Data Integration using Semantic Technology: A use case

Jürgen Angele, ontoprise GmbH, Germany
Michael Gesmann, Software AG, Germany

Abstract

For the integration of data that resides in autonomous data sources Software AG uses ontologies. Data source ontologies describe the data sources themselves. Business ontologies provide an integrated view of the data. F-Logic rules are used to describe mappings between data objects in data source or business ontologies. Furthermore, F-Logic is used as the query language. F-Logic rules are perfectly suited to describe the mappings between objects and their properties.

In a first project we integrated data that on one side resides in a support and on the other side in a customer information system.

Introduction

Data that is essential for a company's successful businesses often resides in a variety of data sources. The reasons for this are manifold, e.g. load distribution or independent development of business processes. But data distribution can lead to inconsistent data which is a problem in the development of new businesses. Thus the consolidation of the spread data as well as giving applications a shared picture of all existing data is an important challenge. The integration of such distributed data is the task of Software AG's "crossvision Information Integrator" one of the components in the crossvision SOA suite [crossvision].

Information Integrator is based on ontologies. Using ontologies Information Integrator solves three major problems. First of all it provides all means to integrate different information systems. This means that comfortable tools are available to bring data from different systems together. This is partially already solved by systems like virtual or federated databases [Batini et al. 1986]. Information Integrator is more powerful compared to most of these systems as it not only supports databases but additional sources like web services, applications etc. The second problem which is solved is that Information Integrator allows reinterpretation of the contents of the information sources in business terms and thus makes these contents understandable by ordinary end users and not only by database administrators. Finally this semantic description of the business domain and the powerful mapping means from the data sources to the business ontology solves the semantic integration problem which is seen as the major problem in information integration. It maps the different semantics within the information sources to the shared conceptualization in the business ontology.

Within Software AG Information Integrator was used for a first project Customer Information Gateway (CIG) whose mission was to integrate data that on one side resides in a support information system and on the other side is stored in a customer information system.

Conceptual Layering

Conceptually Information Integrator arranges information and the access to information on four different layers (cf. fig 1):

- The bottom layer represents different data sources which contain or deliver the raw information which is semantically reinterpreted on an upper layer viz. ontologies. Currently relational databases, Adabas databases and web services are supported.
- The second layer assigns a so called "data-source ontology" to each of the data sources. These "data-source ontologies" reflect only database or WSDL schemas of the data sources in terms of ontologies and can be created automatically. Thus they are not real ontologies as they do not represent a shared conceptualization of a domain.
- The third layer represents the business ontology using terminology relevant to business users. This ontology is a real ontology, i.e. it describes the shared conceptualization of the domain at hand. It is a reinterpretation of the data described in the data-source ontologies and thus gives these data a shared semantics. As a consequence a mental effort is necessary for this reengineering of the data source contents which cannot be done automatically.
- On a fourth layer views to the business ontologies are defined. Basically these views query the integration ontology for the needed information. Exposed as Web services they can be consumed by portals, composite applications, business processes or other SOA components.

The mappings between the data-sources and the source ontologies are created automatically, the mappings between the ontologies are manually engineered and the views are manually defined queries. Mappings provide ways to restructure information, to rename information or to transform values. Up to now, we do not consider and do not plan to consider approaches which try to automatically derive such mappings [Rahm and Bernstein 2001].

This arrangement of information on different layers and the conceptual representation in ontologies and the mediation between the different models by mappings provide various advantages:

- The reengineered information in the business ontology is a value on its own. The representation as an ontology is a medium to be discussed easily by non-IT experts. Thus aggregating data from multiple systems this business ontology provides a single view on relevant information in the user's terminology.

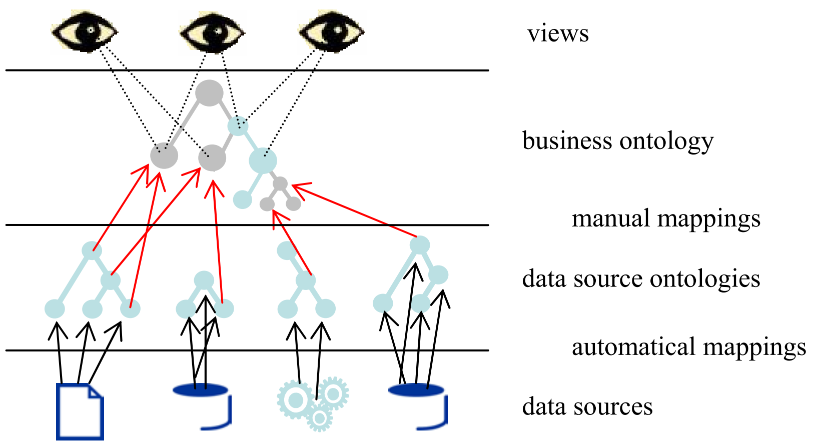

Fig 1. Conceptual Layering of Ontologies

- It is easy to integrate a new data source with a new data schema into the system. It is sufficient to create a mapping between the corresponding source ontology and the integration ontology and thus does not require any programming know-how; pure modelling is sufficient.
- The mediation of information between data sources and applications via ontologies clearly separate both. Thus changes in the data source schemas do not affect changes in the applications, but only affect changes in the mediation layer, i.e. in the mappings.
- This conceptual structure strongly increases business agility. It makes it very easy to restructure information and thus to react on changing requirements. Only the business ontology and the mappings have to be modified. Thus it minimizes the impact of change, eases maintenance and allows for rapid implementation of new strategies
- Ontologies have powerful means to represent additional knowledge on an abstract level. So for instance by rules the business ontology may be extended by additional knowledge about the domain. Thus the business ontology is a reinterpretation of the data as well as a way to represent complex knowledge interrelating these data. So business rules are directly captured in the information model.

Tool Support / Architecture

The crossvision Information Integrator provides a full fledged tool environment for defining models, for mappings between these models and for running queries (cf. fig 2). IntegratorStudio is an ontology engineering environment based on OntoStudio^{TM}.

It allows for defining classes with properties, instances of these classes and rules. Import capabilities generate "source ontologies" from underlying data sources. A powerful mapping tool allows users to interactively define mappings between ontologies by graphical and form based means (cf. fig. 3). Rules may be defined with graphical diagrams. IntegratorStudio supports F-Logic [Kifer, Lausen, Wu 1995], RDF(S), OWL for import and export. Queries which define the mentioned views can be generated and may be exported as web services.

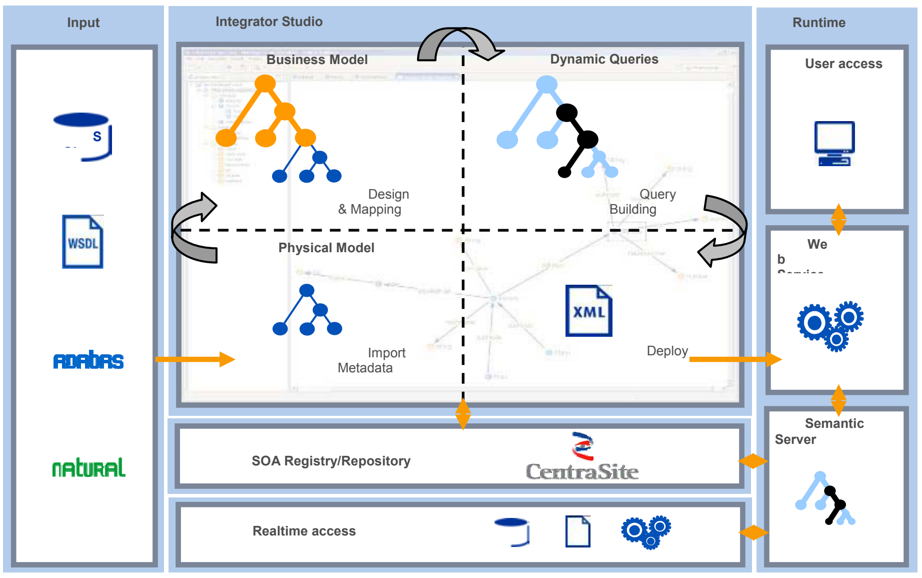

Fig. 2 Architecture of the crossvision Information Integrator

SemanticServer, the reasoning system, provides means for efficient reasoning in F-Logic. SemanticServer performs a mixture of forward and backward chaining based on the dynamic filtering algorithm [Kifer, Lozinskii 1986] to compute (the smallest possible) subset of the model for answering the query. The semantics for a set of F-Logic statements is the well-founded semantics [Van Gelder, Ross, Schlipf 1991].

Meta data like ontologies, their mappings, web service descriptions and meta information about data sources are stored in the CentraSite repository. Also, IntegratorStudio stores information about exported web services in CentraSite. During startup the inference engine SemanticServer which is based on OntoBroker^{TM} loads the ontologies from the repository and then waits for queries from the exported web services. These queries are evaluated by SemanticServer and are online translated into calls to access connected data sources.

Thus SemanticServer represents the run-time engine, IntegratorStudio the modelling environment and CentraSite the meta data repository. SemanticServer is also integrated into IntegratorStudio thus enabling immediate execution of queries to the ontologies.

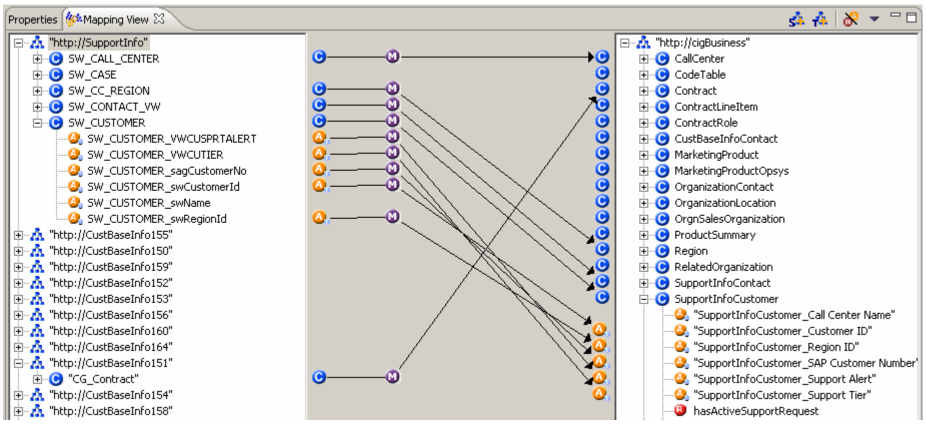

Fig. 3 Mapping Tool in crossvision Information Integrator

Use Case: Customer Information Gateway

Within Software AG the Information Integrator was used for a first project whose mission was to integrate data that on one side resides in a support and on the other side in a customer information system. The support system stores customers, their contact information and active or closed support requests in an SQL server. The customer system provides information about clients, contracts etc. in an Adabas database. The integrated data view is exposed in a browser based application to various parties inside the company, for instance to support engineers.

For illustration purposes we first sketch a very simplified excerpt of imported data and the business ontology. Throughout the following examples we use F-Logic syntax.

First of all there are two classes which have been generated by the mentioned automatic mapping from Adabas files:

F151CONTRACT [F151AA=>string; F151AE=>date].
F87CLIENT [F87AA=>number; F87AB=>string; F87AC=>string].

The cryptic names reflect the internal structure of Adabas files. The names “CONTRACT” and “CLIENT” have been specified by the user during the mapping process. Currently, the semantics of properties is only application knowledge.

Furthermore, we consider two tables from the SQL database. The generated classes are:

CUSTOMER [id=>number; name=>string; addr=>string].
CASE [caseId=>number; customerId=>string; forCustomer=>CUSTOMER].

The business ontology shall contain three classes:

Customer [name=>string; address=>string].
SupportRequest [id=>number; status=>string; issuedBy=>Customer].
Contract[contractId=>string;contractEnd=>date;contracEndFormatted=>string].

In the sequel we present some examples on how we used rules within our ontologies and derive some requirements and use cases for rule languages to be used in such a project.

Data source import

In Information Integrator user-defined built-in predicates implement access to external data sources. In the sequel we abstract from a concrete syntax of these built-in predicates. Instead we illustrate this by a generic predicate "dataAccess":

dataAccess(ci, "tablename", "rowid1", X, "rowid2", Y, ...)

where *ci* describes all parameters that are needed to call the data source, *tablename*, *rowid1, rowid2* are names of some database tables or table columns. *X, Y* are the names of variables which are to be bound by the built-in predicate.

In our example there are rules for ever class in the source ontologies which import data from external data sources. Two of these rules are:

FORALL X, Y c("F151",X) : F151CONTRACT [F151AA→X; F151AE→Y]
← dataAccess(ci, "F151", "AA", X, "AE", Y).
FORALL X, Y c("CASE", X) : CASE[caseId→X; customerId → Y]
← dataAccess(ci, "CASE", "caseId", X, "customerId", Y)].

Every functional model needs to describe relations between objects. Object properties are used to express these relationships. Object identifiers serve as object property values which are similar to foreign keys in relational databases. The foreign key definitions in a schema descriptions are used to generate object properties in source ontologies:

FORALL X, Y X[forCustomer→c("CUSTOMER", Y)] ← X:CASE[customerId→Y].

Source to Business model mappings

It is very easy to define that an object in the data source model is also an object in the business model. Similarly mappings between properties in both models can be expressed. The following example combines both mappings for contract objects:

FORALL X, Y, Z X : Contract [contractId → Y; contractEnd → Z]
← X : F151CONTRACT [F151AA → Y; F151AE → Z].

If the underlying data from the external sources contains such information, it is also easily possible to describe that two objects are the same. For example a client in the customer information system and a customer in the support information system represent the same object, if these have the same name and address. Please note, surrogate values as unique keys are typically not viable object identifiers across independent data sources. Therefore, we need to identify new identifiers:

FORALL X, Y, Z c("Customer", Y, Z) : Customer [name → Y ; address → Z]
← X : CUSTOMER [name → Y; addr → Z].
FORALL X, Y, Z c("Customer", Y, Z):Customer[name → Y; address → Z]
← X:F87CLIENT[F87AB→Y; F87AC→Z].

Often in independent data sources similar data can be encoded in a different ways, e.g. different data types or type systems. Then functions are needed which implement transformations:

FORALL X, Y Y[contractEndFormatted → X]
← EXISTS Z (Y : Contract [contractEnd → Z] and date2string(Z, X)).

where date2string() transforms a date from one format into another one.

Also, object properties need to be mapped to the business ontology:

FORALL X, Y, Z1, Z2 X : SupportRequest [issuedBy → c("Customer",Z1,Z2)]
← X : CASE [forCustomer → Y]
and c("CUSTOMER",Y) [name → Z1; addr → Z2].

The inverse reference is also often needed. But because the foreign key constraint in SQL systems does not provide a name for the inverse relation this is currently postponed to application development. N:M relationships, implemented by two 1:N foreign key relations in SQL systems, could also be expressed directly.

All these simple types of mappings are essential for specification of business ontologies on top of data source or other business ontologies. Most of them can be described in the Information Integrator with graphical means, i.e. developers do not need to see the F-Logic syntax.

Queries

To lower investments for learning new languages and to avoid impedance mismatches rule- and query-language should be the same. Information Integrator uses F-Logic for ontology definitions and as the query language. But, queries in the data integration scenario are much like database queries. Primarily we want to retrieve data. We are not so much interested in explanations or in information about which variable bindings lead to a result. This focus on data access requirements sometimes leads to quite complex query formulations. One example is different handling of not existing values (null values) in SQL and F-Logic. Another example are user defined projections. In order to minimize the number of expensive interactions between client and server we database folks tend to create queries which return complex structured results. Object relations should be contained in the result. E.g. for one customer having multiple contracts each having contract items, then the query result should contain the information which contract item belongs to which contract within a single result per customer.

Performance

Because the integrated view is used in an application where e.g. support engineers expect fast answers for even complex queries while talking to a customer, the performance of the rule and query processing is extremely important. In some cases response times in the range of a few seconds are not accepted. In our first project a lot of effort was spent to improve the responsiveness of the system. Problems that showed up here are very similar to query optimization problems in database systems.

Just for illustration we give two examples. First, the data source mappings as shown above always addressed only a single database table or file. However, a system that implements access to external data sources only via such single-table access rules will not achieve sufficient performance. Instead access operations should use the data source's query capabilities like join-operations. As a second example, the rule engine sometimes first retrieved all data from a table and then continued with the evaluation of filters. Instead, filters need to be identified first and given to the query which reads data from the database.

Summary and Outlook

A data model in Information Integrator consists of ontologies. Data source models describe the structure of data that resides in external data sources. Business ontologies provide a conceptualization of business entities. F-Logic rules are used to define mappings between ontologies. Furthermore, rules are the first choice to express semantics that is not immediately available within the data and otherwise had to be implemented in queries or applications. F-Logic is also used as the query language.

With the exception of mapping rules the business ontology of our first project does not contain many other rules. Access to information in these models is more data retrieval and not so much knowledge inference. Much effort during this project was spent on performance improvements.

With an increasing number of web services where some simply expose data, we also need to support data integration for such web services in our crossvision SOA suite. We are currently working on the mapping of web services and their structured XML data to source ontologies.

The crossvision Information Integrator based on ontoprise OntoStudio^{TM} and Ontobroker^{TM} is the first step for Software AG in the field of semantic technologies. Recently we joined various EU research projects like NeOn (Lifecycle Support for Networked Ontologies) [NEON], “Business Register Interoperability Throughout Europe” and “SemanticGov: Services for Public Administration” [SemanticGov]. All these projects address concrete business cases. With our participation in these projects we intend to achieve deeper understanding of needs for adequate tooling and runtime systems when using semantics technologies for data integration. On the other hand we will contribute our knowledge about data-intensive processing.

References

[Batini et al. 1986] Batini C., Lenzerini M., Navathe S.B. *A Comparative Analysis of Methodologies for Database Schema Integration.* ACM Computing Surveys Vol. 18(4):323-364, 1986

[Belkin 1980] N.J. Belkin. *Anomalous states of knowledge as a basis for information retrieval*. The Canadian Journal of Information Science, 5:133--143, 1980.

[crossvision] http://www.softwareag.com/crossvision

[Kifer, Lausen, Wu 1995]. Logical foundations of object-oriented and framebased languages. Journal of the ACM, 42; (1995) 741–843

[Kifer, Lozinskii 1986]. A framework for an efficient implementation of deductive databases. In Proceedings of the 6th Advanced Database Symposium, Tokyo, August (1986) 109–116

[Jaro 1989] M. A. Jaro. *Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida*. Journal of the American Statistical Association 84:414–420, 1989.

[Jaro 1995] M.A. Jaro. *Probabilistic linkage of large public health data files (disc: P687-689)*. Statistics in Medicine 14:491–498, 1995.

[NEON] http://www.neon-project.org

[Rahm and Bernstein 2001] E. Rahm, P. Bernstein. *A survey of approaches to automatic schema matching*, VLDB Journal 10(4):334-350, 2001

[SemanticGov] http://www.semantic-gov.org

[Van Gelder, Ross, Schlipf 1991]. The well-founded semantics for general logic programs. Journal of the ACM, 38(3); July (1991) 620–650