Information System Design Based on a Domain Ontology and User Requirements

Ana Simonet[1], Michel Simonet[1]

[1] Agim laboratory, Faculté de Médecine,
38700 La Tronche, France
{Ana.Simonet, Michel.Simonet}@agim.eu

Abstract. Domain ontologies can support the production of the conceptual schema of a database but lack the behavioral properties that are necessary to generate an operational Information System. We present an approach to enrich a domain ontology by properties extracted from the User Requirements. Besides determining the required subset of the ontology, we produce the database of the Information System and its API. This approach is implemented by the ISIS (Information System Initial Specification) platform that generates automatically an operational Information System, including a prototypical Graphical User Interface that enables the users to validate the expressed needs and refine them if necessary. Thanks to a reduction of the cycle « expression-refinement of needs / production of target system / validation » the number of these cycles can increase without impacting the global cost of the project and the final result is closer to the real needs of the users.

Keywords: Ontology, Information System, User Requirements, Database Design.

1 Introduction

The term « Information System » is employed in several domains such as database management, knowledge management, information retrieval, with different meanings [7]. In this paper, we consider the point of view where an Information System (IS) is dedicated to the storage and management of data of a company and its accessibility to other company systems [8]. For this kind of system, considerable involvement of end-users during the analysis process of the Information System and a clear formulation of its objectives are two factors that contribute to the production of software compliant with the users' expectations [2][5].

Another factor that contributes to the production of quality software is the use of a domain ontology. A domain ontology (hereafter *an ontology*) expresses an agreement of the actors of a domain upon the concepts of the considered domain and their relationships, and thus provide a shared conceptualization of a domain. It can play the role of a semantic referential by the companies working in the same domain or be used as a referential for the building of a new IS. The use of domain ontologies for IS design is considered as a way to reuse the knowledge about a domain, and thus allow a quicker appropriation of the domain by the analyst. However, reusing such knowledge in the design of an IS is most often limited to the construction of the conceptual schema of its database. This is the case for example of the systems OMMDE [13] and SISRO [6].

As an ontology tends to have a larger number of concepts than a class diagram, the designer has to select those of interest for his particular case. However, mastering thousands of concepts is beyond human capacities. Thus, extracting the subset of concepts that is pertinent in a given situation has become a problem in itself.

Moreover, producing a sole conceptual database schema is not sufficient to guarantee that the generated software will be compliant with the expectations of end-users. Consequently, other aspects of the domain must be represented by using other models of an analysis method (e.g., UML) [2].

In the design of an IS we have chosen to exploit the User Requirements in order to select the sub-ontology that contains all the concepts needed by the IS. Moreover, enriching the ontology with properties deduced from the User Requirements enables us to produce not only the database schema but also an operational IS with a prototype GUI.

This approach is implemented in the ISIS system, a platform for the specification of Information System from a domain ontology. The ISIS system aims at implementing the equation *Ontology + User Requirements = Information System*. In practice, given an ontology, represented by a graph, and User Requirements, represented by the Use Cases of the application, it identifies the sub-graphs needed for each Use Case. Then it proposes the sub-ontology by suppressing the concepts and the relationships that are not pertinent to the business processes[1]. It then produces a database that is optimized for the User Requirements, its API, i.e., the SQL code for the queries corresponding to the Use Cases, and a prototype GUI that makes possible the immediate validation of the User Requirements.

The paper is organized as follows. We firstly we examine the ISIS project, the meta-concepts of its model and the principal behavioral properties. Then, the ISIS methodology and tool are presented through an example.

2. The ISIS Project

ISIS is the acronym for *Information Systems Initial Specification*. It is a model, a methodology and a tool for the design of an Information System (database, API and prototype GUI), from a Domain Ontology and a set of Users Requirements. All the IS design methodologies use a central diagram[2], which we refer to as a *conceptual diagram*. We use a unique diagram – the so-called *conceptual diagram* – to express the static properties of the entities of the domain as well as their dynamic (behavioral) properties.

Initial in its acronym means that the starting point of a modeling in ISIS is close to the users' *natural* perception of their business and its expression in natural language.

[1] It also enables the addition of new concepts if necessary.

[2] *class diagram* in object methods, *conceptual schema* in E-R models, *logical schema* in relational databases

Naming the concepts (e.g., vendor, customer, product, price, quantity, address …) of the domain and their interrelationships is the first step in the design of an IS following the ISIS approach. This is ontological work, and the input to ISIS can be an existing domain ontology or a micro-ontology of the considered application domain that is built at the time of the design. In both cases, we will call DO (Domain Ontology) the ontological structure that constitutes the input to ISIS. A DO is represented by a graph where the nodes are the concepts of the domain and the arcs are the relationships.

2.1 User Requirements

The second step in the ISIS approach is to enrich the DO with the behavioral properties of the entities of the real world. These properties are deduced from the User Requirements, expressed by the Use Cases needed for the business tasks. We have classified the Use Cases into two categories: those whose objective is to consult existing data (e.g., *titles of the books borrowed by a reader*), and those in which objects can be created, modified or suppressed (e.g., *create a new reader*). We call the former Sel-UC (for Selection Use Case) and the latter Up-UC (for Update Use Case). Each Use Case has input and output data, represented by concepts of the DO. For example, the above Sel-UC is interpreted as (**input**: reader, **output**: title) according to the DO of Fig. 3. In the context of a DO, the set of Sel-UC enables ISIS to determine the subgraph of the DO that is actually needed to produce the Information System of the application. The set of Up-UC enables ISIS to deduce the implementation (physical schema, API and prototype GUI).

2.2 The ISIS Model

The ISIS model belongs to the family of *binary relational* models [1], which are among the simplest to represent the knowledge about a domain. Its meta-concepts are *concept*, *binary relation* (or simply *relation*), and *subsumption* (or *specialization*) *relation*. The concepts of a given domain and their relationships are represented through the DO graph.

Definitions

- A *concept* is an intensional view of a notion whose extensional view is a set of instances.
- A *binary relation* R between two concepts A (domain) and B (range), noted R(A,B), is considered in its mathematical sense: a set of pairs (a, b) with $a \in A$ and $b \in B$.
- the *image* of x through R, noted R(x) is the set of y such that R(x,y).
  $R(x)_t$ is the image of x through R at time t.
- An *association* is a pair of binary relations, reverse of one another.
- A *subsumption relation* holds between two concepts A and B (A subsumes B) iff B is a subset of A.

In a DO, *static properties* (or *constraints*) of concepts and relations are given. Among these, only the minimal (generally 0 or 1) and maximal (generally 1, * or n) cardinalities of relations and unicity constraints are mandatory. Other constraints, such as Domain Constraints and Inter-Attribute Dependencies may be considered for the production of Intelligent Information Systems under knowledge-based models such as Description Logics [10] [11]. As in other models, the static constraints govern the production of the logical relational database schema. Behavioral constraints are necessary to automatically produce the logical object database schema, the physical relational (or object) database schema, the API of the functional kernel and the prototype GUI. In ISIS, the main behavioral properties that are considered are [12]:

- The **criticity** of a relation for a given application. Critical relations are the relations needed in the queries of Sel-UC (§2.3).
- The **modifiability** of a relation in a given application. A modifiable relation is a non-monotonous relation. It is deduced from the concepts used in Up-UC (§2.4).

These properties and the knowledge of the input and output parameters of the UCs enable the automatic production of an operational IS with a prototype GUI.

In ISIS we consider three categories of concepts: *predefined concepts*, *primary concepts* and *secondary concepts*. Predefined concepts correspond to predefined types in programming languages, e.g., string, real, integer. Primary and secondary concepts are *built* to represent the concepts specific to a domain. Primary concepts correspond to those concepts whose instances are usually considered as atomic. A primary concept is connected to one and only one predefined concept (and to one or several secondary concepts). Contrary to primary concepts, a secondary concept is connected only to *built* concepts, i.e., primary or secondary concepts. We name valC the relation whose domain is a primary concept C and whose range is a predefined concept (e.g., valAge, valName). Fig. 1 represents the ISIS *complete* diagram designed to model persons with a name and an age.

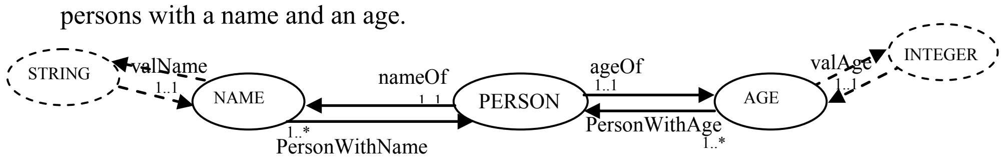


Fig. 1. ISIS complete DO modeling person name and age.

Representing the predefined concepts increases the complexity of a DO. Except in particular situations, e.g., to support the training for the foundations of ISIS, the predefined concepts and their relations with primary concepts are masked in the ISIS diagrams.

2.3 Critical Relations

Our main criterion to select which binary relations, and consequently which concepts, are needed in the informatics system is to consider the relations present in the selection queries of the Sel-UC and which are consequently considered important (critical) for the company. **Critical relations** are those participating in at least one query involved in one of these Use Cases[3]. However, the designer can decide to make any relation critical, independently from critical queries.

[3] In the current version of ISIS all the queries in Sel-UC are considered critical.

The ultimate purpose of an IS is expressed through its selection queries, hence our choice of these queries to decide which relations are critical. The update queries are only needed to ensure that, at every moment, data in the IS are complying with data in the real world; they are not considered in the determination of the critical relations.

Definitions

- A selection query is ***critical*** iff it is part of a Sel-UC.
- A selection query Q is defined by a triple (I, O, P) where:
  - I is the set of *input* concepts of Q
  - O is the set of *output* concepts of Q
  - P is a set of paths in the OD graph.
- The triple (I, O, P) defines a subgraph of the OD.
- A path p(i, o) in a query (I, O, P) is an ordered set of relations connecting $i \in I$ to $o \in O$.
- A binary relation is critical iff it belongs to at least one critical selection query or if it has been explicitly made critical by the designer.
- An association in a DO is critical iff at least one of its relations is critical.

2.4 Modifiable Relations

Definition

- Given a relation R(A, B), R is modifiable iff there exists $a \in A$ such that $R(a)_t$ is different from $R(a)_{t+1}$.

To express the **modifiability** property we had to extend the classical binary relational model, which has only two categories of nodes[4], to a model with three types of nodes (§2.2). This difference is illustrated by Fig. 1 and Fig. 2. Fig. 2 represents the Z_0 schema of a set person with two access function[5], *ageOf* and *nameOf*, where the notion of access function is derived from that of relation. The transformation of this schema into a schema in relational (object/E-R) model induces a representation of *ageOf* and *nameOf* as attributes of a table (class/entity) person.

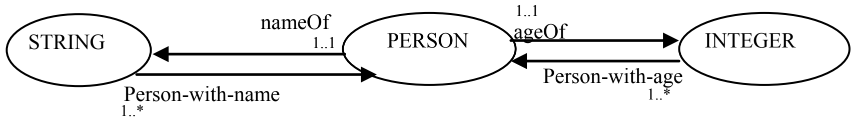


Fig. 2. Z_0 schema modeling persons with name and age [1].

In ISIS (Fig. 1), person, age and name are *concepts*, and so are string and integer. person, age and name are concepts specific to the application domain, whereas integer and string may belong to any application. Thanks to the three sorts of concepts and to the behavioral properties added to the diagram, ISIS will propose that a given concept such as age, name or person be represented as an object or as a

[4] E.g., in Z_0, concrete (structured) and abstract (atomic) sets.

[5] « An access function is à function which maps one category into the power set of another (the set of all subsets) ».

literal, according to the ODMG classification [4], and among the objects propose candidates to become database indexes.

Let us consider the Use Case *change the age of a person* in the context of Korea and other Asian countries, where a person changes his age on Jan. 1st at 0h [9]. If the designer is conscious that an age update on Jan. 1st will concern millions or billions of persons, he will choose an object representation, which leads to at most 140 updates (if the age ranges from 0 to 140) instead of millions or billions with a representation as an attribute. This «best» solution cannot be **automatically** produced from the diagram of Fig. 2 where the only relation[6] that may be considered as modifiable is *ageOf*. In the ISIS representation (Fig. 1) two relations are potentially modifiable: *ageOf* and *valAge*. Making *ageOf* modifiable models the update of **one** person, whereas making *valAge* modifiable models the update of **all** the persons with a given age. The best modeling for Korea is then to consider *valAge* as a modifiable relation, and then model age by a new table (class/entity) referenced by the table person. Moreover, the best modeling for Europe is to consider *ageOf* as the modifiable relation, which makes person the only necessary table (class/entity). Distinguishing primary concepts and predefined concepts is necessary to differentiate these two ways of modeling the update of the age of a person, or of other primary concepts such as salary: either change the salary of one person (promoted individually) or change the salary of all the persons belonging to a given category.

3 ISIS Methodology and Tool through an Example

3.1 Domain Ontology (DO)

The first step in the ISIS approach is the design or the import of a DO. The DO can be checked for well-formedness (absence of cycle, no relation between two primary concepts …). Fig. 3 shows a partial DO of a library management application:

- book, childBook, isbn, … , age and type are *built* concepts for this IS.
- isbn, title, author, age and type are primary concepts: the designer must choose the predefined concept related to a primary concept.
- book and childBook are secondary concepts.
- childBook is a subconcept of book. It is connected to book by a subsumption relation. Every instance of childBook is also an instance of book.
- pairs such as (1,1), (0,*) represent respectively the minimal and maximal cardinalities of a relation.
- the relation corresponding to a semantic identifier (key) of a secondary concept is represented by a double line plus a key symbol attached to the concept.

[6] Called *access function* in Z_0.

- *categoryOfBook*[7], *copyOfBook* … are binary relations. Their extension is a subset of the Cartesian product of their origin and end concepts. *categoryOfBook* and *bookOfCategory* constitutes a *binary association*.

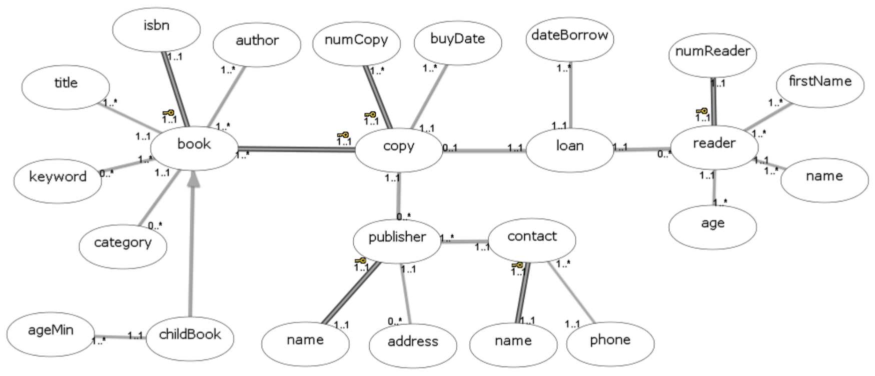


Fig. 3. DO of a library management application.

3.2 Behavioral Properties

The second step in the ISIS methodology is the **enrichment** of the DO with the behavioral properties of the application. Let us consider the following Use Cases:

UC1 - *books of a reader*: given a reader (identified by its numReader), find the books he has borrowed; for each book, display its title, its authors and its isbn.

UC2 – *loans of a reader*: given a numReader, display his firstName and his name, and for each loan, display the dateBorrow and the title and author.

UC3 - *list of the books of the library*: for each book in the library, display its isbn, title, authors, keywords and category of the book.

UC4 - *readers of a book*: for a given book (identified by its isbn) display its borrowed copies; for each copy, display its numCopy and its reader (numReader, firstName, lastName).

UC5 - *new book*: create a new book. **UC6** - *new copy*: create a new copy.

UC7 - *new reader*: create a new reader. **UC8** - *new loan*: create a new loan.

UC9 - *update book*: modify the keywords and/or the category of a book.

The first four are Sel-UC and the last five are Up-UC. For each query of an UC the designer identifies the concepts it concerns and annotates them as **input**, **output**[8] concepts.

UC1: **in**{numReader}, **out**{title, author, isbn}; **UC2**: **in**{numReader}, **out**{firstName, lastName, dateBorrow, isbn, title, author}; **UC3**: **out**{isbn, title, author, keyword, category}; **UC4**: **in**{isbn}, **out**{numCopy, numReader, firstName, lastName}; **UC5**: **in**{book}; **UC6**: **in**{copy}; **UC7**: **in**{reader}; **UC8**: **in**{loan}; **UC9**: **in**{isbn, keyword, category}.

[7] *categoryOfBook* denotes the relation of domain book and range category. The name aaOfBb is produced by ISIS for a relation with domain aa and range bb and may be changed.

[8] input and output concepts correspond to input and output concepts of the procedures of the functional kernel of the IS.

Each query of a Use Case is used to annotate the DO by its input and output concepts. Fig. 4 illustrates UC2. The designer annotates the concept numReader as **input** concept (downward arrow ⬇) and the concepts firstName, lastName, dateBorrow, isbn, title and author as **output** concepts (upward arrow ⬆). ISIS calculates and presents the paths between the input and the output concepts. The intermediate concepts, such as copy, are automatically annotated with an orange flag (⚑).

All the relations of a path in a query of Sel-UC are automatically annotated by ISIS as *critical*. In the example of Fig. 4, readerOfNumreader, firstNameOfReader, nameOfReader, loanOfReader … are critical.

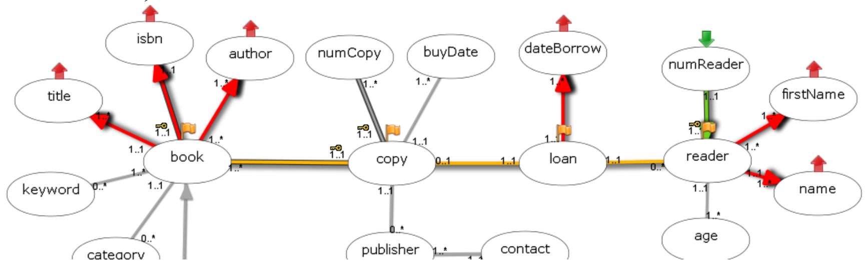


Fig. 4. Subgraph of the Use Case UC2.

When different paths are possible between an input concept and an output concept, the designer must choose the intermediate concepts by moving the orange flag(s). When several relations hold between two concepts, one of them must be selected.

4.3 Sub-Ontology Extraction

In the third step ISIS deduces a diagram that is the smallest subgraph that contains all the Sel-UC subgraphs and proposes[9] to suppress the concepts that are not needed.

For example, if the query presented in Fig. 4 is the **only** query of the IS, the relations of the associations *book-category* and *book-keywords* are not critical. Thus, the designer must decide either to suppress the association or to make critical one of its relations. When a concept becomes disconnected from the other concepts of the DO, it is suppressed. This constitutes the first phase of the *simplification* process where the objective is to determine the sub-ontology of an application.

Before generating the IS, ISIS proceeds to a second phase of simplification, by proposing to eliminate the concepts that are the domain or the range of a critical relation but that do not bear information significant for the business process. For example, considering the query of Fig. 4 as the **only** query of the IS, the concept **copy** can be eliminated, as it acts only as an intermediate concept linking book to loan.

Fig. 5 shows the simplified sub-ontology obtained by taking into account the whole set of Sel-UC. The concepts buyDate, publisher, contact, age (reader) … have been suppressed and will not appear in the generated application, because they are not used in any of the four Sel-UC (and the designer has agreed with their suppression).

[9] The designer decides to keep a concept or not.

4.4 Object, Value and Index Deduction

From the update queries of Up-UC, ISIS deduces which relations are modifiable (cf. § 2.4). For example, UC5 (*new copy*) enables ISIS to deduce that the relation copyOfBook (Fig. 5) is modifiable.

Considering the critical modifiable relations, ISIS deduces which concepts should be represented as values or as objects, and among the latter which ones are proposed to become indexes of the generated database. Again, the designer may decide to make other choices. For example, in the relational model, an object concept is represented by one or several tables; a value concept is represented as the domain of an attribute; an attribute represents a relation whose domain is a value concept. Fig. 5 shows the object-value-index deductions on the library example.

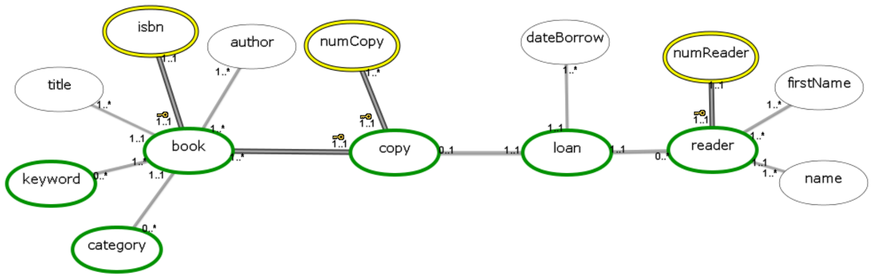
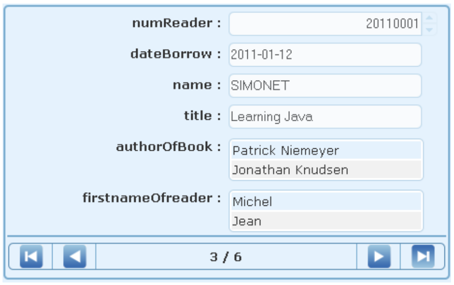


Fig. 5. Simplified DO with Object-Value-Index deductions for the library example.

Object concepts: book, copy, loan, reader, keyword, category[10].
Potential indexes: isbn, numCopy, numReader.
Value concepts: title, author, dateBorrow, firstName, Name.
To make these deductions, ISIS uses a set of rules, among which:
R01: C is an object concept if it is the domain of at least one modifiable critical relation.
R02: C is an object concept if it is the domain of a critical relation to an object concept.

4.6 Generation of Software Artifacts



numReader :	20110001
dateBorrow :	2011-01-12
name :	SIMONET
title :	Learning Java
authorOfBook :	Patrick Niemeyer Jonathan Knudsen
firstnameOfreader :	Michel Jean

Fig. 6 shows the GUI corresponding to UC2 (*borrowals of a reader*) in the PHP-MySQL application that is automatically generated. In spite of its basic ergonomics, the generated GUI enables users to verify the presented items and their type, and check whether the dynamics of windows corresponds to the needs of their business process.

Fig. 6. Prototype GUI: UC2 query.

[10] Note that keyword and category are primary, i.e., non-structured, concepts.

6. Conclusion and Perspectives

The design of Information Systems is confronted with several problems: domains become more and more complex; users belong to several categories and have distinct knowledge and needs; collaboration between IS is needed to ensure the survival of the enterprise. Various solutions aiming at a better acceptation of the final system have been proposed, among which the use of a domain ontology and an active collaboration between analysts and end-users during the design phase of the IS.

A domain ontology can support the design of the conceptual schema of the IS database [6] [13]. From our experience, a conceptual database schema (e.g., class diagram or E-R schema) concerns analysts rather than end-users, who do not have the knowledge necessary to understand the meta-concepts chosen by the designer and consequently are only able to validate the terms used in the schema. Moreover, they interpret them in their own cultural context and two users validating the same conceptual diagram may expect different systems.

An effective collaboration between designers and end-users necessitates a common language that is mastered by both parties [5], so that one can quickly identify possible mutual misunderstandings. Active collaboration also requires a high degree of availability of both parties in order that user requirements and business rules be well understood by the designer. Such availability increases the cost of the software and thus may question the feasibility of the project [3]. In order to favor an active collaboration, we have aimed at the automatic production of an operational IS that can be immediately tested by end-users. Quickly reacting to the generated IS enables them to refine the user requirements.

We have proposed a binary relational model, which has a limited number of meta-concepts, making it easier to be understood by end-users: *concept* (built and predefined), *binary relation* (and *association*) and *ISA relation*. Contrary to other methods that use several diagrams, e.g. class diagram, sequence diagram, activity diagram, etc. in ISIS we chose to enrich a DO with the behavioral properties of the instances of its concepts. The main two behavioral properties we have identified are the *criticity* and the *modifiability* of a relation. However, deciding which relations are critical or modifiable is outside the capabilities of end-users, whereas they know the data necessary for each Use Case, which is made explicit in ISIS through its *input* and *output* parameters. The representation of the input and output parameters of the UC makes it possible to deduce the behavioral properties of the instances of the application. From these parameters ISIS deduces which relations are critical or modifiable. ISIS then deduces and proposes the concepts that – for this application – should be omitted. Finally ISIS proposes the concepts that should be represented as objects, values or indexes in the application implementation. The designer can accept or refuse these proposals. ISIS then proceeds to the automatic generation of the database, the API and a prototype GUI of the IS. These software artifacts enable end-users to verify the adequacy of the IS for their needs and refine them if necessary.

The current ISIS tool has been developed in Java with a dynamic web interface. It generates a PHP-MySQL application. A console enables the programmer to write SQL code, which makes it possible to write more complex queries (e.g., recursive queries). Future work encompasses the generation of UML and E-R diagrams, and the use of the ISIS methodology for the integration of heterogeneous databases.

Acknowledgments. The authors want to thank Eric Céret, who designed the current web version of ISIS, and Loïc Cellier who continues its development. They are also grateful to Cyr-Gabin Bassolet, who designed and implemented the early prototypes and participated actively in the first phases of the project.

References

1. Abrial, J.R.: Data Semantics. in J.W. Klumbie and K.I. Koffeman (Eds), Database Management, North-Holland, Amsterdam, 1-59 (1974)
2. Burton-Jones, A., Meso, P.: Conceptualizing Systems for Understanding: An Empirical Test of Decomposition Principles in Object-Oriented Analysis. Information Systems Research vol 17, No1, pp 38-60 (2006)
3. Butler, B., Fitzgerald, A.: A case study of user participation in information systems development process, 8th Int Conf on Information Systems, pp 411-426 Atlanta (1997)
4. Cattell, R. G. G., Atwood, T., Duhl, J., Ferran, G., Loomis, M., Wade, D.: Object Database Standard: ODMG-93, Morgan Kaufmann Publishers (1994)
5. Cavaye, A.: User Participation in System Development Revisited, Information and Management (28) pp 311-323 (1995)
6. Fankam Ch., Bellatreche L., Dehainsala H., Ait Ameur Y., Pierra G. SISRO : Conception de bases de sonnées à partir d'ontologies de domaine. Revue TSI vol.28 pp1-29 (2009)
7. Guarino, N.: The Ontological Level: Revisiting 30 Years of Knowledge Representation. In: Conceptual Modeling: Foundations and Applications. Essays in Honor of John Mylopoulos, A. Borgida et al. (eds.), Springer Verlag, pp 52-67 (2009)
8. Matheron, J.P.: Comprendre MERISE, outils conceptuels et organisationnels, Ed. Eyrolles (1987)
9. Park J., Ram S., Information Systems: What Lies Beneath, ACM Transactions on Information Systems, Vol. 22, No. 4 pp 595–632, (2004)
10. Roger M., Simonet A., Simonet M.: A Description Logic-like Model for a Knowledge and Data Management System. DEXA 2000, Greenwich (UK) (2000)
11. Simonet A., Simonet M.: Objects with Views and Constraints: from Databases to Knowledge Bases. Object-Oriented Information Systems OOIS'94, D. Patel, Y. Sun and S. Patel eds, London, Springer Verlag, pp 182-197 (1994)
12. Simonet A.: Conception, Modélisation et Implantation de Systèmes d'Information. Habilitation à Diriger des Recherches. Université de Grenoble (2010)
13. Sugumaran V., Storey V. C.: The role of domain ontologies in database design: An ontology management and conceptual modeling environment. ACM Trans. Database Syst., vol. 31, ACM Press, New York, NY, USA, pp 1064-1094 (2006)