ComeIn 4.1 Semantic Interfaces

run Software-Werkstatt GmbH
Köpenicker Strasse 325
12555 Berlin

www.run-software.com
Tel: +49 (30) 65762791
e-mail: run@run-software.com

Berlin, August 2010

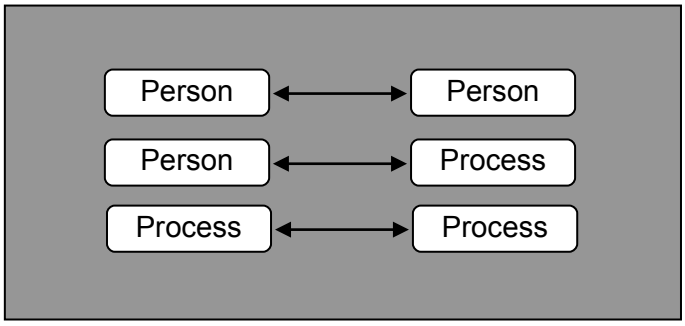
1 Semantic Interfaces

The complexity of communication in information technology is characteristic for the last ten years. A number of technical solutions have been provided as technical standards to improve the communication facilities especially between processes. However, standards as SQL, CORBA, DCOM define a common syntax, only. There is no way to support semantic standards based on standard terminology.

This paper gives a short introduction about the meaning of semantic interfaces. Several reasons for the urgent need of semantic interfaces are shown. Finally, principles for implementing a semantic interface are shown referring to ComeIn as an example for a semantic interface.

1.1 Why do we need a semantic interface?

Communication requires always an interface, an agreement that allows exchanging information between objects. We consider here only persons and processes as communicating objects, even though there exist many other communicating objects. With this restriction we can differ between three types of communication: "Person to Person", "Person to Process" and "Process to Process" communication.



“Person to Person” communication has been highly standardized during human’s history by means of natural language. There are about hundred local standards (languages) for local communications. During the last three centuries three languages have become a type of worldwide standard. Once more, during the last century English became more and more an international standard.

“Process to Process” communication is not well developed from this aspect, however, at least common syntaxes as C, SQL have been defined. In contrast to natural language there is no general agreement on semantics in process languages. Programmers choose names for program variables or database attributes according to their taste. When reading an SQL statement or a C-program it is in general not possible to interpret the meaning of the syntax because of using very specific technical names (terms). This causes problems not only in “Person to Process” communication but also in ”Process to Process” communication. You cannot run the same SQL query on Swedish census data and on US census data because of using different names, even though these databases may display the same variables.

⇒ *A semantic interface is an agreement on terms and meaning, a language within a group of communicating objects. In this case the language is not only defined by its syntax but also by its semantic. Without semantic interface communication is impossible.*

1.2 Communication becomes more complex

Considering “Person to Process” and Process to Process” communication we can see that communication became much more complex during the last 5-10 years. New information technologies allow us building quite complex systems as required for different purposes (e.g. to build complex administration or metadata systems).

Common technical interfaces and object-orientation allow us increasing the development speed by 10-20 times during the last ten years. Development of complex applications becomes possible in a very short time.

> **Example** The Bridge System as metadata system for statistical offices has been developed with resources of 3 PY (person years). With this human resources
>
> - 120 persistent object types with about 1,000 attributes and 300 relationships
> - 350 internal object classes with more than 3,000 functions
> - 350 dialogues and forms
>
> have been implemented (including analysis and documentation). Just the Bridge model definition describing database objects and properties contains more than 150 pages of documentation, i.e. 5 PD (person days) for implementing one persistent object type.

Today we count with 2-5 days for developing a new persistent object with about 20 properties (attributes and relationships). This includes analyzing, documentation and implementation (general behavior, GUI functions, document generation).

There is not only the possibility to build complex systems but also a requirement to do so. A good example are metadata systems that are providing descriptive information about data (variable definitions, classifications etc.) as well as technical information (record structures, data types). Metadata databases become a type of knowledge databases where most of conceptual information for an organization is stored. Metadata systems are complex systems by nature. The Bridge system as a statistical metadata system is based on a database with about 120 conceptual object types and it will grow up to 300 or 500 conceptual object types during the next years.

Hence, today it is necessary and possible to develop systems with more than hundred objects within a few month – with the result that nobody understands the system. The reason is that the model uses special names invented by programmers. This terminology does not correspond to the conceptual terminology of users (also application programmers in statistical offices found it very hard to understand the Bridge model even though it seems to be well documented and quite clear from a logical point of view).

⇒ *Complex data models require a semantic interface to enable technical and subject oriented persons to communicate with the database. The database must be accessible by means of conceptual terminology.*

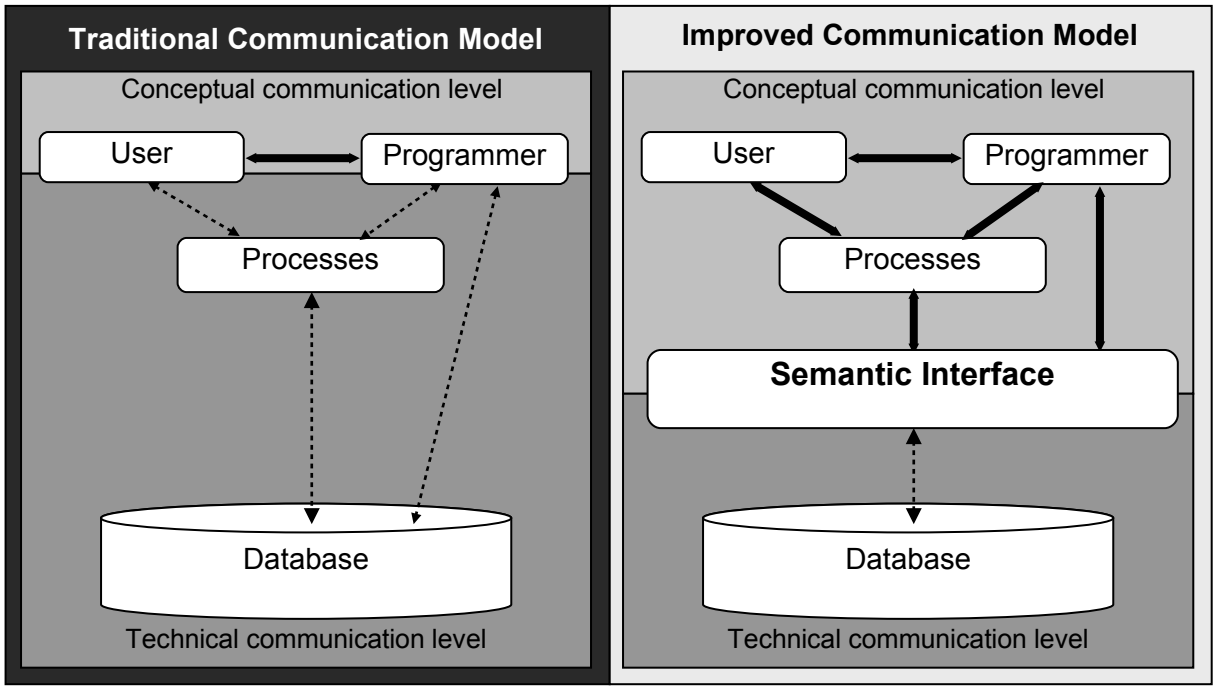
1.3 Requirements for data models and communication

Differences between data model and terminology are not only a question of names for object types and attributes. Communication reflects different views to reality.

> **Example** A table generation process has a quite different view to metadata than an interviewing process and this differs again from a maintenance process view. What is called dimension in the table generation process could be called variable in the interviewing process just expressing different views to the same phenomenon. In the database it may have the name tab_dimension.

To support different views the data model must reflect the reality. Hence, data model requirements will always differ from communication requirements and the technical (database) terminology (names for object types/relations and properties/attributes) will differ from the conceptual terminology.

We can consider a database to be a special process that provides information to other processes or persons. Hence, databases will be involved in communication in different ways. Unfortunately, data models will change frequently to correct mistakes (that always happen), to introduce new features and to reflect changes in reality. However, semantics as base for communication are quite stable and change slowly. The difference in database and communication requirements can be solved by an appropriate semantic interface that acts as a translator between persons and processes and databases



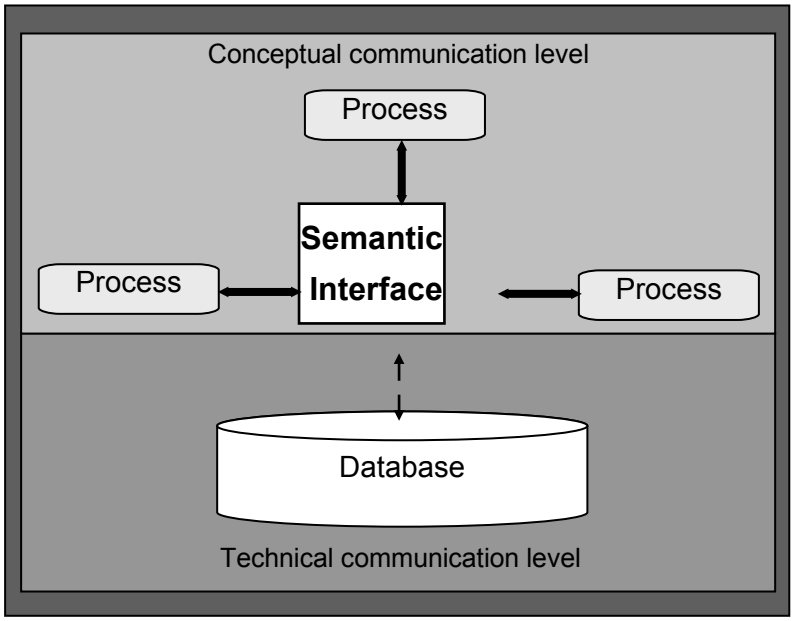
The conceptual communication level is based on natural language terms and semantics. This is used to persons and much easier to handle then communication on the technical level. Introducing a semantic interface directly on top of the database we can improve communication between persons and processes by communication on the conceptual level, i.e. by means of a defined terminology. Moreover, processes become database independent and databases can be replaced easily.

⇒ *Because the database terminology is different from natural language (person's terminology) a semantic interface is required at least between persons and databases.*

1.4 Communication between processes

This is not only a problem for “Person to Process” communication but also for “Process to Process” communication. Because each process has its own technical language processes can communicate only when they belong to the same family or when one process knows the semantics of the other one.

The situation has changed because processes refer more and more to metadata instead of using process inherent meta-information. But how shall one process ask the other one to get information about a variable without knowing the process specific term for variable (that could be “dimension”, “indicator”, “tab_var” or anything else). Thus, also processes will benefit from a semantic interface that defines the semantic rules for processes.



On the one side processes are able to communicate via semantic interfaces with each (meta) database by means of defined conceptual terms. Because the semantic interface translates conceptual terms into appropriate technical terms the processes can communicate independently on the specific data model that defines semantics on the technical level.

⇒ *Designing processes based on a semantic interface creates a family of universal processes. These processes can communicate with any process and with any database supporting the semantic interface.*

1.5 Conclusions

1. A semantic interfaces is a natural way to handle the increasing complexity of information structures.
2. Semantic interfaces are one step to “Person to Process” communication via natural language.
3. Semantic interfaces used in “Process to Process” communication will increase the flexibility of standard software. Metadata driven standard software becomes universal and fits into any environment.
4. Metadata driven applications and standard software becomes (metadata) database independent.

5. Modern technologies as object orientation (object-oriented databases) and standard interfaces (COM, CORBA, XML) are the technical background that makes it possible to build and to use semantic interfaces.

2 Implementation of a Semantic Interface

Term based semantic interfaces could be a good starting point for developing semantic interfaces, even though semantics are not only determined by the meaning of terms. Providing a term based semantic interface requires two steps:

- Agreement on standard terminology in a communication group
- Providing a mapping between technical and conceptual communication level

For defining the terminology a simple method is suggested that defined global terms for basic concepts (logical object types) and context related terms to define properties for basic concepts. As a technical solution and an example for mapping technical and conceptual terms we will refer to ComeIn (Common Metadata Interface) in this section.

2.1 Defining conceptual terms

The main task is to define standard concepts and associated terms (standard terminology). Terminology has to be agreed upon in the family of users that are going to communicate with each other. That is not always simple because different terms are used with different meanings and a standard for internal communication has to be defined. Thus, it is not the intension to provide a worldwide terminology standard but to start with groups of communicating objects (persons).

In this chapter we will give a short explanation of the method defining concepts and terms. Concepts are associated with standard terms. For each term that defines a basic concept a number of concept related terms are used to define properties for the basic concept. While terms referring to basic concepts should be unique concept related terms need not, i.e. concept related terms may differ in meaning when defined as properties for different basic concepts.

The following example is part of the terminology definition for classifications defined by several national statistical offices in Neuchatél, June ’99:

> **Classification** Classification defines the general idea of classifying statistical *observation units* in a given population. A classification is a structured list of mutually exclusive categories to describe all units of a defined population according to a defined property.
>
> A classification is represented by one or a number of consecutive classification versions which may be linked over time by correspondence tables.
>
> > **name:** A classification is identified by a unique name which may typically be an abbreviation of its title.
> >
> > **title:** A classification has a title as provided by the owner. The title usually indicates the scope of the classification.
> >
> > **owners:** The statistical office or other *authority* which created and maintains the classification. A classification may have several owners.

Each term is described by a textual definition. This may refer to other concepts as e.g. *observation unit*. Concept related terms that refer to other concepts are usually referenced by the concept name except the property expresses a special rule of the referenced concept (as e.g. “owners” that reflect the special rule of an authority as property in a classification).

Here it is also the possibility to define synonyms for basic concept and property names. This makes it easier to find agreements in terminology but it makes it more difficult to find appropriate terms for new basic concepts that must be unique again.

2.2 Mapping conceptual terms to technical terms

One solution of mapping conceptual terms to technical has been provided with ComeIn (Common Metadata Interface). ComeIn is a running example for semantic interfaces that is used to interface statistical metadata by standard software, statisticians and costumers of statistics. However, ComeIn provides a technique that allows defining a semantic interface related to any type of communication group. The conceptual level (terminology) of the current ComeIn interface is based on agreements between several national statistical offices in Europe.

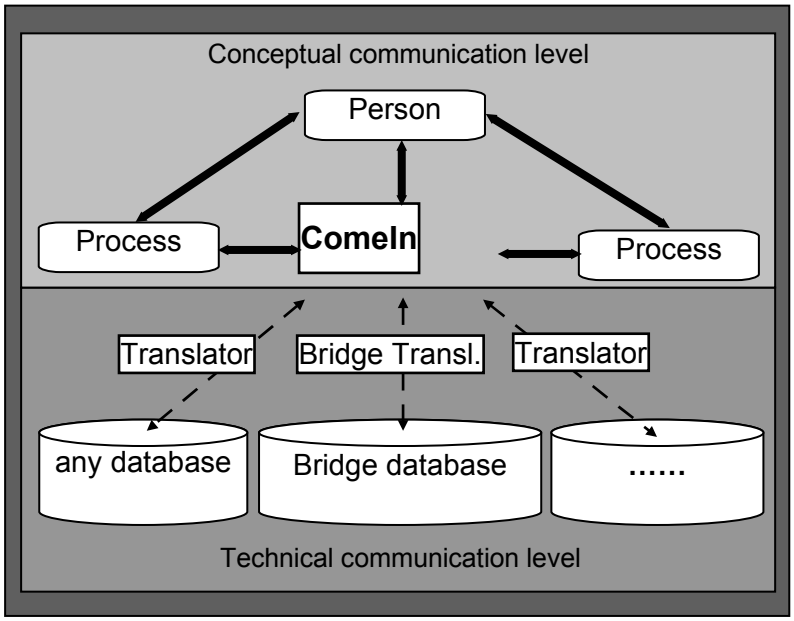
In ComeIn terms for basic concepts result in **logical object types**, terms for concept related properties become **logical properties**. ComeIn is using the same terms as defined in the conceptual definition (agreement on terminology). The minor changes on terms are:

1. Terms for basic concepts that consist of several words are written without spaces (e.g. the conceptual term “Classification Version” is referenced as logical object type “ClassificationVersion” in ComeIn).
2. Terms for concept related properties that consist of several words are connected by underline characters (e.g. the concept related term "units classified" referenced as logical property name “units_classified” in ComeIn).
3. Names are limited to 40 characters.

Thus, the logical ComeIn data model corresponds exactly to the terminology agreement. In the second phase the logical ComeIn data model might be expanded by additional logical properties that result from another basic rules.

1. Each relationship to another concept (e.g. “classification” refers to “authority” as “owner”) must have an inverse relationship in the referenced concept (e.g. “classifications” as property for the basic concept “authority”).

Each database supporting ComeIn has to provide a translator module that translates conceptual terms in relation or object type names and conceptual property names in technical property names. The translator module translates the conceptual term into a technical term. This can be a database object or attribute name as well as a function. The concept related term “number of variables” for an “Administrative Register” (basic concept) could be



translated into **CL_Register.items.GetCount()** referring to a function instead to a stored attribute. In other cases the translator may translate directly to a property or return a non-response if the property for a basic concept is not available in the database.

This is a simple implementation of a translator module that just maps the conceptual terms to technical property names or methods. However, the conceptual level is more sophisticated and required enhanced methods for translating conceptual terms to technical ones.

Context sensitive interfaces

A semantic interface must react in a context sensitive way. Asking for a “variable” in a general context we can expect to get the “definition” for the variable but not the “position” in a data record. Only when retrieving “variable” information in the context of a “Record” we can expect a value for the “position”. This is not a problem on the conceptual level but the database and the translator module have to handle such situations.

Extended property search

In some cases a basic concept (e.g. “Person”) has just a limited number of properties defined. But more special concepts as e.g. “Employee” may have additional properties as “income”. Asking a “Person” for its “income” the translator will probably return a non-response. Extended property search is a feature that searched for all possible properties that are associated with a concept even though the property is not directly defined for the current concept (or logical object type).

Moreover, Extended property search allows to search for extension attributes. This are attributes that are added by the user to store information that are not supported by the semantic interface or by the data model. Extension attributes are attributes that are added on demand, i.e. if a user finds it remarkable that a person owns three cars he can add a property “number of cars” to store this information. This can be retrieved later on as a normal attribute even though it is neither part of the logical ComeIn data model nor of the physical database model.

Those features are not yet supported by the ComeIn-Bridge translator but will be supported in the near future.