Integrating Behavioral Properties in an Ontology to Automatically Produce an Information System

Ana Simonet
AGIM laboratory
Université de Grenoble
38700 La Tronche - France
Ana.Simonet@imag.fr

Abstract— **The use of domain ontologies lies at the very heart of Information Systems, as they provide a shared conceptualization of domains. However, using an ontology, which by essence embraces a wide range of concepts, poses several questions among which the determination of the subset of its concepts that is actually needed for an application and how this sub-ontology can support the production of Information System artifacts compliant with user requirements. We present an approach based on the enrichment of an ontology with properties deduced from the user requirements to determine the subset of the ontology that is necessary for a new Information System. These properties also make possible the automatic production of an operational IS with a prototypical GUI. This approach, implemented by the ISIS platform, can support a high number of specification-implementation-validation cycles without increasing the global cost of the project.**

Keywords- Ontology, Information System Design, User Requirements

I. INTRODUCTION

Information System (IS) design involves the representation of the knowledge of a domain through three families of models in order to represent the expected functional, static and dynamic aspects of the IS. For example, when using UML, a use case diagram, a class diagram and one or several dynamic diagrams (e.g., state transition diagrams, sequence diagrams, activity diagrams, collaboration diagrams …) must be designed by the analyst [4][19]. These diagrams allow the analyst to increase his understanding of the domain, help communication between developers and users and facilitate the implementation of the information system. Today, new needs arise and question the classical process. Among these we underline the need for the company to rely on accepted referentials in order to ensure the usability of the IS over time and the need for the analysis process to take in account the dynamics of the company, which may require changes in the user requirements, even during the design process [17].

The enterprise global schema is an attempt to answer the first question [12]. This schema represents the entities specific to the company and their associations, whether they are represented in a computer system or not. The global schema may be used as a referential, both terminological and semantic, for the company, from which sub-schemas can be derived in order to implement different IS. However, the cost for building a global schema is high. Moreover, the existence of such a referential within a company does not ensure it is a referential for the external actors with which it must communicate. The solution to this problem is that all the actors agree upon a common referential, independent from a particular company. This is actually the role of a domain ontology.

A domain ontology (in short an ontology, in the context of this paper) expresses an agreement of the actors of a domain upon the concepts of a domain and their interrelations [11]. It can play the role of a semantic pivot for the different IS of a company and can also be used as a semantic referential by the companies working in the same domain, thus favoring a non ambiguous communication inside and outside a company. However, as an ontology has a universal purpose it tends to have a bigger size than a class diagram and to grow rapidly, especially when it becomes a standard in a given domain [2]. Thus, when using an ontology as a basis for IS design, one is faced with the question of determining the subset of the ontology which is concerned with the concrete IS. Another question concerns the artifacts (diagrams and software) that may be (semi-)automatically derived from an ontology.

Regarding the first question, as many of the concepts of a domain ontology will be of no use for a given application, one has to select those of interest for this application and enrich them if necessary. However, mastering the hundreds or thousands of concepts of an ontology is beyond the capacity of a human IS designer. Thus, extracting the sub-ontology that is pertinent in a given situation has become a problem in itself. This problem has been dealt with not only in IS design [9][20], but also in different contexts such as Information Retrieval [3], computational biology [2], building ontologies from texts [10] and ontology evolution [13].

In the area of IS design, most authors have used ontologies to favor the reuse of the knowledge contained in the ontology. Reusing this knowledge reduces the time needed by the analyst to assimilate the knowledge of the studied domain and enables him to define a conceptual data schema more quickly. According to some authors, the conceptual schema produced this way is more understandable and more consistent than a schema produced without the support of an ontology [20]. However, the contribution of ontologies is most often limited to the (semi-automatic) design of the sole conceptual data schema, all others diagrams and software artifacts being produced manually.

In this paper we propose an approach that enforces the use of a domain ontology in IS design. This approach is implemented by the ISIS system, a platform for the

specification of Information System that aims at implementing the equation:

Ontology + User Requirements = Information System

In practice, given an ontology and user requirements represented by the use cases of the application, we first propose to enrich the ontology with information derived from the use cases. Then, the extraction of the useful part of the initial ontology by "hiding" the concepts and relationships that are not pertinent for the business processes is realized. The concepts that must be represented by objects, literals and index at the implementation level [6][18] are then deduced. Finally, a database optimized for the user requirements, the SQL code for the queries corresponding to the use cases, and a prototype GUI (Graphical User Interface) are automatically generated.

The paper is organized as follows. We first present some works on sub-ontology extraction, then the ISIS model. Finally, we present our approach to enrich the sub-ontology based on the user requirements.

II. From Ontological to Implementation Level

A domain ontology aims at universality and application independence [8]. Hence, when using an ontology in an application, it is necessary to first determine the sub-ontology needed for this application. This process, named sub-ontology extraction, has as main objective to obtain a consistent sub-ontology, i.e., an "ontology" that is valid and autonomous [10]. This process can be decomposed into three steps: identifying a family of relevant concepts, labeling them and applying algorithms to produce a consistent sub-ontology. Such a process has been dealt with in various domains [16]: ontology design from texts [10], Information Retrieval [2][3], visualization adapted to a particular category of end-users of large ontologies [10]. For example, Navigli et al. use statistic methods during ontology design to label redundant and much specific concepts, so that they can be eliminated by pruning algorithms to produce the right ontology [13]. In the OntoMove system, proposed by Bhatt et al. [3], a sub-ontology is defined for every category of users in order to reduce the search space and consequently reduce the query response time. For each category of users there is a manual phase to annotate each concept of the initial ontology as *selected*, *deselected* and *void*. The result of this phase, the annotated ontology, is considered by the authors as the user requirements for the considered category of users and the extracted sub-ontology as a persistent view of the initial ontology.

The use of domain ontologies for database design is now considered as a way to reuse the knowledge about a domain, which is made explicit in the ontology. OMMDE [20] and SISRO [9] are two systems which use ontologies to support the design of a database conceptual schema.

OMDDE uses a domain ontology enriched with certain particular relationships (pre-condition, mutually inclusive, mutually exclusive) that are used to help the design of the conceptual schema of a database. Thus, considering a fragment of an existing E-R schema, OMDDE checks its validity and proposes an extended schema according to the ontology enriched with the particular relationships. For example, it compares the names of the entities with the names of the concepts or their synonyms and proposes to use concept names whenever possible; it checks that pre-conditions and mutually inclusive concepts have actually been used and if not it proposes their insertion …

In the SISRO approach the designer may work with one or several domain ontologies from which he selects the classes he wants to be present in his final class diagram. For each selected class, SISRO builds a local class that is related to the ontology class through an ISA relationship. When designing a local class, the designer can add new attributes or suppress attributes inherited from the ontology. Thus, the classes of the domain ontology are used as patterns for the design of the classes of the local schema.

However, in IS design, the sole E-R schema or the class diagram alone are not sufficient to ensure the production of an IS compliant with the needs of the end-users: two end-users can validate the same diagram while expecting different final IS, as their knowledge is insufficient to validate anything other than the terms used in the diagram. Moreover, they interpret these terms in their own culture, and two users validating the same diagram may actually expect different systems. In order to ensure an IS compliant with the actual needs of the company, the functionalities of the IS and the dynamics of the objects must also be validated by the end-users, which necessitates a representation they can understand.

With the ISIS project we wanted to propose an approach to enrich an ontology with knowledge derived from the user requirements, in order that both the extraction of the sub-ontology needed by the IS and that the transformation of this sub-ontology into an operational IS with a prototype interface be automatic. An important objective of our approach was to allow the user to validate the user requirements and refine them if necessary, in a way that several <*specification-implementation-validation*> cycles can be performed without increasing the global cost of the project. Thus, the final IS can be closer of the real needs of the company. As this methodology also supports the modification of the user requirements during the design step, it can take into account the evolution strategy of the company itself.

Another objective of the ISIS project is to favor the collaboration between analysts and end-users. Thus we chose to limit the number of models, and consequently the number of meta-concepts an end-user has to master. In particular, we chose to favor the enrichment of an OD with behavioral properties rather than producing several dynamic diagrams and verifying the global consistency of all the produced diagrams.

The ISIS data model is a binary relational model [1][14], which is simultaneously simple, formal and powerful. Its main meta-concepts are *concept*, *binary relationship* and *ISA* relation, which may be readily mastered by the end-users. Consequently, it favors a better collaboration between end-users and analysts, which facilitates the acceptation of the final IS [5][7]. In this model, an ontology is represented by a graph which we call an Ontological Diagram (OD).

To automatically produce an operational system from the conceptual level, one must know how to transform a notion at the conceptual level into concepts at the physical level, e.g.,

class, *index* and *attribute* in the case of an object model. In the transformation from the conceptual to the physical level, a *concept* is transformed into a *class* and a *binary relationship* is transformed into an *attribute* of the class representing its concept domain. Two kinds of classes are possible: *object class* and *value class*. Formally, a class is an *object class* if and only if there is at least one of its instances whose value changes over time [6]; otherwise, the class is a *value class*. Thus, the automatic production of an operational IS requires the automatic determination of the mutability of class instances.

The two main behavioral properties we have identified to support the transformation from the ontological level into the implementation level, are the *criticity* and the *modifiability* of the relations. The criticity of a relation expresses that this relation is necessary for at least one of the use cases modeling the user requirements. The modifiability of a relation with domain A and range B expresses that, in the context of an IS, there exists at least one instance of A where its image in B changes over time. However, deciding on the criticity or the modifiability of the relations is outside the capabilities of end-users, whose knowledge is directly related to the way they achieve their business tasks, particularly the data and the business rules they use to perform them. This data, made explicit in the ISIS methodology through the *input* and *output* parameters of each use case, enables the system to infer which relations are critical and/or modifiable. We can then deduce the concepts that can be omitted and the best physical implementation for the remaining ones.

III. THE ISIS SYSTEM

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, from a Domain Ontology and a set of user requirements. Our strategy is to enforce the use of a unique diagram to represent the static properties of the entities of the domain as well as their dynamic (behavioral) properties.

A. The ISIS Use Case Model

The ISIS use case model allows the representation of user requirements, expressed in natural language, by a set of use cases. Each use case is represented by a collection of queries, where each query is represented by a tuple <*name, type of query, input concepts, output concepts, relations, business rules*>. For a query, only its type (selection, creation, modification, suppression) and its input concepts are mandatory.

We have distinguished two kinds of use cases: those whose all queries are selection queries (e.g., *for a patient, date of its consultations and name of the doctor*), and those in which at least a query is an update query (e.g., *create a new consultation*). We call the former Sel-UC (for Selection use case) and the latter Up-UC (for Update use case). For example, the above Sel-UC is interpreted as <selection, **in**: {patient}, **out**: {consultation date, name of doctor}>. The set of Sel-UC enables ISIS to determine the subgraph of the OD that is actually needed to produce the IS. The set of Up-UC enables ISIS to determine which relations are mutable and consequently which concepts of the ontology have mutable instances. Thus, for each concept of the OD, ISIS can propose the best physical representation according to the users requirements.

B. The ISIS Model

The concepts of a given domain and their relationships are represented through an OD 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 an OD, *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. As in other models, the static constraints govern the production of the logical database schema. Behavioral constraints are necessary to automatically produce the physical database schema and the associated software. The main behavioral properties we consider are the **criticity** and the **modifiability** of a relation for a given application. Critical relations are the relations needed in at most one query of Sel-UC. A modifiable relation is a non-monotonous relation. It is deduced from the update queries of Up-UC.

Critical Relations. The ultimate purpose of an IS is expressed through Sel-UC, hence our choice of the queries of Sel-UC to decide which relations are critical. Our main criterion in selecting the critical relations is to consider the relations participating in at least one selection query of Sel-UC (critical query). However, the designer can decide to make any relation critical, independently from critical queries.

The update queries of Up-UC are 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 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 OD is critical iff at least one of its relations is critical.
- Given a concept CC, domain of the critical relations r_1, r_2,…, r_n, and C_1, C_2, …, C_n the range concepts of r_1, r_2,…, r_n. The *value* of an instance $cc_k \in CC$ is an element of the Cartesian product $C'_1 \times C'_2 \times \ldots \times C'_n$, where $C'_i = C_i$ if r_i is monovalued (max. card. = 1) and $C'_i = \mathcal{P}(C_i)$ [1] if r_i is multivalued (max. card. ≥ 1).

Modifiable Relations and Modifiable Concepts

Definition: Given a relation R(A, B), R is modifiable iff there exists a ∈ 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[2], to a model with three types of nodes: *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 secondary concept corresponds to concepts whose instances are « structured ». We name *valC* the relation whose domain is a primary concept C and whose range is a predefined concept (e.g., valAge, valName).

In Fig. 1, three categories of nodes are represented: predefined concepts (integer, string), primary concepts (name, age) and secondary concepts (person). Primary and secondary concepts are concepts built for an application.

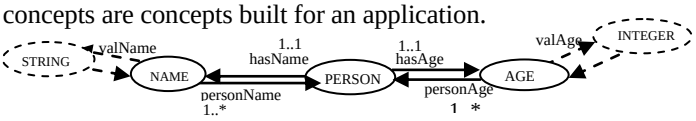

Fig. 1. Built and predefined concepts in an OD.

Distinguishing these three categories of concepts in the model is necessary to support the automatic production of an IS. However, in the ISIS tool predefined concepts are hidden in order to make the presentation of an OD simpler (see Fig. 2).

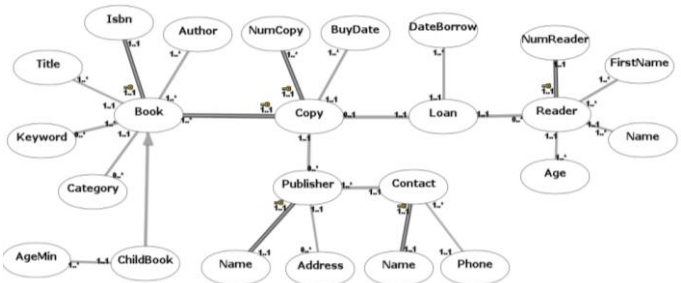

Fig. 2. Partial ISIS diagram of a library management application.

Fig. 3 illustrates a data schema represented with a classical binary relational model, which has only two categories of nodes. This schema, extracted from [1], represents the Z_0 schema of a set person with two access functions[3], *ageOf* and *nameOf*, where the notion of access function is derived from that of relation. This representation induces a representation of *ageOf* and *nameOf* as attributes of a class/entity (in the object/E-R model) or of a table (in the relational model) person.

Fig. 3. Z_0 schema modeling persons with name and age [1].

Thanks to the behavioral properties added to the diagram, ISIS will propose that a concept such as age, name or person be represented as an object or as a literal, according to the ODMG classification [6], and among the objects propose candidates to become database indexes.

Let us consider the concept person represented in Fig. 1 and Fig. 3, and 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 [15]. In common design situations, representing age by a literal or by an object is (manually) decided by the designer; as it is atomic, it is generally considered as a literal. However, 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 for age, 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. 3 where the only relation 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, while the best modeling for Europe is to consider *ageOf* as the modifiable relation. Distinguishing primary concepts and predefined concepts is necessary to differentiate these two ways of modeling the update of the age of a person. The same reasoning applies to other primary concepts such as salary: either change the salary of one person (who has been promoted individually) or change the salary of all the persons belonging to a given category.

Definition: a concept is said to be a *concept with instances with modifiable values*, or simply *modifiable concept*[4], iff it is the domain of at least a binary relation that is both *critical* and *modifiable*.

Considering the example of Fig. 1, person is a modifiable concept for European countries, whereas in the Korean context age is a modifiable concept

[1] $\mathcal{P}(E)$ represents the set of parts of E.

[2] E.g., in Z_0, concrete (structured) and abstract (atomic) sets; lexical and not-lexical in Niam.

[3] « An access function maps one category into the powerset of another (the set of all subsets) ».

[4] Note that it is not the concept itself that is mutable, but its instances.

C. *Enriching an OD with Behavioral Properties*

Let us consider the following use cases in a library management application:

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.

These user requirements are rewritten in the following ISIS UC. The first four UC are Sel-UC and the last five are Up-UC. For each query of a UC the designer identifies the concepts it concerns and annotates them as **input or output** 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}, **out** {book}); (**in** {book}, **rel** {*keywordOf*, *categoryOf*})}.

From Sel-UC to Critical Relations. The first step in the identification of the concepts of an OD that must belong to the sub-ontology is the identification of the *input* and *output* concepts of the queries of Sel-UC. All the relations of the subgraph of a critical query are marked as critical. For example, to enter UC4 in the ISIS tool, the designer annotates the input concepts (downward arrow) and the output concepts (upward arrow). The ISIS system calculates the paths between the input and the output concepts and marks the intermediate concepts with flags (Fig. 4).

The subgraph of this UC is made of three paths: (**isbn**, book, copy, **numCopy**), (**isbn**, book, copy, loan, reader, **name**) and (**isbn**, book, copy, loan, reader, **firstName**).

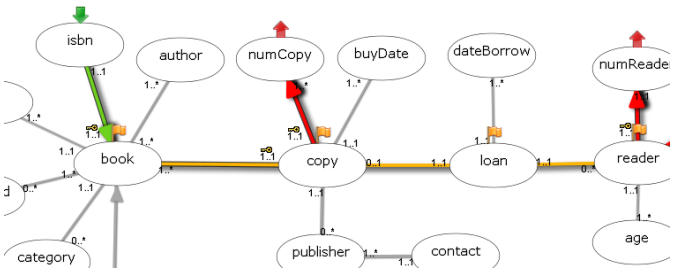

Fig. 4. Part of the subgraph of UC4 (*readers of a book*).

In this example, the relations (isbn, book), (book, copy), (copy, numCopy) etc. are critical. When several paths are possible between an origin and an end concept, the designer must choose intermediate concepts (by annotating them with a flag) in order to ensure the semantics of the query. When several candidate relations have the same pair of domain and range concepts, the user can select one of them.

Definition: the sub-ontology needed for the IS is represented by the smaller subgraph containing the subgraphs of all the Sel-UC. It contains all the binary relations (and their corresponding domain and range concepts) marked as critical.

When none of the relations of an association is marked as critical, two solutions are proposed by ISIS: remove the association or make critical one of its relations. Removing an association usually leads to removing concepts from the OD. For example, if the query presented in Fig. 4 is the **only** selection query of the IS, the relations of the associations *book-category* and *book-keywords* are deduced as not critical. Thus, the designer must decide either to suppress the association or to make one of its relations critical. When a concept becomes disconnected from the others concepts of the OD, it is suppressed. This constitutes the first phase of the *simplification* process.

From Up-UC to Modifiable Relations. The second step in the ISIS methodology is the enrichment of the OD with the Up-UC. Again, the user must identify the concepts of the update queries of each Up-UC, and, for modifiable queries, the relations that must be changed. ISIS deduces which relations are modifiable. For example, UC6 (*new copy*) enables ISIS to deduce that the relation *copyOfbook* [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. In the library example, based on UC1-UC9, the ISIS proposals are:

Object concepts: book, copy, loan, reader, keyword, category[6].

Potential indexes: isbn, numCopy, numReader.

Value concepts: title, author, dateBorrow, firstName, Name.

Generation of Software Artifacts. In the last step ISIS generates the application, i.e., the database, the code of the queries of the use cases and a prototype GUI. Fig. 5 shows the GUI corresponding to UC2 (*loans of a reader*) in the PHP-MySQL application that is automatically generated.

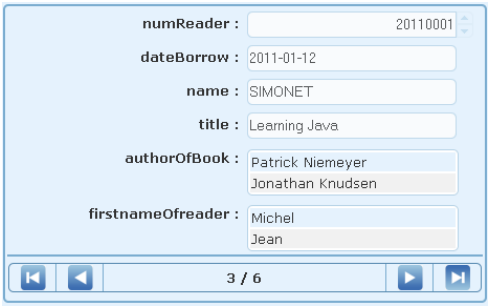

Fig. 5. Screen copy of the window generated for UC2

The generated GUI has Spartan ergonomics: first the monovalued attributes are presented in alphabetical order, then

[5] Relation with domain *book* and codomain (range) *copy*.

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

the multivalued attributes. In spite of these basic ergonomics, it enables the users to verify the presented items and their type. They can also check whether the dynamics of the windows corresponds to the needs of their business process

Import of a domain ontology. When a domain ontology is imported into ISIS, its concepts are rewritten in terms of the ISIS meta-concepts. Thus, for a class C we create a secondary concept C and as many primary concepts as C has attributes. For example, the OD representing a class PERSON with two attributes *hasName* and *hasAge* is defined by a secondary concept PERSON and two primary concepts NAME and AGE, created to represent the domains of these attributes (cf. Fig.1). Each primary concept is linked to a predefined concept.

IV. CONCLUSION AND PERSPECTIVES

The ISIS project has been designed to support the reuse of the knowledge expressed in a domain ontology and favor an active collaboration between end-users and analysts, in order to produce a system compliant with the real needs of a company. We have chosen a binary relational model, with a limited number of meta-concepts, thus making it easier to apprehend by end-users

To enable the reuse of an existing domain ontology we propose an approach that takes an ontology, transforms it into the ISIS internal format and enriches it with knowledge implied by user requirements. From the input and output concepts of the use cases, ISIS can identify the critical and the modifiable relations. In a first simplification phase it proposes to suppress the binary associations where both constituent relations are non-critical, and then the isolated concepts. The resulting concepts and relations constitute the sub-ontology of the IS. Some of these concepts can still bear non-pertinent information for the business processes and can be eliminated in a second simplification phase, which is not presented in this paper.

After its enrichment with behavioral properties and the sub-ontology deduction, model transformations enable the automatic generation of the database, the API and a prototype GUI of the IS. These software artifacts enable the end-users to verify the adequacy of the IS to their needs and refine them if necessary.

The ISIS tool has been developed in Java with a dynamic web interface; the OD are stored as XML files. It currently produces a PHP-MySQL application. Future work encompasses the establishment of two-way relationships between ISIS diagrams and UML diagrams, and the use of the ISIS methodology for the integration of heterogeneous databases.

REFERENCES

[1] J.R. Abrial, Data Semantics, in J.W. Klumbie and K.I. Koffeman (Eds), Database Management, North-Holland, Amsterdam, 1-59, 1974.

[2] S.Bauer, S. Grossmann, M. Vingron & P. N. Robinson, Ontologizer 2.0: a multifunctional tool for GO term enrichment analysis and data exploration, Bioinformatics, 24(14), 1650-1651, 2008.

[3] M. Bhatt, A. Flahive, C. Wouters, W. Rahayu , D. Taniar & T. Dillon, A distributed approach to sub-ontology extraction, 18th International Conference on Advanced Information Networking and Applications, Vol 1, pp 636-641, 2004.

[4] A.Burton-Jones, P. Meso, Conceptualizing Systems for Understanding: An Empirical Test of Decomposition Principles in Object-Oriented Analysis, Information Systems Research, vol 17, No1, pp 38-60, 2006.

[5] B. Butler, A. Fitzgerald, A case study of user participation in information systems development process, 8th Int Conf on Information Systems, pp 411-426, Atlanta, 1997.

[6] R.G.G. Cattell, T. Atwood, J. Duhl, G. Ferran, M. Loomis, D. Wade, Object Database Standard: ODMG-93, Morgan Kaufmann Publishers, 1994.

[7] A. Cavaye, User Participation in System Development Revisited, Information and Management (28) pp 311-323, 1995.

[8] T. Dillon, E. Chang, M. Hadzic, P. Wongthongtham, Differentiating Conceptual Modelling from Data Modelling, Knowledge Modelling and Ontology Modelling and a Notation for Ontology Modelling, Proc 5th Asia-Pascific Conf on Conceptual Modelling, 2008.

[9] Ch. Fankam, L. Bellatreche, H. Dehainsala, Y. Ait Ameur, G. Pierra, SISRO : Conception de bases de sonnées à partir d'ontologies de domaine, Revue TSI, vol.28 pp1-29, 2009.

[10] A. Flahive, W. Rahayu, D. Taniar , B. O. Apduhan , C. Wouters & T. Dillon, A service oriented architecture for extracting and extending sub-ontologies in the semantic grid, 21st International Conference on Advanced Networking and Applications, pp 831-838 , 2007.

[11] N.Guarino, 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.

[12] D. Moody , A.R. Kortink, From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design, Proc. Int. workshop on Design and Management of Data Warehouses (DMDW'2000), 2000.

[13] R. Navigli, Automatically extending, pruning and trimming general purpose ontologies, IEEE International Conference on Systems, Man and Cybernetics, Tunisy, 2002.

[14] G.M. Nijssen, Current Issues in Conceptual Schema Concepts. In G.M. Nijssen (Ed), Architecture and Models in Data Base Management Systems, North-Holland, Amsterdam, pp 31-65, 1977.

[15] J. Park, S. Ram, Information Systems: What Lies Beneath, ACM Transactions on Information Systems, Vol. 22, No. 4 pp 595–632, 2004.

[16] V. Ranwez, S. Ranwez, S. Janaqi, Extraction de sous-ontologies autonomes par fermeture des opérateurs hyponymie et hyperonymie. Journées francophones sur les ontologies JFO 2009, ACM 978-1-60558-842-1, pp. 45-55, 2009.

[17] C. Rolland, From Conceptual Modelling to Requirements Engineering. ER'2006, LNCS 4215, pp5-11, 2006.

[18] A. Simonet, Automatic Production of an Operational Information Ssytem from a Domain Ontology Enriched with Behavioral Properties, 1st Int. Conf. on Model & Data Engineering (MEDI'2011), LNCS 6918 pp 4-17, 2011.

[19] P. Spyns, R. Meersman., M. Jarrar, Data modeling versus Ontology engineering. SIGMOD Record, vol 31, no 4, pp 12-17, 2002.

[20] V. Sugumaran, V. C. Storey, 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.