From: Proceedings of the Eleventh International FLAIRS Conference. Copyright © 1998, AAAI (www.aaai.org). All rights reserved.

Towards a cooperative information system modeling method based on integration of conceptual representations

Nahla Haddar, Faïez Gargouri, Abdelmajid Ben Hamadou
Laboratoire de recherche LARIS, FSEGS, B.P. 1088 Sfax - TUNISIA
Fax (2164) 279 139
Abdelmajid.Benhamadou@fsegs.rnu.tn

Abstract

This paper proposes a method for cooperative information system modeling based on integration of conceptual representations. Starting from existing information systems, a global information system is developed by integrating the initial ones. The originality of the approach is the use of a unification model from which the cooperation process is started, and the comparison of classes basing on static and behavioral aspects of objects. The last aspects are given by their life cycles.

Introduction

Nowadays, cooperation in information system (IS) modeling has become of great necessity due to the increasing need of enterprises and organisations to share and to reuse information. Besides, numerous methods, technics and tools coexist in the IS field all aiming at the restructuring and support of IS adaptation or development process. So, it is difficult for a large enterprise having different autonomous services, for instance, to choose or to apply a single method to design its IS. The design can be then carried out by a group of designers which work separately and use eventually a different analysis and design method to model each department IS. This helps to work concurrently on the one hand, and to overcome the problem of complexity on the other. The global system becomes an assemblage of units (sub-systems) endowed with various resources. But without concertation it is improbable that the global IS can be used in practice. The approach that we propose in this paper to realise this concertation consists in integrating the resulting subsystems in order to construct a global system. The integration is carried out in three steps deduced from the database field (Spaccapietra, Parent, and Dupont 1992) and adapted to the design field.

The integration of information systems is a difficult task since the global IS has to satisfy the following properties:

- Correctness: The common system has to contain all concepts present in any component subsystem correctly.
- Minimality: If the same concept is represented in more then one component subsystem, it has to be represented only once in the integrated system.
- Understandability: The integrated system must be easy to understand for the user.

It is especially the second property which makes the integration a non-trivial task.

Our approach differs from other approaches in the following points:

- The framework which we use for the integration aims to conceal the dissimilarity between object-oriented methods and tools and to be able to integrate and reuse modules which are developed through the use of different design methodologies.
- The use of object behavioral aspects given by their life cycles to detect similarities and conflicts between classes.

The remainder of this paper is organised as follows: section 2 is devoted to the presentation of our approach. Section 3 describes in detail the different integration steps. Section 4 presents our conclusions and future works.

IS Modeling by Integration

Modeling a complex IS by considering it as a single entity does not help to express its intrinsic complexity. This is why the classic modeling methods -whether they are organisational, functional or object oriented - are far from being suitable to model it. As information systems become more complex, researchers are trying to elaborate efficient methods to model them. In the last decade several approaches have been proposed (Ducateau, 1995), (Koriche, 1996).

Our proposal to model a complex IS is to start with modeling subsystems which cover all the domain to be modelled and then to integrate them. This approach has many advantages. First, the modeling problem is divided into less complex ones and therefore the complexity of the system can be controlled. Second, the modeling becomes a cooperative task which can be performed by designer teams who work concurrently and can use different analysis and design methods.

Our modeling method is based on three steps. First, information subsystems are translated using a generic model to obtain a unified representation of them. This step is called preintegration. Then, elements of the different information subsystems are compared to find similarities

Copyright © 1998, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

or conflicts between them. Finally the subsystems are merged after resolving conflicts.

The following figure shows these three steps. In this figure, different sub-systems IS1, IS2, ..., ISn are modelled using a given analysis and design method M1, M2, ..., Mn. After the preintegration step, each sub-system is modelled in the Unification Model (UM).

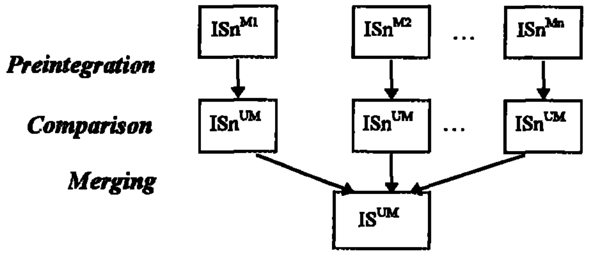


Figure 3: The three integration steps

The generic model which we have used as a unification model is called MGCO2 (Gargouri, 1995).

Integration Process

In the following, we describe each of the integration steps in detail.

Preintegration

In order to translate the subsystems conceptual representations (realised using OMT, O*, ...) into MGCO2 equivalent ones, (Gargouri, 1997) proposes an hybrid approach. It consists in transforming a conceptual representation obtained by applying any object oriented analysis and design method into its MGCO2 equivalent one. The transformation mapping T is defined as follows:

$$T: M \times C(M)^* \rightarrow C(MGCO2)^*$$

where M={OOAD, OMT, O*, ...}, C(M) is the set of M concepts and C(MGCO2) is the set of MGCO2 concepts. The resulting conceptual representation is independent from M. Moreover, an MGCO2 concept may be split into several M concepts and, inversely, an M concept may correspond to a sequence of MGCO2 concepts.

Example. Let's take a complex application that we call 'Clinic' concerning the data management of a clinic. It is decomposed into many sub-applications from which we distinguish the two sub-applications 'Administration' and 'Surgery' concerning respectively the management of information of the administration and those of the general surgery department. Figure 2 shows their conceptual representations. The first sub-application is modelled with OMT (Rumbaugh, 1991), while the other with O* (Brunet, 1993). The class 'Physician' of the system S1 represents all the clinic physicians while that of S2 represents only surgeons. Besides, the class S1.Patient represents all the patients admitted in the clinic and S2.Patient represents only patients who undergo operations. This example is easier than a real situation. However, it illustrates our method fully.

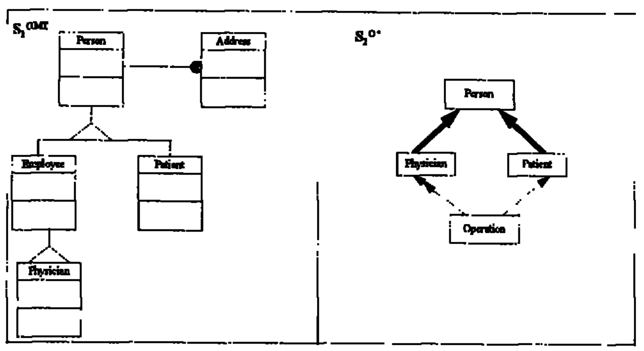


Figure 2: Conceptual representations of two subsystems

The transformation of both information systems leads to the following conceptual textual representations

```
Class Person: Abstract
Properties:
    number: INTEGER
    f_name: STRING
    name: STRING
    age: INTEGER
    address: Address
 Static constraints:
    Uniqueness: number
Methods:
    create()
    destroy( )
 ENDCLASS Person
Class Address: Abstract
 Properties:
    street: STRING
    nr: INTEGER
    city: STRING
Methods:
    change_adr()
ENDCLASS Address
```

```
Class Patient: Concrete
Inherits Person
 Properties:
    dateadm: DATE
    service: STRING
 Methods:
    change_ser()
ENDCLASS patient
Class Employee: Concrete
Inherits Person
Properties:
    entrance_date: DATE
    salary: INTEGER
Methods:
    change_sal()
ENDCLASS Employee
Class Physician: Concrete
Inherits Employee
Properties:
    specialty: STRING
ENDCLASS Physician
```

Figure3. Textual representation of S1

```
Class Operation: Concrete
Properties:
    nr: INTEGER
    op_date: DATE
    responsible: Physician
    patient: Patient
 Methods:
    create()
    destroy()
    change()
ENDCLASS Operation
Class Person: Abstract
 Properties:
    number: INTEGER
    f_name: STRING
    name: STRING
    age: INTEGER
    address: aggregation of
```

```
{street: STRING, nr: INTEGER,
city: STRING}
 Static constraints:
    Uniqueness: number
 Methods:
    create()
    destroy( )
    change_adr()
 ENDCLASS Person
 Class Patient: Concrete
 Inherits Person
 Properties:
    adm date: DATE
 Methods:
    change_state()
 ENDCLASS Patient
 Class Physician: Concrete
 Inherits Person
 ENDCLASS Physician
```

Figure 4. Textual representation of S2

Comparison

Once the translation is done, the next step in the integration process is to find common elements between the original subsystems.

Although an IS represents objects of the real world, with their properties and their behavior, nevertheless the integration process exceeds representations to consider first what is represented (the semantic aspects) rather than how it is represented (the syntactic aspects). Thus, we say that two information systems have some common things, if objects of the real world that they represent have common elements. The determination of correspondences between elements of information systems is therefore based on the real world objects semantics. We define the real world semantics (RWS) of an object class as being the set of real world object properties and behaviour represented by this class. Our definition of the RWS is different from that proposed in (Larson, Navathe, and Elmasri 1989) or (Spaccapietra, Parent, and Dupont 1995) concerning database schema integration in the fact that the later one is based on the class extension occurrences.

Basing on this definition of the RWS, we compare two classes from different information systems by comparing their attributes first. Once some similarities are detected, we compare the methods and then their state graphs.

Attribute comparison. The attribute comparison is based on the comparison of class structures. An attribute set Att(C) of a class C is formed by the attributes inherited from the superclasses and the specific class attributes. Before comparing two classes, it is therefore necessary to determine all their attributes. The structure of a class can be represented by a labelled infinite tree whose nodes are attribute types and whose leaves are predefined types. The relationship between two class types is then expressed in terms of homomorphism between trees representing their structures. From this mapping, the common attributes between classes are deduced. A detailed study of this formalism can be found in (Thieme and Siebes, 1995).

Example. The Structures of S1.Person and S2.Person will be represented by the two trees given by figures 5 below. Note that the trees are not infinite because we have no recursive types. The two trees are isomorphic since there is a bijective map between their nodes and edges. Consequently two classes have the same static structure and share the same attributes.

The attributes of a class represent its static aspect. Class methods are part of its dynamic aspect. If two classes share a significant number of attributes, they will probably share some methods which use some of the common attributes. The method comparison is presented in the next sub-section.

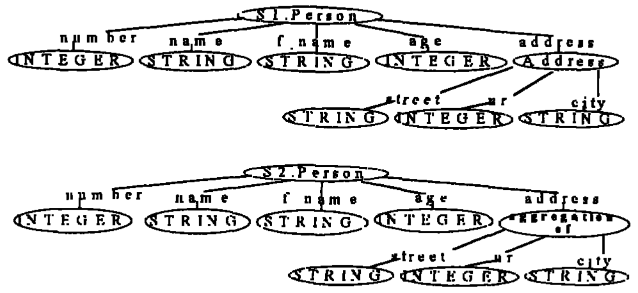


Figure5: Structure of S1.Person and S2.Person

Methods comparison. Each method of an object class has a body, a type (result provided by the method) and a signature giving the name and the set of attributes involved by the method. The method set Meth(C) of a class C is formed by the methods inherited from the superclasses and the specific class methods.

Methods of a class can be modelled using so called class-methods whose attributes represent the arguments, the body and the type of a method. (The name of the method is given by the name of the class). This concept is used also in Shood (Escamello De Los Santos, 1993).

Links between the class-methods are semantic links between the different methods. A definition of the different types of links is given in (Ravat, 1996). Especially, one distinguishes the inheritance link which models an overload. Indeed, an inheritance link between a superclass-method X and a sub-class-method indicates that the method Y overloads the method X.

So, similarly to class structures we can define structure of class-methods. Thus, comparison of methods of two classes of different information systems leads to attribute comparison of class-methods. The difficulty here resides in the comparison of method bodies. From the theoretical point of view, this comparison is possible. By choosing an appropriate formalism to specify method bodies this comparison can also be carried out automatically.

A class state graph describes the behavior of its objects. This behavior is expressed by the set of methods changing an object from a state to another. Thus, if two classes have some common methods, it is probable that they share also some parts of the state graphs of each other. So, the study of the relation ship between state graphs is important.

State Graph Comparison. A class state graph describes the behavior of its objects in response to an event occurrence. This behavior is the same for all the class objects. Basing on this property, we try to translate semantic relationships between objects of two classes belonging to different information systems. Each link will be expressed in terms of relationship between the state graphs of the considered classes. This helps, given two state graphs, to determine the relationship existing between the corresponding classes. This relationship can be either an inclusion, a strict intersection, or a disjunction.

Let C be a class of an IS S, and O an object of this class. Let st(C) be the set of states that an object O can take during its life cycle. st(C) contains a particular state state0 that corresponds to the state taken by O before its creation and after its destruction. The C state graph, G(C), can be defined as follows:

$G(C) = (Evt(C), Const(C), Meth(C), st(C), state0, \delta, \lambda)$ where the functions δ and λ are defined by:

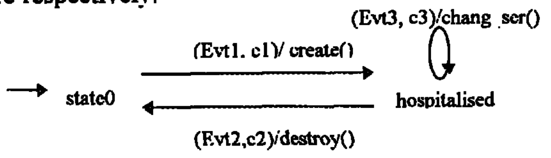
$\delta: st(C) \times Evt(C) \times Const(C) \rightarrow st(C)$

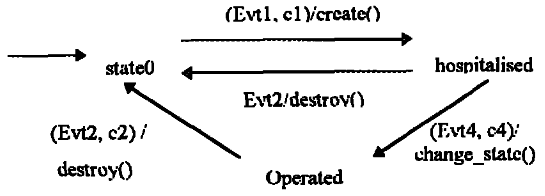
$\delta(q, e, c)$ = the state that takes an object of C being in the state q in response to an event e when the condition c is satisfied.

$\lambda: st(C) \times Evt(C) \times Const(C) \rightarrow Meth(C)$

$\lambda(q, e, c)$ = the action activated on an object of C being in the state q in response to an event e when the condition c is satisfied.

Example. The state graphs of S1.Patient and S2.Patient are respectively:





where Evt1: appoint patient to a service;
Evt2: patient leaves the clinic;
Evt3: patient changes services;
Evt4: operate patient.
c1, c2, c3, c4 are constraints.

Remark that the event sequence relative to a path p(q0, qn) in G(C) represents a scenario of the life cycle of a C object.

In the following, Let S1 and S2 be two distinct information systems, C1 and C2 be two classes belonging respectively to S1 and S2. We study the relationship that can exist between two state graphs. The definition of inclusion or intersection of two state graphs is provided by the theory of graphs.

If the objects behavior of a class is the same as a part of the objects behavior of another class, then the former contains the first. This fact can be demonstrated by the following lemma.

Lemma 1. We have $G(C1) \subseteq G(C2) \Rightarrow C1 \subseteq C2$

Proof. Suppose that $G(C1) \subseteq G(C2)$. Then all paths of G(C1) are paths of G(C2) too. If $C1 \not\subset C2$ then one could find an object O such that $O \in C1$ and $O \notin C2$. Since $O \in C1$ means that there exists a path p(q0, q0) in G(C1) whose sequence of events is a scenario of LiC(O) and $O \notin C2$ means that no path in G(C2) corresponds to a scenario of LiC(O), we obtain a contradiction with our hypothesis because p(q0, q0) belongs to G(C2).□

If the behavior of a subset of class objects is the same as a subset of objects of another class, then the two classes have a non empty intersection. This fact be demonstrated by the following lemma.

Lemma 2. We have $G(C1) \cap G(C2) \neq \phi \Leftrightarrow C1 \cap C2 \neq \phi$.

Proof. $G(C1) \cap G(C2) \neq \phi \Leftrightarrow Evt(C1) \cap Evt(C2) \neq \phi \wedge st(C1) \cap st(C2) \neq \phi \wedge Const(C1) \cap Const(C2) \neq \phi \wedge Meth(C1) \cap Meth(C2) \neq \phi \wedge \forall p(q0, q0)$ in G(C1), $length(p(q0, q0)) > 1$: p(q0, q0) is in G(C2). $\Leftrightarrow \exists O$: the event sequence relative to p(q0, q0) is a scenario of $LiC(O) \wedge O \in C1 \wedge O \in C2 \Leftrightarrow C1 \cap C2 \neq \phi$. □

Conflicts taxonomy . When a correspondence describes some elements as being identical (i.e., they have the same representation and the same semantics) their integration is then obvious: the integrated element (which will be present in the final system) will be identical to the input elements. But, in most cases, the corresponding elements present some differences in their representations or in their semantics. This case leads to a conflict situation. We give hereafter a taxonomy and some examples of conflicts occurring when comparing two subsystems.

- Classification conflicts: A classification conflict occurs when the corresponding classes describe object sets which are different but semantically linked. In our example Patient in S1 describes patients that have been admitted in the clinic while Patient in S2 describes only patients who undergo operations.
- Structural conflicts: A structural conflict occurs when the corresponding elements are described using different concepts belonging to different abstraction levels, for example a class and an attribute. In our example, there is a structural conflict between the S1 class Address and the S2 attribute Person.address.

Other types of conflicts can be met, namely, descriptive conflicts and dynamic aspects conflicts.

Conflict resolution and merging

Once the correspondences between systems are established, the integration can begin. Every correspondence is analysed in order to determine which integration rule will be applied to obtain the corresponding final system elements. The major difficulty of this step to resolve is the emphasis on the different conflicts and the semantic problems detected in the comparison step and their resolution. Some merging rules must be therefore established .

As a solution for the classification conflicts we propose to include in the integrated information system an appropriate generalisation-specialisation hierarchy as detailed by the following rules:

Rule1: $G(C1) \subseteq G(C2) \Rightarrow$

$Generalise(C2) \wedge Mask(Meth(C2) \setminus Meth(C1)) \wedge Mask(Att(C2) \setminus Att(C1)) \wedge Specialise(C1, C2)$

Rule2: $G(C1) \cap G(C2) \neq \phi \Rightarrow$

$Generalise(C1 \cap C2) \wedge Specialise(C1 \setminus C2, C1 \cap C2) \wedge Specialise(C2 \setminus C1, C1 \cap C2)$

where Generalise(C) is a method which creates a generalisation class C; Specialise(C, C') is a method which makes from C a specialisation of class C'; and Mask is a function defined on the set of attributes and methods of a class and it allows the masking of some properties of classes (Castellani, 1993). By comparing both systems of our example we obtain following similarities and conflicts:

- the classes S1.Person and S2.Person are equivalent because they have the same attributes. the same methods and the same state graphs;
- the classes S1.patient and S2.patient have a not empty intersection;
- the class S2.physician is included in S1.physician.

Conclusion

In this paper we have given a cooperative IS modeling method based on the integration of systems conceptual representations. The method decomposes the cooperation realisation process into three steps. In the first step the initial systems are translated by a generic model in a unified representation. The second step, consists in searching for the similarities between the elements of the different systems and in detecting eventual conflicts. In the last step the conflicts must be resolved and the systems are merged by using some integration rules.

When comparing elements of the initial systems, we have used the state graphs of the systems classes. We have proved that the relationship between these graphs can give information about the semantic relationship between classes.

The formalisms used for element comparison are only proposals that have to be developed. In future works, we will study the different possible conflicts between information systems elements in more detail and we will try to extend the set of merging rules.

Our approach can be extended to deal with complex system modeling (and not only complex IS modeling) without any additional reflections. Moreover, this approach is perhaps a first step towards the reuse of conceptual representations. Indeed, it facilitates IS modeling by integrating some existing conceptual representations. In addition to saving realisation time and reducing costs, the reuse of conceptual representations results in the generalisation or even the standardisation of conceptual constituents capable of leading to the use of real conceptual components.

References

Brunet, J. 1993. Analyse conceptuelle orientée objet. Ph.D. diss., Paris 6 University. France.

Castellani, X. 1993. *MCO: Méthodologie générale d'analyse et de conception des systèmes d'objets Tome1: Ingénierie des besoins*. Edition Masson.

Ducateau, C. F.; and Picavet, M. 1995. Progressive adjusting Process for Data Modeling. In Proceedings of the International Conference on Industrial Engineering and production Management. 235-244. Maroco.

Dupont, Y. 1995. Problématique de résolution contextuelle des conflits de fragmentation dans l'intégration de schémas. *Ingénierie des Systèmes d'Information* 3(1):29-58.

Escamello De Los Santos. J. G. 1993. Shood, un modèle méta-circulaire de représentation des connaissances. Ph.D. diss., Institut National Polytechnique de Grenoble. France.

Gargouri, F. 1995 Proposition d'un modèle générique pour la conception orientée objet des systèmes d'information et étude de son implantation. Ph.D Paris 5 University.

Gargouri, F.; Ducateau, C. F.; Gargouri. W. ; Haddar, N. 1997. About cooperation for complex applications and information systems modeling. In Proceedings of the International Conference on Industrial Engineering and Production Management, 258-267. Lyon. France.

Koriche, F. 1996. Une méthode de modélisation des systèmes d'information coopératifs. *Ingénierie des Systèmes d'Information* 4 (2):195-218.

Larson, J. A.; Navathe, S. H.; Elmasri, R. 1989. A theory of attribut equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering* 15(4):449-463.

Parent, C.; Spaccapietra, S. 1996. Intégration de bases de données: panorama des problèmes et approches. *Ingénierie des Systèmes d'Information* 4(3):333-358.

Ravat, F. 1996 La fragmentation d'un schéma conceptuel orienté objet. *Ingénierie des Systèmes d'Information* 4(2): 161-193.

Rumbaugh, J.; Blaha, M. ;et al. 1991. *Object oriented modelling and design*. Printice Hall publishing company. Engelwood Cliffs.

Thieme, C.; Siebes. A. 1995. Guiding schema integration by behavioral information. *Information Sytems*.20(4):305-316.