Context Comparison for Object Fusion

Fabrice Jouanot[1], Nadine Cullot[2], and Kokou Yétongnon[2]

[1] EPFL, Database Laboratory
Fabrice.Jouanot@epfl.ch
[2] University of Burgundy, LE2I
{Nadine.Cullot,Kokou.Yetongnon}@u-bourgogne.fr

Abstract. We propose a solution to help the integration of heterogeneous sources based on the fusion of objects according to their context. New requirements for information exchange have emerged with all the developments around Internet. Information consumers want to access and combine data from remote and heterogeneous sources in a transparent and dynamic way. To achieve this level of interoperation is yet a real challenge. We present a model to define local data as informative objects with a contextual representation associated to them. A semantic context comparison mechanism, based on a semantic distance, reconciles context of applications and constructs virtual objects in which rules make the fusion of informative objects. Virtual objects play the role of interface for transparent user access.

1 Introduction

Cooperation of information systems is greatly influenced by the development of the World Wide Web. Models, methodologies and tools have to be adapted to the new requirements of the cooperation end-users. Help them to find their way in the jungle of the knowledge providing by the Web is really suitable. To reach this major aim, it is necessary to deal with the different kinds of heterogeneities that can be encountered in a cooperation of information systems. We agree that syntactic and structural heterogeneities which concern model and data structure differences can be solved using translator tools, pivot models or other solutions proposed in the literature. The real challenge is now to take into account data semantics to find relevant information. In IS cooperation, different approaches have been developed based on federated or mediator/wrapper architectures. Mediation approaches are well adapted to dynamic environments. In schema mediation, the mediator localizes and integrates relevant data to construct a mediation schema which can be queried by end-users. Projects such as DIOM [1], DISCO [2], DOK [3] are schema mediation solutions. In context mediation, the mediator dynamically integrates information to answer a end-user query. SemWeb [4], On2Broker [5, 6], Observer [7], WebFindit [8] or InfoSleuth [9, 10] are context mediation solutions. These projects propose different solutions to modelize data semantics. On2Broker is interested in the semantics of Web documents which can be in different formats (RDF, XML-like, ...). The solution includes four basic engines (query engine, info agent, inference engine, and database manager)

J. Eder and M. Missikoff (Eds.): CAiSE 2003, LNCS 2681, pp. 536–551, 2003.

which are all relied on ontologies. WebFindIt defines a complete architecture to access to different databases by browsering their application domain using their associated metadata (co-databases). Metadata, ontologies and contexts are tools to modelize data semantics. Metadata describe intrinsic properties of data. The Dublin Core [11] is an example of metadata about document exchange. Ontologies are more and more recognized as a promising solution to modelize data semantics. Ontologies can be viewed as the formal modelization of shared concepts. Projects such as DOGMA [12] or KAON [13] propose formal models to modelize ontologies. Nicolas Guarino [14] classifies ontologies according to their task and point of view. Top-level ontologies (1) describe very general concepts. Domain ontologies (2) describe a vocabulary related to a generic domain. Task ontologies (3) describe a task or an activity. And application ontologies describe concepts depending on a particular domain and a task. They can be viewed as a specialization of task and domain ontologies. Contexts are used to describe semantics of data relied to one or more application in a specific domain. With the Guarino's classification, contexts can be compared to application ontologies. The development of metadata, contexts or ontologies leads to other related requirements which are needs to compare these ontologies or contexts, to be able to communicate in a large way. Some recent works focus on the definition of measures to compare terms [15], concepts or ontologies [16].

The DILEMMA (Dynamic and Interoperable Logical Extended Mediation Model Architecture) [17] project described in this paper is an hybrid approach of schema and context mediation which allows information system (IS) cooperation using contexts to modelize data semantics. The main idea is to provide tools to build a cooperation which can be updated in an incremental manner according to end-user requirements. DILEMMA proposes a model to describe data with their meanings and a semantic reconciliation method which is based on the definition of a semantic distance to compare contexts. In this approach, a cooperation between two IS can be summarized in four steps. In the first one, provider and consumer systems have to describe their own application domain to make the meaning of their data or requirements understandable and unambiguous. This is done using a model (section 2) which allows the definition of information via Informative Objects (I-Objects) and the description of a context for these I-Objects via the definition of concepts. The second step is a context comparison between two contexts which aims to identify their similar concepts using a semantic distance (section 3). The third step considers the results of this comparison to choose relevant I-Objects which can be interpreted in a consumer context (section 4.1). The last step consists in an object fusion process (section 4.2) which integrates semantically related I-Objects into virtual I-Objects to access to all the data of the cooperation in a transparent way.

2 Objects and Contexts

The data description is made up of three layers with a progression in the semantic granularity from high level terminological concepts to value understanding. The

first layer consists in mapping the local data (if exist) into a pivot model to structure information and to simplify exchanges. This layer defines I-Objects as interface to access data. The second layer is the definition of a local context to deal with the semantics of the application domain. It is built using a set of terminological concepts and roles. The third layer clarifies the I-Object meanings on the defined context. Interpretation rules allow to interpret I-Objects on the context and characterization rules allow to explicit intrinsic properties of data values (as unit, scale, precision).

2.1 Definition of I-Objects

An I-Object class definition is composed of a profile and may be a deductive rule description. A profile $Classname[Method_1 => Type_1; ...; Method_n => Type_n]$ defines a set of attributes called methods which can be mono-valued $Method => Type$, multi-valued $Method =\gg Type$ or parameterized $Method(Parameter -> Type) => Type$. Access rights (*public, private and shared*) can be added to control importation, exportation and reuse of the I-Object class. All defined I-Objects inherit from a basic class called Very First Object (*vfo*) which provides four common methods : *repository* to precise the URL of the source, *description* to give a literal definition of the class, *version* number to find right I-Objects and *Mproperty* a method used for the characterization of data values. An I-Object is an instance of an I-Object class. The term I-Object is sometimes used instead of I-Object class by misuse of language. If the I-Object class is a virtual I-Object class then a deductive rule completes the definition of the class. The rule allows to define how to combine I-Object instances of existing classes to build this new virtual one.

The first example below shows the definition of an I-Object class *person*. It inherits from vfo, has a parameterized method *firstName* and has a complex method *address*.

```
class person::vfo[ repository -> '192.52.237.1';
                   description -> 'A person is a human described by ...';
                   version -> 1].
class person[ name => string; firstName@(num => integer) => string;
              ssn => string; address => Address; tel =>> string ].
class Address::vfo[ repository -> '192.52.237.1'; ... ]
```

The second example below shows the definition of a virtual I-Object class *lausannois* which inherits from the class *person* and describes the inhabitants of the city of Lausanne.

```
lausannois::person.
rule(X,A) X:lausannois :- X:person[ address -> A ],
                          A:Address[ city -> 'Lausanne' ].
```

In addition to these I-Objects, the model defines function libraries where specific transformation functions can be stored. These functions are useful to resolve semantic mismatch of method values between different I-Objects and they

play an important role in the object fusion process. The example below shows the definition of a library *DateConverter*. It contains one function *age2date* which returns a date of birth from an age. A class *student* is also defined. It inherits from a class *person* and has a method *ybirth*. This method is calculated from the value of the method *age* using a deductive rule. The rule gives the value of the method *ybirth* for a *student* using the function *age2date*.

```
library DateConverter[ age2date@(age -> integer)=> date ].
class student::person[ age => integer; Netud => string;
                       inscrip =>> Inscription; ybirth => integer].
rule{A, AN, X} X[ ybirth -> AN ] :- X:student[ age -> A ],
               DateConverter[ age2date@(age -> A) -> AN ].
```

2.2 Definition of a Local Context

A context is composed of concepts called conceptual classes. A conceptual class is defined by its name and a list of its roles. They describe the roles played by the other classes for this conceptual class. If $\Phi_{\mathcal{D}}$ the set of conceptual classes on a domain $\mathcal{D}$, $Cc \in \Phi_{\mathcal{D}}$ is defined by :

$$\mathbf{Cc} = \mathbf{concept\ name}\{+\mathbf{synonym\_list}, -\mathbf{antonym\_list}\}[\ \mathbf{role\_list}\ ].$$

- **name** the name of the conceptual class. It represents a terminological definition of the concept and it depends on the language;
- **role_list** describes the semantic surrounding of a concept. A role defines a semantic relation between the concept which owns it and concepts which play this role. The set of roles played by other classes for Cc is noted $\mathcal{R}_{Cc}$. If $r_i \in \mathcal{R}_{Cc}$, r_i is defined as below :
  - $\mathbf{r_i}\{+\mathbf{synonym\_list}, -\mathbf{antonym\_list}\} => \mathbf{Cc_{r_i}}$ with $Cc_{r_i} \in \Phi_{\mathcal{D}}$ describes that Cc_{r_i} plays the role r_i for Cc.
  - $\mathbf{r_i}\{+\mathbf{synonym\_list}, -\mathbf{antonym\_list}\} => \{\mathbf{Cc_{r_ij}}\}$ with $\forall j = 1..n, Cc_{r_ij} \in \Phi_{\mathcal{D}}$ describes that Cc_{r_ij} plays the role r_i for Cc.
- **synonym_list** defines a set of terms which are synonyms of a concept or a role. It allows to enlarge the meaning of an entity and can optimize the comparison between two entities.
- **antonym_list** defines a set of terms which could be consider as synonyms but have an opposite meaning in this context.

Finally conceptual classes can be organized in inheritance hierarchies which allow role inheritance with some constraints. The example below depicts a part of a context definition. The concepts *life form, human, individual, etc.* are introduced. They are linked togother with roles. For example, the concept *birth date* plays the role *birth* for the concept *life_form*. The context can be viewed as a local ontology of the application domain.

```
concept life form[ birth => birth date; death => decease date; age => Age ].
concept human{+man}::life form.
```

```
concept individual{+person,+guy}::human
                [ identifier => identity; home{+house} => address ].
concept address{+place,+localization}
               [ component => {number, street, post code, town} ].
concept identity[ identifier => social number;
                  name => {first name, last name} ].
concept christian name::first name.
```

2.3 Definition of Interpretation Rules

Rules allow to interpret the profile of an I-Object class i.e. to describe its meaning. Three categories of rules can be distinguished : simple rules, method rules and rules with constraints.

A simple rule specifies that the meaning of an I-Object class can be found in a particular concept. It links an I-Object class to a conceptual class. For example, the class *person* can be interpreted on the concept *individual* as follows.

```
inter person:::individual.
```

A method rule specifies the meaning of the values returned by a method. It links a method of an I-Object to a conceptual class. For example, the methods *name* and *ssn* of the class *person* can be interpreted on the conceptual classes *last name* and *social number*. Parameters of a method have also to be interpreted in a similar way. The example below lists some interpretations of methods.

```
inter person.name:::last name.
inter person.ssn:::social number.
inter address.cp:::post code.
```

An interpretation rule with constraints allows to interpret any sub-set instances of an I-Object class. The example below specifies that the first name of a person can be interpreted as a Christian name.

```
inter{X,Y} X{Y:person[firstName@(num->1)->X]}:::christian name.
```

Moreover, interpretation rules can be completed by an interpretation path which can avoid ambiguities of interpretation in complex contexts. This expression below specifies that the method *name* of the class *person* is interpreted on the concept *last name* when it plays the role *name* for the concept *identity*, which itself plays the role *identifier* for the concept *individual*.

```
inter person.name:::last name <= identity (<= individual.identifier).name
```

2.4 Definition of Characterization Rules

The model provides some metadata which represent the most used types of data in usual information systems. Metadata are represented with meta conceptual classes, called MCc. Two hierarchies of MCc are pre-defined : one for spatial data representation and another for classical data representation such as time,

geometry, mass, electricity and many more-sub types of numerical values. Part of the hierarchy for classical data is presented in the figure 1. The meta conceptual classes Geometry, Mass, Time, etc. inherit the roles and semantic surrounding from the root MCc Numerical Datum Reference Information.

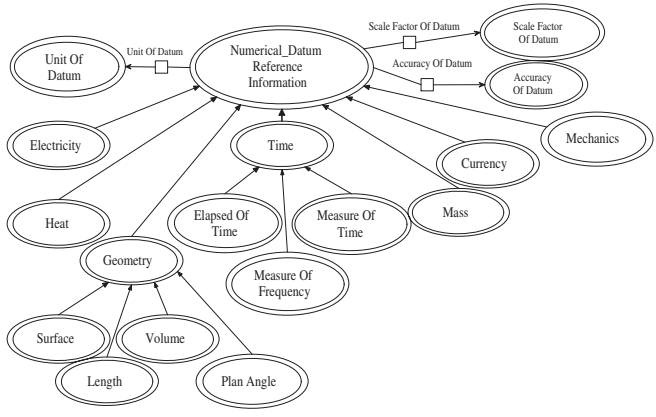

Fig. 1. Part of the hierarchy of classic meta conceptual classes.

This root MCc is relied using roles to three MCcs Unit Of Datum, Scale Factor Of Datum and Accuracy Of Datum to define the intrinsic properties of a numeric value which are respectively its unit, scale and precision. These MCcs are called terminal classes and have got a list of values, terms or expressions which can be used to parameterize the characterization of the I-Objects values .

The figure 2 presents the characterization of the values of the method *age* of the class *student* upon these metadata. The method *age* is interpreted on the conceptual class *Age* which derives from the from MCc *Elapsed Of Time*. The MCc *Elapsed Of Time* inherits from *Time* which inherits from *Numerical Datum Reference Information*. The method *age* is characterized specifying its unit *year*, scale *1* and precision *0* which means that the value is exact.

Formally, the characterization of a method is achieved giving the characterization of its values on each terminal meta-class (property) which plays a role for the MCc on which this method is indirectly interpreted. The example below gives the formal definition of the method *age* of the I-Object class *student* as described graphically in figure 2. The method *MProperty*, inherited from *vfo*, allows the characterization of the method *age* of the class *student* on its unit (1), scale (2) and precision (3). Each characterization is done with the method Mproperty, giving the name of the method, the terminal class considered and the characterization value.

```
concept Age::Elapsed Of Time.
inter student.age:::Age.
(1)  class student[ MProperty('age','Unit Of Datum') -> 'year' ].
```

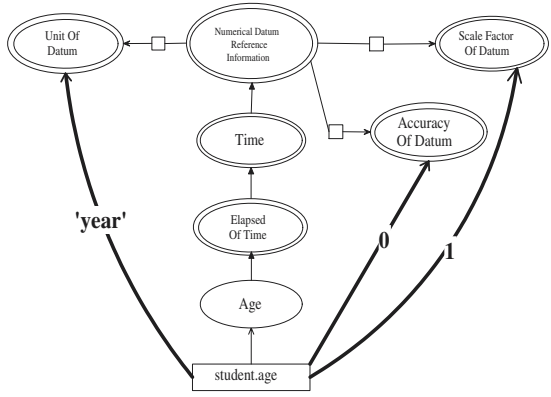

Fig. 2. Characterization of a simple value.

```
(2)  class student[ MProperty('age','Scale Factor Of Datum') -> 1 ].
(3)  class student[ MProperty('age','Accuracy Of Datum') -> 0 ].
```

To ensure the correctness of the model, the method *MProperty* has also to be interpreted on the context as any other method. This interpretation implies a complex interpretation rule as shown below with the interpretation of the method MProperty of the I-Object class *student* for the unit of the method *age*. The result of *MProperty* (identified by variable X) is interpreted on the concept *Unit Of Datum* when it plays the role *Unit of Datum* for the concept *Elapsed Time*.

```
class student[ MProperty('age','Unit Of Datum') -> 'year' ].
int{X,Y} X{Y:student[ MProperty@('age','Unit Of Datum') -> X ]}
           :::Unit Of Datum <= Elapsed Of Time.Unit Of Datum.
```

3 Context Comparison

To be able to define context comparison, it is necessary to first define a semantic distance to measure the similarity between two concepts of two contexts. The calculation of a semantic distance between two conceptual classes can be defined considering the taxonomy and the semantic surrounding of these classes. More formally, if Cc and Cc' are two concepts from different contexts,

- $D_{cc}(Cc, Cc')$ is a basic distance which gives the taxonomic equivalence between Cc and Cc',
- $\boldsymbol{R}_{Cc,Cc'}$ is a vector which gives a summary of roles played for Cc and Cc',
- $D_{rj}(Cc, Cc')$ is a basic distance which evaluates roles played by Cc and Cc' (in a matrix),

then the semantic distance between Cc and Cc', noted $D(Cc, Cc')$, is defined as follows :

- **case 1,** if $D_{cc}(Cc, Cc') = 0$ then $D(Cc, Cc') = 0$
- **case 2,** if exists a set of conceptual classes playing roles for Cc and Cc', if exists a set of conceptual classes for which Cc and Cc' played roles for, then

$$D(Cc, Cc') = \tfrac{1}{2} D_{Cc}(Cc, CC') + \tfrac{1}{2}(\alpha \times \frac{\sum_{i=1..n} \boldsymbol{R}_{Cc,Cc'}(i)}{n} + \beta \times D_{rj}(Cc, Cc'))$$

  with $0 < \alpha < 1,\ 0 < \beta < 1$ *and* $\alpha + \beta = 1$.

  α and β allow to weight the two parts of the semantic expression to give more or less importance between both surrounding parts.
- **case 3 and 4,** if one of the set of conceptual classes "roles played for Cc and Cc'" or "roles played by Cc and Cc'" does not exist, then $D(Cc, Cc')$ is defined respectively as

$$D(Cc, Cc') = \tfrac{1}{2} D_{Cc}(Cc, CC') + \tfrac{1}{2}(\frac{\sum_{i=1..n} \boldsymbol{R}_{Cc,Cc'}(i)}{n})$$

  or

$$D(Cc, Cc') = \tfrac{1}{2} D_{Cc}(Cc, CC') + \tfrac{1}{2}(D_{rj}(Cc, Cc'))$$

- **case 5,** if no set of conceptual classes "roles played for Cc and Cc'" or "roles played by Cc and Cc'" exists, then $D(Cc, Cc')$ is defined as

$$D(Cc, Cc') = D_{cc}(Cc, Cc')$$

This global distance $D(Cc, Cc')$ uses other specific distances to calculate the distance between two concepts. We briefly present all these distances. First $D_{MCc}(Cc, Cc')$ (used in the calculation of D_{cc}) is a basic distance which evaluates the semantic distance between two conceptual classes Cc and Cc' according only to their relationships with meta conceptual classes. In fact, $D_{MCc}(Cc, Cc')$ is true if the meta conceptual classes, from which the compared concepts derived, are semantically compatible, and false if not.

If $Cc_1,\ Cc_2$ two concepts from a context, $Cc'_1,\ Cc'_2$ two concepts from another context, MCc and MCc' two meta conceptual classes, the distance D_{MCc} between Cc_1 and Cc'_1, noted $D_{MCc}(Cc_1, Cc'_1)$, is defined as follows :

$$\begin{cases} D_{MCc}(Cc_1, Cc'_1) = 1 \\ \quad if\ Cc_1 :: MCc\ (direct)\ or \\ \quad\quad Cc_1 :: .. :: Cc_2 :: MCc\ (indirect) \\ \quad and\ Cc'_1 :: MCc\ (direct)\ or \\ \quad\quad Cc'_1 :: .. :: Cc'_2 :: MCc\ (indirect) \\ D_{MCc}(Cc_1, Cc'_1) = 0 \\ \quad if\ Cc_1 :: MCc\ ou\ Cc_1 :: .. :: Cc_2 :: MCc \\ \quad and\ Cc'_1 :: MCc\ ou\ Cc'_1 :: .. :: Cc'_2 :: MCc' \\ \quad and\ MCc \neq MCc' \\ \ else \\ D_{MCc}(Cc_1, Cc'_1) = v\ with\ v = 0\ or\ v = 1 \end{cases}$$

The values of the variable v depend on the particular cases shown in figure 3: this value resolves the uncertainty which appears in these conciliation cases.

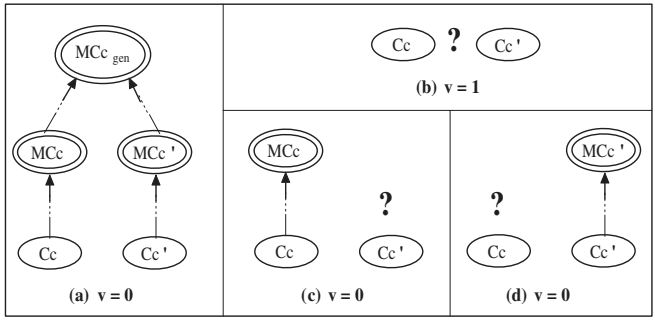

Fig. 3. Particular conciliation cases for D_{MCc} basic distance.

The distance $D_{cc}(Cc, Cc')$ focuses on the taxonomic aspects of concepts Cc and Cc'. Its calculation takes into account three semantic features of a concept: its name with its synonymy and antonymy relationships, its position into the inheritance hierarchy of concepts and its meta-conceptual relationships. In the formal definition below, we consider the general case where the both concepts are placed in a hierarchy. Other definitions exist according to the fact that one of the concepts (or both) is placed or not into a hierarchy. The distance $D_{cc}(Cc, Cc')$ is equal to the previous distance $D_{MCc}(Cc, Cc')$ when all comparison criteria success, and equal to zero if not. In fact the value of $D_{MCc}(Cc, Cc')$ determines the possibility of a similarity between concepts.

If Cc a concept from the consumer context, Cc' a concept from a provider context, $\mathcal{N}(Cc)$ the name of Cc, $\mathcal{S}(Cc)$ the synonyms of Cc and $\mathcal{A}(Cc)$ the antonyms of Cc, then $D_{cc}(Cc, Cc')$ can be formally defined as follows :

$$
\begin{cases}
D_{Cc}(Cc, Cc') = D_{MCc}(Cc, Cc')\ if \\
\qquad card((\mathcal{N}(Cc) \cup \mathcal{S}(Cc)) \cap \mathcal{N}(Cc') \cup \mathcal{S}(Cc'))) > 0 \\
\qquad and\ \forall Cc'_j \in \Phi'_{\mathcal{D}},\ Cc'_j :: .. :: Cc' \\
\qquad\qquad card((\mathcal{N}(Cc) \cup \mathcal{S}(Cc)) \cap \mathcal{N}(Cc') \cup \mathcal{S}(Cc'))) \geq \\
\qquad\qquad card((\mathcal{N}(Cc) \cup \mathcal{S}(Cc)) \cap \mathcal{N}(Cc'_j) \cup \mathcal{S}(Cc'_j))) \\
\qquad and\ \forall Cc_j \in \Phi_{\mathcal{D}},\ Cc_j :: .. :: Cc \\
\qquad\qquad card((\mathcal{N}(Cc) \cup \mathcal{S}(Cc)) \cap \mathcal{N}(Cc') \cup \mathcal{S}(Cc'))) \geq \\
\qquad\qquad card((\mathcal{N}(Cc_j) \cup \mathcal{S}(Cc_j)) \cap \mathcal{N}(Cc') \cup \mathcal{S}(Cc'))) \\
\qquad and\ \forall Cc_i \in \Phi_{\mathcal{D}},\ Cc :: .. :: Cc_i \\
\qquad\qquad card((\mathcal{N}(Cc) \cup \mathcal{S}(Cc)) \cap \mathcal{N}(Cc') \cup \mathcal{S}(Cc'))) > \\
\qquad\qquad card((\mathcal{N}(Cc_i) \cup \mathcal{S}(Cc_i)) \cap \mathcal{N}(Cc') \cup \mathcal{S}(Cc'))) \\
\qquad and\ if\ card(\mathcal{S}(Cc) \cap \mathcal{A}(Cc')) = 0 \\
\qquad and\ card(\mathcal{S}(Cc') \cap \mathcal{A}(Cc)) = 0 \\
\quad else \\
D_{Cc}(Cc, Cc') = 0
\end{cases}
$$

The criteria of this definition precise that a better distance between Cc and a concept inherited from Cc' must not exist, and a better distance between Cc' and a concept inherited from Cc must also not exist.

However the semantics of a concept is contained in its roles : roles played by other concepts for this one, and roles played by this concept to the others. To evaluate this neighborhood similarity, an important operation is the distance between roles D_r. The distance D_r measures the similarity of two roles based on its taxonomy only. If Cc a concept from the consumer context, Cc' a concept from a provider context, r a role of Cc and r' a role of Cc', $D_r(Cc.r, Cc'.r')$ is defined as follows :

$$\begin{cases} D_r(Cc.r, Cc'.r') = 1 \; if \; D_{Cc}(Cc, Cc') = 1 \\ \qquad\qquad\qquad\qquad\qquad and \; card((\mathcal{N}(r) \cup \mathcal{S}(r)) \cap (\mathcal{N}(r') \cup \mathcal{S}(r'))) > 0 \\ \qquad\qquad\qquad\qquad\qquad and \; card(\mathcal{S}(r) \cap \mathcal{A}(r')) = 0 \\ \quad else \\ D_r(Cc.r, Cc'.r') = 0 \end{cases}$$

We will not detail moreover the calculation of the semantic distance. All the information about the calculation of the semantic distance can be found in [17].

4 Object Fusion

We assume that the context comparison process has selected some relevant conceptual classes in the provider context equivalent to some other classes in the consumer context. They have to be imported and interpreted on this consumer context.

4.1 Re-interpretation Problematic

To construct a coherent interpretation of the imported I-Object classes with their profiles and associated function libraries on the consumer context, it may be necessary, to avoid ambiguities, to use interpretation paths. The figure 4 illustrates this problem. The concept *individual* from the consumer context has been evaluated as similar to the concept *person* from the provider context. The interpretation of the method *name* of the class *student* which is *lastname* <= *person.identity* on the provider context cannot be directly imported in the consumer context. A new interpretation path is built using the results of the context comparison. In the example, the method *name* of the class *student* is re-interpreted on the consumer context using the interpretation path *lastname* <= *identity*(<= *individual.identifier*).*name*.

The very first part of a re-interpretation path is given by the step of importation. We suppose the importation of an interpretation on a conceptual class Cc' on Cc (consumer context). The following levels of re-interpretation are constructed in an incremental way, comparing conceptual classes Cc'_j, which appear in the interpretation path of a conceptual class Cc', with conceptual classes Cc_i which appear in the interpretation path of a conceptual class Cc similar to Cc'.

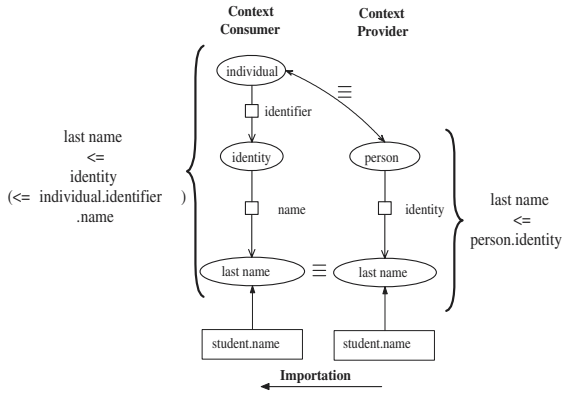

Fig. 4. An example of re-interpretation mechanism.

For each level, we consider two similar conceptual classes Cc_i and Cc'_j, helped with similarities of Cc_{i+1} and Cc'_{j+1} for which Cc_i and Cc'_j play a role. To decide the re-interpretation form, five cases of similarities are considered:

- **case 1 :** conceptual classes for which Cc_i and Cc'_j play a role are similar ($Cc_{i+1} \equiv Cc'_{j+1}$). The level of provider interpretation path $Cc'_j(<= Cc'_{j+1}(...).r'_{j+1}).r'_j$ becomes $Cc_i(<= Cc_{i+1}(...).r_{i+1}).r_i$ in the consumer context.
- **case 2 :** conceptual classes for which Cc_i and Cc'_j play a role are not similar but a unique interpretation path exists in the consumer context which carries out to a conceptual class Cc_{i+n} similar to Cc'_{j+1}. This level of interpretation adds the successive steps to find Cc_{i+n}. The level of provider interpretation path $Cc'_j(<= Cc'_{j+1}(...).r'_{j+1}).r'_j$ becomes $Cc_i(<= Cc_{i+1}(<= ...(<= Cc_{i+n}.r_{i+n})...).r_{i+1}).r_i$ in the consumer context.
- **case 3 :** conceptual classes for which Cc_i and Cc'_j play a role are not similar but a unique interpretation path exists in the provider context which carries out to a conceptual class Cc'_{j+m} similar to Cc_{i+1}. This level of interpretation deletes the successive steps to find Cc'_{j+m}. The level of provider interpretation path $Cc'_j(<= Cc'_{j+1}(...(<= Cc'_{j.m}.r'_{j+m})...).r'_{j+1}).r'_j$ becomes $Cc_i(<= Cc_{i+1}(...).r_{i+1}).r_i$ in the consumer context.
- **case 4 :** conceptual classes for which Cc_i and Cc'_j play a role are similar but two possible roles exist $r1_{i+1}$ and $r2_{i+1}$ which are played by these classes for Cc_{i+1}. The interpretation rule on the provider context is $Cc'_j(<= Cc'_{j+1}.r'_{j+1}).r'_j$. This state gives several possible re-interpretation paths and requires a human decision to define the relevant path.
- **case 5 :** similar classes for which Cc_i and Cc'_j play a role don't exist. The re-interpretation is stopped and the re-interpretation discovered until this case happens is considered as a correct interpretation path on the consumer context. This interpretation can be completed by a user or a domain specialist if needed.

4.2 I-Objects Fusion

With all information correctly interpreted on the consumer context, it is now possible to define virtual I-Objects. This is a semi-automatic process where user has to define its interface by giving the profile of a virtual I-Object class, by interpreting this class, characterizing it and applying a fusion operator to create integration rules. Two kinds of operator exist : the merging operator (Operation $\oplus$), which is detailed below, and the joining operator (Operation $\otimes$).

When **Operation** $\oplus$ is applied to a virtual I-Object, rules are generated to merge instances of all similar classes into the virtual one. All these instances can be accessed using the virtual class. This rule generation is composed of two steps. The first one finds equivalent classes and builds skeleton of rules. Formally, if C_G is a virtual class to group equivalent objects which have an interpretation on the concept Cc and, $C_{i=1..n}$ are classes all having the same interpretation on Cc as on C_G, then the following grouping rules are generated :

$$\mathbf{X : C_G : -X : C_i}$$
with X being a variable for unification process.

The second step finds relevant methods which are equivalent to methods of virtual class and enriches rules from the first step. Formally, if C_G and each equivalent C_i own a method, respectively m_G and m_i, having a compatible interpretation path, then grouping rules can be completed as follows :

$$\mathbf{X : C_G[m_G -> Y] : -X : C_i[m_i -> Y]}$$
with X and Y two variables for unification process.

If C_G and each equivalent C_j own a method, respectively m'_G and $m"_j$, having an indirect compatible interpretation path – i.e. $m"_j$ return objects which belong to classes C_k which own a method m'_k with the same interpretation path than m'_G –,then grouping rules can be completed as follows :

$$\mathbf{X : C_G[m_G -> Y] : -X : C_j[m"_j -> Z], Z : C_k[m'_k -> Y]}$$
with X, Y and Z three variables useful for unification process.

The **Operation** $\otimes$ is more complex than $\oplus$. When it is applied to a virtual class, user has to specify a join criteria and rules are generated to join a similar class (to virtual class) with other classes which have the join criteria as a method and some methods equivalent to methods of the virtual class.

The fusion is user-driven and the example below shows how an interface can be defined to exploit the semantics expressed in this contextual approach. For example, we consider a user-consumer who wants to create an interface to access information about names and sizes of students. We suppose that the context of this consumer is given and that a context comparison has been initiated with two provider sources *source1* and *source2*. This consumer context depicts only concepts that characterize a student : A student is a person who studies at the university, with a first name, a name as identity and a height as characteristic (*height* derives from a MCc *Length*).

The user creates its interface as an I-Object definition and interprets it on its local context. Interpretation rules in this example are simplified and no interpretation path is defined. The user clarifies the semantics of its data defining the characterization of the method *size* specifying that its values has to be expressed in foot (only).

```
concept individual{+person,+human}
                [ identifier => identity, characteristic => height ].
concept height::Length.
concept identity[ name => {first name, last name} ].
concept christian_name::first name.
concept student::individual [ study => university ].

class student[ lname => string; cname => string; size => real ].
class student[ MProperty@('size','Unit Of Datum') -> 'foot' ].
inter student:::student.
inter student.lname:::last name.
inter student.cname:::christian name.
inter student.size:::height.
inter{X,Y} X{Y:student[ MProperty@('size','Unit Of Datum') -> X ]}
           :::Unit Of Datum <= Length.Unit Of Datum.
```

The figure 5 presents the interpretation of the class *student* defined by the consumer and the result of the context comparison between the two providers *source1* and *source2*. Similarities in the structure of I-Objects appear clearly in this simple graph. For example, *student.cname*, *student1.firstname@(1)* and *etudiant2.prenom* (source2 is french) are identified as similar.

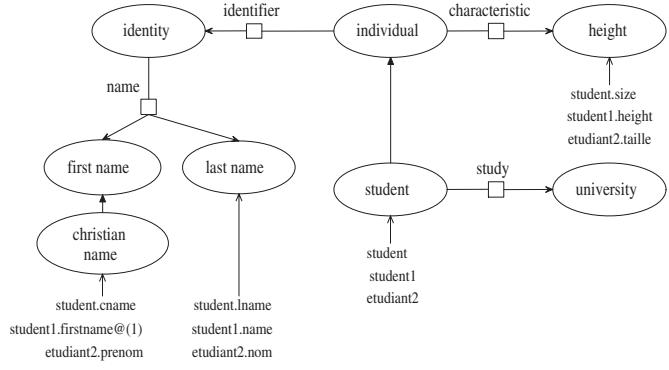

Fig. 5. Reconciliation of different sources on a consumer context

The application of the operation $\oplus$, noted $\oplus$(student),gives the two rules below in the first step. Each I-Object class similar with *student* is unified in a rule which maps I-Objects of *student1* and *etudiant2* to I-Objects of *student*.

Similar methods from each unified class have to be mapped to the profile of *student* in a second step.

```
X:student :- X:student1[ repository -> "source1.fr" ].
X:student :- X:etudiant2[ repository -> "source2.fr" ].
```

The result of this second step of $\oplus$ on this example updates the rules as follows :

```
(1) X:student[ lname -> NF; cname -> NB; size -> SZ ]:-
    X:student1[ repository -> "source1.fr"; name -> NF;
                firstname@(1) -> NB; height -> SZ ].

(2) X:student[ lname -> NF; cname -> NB; size -> SZ ]:-
    X:etudiant2[ repository -> "source2.fr"; nom -> NF; prenom -> NB;
                 taille -> SZ ].
```

The I-Object class *student* is now a well defined virtual I-Object which can plays the role of interface to access data from different sources. A problem keeps unsolved if data from sources are expressed in a different format, for example if *source2* uses the meter unit to express the height of students rather than foot. This problem can be resolved by comparing the characterization of the unified methods and is taken into account with another automatic operation called adaptation operation.

An adaptive operation is automatically applied on the rules which define virtual I-Objects built with $\oplus$ or $\otimes$. This operation is composed of four steps :

- A comparison of the characterization rules for methods which are unified in the virtual I-Object definition rule. Characterization mismatch is retained for the following step.
- A template function is built for each type of mismatch characterization. This function is created in a specific library called *MODEL_LIBRARY*. This template is interpreted and characterized on the local context.
- Calls to these templates are inserted in the rule definition of the virtual I-Object.
- Search for relevant functions in the local context which match with the interpretation of templates and replace these ones by functions if exist.

In the example above, if the characterization of the method *taille* shows that the unit for these data values is *meter*, a characterization mismatch appears between *student.size* and *etudiant.taille*. The adaptation operation generates the template function *template1* in library *MODEL_LIBRARY*. This templates gives a model of function to transform meter value into foot value. The adaptation operation provides a correct characterization for parameter and results of this function as presented below :

```
library MODEL_LIBRARY[ template1@(parameter->real)->real ].
inter MODEL_LIBRARY.template1(parameter:::height):::height
library MODEL_LIBRARY[ MProperty('template1','Unit_Of_Datum') -> 'foot' ].
library MODEL_LIBRARY[ MProperty('template1@parameter','Unit_Of_Datum')
                       -> 'meter' ].
```

The example below shows how a template is inserted in the body of a virtual I-Object definition. The variable which unifies directly methods with mismatch characterization is replaced by a template call.

```
(2bis) X:student[ lname -> NF; cname -> NB; size -> SZ ]:-
            X:etudiant2[ repository -> "source2.fr"; nom -> NF;
                         prenom -> NB; taille -> TA ],
             MODEL_LIBRARY[ template1@( parameter -> TA)-> SZ ].
```

This virtual I-Object can now be used to access student data from two heterogeneous sources in a transparent way.

5 Conclusion and Future Works

DILEMMA proposes three important features to enhance cooperation of information systems : (1) a semantic mediation model for the definition of I-Objects and their meanings on a local context, (2) a reconciliation method which assures the comparison of contexts using the definition of a semantic distance, (3) a mechanism of object fusion to construct virtual I-Object classes which allows a transparent access to the cooperation. In this approach, information do not depend on domain or universal ontologies and the comparison mechanism is always possible. Some recent works also focus on the definition of similarity distances to compare concepts described using ontologies and inter-ontologies mapping may be necessary to assure a complete comparison process. The mechanism of object fusion is based on the results of the context comparison. It allows to group I-Objects which are interpreted on similar concepts. The discovery of relevant I-Objects over the cooperation depends on the quality of the context. Thus, a promising way may be to construct local contexts by specializing and adapting some pre-defined top-level ontologies.

References

1. Lee, Y., Liu, L., Pu, C.: Towards interoperable heterogeneous information systems: An experiment using the diom approach. In: Proceesings of the 12th ACM Symposium on Applied Computing (ACM SAC'97) Special track on Database Technology, San Jose, California, USA (1997)
2. Tomasic, A., Raschid, L., Valduriez, P.: Scaling heterogeneous databases and the design of disco. In: Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS'96), Hong Kong, IEEE Computer Society Press (1996) 449–457
3. Tari, Z., Cheng, W., Yetongnon, K., Savnik, I.: Towards cooperative databases: The distributed object kernel approach. In Yétongnon, K., Hairi, S., eds.: Proceedings of Conference on Parallel and Distributed Computing Systems (PDCS'96), Dijon, France, The International Sociaty for Computers and their Applications (ISCA), isbn 1880843-16-1 (1996) 595–600

4. Bishr, Y., Radwan, M., Pandayaal, J.: Semweb - a prototype for seamless sharing of geoinformation on the world wide web in a client/server architectures. In: Third Joint European Conference and exhibition on Geographical Information. (1997) 145–154
5. Fensel, A.D., Decker, S., Erdmann, M., Studer, R.: Ontobroker: The very high idea. In: Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibal Island, Florida (1998)
6. Goi, A., Fensel, A.D., Angele, J., Decker, S., Erdmann, M., Schnurr, H.P., Studer, R., Witt, A.: On2broker: Lessons learned from applying ai to the web. Technical Report 383, Institue AIFB (1998)
7. Mena, E., Illarramendi, A., Kashyap, V., Sheth, A.: Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. International Journal of Distributed and Parallel Databases (DAPD) (2000) 223–271
8. Ouzzani, M., Benatallah, B., Bouguettaya, A.: Ontological approach for information discovery in internet databases. Journal of Distributed and Parallel Databases **8** (2000) 367–392
9. Jr., R.J.B., Bohrer, B., Brice, R.S., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M.H., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., Woelk, D.: Infosleuth: Semantic integration of information in open and dynamic environments (experience paper). In Peckham, J., ed.: Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD'97), Tucson, Arizona, USA, ACM Press (1997) 195–206
10. Nodine, M., Bohrer, W., Ngu, A.H.H.: Semantic brokering over dynamic heterogeneous data sources in infosleuth. In: Proceedings of the 15th International Conference on Data Engineering (ICDE'99), Sydney, Austrialia, IEEE Computer Society (1999) 358–365
11. Weibel, S., Kunze, J., Lagoze, C., Wolf, M.: Dublin core metadata for resource discovery. Technical Report IETF 2413, The Internet Society (1998)
12. Jarrar, M., Meersman, R.: Formal ontology engineering in the dogma approach. In R. Meersman, Z., ed.: CooPIS/DOA/ODBASE 2002, Springer, Lecture Notes in Computer Science vol. 2519 (2002) 1238–1254
13. Motik, B., Maedche, A., Volz, R.: A conceptual modeling approach for semantic-driven enterprise applications. In R. Meersman, Z.T., ed.: Proceedings of the First Inetrnational Conference on Ontologies, Databases and Applications of Semantics (ODBASE), Springer, Lecture Notes in Artificial Intelligence , vol. 2519 (2002) 1082–1099
14. Guarino, N.: Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In Pazienza, .T., ed.: International Summer School (SCIE'97), Frascati, Italy, Lecture Notes in Computer Science, Vol. 1299, Springer, ISBN 3-540-63438-X (1997) 139–170
15. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. In: Journal of Artificial Intelligence Research (JAIR), vol. 11. (1999) 95–130
16. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Madrid, Spain (2002)
17. Jouanot, F.: DILEMMA : vers une coopération de systèmes d'informations basée sur la médiation sémantique et la fusion d'objets. PhD thesis, Burgundy University - France (2001)