The SILK Information Integration System

Tamás Benkő

IQSYS Information Systems Ltd., H-1135 Budapest, Csata u. 8.
benko.tamas@iqsys.hu

Workshop on Logic-based Methods for Information Integration
Vienna, Austria

23rd August, 2003

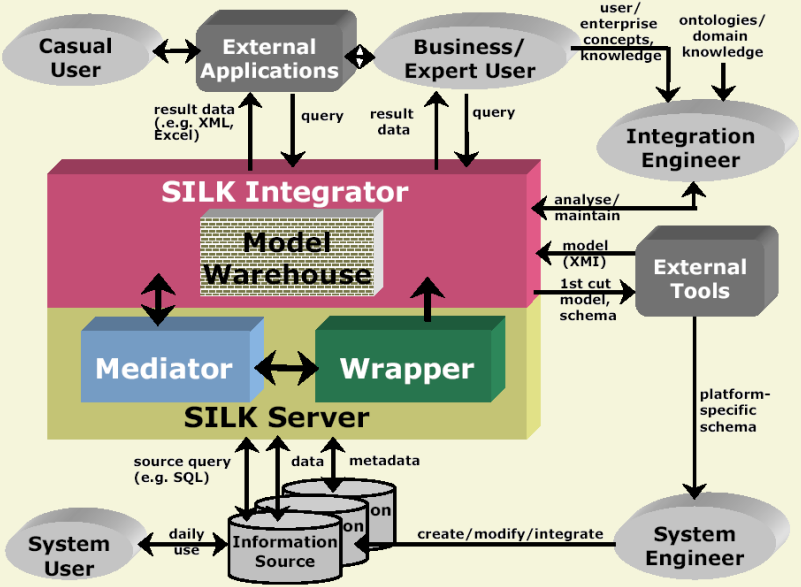
Contents

- SILK overview
- Functionality and architecture
- Model Warehouse
- SILan semantics
- SILan example
- Reasoning capabilities
- Future work

SILK: System Integration using Logic and Knowledge

- An IST, EU 5th Framework programme project
- Partners
  - IQSOFT Ltd. (Hungary)
  - EADS Systems & Defence Electronics (France)
  - National Institute for Research and Development in Informatics (Rumania)
  - Industrial Development and Education Centre (Greece)
- Product: a tool-set for the integration of heterogeneous information sources
  - *Integrator*: building integrated models and customised user views
  - *Mediator*: querying on multiple sources and multiple abstraction levels
  - *Wrappers*: common interface for accessing various information source types
    - relational and object-oriented databases
    - semi-structured sources (e.g., XML, RDF)
    - data provided through applications (e.g., Web-services)

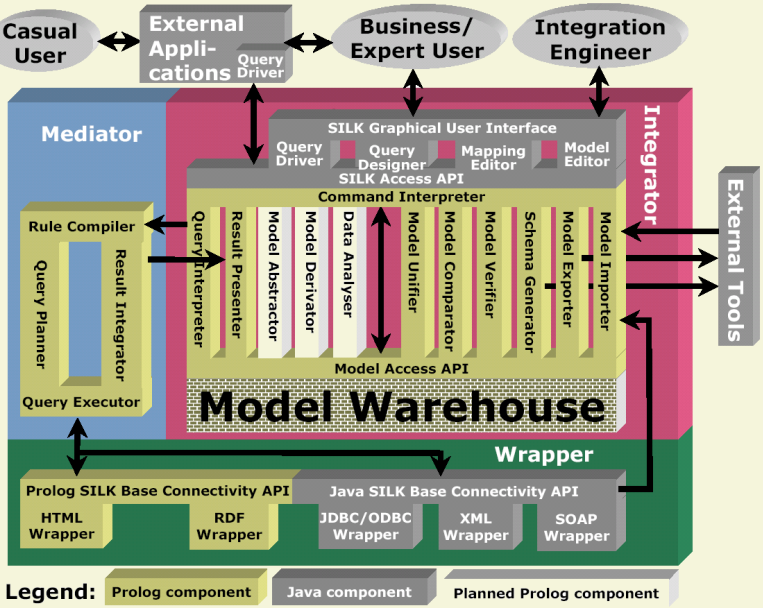
The Context of SILK



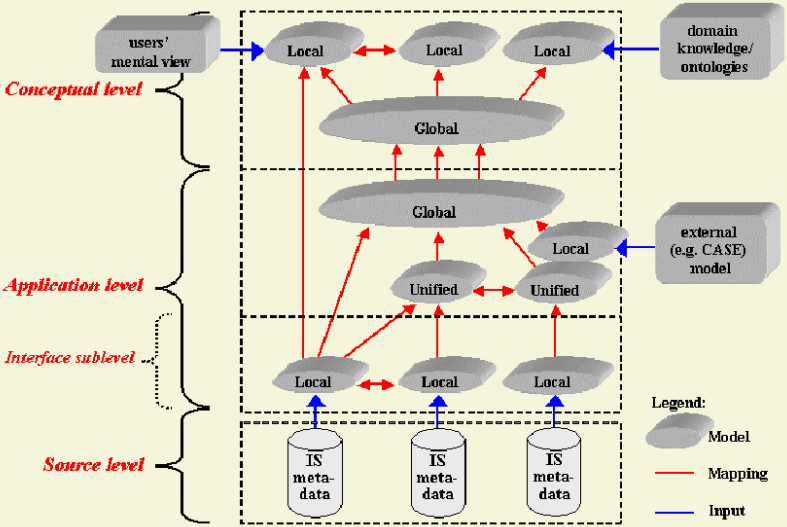
SILK Functionality

- store meta-data, access data on the fly
  - single information source view
  - data location transparency
  - no restriction on the type and structure of the information
- modelling solution
  - import, modify, export object-oriented models
  - establish mappings between models
  - compose queries (and export query results)
- reasoning capabilities
  - model comparison
  - model (in)consistency checking
  - model unification
  - query and mapping composition support

SILK Architecture



Structure of the Model Warehouse



Components of the Model Warehouse

- Models (UML class diagrams)
  - *local*: model of single system or a user view
  - *unified*: model covering several other models
- Mappings (OCL constraints)
  - *correspondence*: constraint linking model elements on the *same* level of abstraction
  - *abstraction*: constraint linking model elements on *different* levels of abstraction
- Queries (SQL with OCL expressions)

SILK modelling example

```
model Factory {
  class Product {
    attribute String serial;
    primary key serial;
  };

  class Car: Product {
    attribute String make;
    attribute Integer price;
    constraint self.wheel.size>2;
  };

  class Wheel: Product {
    attribute Integer size;
  };

  association 'Car-Wheel' {
    connection Car;
    connection Wheel composite;
  };
};
```

Model semantics

- *class*: set of *instances*
- *association*: set of *links*
- *attribute*: function from the class to the type of the attribute
- *operation*: function from the product of the class and the domains of the parameters to the type of the operation
- *connection*: end-point of an association
- *inheritance*: derived class is the subset of the base class
- *primary key*: a set of attributes uniquely identifying an instance of a class
- *constraint*: an invariant for all instances of a class or links of an association

Mapping example (an abstraction)

```
abstraction (s: Peugeot::Vehicle -> c: Factory::Car) {
  constraint s.type = "Car" implies
             s.serial_number = c.serial and
             "Peugeot" = c.make and
             s.price*1000 = c.price;
};
```

- abstractions give the necessary rules to create new instances (links) of classes (associations)
- *suppliers*: the sources of the information (here `s`)
- *clients*: the new instances or links (here `c`)
- in this case we create “abstract” cars from Peugeot vehicles of type `Car`
- OCL constraints for necessary conditions (`s.type = "Car"`) and conversions (`s.price*1000 = c.price`)

Mediator Logical Language

- A simple and uniform representation of the contents (both structural properties and constraints) of the Model Warehouse
- A set of *Description Rules* of the form

$$\forall \overline{X}.(p_0(\overline{X}_0) \wedge \ \ldots \ \rightarrow \ \exists \overline{Z}.q_0(\overline{Y}_0) \wedge \ \ldots)$$

  where $\overline{X} = \bigcup_i \overline{X}_i$, $\overline{Y} = \bigcup_i \overline{Y}_i$, and $\overline{Z} = \overline{Y} \backslash \overline{X}$.
- Quantification is implicit

Example: Abstraction

The condition consists of the existence of the suppliers in conjunction with the condition part of the top-level `implies` (if any). The consequent part is the right hand side of the top-level `implies`

```
'Vehicle'(SNo,'Car',Price)
      ---> 'Car'(SNo,'Peugeot',CPrice), CPrice = 1000*Price
```

Model comparison

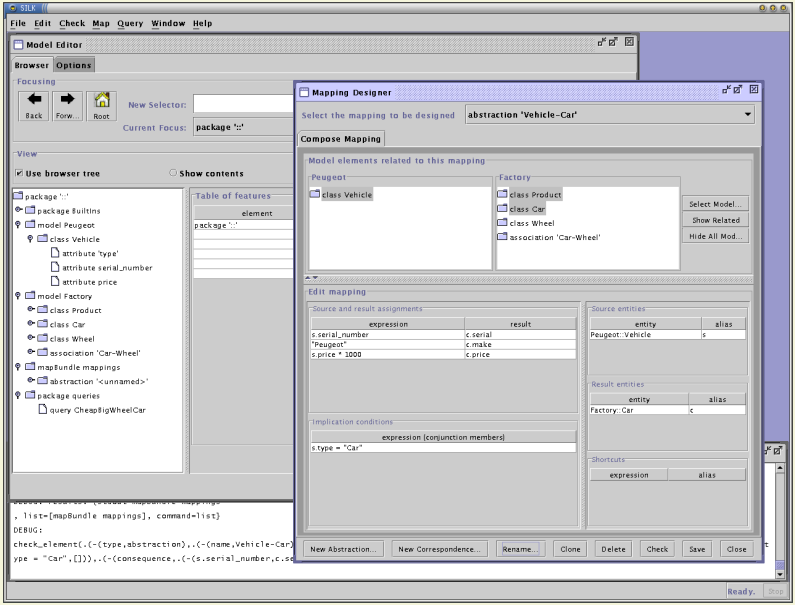
Task

- Find and connect similar elements in two sets of model elements.
- Mathematically: compare two graphs with different kinds of nodes and leaves.

Functionality

- compare models
  - on the same or on different levels of abstraction
  - with a mix of UML and DL constructs
- connect matching elements by *links*
- label links with default constraints (to be refined by the Integration Engineer e.g., by using Mapping Designer)

Result of using the Mapping Designer



Model verification

Task

- check model elements for inconsistency
- provide explanation for the inconsistency

Functionality

- convert OCL constraints and UML structure to first order logic
- check classes and associations for emptiness
- check correspondences and abstractions for satisfiability
- work solely on the meta level, i.e., without considering the data in the information sources

Model unification

Task

- combine several models into a new one
- link the old and new elements to facilitate mediation

Functionality

- create new model elements
- according to unification policy (e.g., create the union or the intersection of models being unified)
- taking into account correspondences
- introducing default abstractions

Future work

Complete DL (Description Logic) support

- extend the RDF support to DL support
- support DL on all levels of SILK

Tighter J2EE coupling

- integrate into J2EE application servers
- provide standard J2EE communication interfaces

Develop an agent architecture

- make the communication with SILK easier for mobile applications
- make SILK-components more autonomous