An Integration Method for Design Schemas

Isabelle Mirbel and Jean-Louis Cavarero

Laboratoire I3S, CNRS-URA 1376
250 Avenue Albert Einstein
Sophia-Antipolis 06560 Valbonne
France

Abstract. The complexity of databases is increasing continually. The work of several designers become necessary. Therefore it is interesting to improve the design process with a new phase devoted to information integration, in order to take into account the designers'viewpoints. In this paper, we present an integration process which allows similarities to be discovered between the schemas under study. It works on object-oriented schemas. Whenever possible, we propose several results for the integration of two given schemas. This then makes it possible to choose the one which is the best adapted to the working context, amongst the result schemas. When design schemas are being integrated, the structural, but also and above all, the semantic part of the schemas are studied. To represent the semantic of the words which are used in a schema, we have defined a model of thesaurus drawn from the domain dealing with the meaning of words : linguistics.

1 Introduction

The complexity of information systems is increasing continually. The work of several designers become necessary. Therefore, during the design phase, several views of the information system are described by various designers belonging to different domains. So, it is necessary to improve the design process with a new phase devoted to information integration, in order to take into account the designers'viewpoints. Moreover, any enterprise needs to evolve in time. This evolution must be passed on to the underlying information systems. So with time, it is necessary to integrate changes resulting from this evolution into the existing design schemas.

The schema-integration process satisfies this dual goal. It allows a federative framework to be given to the designers, and the schema evolution to be taken into account .

In this paper, we present our integration process. The schema comparison is based on the structural, and above all, the semantic aspects of the schemas. We show the richness of these kinds of comparison and how we use them in other phases of the process.

In the next section, we briefly present the existing works on design-schema integration. Then we describe the most important phases of our process, that is to say the comparison phase, the conflict-solving phase and the result-presentation phase. After that, we indicate the strategies for using this process. In the last section, we conclude.

2 Existing Work

A lot of different work has been done on schema integration. This goes from works on the tools to help the designer to get two schemas without conflicts [1], to work on the formal presentation of the integration rules [2]. An overview of the variety of existing work can be found in [3].

Generally, the integration process is divided in four phases. The first phase is the pre-integration one. It consists in translating the schemas under study into a same formalism, and/or into a canonical form [4]. The next phase is the comparison one. It consists in discovering existing similarities and conflicts between the two schemas to be integrated. This is the most difficult part of the integration process [5]. The fundamental problem is that the universe of discourse (UoD) can be modeled in different ways. The same phenomenon of the real word can be described by using different abstraction levels, different properties. These differences react on the structure chosen to represent the phenomenon, but also on the terminology used to name it (homonymy and synonymy conflicts). The less recent approaches describe how to merge several schemas with regard to a set of correspondences between the schemas under study [1, 6, 4]. These approaches were based on relational, functional and entity-relationship models. The inter-schema assertions had to be supplied by the designer, and the conflict solving depended on the designer's common sense. Some recent approaches, in most cases based on object-oriented models, proposed tools to find the likeness between the schema elements with regard to structural criteria [7], behavioral criteria [8] and semantic criteria [9].

The next phase is the conflict-solving phase. This consists in making the schemas compatible, that is to say without conflicts, by applying structural-changing rules to them. The object of this phase is to be able to integrate the schemas [10]. Lastly, the merging phase consists in building a result schema by merging the schemas under study. Several approaches exist. Each of them presents structural-merging rules. Generally, these rules are model dependent [11].

3 The Integration Process

Our integration process allows similarities to be discovered between the schemas under study. The designers do not need to enter the equivalence before doing the integration, as is required in a lot of existing works [6].

The process works on object-oriented schemas. These schemas do not have to be expressed in a particular formalism among existing models, but rather on all the object-oriented concepts used in most of the existing object-oriented design methods [12, 13]: that is to say, concepts of attribute, method, class and reference-and-inheritance links.

The structural, but also and above all, the semantic part of the schemas are used when design schemas are being integrated. The semantic part is used especially when design schemas are being compared, but also when conflicts are being solved and when the result schema is being presented.

Whenever possible, we propose several results for the integration of two given schemas. This then makes it possible to choose the one which is the best adapted to the working context, amongst the result schemas. One could, for example wish to integrate the schemas in order to enrich an existing library of design schemas. In this case, one could compare the result schemas obtained, in accordance with re-use criteria such as those presented in [14].

We will now present the comparison criteria used when design schemas are being compared.

3.1 The Semantic Similarities

To represent the semantics of the words used in the schemas, we have defined a model, drawn from a domain dealing especially with the meaning of words: linguistics [15]. In our data structure, we differentiate words (or terms) from concepts (or meanings), following Quillian's works; he was the first to make this distinction [16]. Each concept is represented by a sentence, the most explicit one, which can be sometimes a definition. We make the distinction between words and concepts so that we can work at a level (the concept level) where knowledge can be defined very precisely, without ambiguity, without synonym and homonym problems. Concepts are linked together by conceptual relationships of a semantic type. Some interpreting relationships between words and concepts allow the expression of meanings given to words.

We define two kinds of relationships in our thesaurus: conceptual relationship of a semantic type, that links concepts together; and interpreting relationship, that links words and concepts together in order to explain their meanings. We will present each of them.

The conceptual relationship of a semantic type: we use two kinds of conceptual relationships which are the generic relationship and the aggregation relationship. These are the relationships most often used. There exists a lot of other relationships linking concepts together[17]. But keeping them inside our thesaurus will bring nothing more, because we cannot exploit them, when design schema are being integrated.
In order to capture more semantics, we graduate the membership in these relationships with likelihood degrees, varying between 0 and 1. So we can represent some categories with ill-defined boundaries, situations between everything and nothing, like the quasi-generic relationship [17], or the fact that a component is optional or not [17].

We differentiate several kinds of aggregation relationships. D. A. Cruse [15] and J. Lyons have shown in their works the existence of transitive and non-transitive aggregation relationships [18], that is to say the preservation, or not, of the part features. (This feature will be used in the thesaurus exploitation phase.)

The interpreting relationships: interpreting relationships make it possible to link words and concepts together. As was done previously, to have a better representation of the real world, we define them like fuzzy relationships which indicate the probability for the concept to be a meaning for the word.

The thesaurus exploitation allows us to know which words are ambiguous, when they can be used with different meanings. It also makes it possible to detect similar words in the schemas under study. Two lexical relationships often appear in linguistic papers: the synonym relationship and the homonym relationship [19]. In our structure, lexical relationships do not appear explicitly, because they can be deduced from interpreting relationships. Moreover, in order to capture more semantics, we speak more globally about semantic likeness and ambiguity. We deduce weighted knowledge from conceptual and interpreting relationships.

The ambiguous word treatment: to know if a word is likely to be ambiguous, we must worry about its connection to several concepts, that is to say to several meanings. In this case, it can be considered as ambiguous. But, we think it will be more interesting to qualify this ambiguity. A word is linked to concepts by an interpreting relationship which has a probability coefficient, varying between 0 and 1, and indicating how much the word is linked to the concept. We can use the information provided by these coefficients to detect words which are likely to be ambiguous. That is why, we compare the probability coefficients of all the interpreting relationships connected to the word. If the coefficients have close values, the different meanings are used as often as each other. If the values are very different, some meanings are used more than others. In this last case, the word is less ambiguous than in the first one. That is why a word's degree of ambiguity is defined by the formula written by Shanon in information theory. So, when we find a same word in several design schemas, we are able to know its ambiguity degree, and make it easier to carry out the schema integration. It will avoid integrating a word used in different meanings in the different schema.

The semantic likeness of word treatment: to know if two words have semantic likeness is more difficult than to determine some ambiguity of one of them, because it requires considering the concepts linked to the two words under study. The most interesting case is when the first word has some concepts which are not directly linked to concepts linked to the second word. In this case, we must first value the proximity between concepts linked to the words under study. That is why we will present the cases where transitivity is admitted. In the next table, we recapitulate the different cases of succession of different relationships: Let c_a,

c_b and c_c be the concepts under study, such as $\exists f_i(c_a, c_b)$ and $f_j(c_b, c_c)$. In lines, we find the different kinds of relationships that can link c_a to c_b. In columns, we find the different kinds of relationships that can link c_b and c_c.

Table 1. The different cases of succession of different relationships

		generic relationship		aggregation relationship			
		$f_{jS^{-1}}$	f_{jS}	$f_{jA_{not}}$	$f_{jA_{\{e,s\}}}$	$f_{jA_{not}}$	$f_{jA_{tran}}$
ge. rel.	$f_{iS^{-1}}$	f_S^{-1}	●	●	●	$f_{A_{not\ tr}}$	$f_{A_{not\ tr}}$
	f_{iS}	●	f_S	$f_{A_{not\ tr}}^{-1}$	$f_{A_{not\ tr}}^{-1}$	●	●
aggr. rel.	$f_{iA_{not\ tr.}}$	$^{-1}$●	$f_{A_{not\ tr}}^{-1}$	●	●	●	●
	$f_{iA_{trans}}$	$^{-1}$●	$f_{A_{trans}}^{-1}$	$f_{A_{not\ tr}}^{-1}$	$f_{A_{trans}}^{-1}$	●	●
	$f_{A_{not\ tr}}$	$f_{A_{not\ tr.}}$	●	●	●	●	●
	$f_{A_{trans}}$	$f_{A_{trans}}$	●	●	●	$f_{A_{not\ tr}}$	$f_{A_{trans}}$

When we substitute two relationships by a single one, one must assign a fuzzy coefficient to it, calculated from the two initial ones. That is why, we have differentiate several cases: let c_a and c_c be the two concepts under study, and c_b be the concept linked to c_a and c_c.

1. If the three concepts are linked together with a same type of relationship, then the semantic distance between the two concepts is:

$$dist = f(c_a, c_b) \times f(c_b, c_c)$$

2. If two relationships are transitive, we see that the aggregation relationship carries more semantics than the generic one. That is why, the semantic distance depends on the aggregation coefficient:

$$dist = 0.9 \times f(x, y),$$

where $f(x, y)$ represents the aggregation relationship.

Now, we can compute the degree of likeness of the two words from the deduced relationship between them. Let m_a and m_b be the two words under study. Let $C_a = \{c_{a1}, c_{a2}, ..., c_{am}\}$ be the set of concepts linked to m_a, and $C_b = \{c_{b1}, c_{b2}, ..., c_{bm}\}$ the set of concepts linked to m_b. In C_a and C_b, concepts from m_a and m_b are directly linked two by two by the new deduced relationships $(\forall k \in [1, m], \exists f = f_S(c_{ak}, c_{bk})$ or $\exists f = f_S^{-1}(c_{ak}, c_{bk})$ or $\exists f = f_A(c_{ak}, c_{bk})$ or $\exists f = f_A^{-1}(c_{ak}, c_{bk}))$. We define the semantic likeness degree D between m_a and m_b by:

$$D(m_a, m_b) = max_{c_{ak} \in C_a, c_{bk} \in C_b} min(f_{Cpt}(m_a, c_{a_k}), f(c_{a_k}, c_{b_k}), f_{Cpt}(m_b, c_{b_k}))$$

where f_{Cpt} represents the interpreting relationship, and f the deduced one.

And we are also able to qualify this likeness by a type of likeness which could be: synonym, but also generalization, specialization, composed or component. More information about this thesaurus can be found in [20].

Thanks to the thesaurus, we are able to know the likeness between two words, and the ambiguity of a given word. Moreover, in design schemas, we note that some words are used more frequently than others. They are used in a more general way, without being attached to a particular domain. They provide less semantics than the others. That is why for example, in order to avoid considering as close (and merge) two classes whose only common elements are *number* or *name*, we have defined the notion of semantic weight. This weight is attached to each word placed inside the thesaurus. The word's semantic weight varies in inverse proportions to the word's rate of appearance in all the design schemas (all domains mixed together). It is updated each time the integration process is used. Let $Nb_A(w)$ be the number of appearances of the word w in the already-studied schemas, n be the total number of already-studied schemas and $P_S(w)$ the semantic weight of the word w,

$$P_S(w) = \frac{n}{Nb_A(w)}$$

3.2 The Structural Similarities

Structural similarities which may exist between two elements depend on the type of elements examined.

At attribute level: similarities deal with the domains (identical, included in one another, intersecting or disconnected), and with the units (identical, compatible or incompatible).

At method level: similarities deal with the attributes used by the methods, and the way they are used (instantiation, modification, consultation). The attribute lists have to be identical or different.

At class level: similarities deal with the attributes and methods included in the classes. The percentage of similar elements, the semantic weights of the similar elements and the semantic weights of the non-similar elements are also considered. The algorithm is as follows:

1. If the average semantic weight of the similar elements is high (higher than the threshold defined by the user)
   (a) If the average semantic weight of the non-similar elements is high
      i. If the percentage of similar elements is high, then the two classes are considered to be similar.

   ii. If the percentage of similar elements is not high, (lower than the threshold defined by the user), the two classes are not considered to be similar.

   (b) If the average semantic weight of the non-similar elements is not high, then the two classes are considered to be similar.

2. If the average semantic weight of similar elements is not high
   (a) If the average semantic weight of non-similar elements is not high
      i. If the percentage of similar elements is not high, then the two classes are not considered to be similar.
      ii. If the percentage of similar elements is high, then the two classes are considered to be similar.
   (b) If the average semantic weight of the non-similar elements is high, then the classes are not considered to be similar.

At link level: the similarities deal with arrival and departure classes (similar, inverted or different), the intervals defined by the minimal-and-maximal cardinality (equal, included in one another, intersecting or disconnected) and the type (weak, if the object instances can be shared or strong, if the object instances cannot be shared). The links may be identical or different.

3.3 The Comparison Phase

In the comparison phase, especially in structural comparison, knowledge about incorporated elements is needed in order to compare the incorporating elements. For example, knowledge about attributes and methods is needed to compare classes. That is why the comparison order is: first, attribute comparison, then method comparison, then class comparison and last link comparison.

It is possible that two designers may have represented the same concept of the real word by using different structures in their schemas. All the cases are possible. So, we also look for comparisons between attributes, methods, classes and links. Of course, these comparisons are limited to a semantic examination, because the element structures are different.

The comparison phase is divided into two parts where similar and conflicting pairs of elements (one of each schema) are detected and memorized.

Part 1: semantic-privileged examination

Step 1: examination of pairs of elements where the structures are different and incompatible: attribute/method, method/class and method/link. Pairs where names are strictly identical are detected and memorized : they are conflicting.

Step 2: examination of pairs of elements where the structures are different but compatible: attribute/class, attribute/link and class/link. Pairs of this type in which there are some semantic similarities (with regard to the criteria defined above) and where the semantic likeness is higher than the threshold defined by

the user are detected and memorized.
This allows the desired semantic-proximity degree to be given for carrying out the integration.
As the examination is only semantic here, pairs whose degree of ambiguity is higher than the threshold defined by the user, are eliminated.
The attribute/link pairs where semantic likeness is of a composed/component type are also eliminated, because they are useless.

Step 3: examination of pairs of elements where the structures are identical: attribute/attribute. Attribute pairs with semantic similarities (with regard to the criteria defined above) and where the semantic likeness is lower than the threshold defined by the user are detected and memorized.
Then, structural similarities are looked for only among the pairs resulting from the semantic examination. Structural similarities between two attributes not having semantic similarities are not sufficient for the attributes to considered as similar. Moreover, there is no point in examining the structural similarities (i.e. the domains) of two attributes, if their semantic likeness is of the composed/component type.
Then, amongst the pairs obtained, pairs whose similarities are only semantic and where at least one of the two words appears with an ambiguity higher than the threshold defined by the user, are eliminated.

Step 4: building the solution tree: some elements of the first schema may be close to several elements of the second schema, and the other way round. It is impossible to merge one element of the first schema with several elements of the second schema, and the other way round. At this stage of the process, we are not capable of choosing the likeness which will allows the best integration, we keep them all and build a result from each one of them. A tree is built. Each node corresponds to a possible choice between several likeness for a given element. In each path which leads from the tree root to one of its leaves, an element only appears once. At the leaf level, several sets of likeness are obtained. The solution tree is built on the pairs obtained through the steps 2 and 3. We call them coherent sets. The likeness related to the pairs where the names are identical, obtained through the step 1, are added in order to solve the homonym conflicts they represent, whatever the coherent set is.

Part 2: structure-privileged examination

When attributes are being examined, the semantic examination prevails over the structural examination. Indeed, a similarity between the domains of two attributes does not imply a similarity between the two attributes. On the contrary, a similarity between the names of two attributes implies a similarity between these attributes. This explains why we first try to find semantic similarities amongst the attributes, before trying to find the structural ones. When method, class and reference links are being examined, it is the opposite, the structural examination prevails over the semantic examination. For example, two classes with only one name in common can not be considered as similar; whereas two classes with only one structure in common must be considered as similar classes. This involves an inversion in the examinations carried out on methods, classes and links: the structural examination precedes the semantic examination. This inversion necessitates adding a new step to try to find elements which are strictly identical from a semantic point of view, but structurally different, and which represent an homonym conflict.

Step 1: examination of the likeness between methods, classes and links: the structural-similarity examination is carried out according to the criteria defined in Sect. 3.2. Then, the semantic examination consists in trying to find the likeness between the names of the elements found during the structural examination. As at the attribute/attribute level, semantic likenesses lower than the threshold defined by the user, are eliminated.
The link/link pairs having a component/composed type of semantic likeness are also eliminated, as they cannot be used during integration of design schemas.
At the class level, our approach consists in examining the different possible combinations between the concept provided by the class (the structural part) and the name given to it (the semantic part), in order to detect any possible conflicts. For that purpose, we consulted the works of B.R. Gaines and M.L.G. Shaw [21] which were carried out in the knowledge acquisition domain. This helped us to show different cases encountered when classes of design schemas are being integrated. They are:

1. Case 1: two classes have the same name; they are composed of the same elements.
2. Case 2: two classes have the same name and they have some common elements (a not insignificant part).
3. Case 3: two classes have different names; they are composed of the same elements.
4. Case 4: two classes have different names but they have some common elements (a not insignificant part).

Step 2: The method/method, class/class and link/link pairs having identical names and different structures are detected and memorized, because they represent homonym conflicts. At the class level (classes having the same name but no common elements), it is a new case, called case 5, in addition to the for cases presented above.

Step 3: building the solution tree: if the elements under study (method, class, link) are close to elements which are already in the solution tree (through the step 4 of the part 1), we continue building the solution tree, by adding this new choice discovered in part 2. Otherwise, it is not possible to build the solution tree. There is simply an enrichment of the existing coherent sets. Indeed, inside

each set, either there are all the necessary likenesses for the new pairs of likeness (attributes and methods for the classes, classes for the links) in which case we add the new likeness to the set; or not all the necessary likenesses are there and we do not add the new likeness to the set.
We also add likeness related to pairs of elements with strictly identical names and different structures (step 2) to each coherent set, in order to solve the homonym conflict they represent, whatever the set.
When classes are being examined, in addition to the presence of attributes and methods in the coherent sets, the following rules have to be respected. The object here, is that the elements of a given class are not involved in more than one likeness, at the same time allowing a class from one schema to be merged with several of the other schema.

1. If the type of likeness between two classes is 1 or 3, no other likeness involving these two classes is allowed in the same branch;
2. If the type of likeness between the two classes is 4, and if other likeness involving these classes exist, the type of likeness must be 4 and/or only one case 2 for each class involved; the likeness between classes must involve different attributes and methods;
3. If the type of likeness between the two classes is 2, and if other likenesses between the two classes exist, the type of these likeness must be 4 and must involve different attributes and methods.

3.4 The Conflict-solving Phase

One concept of the real world can be represented in several ways in two different schemas. It is a conflict between the two schemas. In order to solve this conflict, we must choose one of the two representations. In order to know which representation must be chosen, we assign a credibility coefficient (varying between 0 and 1) to each schema. It represents the expert credibility. Its value is propagated on each attribute, each method, each class and each reference link of the schema. If a conflict between two elements is encountered, we know which representation must be chosen upon the others, by comparing the credibility coefficients of the elements. Each time a choice has to be made, if one schema has to be privileged, the structure of the privileged schema is chosen and used to express similar information in the other schema. If there is no schema to privilege, the richest structure is always chosen.

Each time a conflict is solved, the maximum of the values of the two credibility coefficients of the two structures under studies is assigned to the credibility coefficient of the structure of the result schema.

This way of proceeding, added to the fact that the order of rule-application is well defined, makes it possible to justify the unicity of the integration result. In addition, because the integration is carried out from coherent sets, and because these sets are strictly identical whether they are the result of the comparison of schema 1 with schema 2 or whether they are the result of the comparison of schema 2 with schema 1, there is commutativity.

Part 1: conflict solving of pairs of elements with different and incompatible structures

Here we examine the attribute/method, method/class and method/link pairs, resulting from step 1 of part 1 of the comparison phase. At least one of the two names has to be changed, in order to eliminate the homonymy conflict existing between the two elements. If one of the two schemas has to be privileged, the name of the element of the other schema is changed; otherwise, the names of the two schemas are changed for different names. Then the likeness between the two elements is eliminated, because they are now different elements.

Part 2: conflict-solving in pairs of elements with different but compatible structures

Step 1: attribute/link, attribute/class and class/link pairs: If one schema has to be privileged, the structure of the element of the other schema is changed for the element structure of the privileged schema; otherwise the structures are modified as shown in Fig. 1, 2 and 3 .

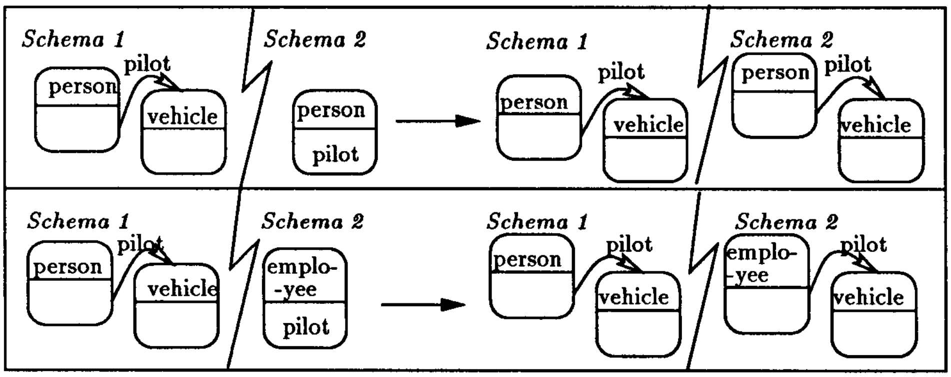

Fig. 1. Conflict-solving in attribute/link pairs

Part 3: conflict solving in pairs of elements with similar structures
Conflict solving dealing with attributes and methods is done through the step 3 of this part of the conflict-solving phase, except the method homonym conflicts, solved here.

Step 1: methods: if the two methods have identical names and different struc-

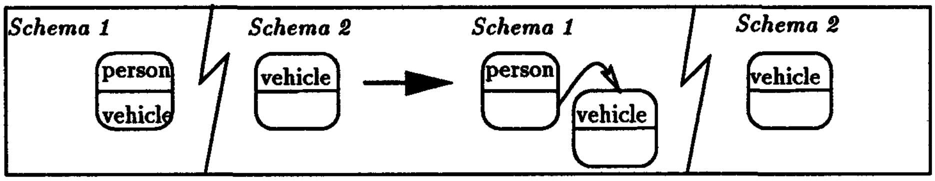

Fig. 2. Conflict-solving in attribute/class pairs

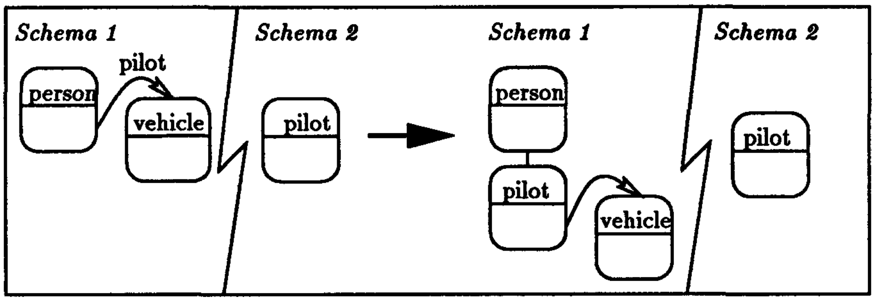

Fig. 3. Conflict-solving in classes/links pairs

tures, at least one of the two names must be changed, in order to eliminate the homonymy conflict that they represent. If one schema has to be privileged, the name of the method of the other schema is changed; otherwise the two names are changed for different names. Then the likeness between the two methods is eliminated, because it has no more reason to exist.

Step 2: classes:

1. case5 : at least one of the two names has to be changed, in order to eliminate the homonymy conflicts existing between these two classes. If one of the two schemas has to be privileged, the name of the class of the other schema has to be changed; otherwise the names of the two classes has to be changed. Then the likeness between the two classes has to be eliminated, because they are now different classes.
2. case 3 : at least one of the two names has to be changed, in order to solve the synonymy conflict existing between the two classes. If one of the two schemas has to be privileged, the name of the class from the other schema is changed for the name of the element of the privileged schema; otherwise the name of the two classes are changed for a common name. The nearness kind of the two classes is changed to case 1.

At this stage, two classes are either identical (with regard to their names and contents) (case 1); or close with regard to their contents (some common elements) but different with regard to their names (case 4); or close with regard to their contents (some common elements) and similar with regard to their names (case 2).

The rules for solving conflicts, according to the different cases are as follows:

1. case 1 : the two classes are strictly identical. There is no conflict between these classes. Only the conflicts between the attribute-and-method descriptions have to be solved.
   (a) Attribute conflict-solving: if the domains are compatible, the union of their intervals is made. If the domains are incompatible, one of them is chosen (the one from the privileged schema if it exists, otherwise the largest one).
   (b) Method conflict-solving: if the two methods have different names but strictly identical structures, they represent a synonymy conflict which must be solved. If one of the two schemas has to be privileged, the name of the method of the other schema is changed for the name of the method privileged schema; otherwise the two names are changed for an identical name.
2. case 2 : our idea is to assume that the two designers have seen the same reality, but from two different points of view. So, we must enrich each point of view with the other one in order to obtain the same class representing the same concept in the most complete way (see Fig. 4).

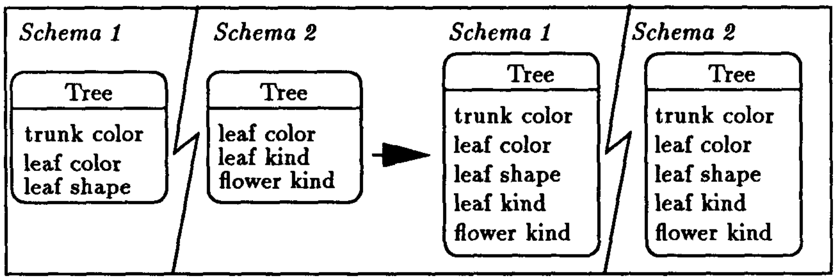

Fig. 4. Example (Case 2)

3. Case 4: the changes to be made in order to resolve conflicts are not the same, depending on whether the class belongs to a inheritance hierarchy or a reference hierarchy. If the two classes belong to inheritance hierarchies, we can think that one or several common abstraction levels have been omitted in the two schemas (see Fig. 5). If the two classes belong to reference hierarchies, it is a the decomposition level which is missing.

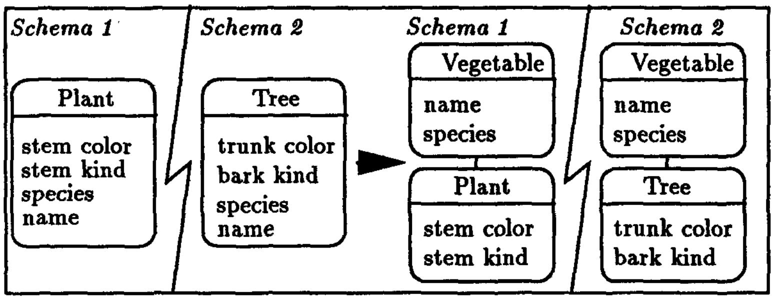

Fig. 5. Example (Case 4a)

What makes our approach worthy of interest is the discovery of dependencies (through references) and generalization/specialization relationships (through inheritance) between elements of the two schemas, even if there are no equivalent classes in the two schemas.

Step 3:links.
The algorithm for solving conflicts between reference links is as follows:

1. the two links have identical names and different structures: one of the two names has to be changed in order to eliminate the homonymy conflict existing between the two classes. If one the two schemas has to be privileged, the name of the other schema is changed; otherwise, the names of the two schemas are changed for two different names. Then the likeness between the two links is eliminated, because they are no longer similar.
2. the two links have different names but similar structures : the following changes must be done.
   (a) change at least one of the two names, in order to solve the synonymy conflict they represent. If one of the two schemas has to be privileged, the name of the link from other schema is changed for the name of the link from the privileged schema; otherwise, the two names are changed for a common one.
   (b) if the departure and arrival classes are inverted and one schema has to be privileged, then its departure and arrival classes are chosen and the reverse-link created.
   (c) if the cardinalities are different and if one schema has to be privileged, then the cardinality of the privileged schema is chosen for the two schemas, otherwise
      i. if there is an inclusion, the largest interval is chosen,
      ii. if there is an intersection or a disjunction, the union of the two intervals is chosen.
   (d) if the types are different and if one schema has to be privileged, the type of the privileged schema is chosen, otherwise the weak type is chosen.

Now that the conflicts are solved we are able to merge the two schemas at the link level, that is to say to superpose the link hierarchies. This step is not explain here, because it is only structural. More information about this step can be found in [22].

3.5 Choice of the Result Schema

One of the advantages of proposing several schemas as result of the integration process, is the possibility of choosing the best schema according to the use one has for it. For example, with regard to quality criteria [23] or re-use criteria [14], if the integration has been carried out with a view to enriching an existing library of design schemas.

3.6 Presenting the Result

At this stage of the process, we superpose a vocabulary level over each result schema, in order to present it to each designer in his own vocabulary when there is a difference in words used from one designer to another.

We will now present strategies for using the integration process, when more than two design schemas are being integrated.

4 Integration Strategies

Two contexts exist when more than two schemas are being integrated. When we need to integrate a schema into a set of already-integrated schemas, it is the incremental context; and when we need to integrate more than two schemas at any given time, it is the multi-expertise context. We will present the integration strategies in both contexts.

4.1 The Incremental Context

In the incremental context, at a given time t, we try to integrate a new schema with a set of already-integrated schemas, which was carried out at time $t-1$. To integrate the schemas in the right way, it is necessary to take into account the time passed between $t-1$ and t, as well as the evolution of the information system represented by the schemas to be integrated. This is done by a robustness coefficient, k. When k leans towards 0, the system is not considered as a robust one, and the new schema is privileged compared to the schema resulting from the previous integration. On the other hand, when k tends to 1, the system is considered to be robust and is privileged compared to the new schema. Therefore, as time goes by, a convergence may be found, a stable schema may be obtained as the result of the incremental integration of different expertises, and of the stability of the information system observed. But this stability also depends of the time elapsed between the two integrations under study. The system can be stable for a while, and then become unstable because of the deep modifications of the information system in question.

The conflict solving is local : if the starting coefficients of the elements have all the same value defined at the schema level, these values will become different one from the others while several integrations are being processed, because they change with regards to the encountered local clashes. The comparison is based on the coefficients assigned to the elements of the conflicting schemas (attributes, methods, classes and links). The weighting by the robustness coefficient k (which is a global coefficient) is done on each local coefficient. The robustness coefficient is assign to the schema which is the result of the already-integrated schemas.

Let S_r be the result schema of the already-integrated schemas at $t-1$, S_n be the schema integrated with S_r at t, E_r be an element of S_r clashing with an element E_n of S_n, C_{E_r} be the credibility coefficient of E_r in S_r, C_{E_n} be the credibility coefficient of E_n in S_n, and k be the robustness coefficient assign (globally) to S_r, then the weights P_{E_r} and P_{E_n} respectively assigned to the elements E_r and E_n in order to know which one must be privileged when conflicts are being solved are as follows :

$$P_{E_r} = C_{E_r} * k \text{ and } P_{E_n} = C_{E_n}$$

Each time a conflict is solved, the value of the maximum of the credibility coefficients of the structures of the elements under study is assigned to the credibility coefficient of the structure of the element of the result schema. It is important to emphasize that it is the maximum between C_{E_r} and C_{E_n} which is kept, and on no account the maximum between P_{E_r} and P_{E_n}.

Is is important also to observe that when the value of C_{E_r} is very low, compared to the value of C_{E_n}, even a high value of k will not allow C_{E_r} to be privileged on C_{E_n}; and it seems to be reasonable. A very-low local credibility cannot be privileged on a very-high local credibility, even if the schema is globally robust. In the same way, if the value of C_{E_r} is very high, compared to the one of C_{E_n}, even a low k will not enable C_{E_n} to be privileged on C_{E_r}.

Lastly, we can see the interest of local work: the choice of the privileged structure is more precise. And while several integrations are being processed, it allows to improve the credibility of a given representation for a given concept. In addition, it allows to distinguish several sub-schemas inside the result schema, each of them having a different credibility level. We can, for example, distinguish a stable and very-credible kernel.

4.2 The Multi-expertise Context

There exists two strategies [3]: the ladder strategy and the balanced strategy. The first one consists in the integration of two schemas, then the result of the integration with a third one and so on until all the schemas have been integrated. The second consists in a parallel integration of pairs of schemas, then a parallel integration of pairs of result-schemas, and so on until a single schema is obtained. The latter strategy does not correspond to a natural process. Moreover, it seems difficult to choose (and to justify) which pairs of schemas to start with. That is why we have chosen the ladder strategy.

The integration process in a multi-expertise context is equivalent to the integration process in a incremental context but without taking time into account, that is to say without using a robustness coefficient. Integrations are made at the same time (the incremental context corresponds to a multi-expertise context in which we have decided to delay the integration). In this case, the weights assigned to the elements through the integration are the credibility coefficients.

Let S_r be the result schema of the integration of n schemas, let S_n be the $n+1$st schema to be integrated, let E_r be an element of S_r clashing with an element E_n of S_n, let C_{E_r} be the credibility coefficient of E_r in S_r and C_{E_n} the credibility coefficient of E_n in S_n, then the weights P_{E_r} and P_{E_n} respectively assigned to the elements E_r and E_n, in order to know which one will be privileged when conflicts are being solved are defined as follow:

$$P_{E_r} = C_{E_r} \text{ et } P_{E_n} = C_{E_n}$$

5 Conclusion

We have presented a process which allows to perform the integration of object-oriented schemas. This process is based on comparison rules, conflict-solving rules and superposition rules. The integration is structural but essentially semantic. To take into account the semantics provided by the design schemas, a tool like the fuzzy thesaurus is quite a valuable asset at all the process levels, because it makes it possible to find similarities between the schemas, but also to solve conflicts and to present the result schema.
This process also propose several solutions to the integration of two given schemas, in order to allows the choice of the best result, with regards to context-dependent criteria, like the reuse criteria for example.

A prototype of this integration method is under development.

We would like to improve our thesaurus by taking into account domain applications, during the acquisition phase, but also during the exploitation phase. We also would like to complete our integration process with specific modules, each on dedicated to one existent object-oriented model (like OMT or MCO). First, it will allow to integrate schemas expressed with one of these formalisms, and it will then become a federated framework for all these formalisms.

References

1. A. Motro. Superviews : virtual integration of multiple databases. *Transaction on software engineering*, SE-13(7):785–798, July 1987.
2. S. Navathe, R. Elmasri, and J. Larson. Integrating user views in database design. In *IEEE Computer*, pages 50–62, January 1986.
3. C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, December 1986.
4. P. Johannesson. Schema transformations as an aid in view integration. In *Advanced information systems engineering : proceedings of the 5th international conference on advanced information system engineering (CAISE'93), June 8-11, 1993*, volume 685 of *Lecture Notes in Computer Sciences*, pages 71–92. Springer Verlag, 1993.
5. P. Johannesson. Supporting schema integration by linguistic instruments. In *First workshop on applications of natural language to data bases(NLDB'95)*, pages 41–55, Versailles, France, 28-29 June 1995.
6. S. Spaccapietra, C. Parent, and Y. Dupont. Model independent assertions for integration of heterogeneous schemas. *Very Large Data Bases Journal*, 1:81–126, 1992.
7. I. Comyn-Wattiau. *L'intégration de vues dans le système SECSI*. PhD thesis, Université Paris VI, France, Octobre 1990.
8. C. Thieme and A. Siebes. An approach to schema integration based on transformations and behaviour. In *Advanced information systems engineering : proceedings of the 6th international conference on advanced information system engineering (CAISE'94), June 6-10, 1994*, volume 811 of *Lecture Notes in Computer Sciences*, pages 297–310. Springer Verlag, 1994.
9. P. Fankhauser, M. Kracker, and E.J. Neuhold. Semantic vs. structural resemblance of classes. *Sigmod record*, 20(4):59–63, December 1991.
10. C. Thieme and A. Siebes. Schema integration in object-oriented databases. In *Advanced information systems engineering : proceedings of the 5th international conference on advanced information system engineering (CAISE'93), June 8-11, 1993*, volume 685 of *Lecture Notes in Computer Sciences*, pages 54–70. Springer Verlag, 1993.
11. P. Buneman, S. Davidson, and A. Kosky. Theorical aspects of schema merging. In *International conference on extending database technology (EDBT'92), Vienna, Austria, March 23-27, 1992*, volume 580 of *Lecture Notes in Computer Sciences*, pages 152–167. Springer Verlag, 1992.
12. J. Rumbaugh, M. Blaha, W.Premerlani, F. Eddy, and W. Lorensen. *Object-oriented modelling and design*. Prentice Hall, New Jersey, 1991.
13. X. Castellani. *MCO, méthodologie générale d'analyse et de conception des systèmes d'objets*. Masson, 1993.
14. S. Castano and V. De Antonellis. Reuse in object-oriented information systems development. In *Object-oriented methodologies and systems, International symposium (ISOOMS'94), Palermo, Italy, September 1994*, volume 858 of *Lecture Notes in Computer Sciences*, pages 346–358. Springer Verlag, 1994.
15. D.A. Cruse. *Lexical semantics*. Cambridge textbooks in linguistics. Cambridge university press, 1986.

16. M. W. Evens. Relational models of the lexicon : introduction. In M. W Evens, editor, *Relational models of the lexicons*, pages 1–37. Cambridge university press, 1988.
17. J.C. Sager. *A practical course in terminology processing.* John Benjamins, 1990.
18. M. A. Iris, B. E. Litowitz, and M. W. Evens. Problems of the part-whole relation. In M. W Evens, editor, *Relational models of the lexicons*, pages 261–288. Cambridge university press, 1988.
19. G. Hirst. *Semantic interpretation and the resolution of ambiguity.* Cambridge University Press, 1987.
20. I. Mirbel. A fuzzy thesaurus for semantic integration of design schemes. In J. Sharpe, editor, *AI System Support for Conceptual Design (LIWED'95)*, pages 319–335, Ambleside, United Kingdom, 27-29 March 1995. Springer.
21. B.R. Gaines and M.L.G. Shaw. Comparing the conceptual systems of experts. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 633–638, Détroit, 1989.
22. A. Cavarero and E. Vittori. COD, un système de définition de classes d'objets. In *Inforsid*, Lille, France, Mai 1993.
23. S.R. Chidamber and C.F. Kemerer. A metrics suite for object-oriented design. *IEEE Transactions on software engineering*, 20(6), June 1994.