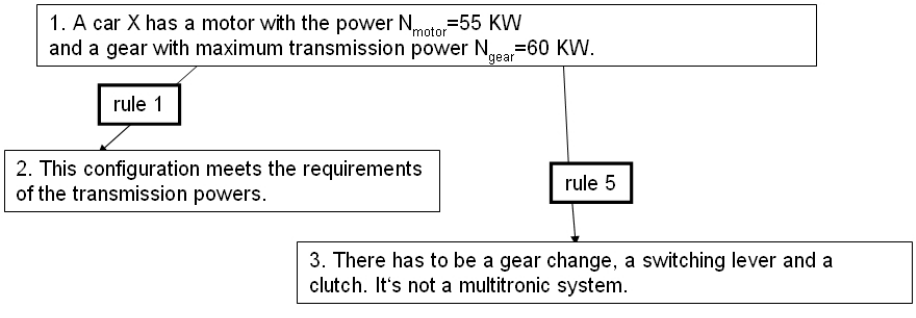


**Fig. 3.** The inferencing process

### 3.3 Semantic Guide for Test Car Configuration

On top of OntoBroker equipped with our ontology, a first prototype of a web-based user interface for the semantic guide has been developed (see fig. 4). In the left frame the user navigates within a part-of hierarchy of the components. This view may be switched to an is-a hierarchy. The attribute values of a selected component are editable in a form in the middle frame. The current configuration with all its components is shown in the right frame. If a configuration contains inconsistent components which is checked by applying consistency rules, appropriate error and warning messages are immediately given and alternatives for a selected component are presented to the user which make the configuration consistent. In our screenshot a special motor is selected and presented together with its attribute values.

The current configuration contains two incompatible components: gear 0815 and a switching lever must not be used together. This is indicated by the error message at the bottom. If the system knows about options which make the configuration consistent, these options are shown in the right window. A user can now exchange a component by the suggested option. It is clear that configuration on this way is a mixed-initiative approach between the computer and the user. This is in contrast to problem-solving methods for configuration like [13].

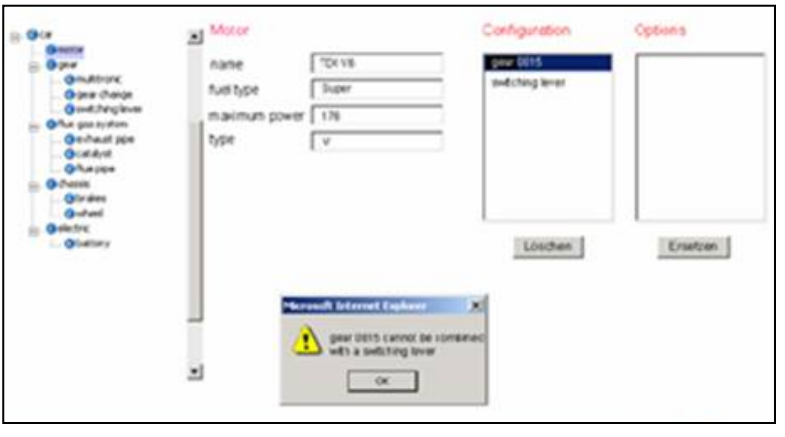


**Fig. 4.** Prototype of the Semantic Guide

## 4 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 common and 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 [8], 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 and to avoid additional effort.

An ontology could now catch up these different sources and integrate them in a common logical model. This goes much beyond building just connectors [9] 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*, with 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.

### 4.1 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 [6], [10]. Beneath relational database schemas OntoStudio has also import filters for other schemas like RDF [11], [12] 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 5. It contains information about motors like the *fuel type, power* etc.

2:Daten in Tabelle 'engine' in 'demo' auf 'WALLDORF-NEU'

| 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. 5.** Database table engine

### 4.2 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. 6 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").
```

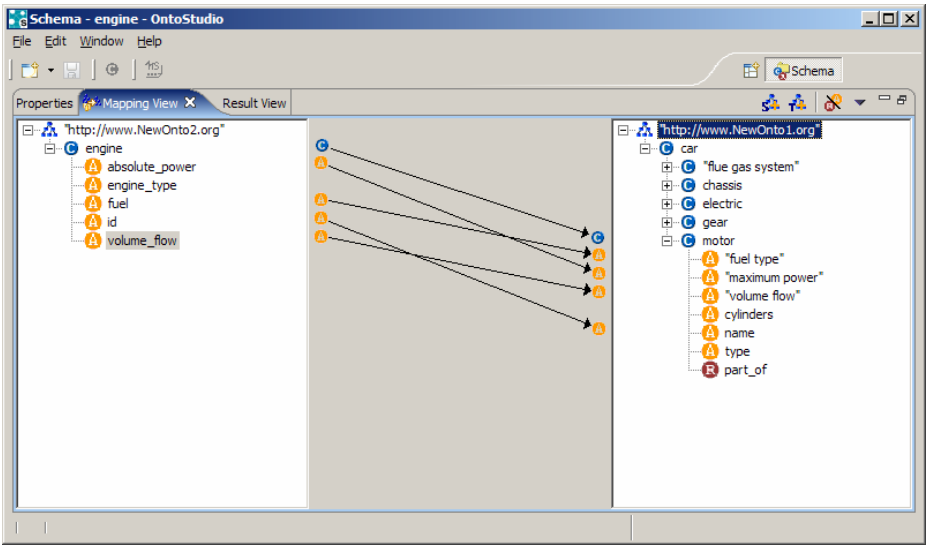


**Fig. 6.** Database Mapping

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.

### 4.3 Querying the Integration Ontology

Mappings as described in section 4.2 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 are stored in other databases than information about mechanical parts. Information about the 3-D geometry of objects is separated from their mechanical properties etc.

### 4.4 How to Handle Inconsistencies

It is clear that in practice the different information sources contain redundant ore even inconsistent information. For instance in our scenario car types have not been represented in a unique way. The assignment of properties to car types has been described with different keys for one and the same car type. E.g. Keys like *A3/A4* have been

used to describe common properties of two car types while unique properties have been assigned to the car type by a key *A3*. We again use rules and thus inferencing to solve such problems.

```
FORALL X, Part, Type
X:Car[carType->>Type; has_part->>Part]
<-
dbaccessuserid("car", X, F( "id", T, "part", Part),
"mssqlserver2000", "car database","server:1433") and
T is substring(T,0,indexof(”/”)).
```

## 5 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 (complex) 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 which is based on F-Logic. This semantic guide is a prototype which is currently evaluated at our customer. It has been shown that it accelerates the configuration of test cars at our customer and thus accelerates the development of new cars as well. This will reduce time-to-market in the end.

## References

1. M. Kifer, G. Lausen, and J.Wu. Logical foundations of object-oriented and framebased languages. *Journal of the ACM*, 42; (1995) 741–843
2. 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
3. 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)
4. M. Kifer and E. Lozinskii. A framework for an efficient implementation of deductive databases. In Proceedings of the 6th Advanced Database Symposium, Tokyo, August (1986) 109–116
5. Andreas Maier, Mike Ullrich, and Hans-Peter Schnurr. Ontology-based Information Integration in the Automotive Industry. Technical report, ontoprise whitepaper series, (2003)

6. 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.
7. S. Decker, D. Brickley, J. Saarela und J. Angele: A Query and Inference Service for RDF. In *Proceedings of the W3C Query Language Workshop (QL-98)*, Boston, MA, 3.-4. Dezember, (1998)
8. 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)
9. D. Kreuz. Formale Semantik von Konnektoren. PhD thesis, Technische Universitaet Hamburg (1999)
10. Y. Sure, S. Staab, J. Angele. OntoEdit: Guiding Ontology Development by Methodology and Inferencing. In: R. Meersman, Z. Tari et al. (eds.). Proceedings of the Confederated International Conferences CoopIS, DOA and ODBASE 2002, October 28th - November 1st, 2002, University of California, Irvine, USA, Springer, LNCS 2519 , (2002)1205-1222.
11. Richard Fikes. Ressource Description Framework (RDF). http://www.stanford.edu/class/cs222/slides2/RDF.PDF. (2002)
12. Steffen Staab, Michael Erdmann, Alexander Mädche, Stefan 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 / London, UK. December (2001)
13. Mittal, S., Frayman, F. Towards a generic model of configuration tasks. In *Proceedings of IJCAI'89*, (1989).