Requirements Ontology and Multi-representation Strategy for Database Schema Evolution[1]

Hassina Bounif, Stefano Spaccapietra, Rachel Pottinger

Database Laboratory, EPFL, School of Computer and Communication Science
Lausanne, Switzerland
Data Management and Mining Laboratory, University of British Columbia
Vancouver, Canada

hassina.bounif@epfl.ch, stefano.spaccapietra@epfl.ch, rap@cs.ubc.ca

Abstract. With the emergence of enterprise-wide information systems, ontologies have become by definition a valuable aid for efficient database schemas modeling and integration besides their disseminated use in other important disciplines such as semantic web and natural language processing. This paper presents another important utilization field of ontology related to database schemas and which is schema evolution topic. More specifically, our research work concentrates on a new three-layered approach for schema evolution based on domain ontology that we have called a requirements ontology and multi-strategies to a powerful change management and cost-effective evolution. This a priori approach for schema evolution, in contrast with existing a posteriori solutions, can be employed for any data model and for both 1) design from scratch and evolution and 2) redesign and evolution. The paper focuses on the two main foundations of this approach which are the requirements ontology and the multi-representation strategy based on stamping mechanism.

Keywords: Ontological Engineering, multi-representation strategy, Schema Evolution

1 Introduction

With the emergence of enterprise-wide information systems, the number of ontologies in semantic-driven data access and processing is increasing. It is the case of the use of ontologies in semantic web and natural language processing. In addition to that, ontologies have become by definition a valuable aid for efficient database schemas modeling and integration. In this work, we have investigated another area in which ontologies have a colossal potential of utilization and which is related to information systems. That is database schema evolution, an important, complex and very active research issue. Several solutions have been proposed and much progress has been made in data structures, rules, constraints, schemata models and meta-models. In this

[1] This work is carried out as part of IM2 (Interactive Multimodal Information Management) (http://www.im2.ch), Swiss National Competence Centre in Research (NCCR), supported by the Swiss National Research Fund

context, we advocate a new approach for schema evolution in which potential changes are inspected and integrated into the schema for future use. This approach is based on prediction. Generally, prediction means to envision with exactitude or not the changes that could occur over time. However, in this research work, we intend to detect the changes that are plausible to carry out on a database schema and are important for the database users. Our intent is to move one step towards, developing multi-disciplinary and a priori approaches for database schema evolution, in contrast with existing a posteriori solutions that track the change instead of planning it.
Our approach relies on the use of both 1) requirements ontology that contains the changes that are plausible to carry out on a database schema and are important for the database users and 2) multi-strategies to a powerful Change management and compatibility with any conventional data model. We have chosen to focus on one of the multiple strategies we have defined, i.e. the multi-representation strategy based on stamping mechanisms. The objectives of this section is to explain the problem of schema evolution as well as the principles and the characteristics of a new approach that we propose to resolve it

1.1 Problem Description

Intuitively, schema evolution means the ability of a schema to undergo changes over time without any loss of the extant data. However, besides managing the changes to the schema, applications and data linked to it need to be adapted as well. Changes to the schema are divided into three categories depending on their consequences on it[1]:
1- Additive : extra semantic knowledge is modelled
2- Subtractive: less semantic knowledge is modelled
3-Descriptive: the same semantic knowledge is modelled in a different manner

To better understand the general problem of database schema evolution, we need to consider it from two different sides, depending on the kind of solution we choose: 1) From a posteriori solution side, 2) From a priori solution side.

1) From a posteriori solution side:
Historically, from this side, to resolve the schema evolution problem, one should take into consideration two major criteria, which are respectively [2]: a) the semantic of change, i.e. the understanding of the change that has taken place because of several reasons such as the new perceptions of the real world over time and technology development and performance strategies and b) the propagation of this change on the schema immediately or at deferred time fixed by the database administrator. There is a posterior order in which the change must be received after by the schema and its components. Schema evolution is resolved either by versioning the original schema, by modifying it using restricted evolution primitives, by adopting views on the top of it or by refining it by accommodating the exceptional information in the database [3]. All these solutions react to changes that could occur on the schema. However, they are insufficient solutions, especially when the schema is facing complex changes. For instance, in the modification approach, changing the schema may lead to a loss of information. Whereas in the versioning approach, replication of the schema avoids

data loss; it however creates complex navigation through the different generated versions and slows down the DBMS (Database Management System). As for the combining solution, i.e. a solution that includes two existing approaches, for example the work presented in [4], it allows to avoid the above mentioned problems. It is at the same time, characterized by the complexity and the onerous mechanisms to be executed.

2) From a priori solution side:
To resolve the problem of schema evolution from this perspective, one must clearly take into account these imperative criteria:

- Understand the current database structure and content
- Identify the dependencies among the current database schema, data and applications because the impact of one element on another needs to be known and accounted for before making changes on the database
- Detect potential changes that are plausible to occur on the schema
- Understand the potential future changes and new applications and identify their impacts on the current schema
- Consider the two possible situations, related to the database, that are respectively: 1) the situation in which the initial database schema is not created yet and 2) the situation in which the initial schema has already been created; however, it needs to be redesigned.

Compared to the previous side, here, the order of applicability of the changes has been modified. The changes are incorporated before they really occur. There is what is called an a priori order in which the potential change must be received before by the schema and its components.

1.2 Predictive Approach for Schema Evolution

This new a priori approach for schema evolution is characterized by the following:

- A priori Approach
- Dual Applicability for 1) design from scratch and evolution and 2) re-design and evolution
- Requirements analysis and their categorization into two categories: current and future
- Multiple strategies for data management and compatibility with any data
- Evolution cost evaluation

a) A priori Approach
This approach, in contrast with existing posterior solutions for evolution such as modification or versioning approaches, tackles the problem of evolution before the changes occurred effectively on the database schema. It tends to react preventively by incorporating the changes in the schemas for future use. However, it can also be applied when the changes need to be materialized on the schema.

b) Dual Applicability

This approach takes into consideration two situations of the database: 1) the situation in which the schema is designed from scratch and 2) the situation in which the schema is redesigned. Therefore, the proposed approach can be applied for both:

- Design and evolution of the schema
- Re-design and evolution of the schema

c) Requirements Analysis and Categorization

The predictive approach does not work in the same way as existing a posteriori solutions for evolution. As a matter of fact it explicitly includes a requirement analysis phase in which, besides the current user requirements that are performed with the help of database user's feedbacks and comments, additional requirements called potential future requirements are investigated. These new requirements, representing potential future needs that might emerge during the lifecycle of the database are inspected inside a schema repository. We propose to model all these requirements with a structured method based on domain ontology called a requirements ontology. This method is complex and includes the use of data mining techniques and the schema repository. We are not interested in the requirements process and their associated specification templates that are applied usually in the construction of a new system. However, we require that the resulting output of the requirements modeling be defined in terms of requirement ontology.

d) Multiple Strategies to Change Management and Compatibility with any Conventional Data Model

This approach for evolution is based on three modeling levels: the ontological level, the conceptual level and finally the logical level.

- At the ontological level, requirements ontology is used. This is illustrated in figure 1. A detailed description of the role, the structure and the construction of this ontology is presented later in this paper.

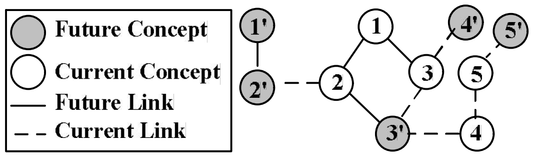

Fig. 1. The requirements ontology with the two kinds of concepts and relationships (links)

- At the conceptual level, a part or selected parts of the requirements ontology representing the potential future requirements are included into the conceptual schema of the database to be designed or redesigned using one of the strategies in figure 2 presented below.

1- Direct conversion strategy: in this strategy, the selected part(s) of the requirements ontology are converted to a conceptual schema with a direct mapping.

2- Multiple representations based on view or stamping mechanisms

3-Combined approach (Direct and Multi-representation strategies)

The formalism presented in the figure 1 is used in the Motivating Examples section to show how the database designer takes into account the ontological level represented by the requirements ontology for an anticipated schema evolution.

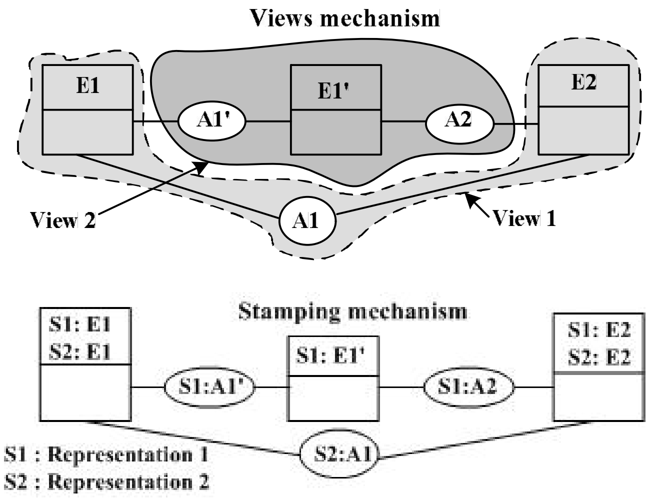

Fig. 2. The multi-representation strategy with views and stamping mechanism at the conceptual layer

- At the logical level, the conceptual schema is converted as it is into a logical schema. However, the tables and attributes that correspond to future requirements are not functional yet. They are included in the schema but cannot be acceded to by the DBMS. However, in the evolution phase, i.e. when the database schema needs to be changed, these accompanying structures are accepted or rejected based on their relevance with the future state.

e) Evolution Cost Evaluation
Another important objective of this approach is to estimate the cost related to the planning of the evolution using requirements ontology. We have defined a simple cost model based on two factors: the importance of a concept, respectively a relationship in the requirements ontology and their tendencies to the changes. The details of this cost model are not developed in this paper due to the lack of space
The advantages of this approach are various. Indeed, it reinforces the conceptual schemas and makes them ready for evolution, it is compatible with any data model and finally, it facilitates the work of database designers and helps them save time and fund on the evolution of their databases.

1.3 Contribution of the Paper

The contributions of this paper are as follows:

1. Presentation of the predictive approach for database schema evolution, including the characteristics and the differences with other existing approaches for schema evolution
2. Presentation of the requirements ontology, including its role, construction and structure
3. Presentation of the multi-representation strategy with the two defined mechanisms views and stamping
4. Presentation of examples showing how the predictive approach works and outlining the role of both the requirements ontology and the multi-representation strategy.

1.4 Outline of the Rest of the Paper

This paper explains the articulation of this new approach for database schema evolution and outlines the use of domain ontology and a multi-representation strategy. The paper comprises five main sections. Section 2 presents 1) the role of the requirements ontology in the predictive approach for evolution and 2) the structure and how it is built from the schema repository. Section 3 describes the multi-representation strategy and how it is used for schema evolution. Section 4 presents some motivating examples to demonstrate the feasibility of the proposed approach as well as the two categories of database designers that can use it. Section 5 is a conclusion and a summary of the important points dealt with in this paper and introduces perspectives on the future work.

2 Requirements Ontology for Schema Evolution

The requirements ontology is a domain ontology in which requirements are expressed with concepts (terms), relationships and constraints. For example, if a database designer needs to create a database for meetings, the requirements ontology associated to this database contains concepts, relationships and constraints related to meeting domain such as *MEETING*, *PARTICIPANT*, *ROOM* and *AGENDA* concepts and *IS*, *HAS* relationship types and so on. This is illustrated in figure 3.

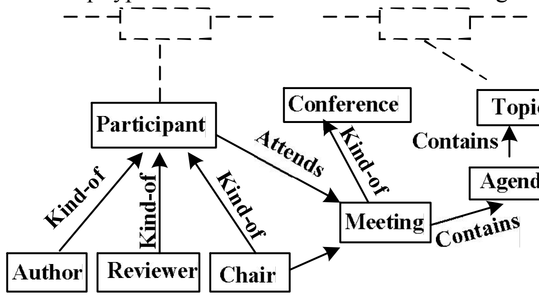

Fig. 3. Presentation of a part of requirements ontology of meeting domain

The requirements ontology looks like a global entity relationship model however it is not an entity relationship model because it contains more semantics related to a specific domain.

2.1 Requirements Ontology Role

There are two ways in which this ontology is used in the predictive approach for evolution. These two primary functions are 1) design and evolution and 2) redesign and evolution.
1) Design and evolution
In case the initial database schema is not created yet, the ontology fulfils several tasks, as presented in [5] and [6]: it generates a design "from" scratch using the defined terms and relationships as a representative model of the domain. It suggests possible missing entities and relationships in the case just a part or selected parts of it is/are considered by the database designer. In addition, the inclusion of synonyms helps to identify the most appropriate label for an entity or a relationship. The requirements ontology offers additional features; it includes terms, relationships and constraints that might represent potential future requirements and identifies in advance their dependencies with terms and relationships representing current requirements. Consequently, it facilitates the work of database designer when changes should be made on the schema.

2) Redesign and evolution
In case the initial schema has already been created and however needs to be redesigned, then the ontology fulfils other important tasks as presented in [5] and [6]. For example, it is used to check for missing entities or relationships or inconsistencies in an existing or partial design because the data model produced in the redesign process, called a reverse engineered (RE) data model [7], cannot be considered as a conceptual schema. The RE model converts all the logical schema tables to entities without making distinction between data tables and the other tables of the schema whose function is to join tables. This model is not a logical schema because some important schema information is lost during the conversion process, such as in the case of foreign keys. Besides all these tasks, the requirements ontology allows to understand the entities that have undergone changes and to identify their dependencies with existing entities that have not changed in the schema.

2.2 Requirements Ontology Construction

The requirements ontology is developed using both a schema repository in which the main concepts are extracted and WordNet ontology [8] for extracting their corresponding synonyms and antonyms. The requirements ontology consists of two kinds of partitions, the ones representing current requirements called Current sub-domains and the ones corresponding to potential future requirements called Future sub-domains. This is illustrated in figure 4.

The process of the requirements ontology creation is iterative and complex some how compared to existing approaches. It consists on four main phases: *knowledge acquisition*, *Data mining and informal conceptualisation*, *Evaluation for Refinement or Revision* and *Formal Conceptualization*

1 -Knowledge acquisition and pre-processing: consists of schemas collection and preparation

2 -Data mining algorithms and informal conceptualization: in which concepts and relationships are extracted from schema data sets repository in an unsupervised way and used as output for the informal conceptualization of the ontology from scratch.

3 -Evaluation for Refinement or Revision: means to test the validity of the concepts belonging to the taxonomy and to decide to keep or reject them using qualitative and quantitative methods.

4 -Formal Conceptualization: consists in building formally the requirements ontology using OWL and description logic.

These phases are not very developed in this paper because they necessitate a considerable space.

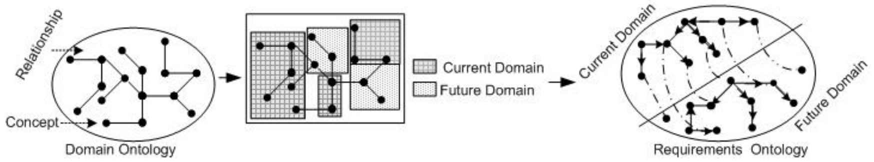

Fig. 4. The requirements ontology divided into two main domains: the current domain for the current requirements and the future domain for the future requirements

The schema repository contains many different schemas that model a specific domain. These schemas and their related versions may be of different types, such as ER, relational, object and object-relational schemas. The XML databases can be included as well as the ontological schemas expressed in OWL technology [9]. The schema repository has a dual role in building the requirements ontology [10]: (1) the repository serves in the data-mining process to identify and analyze trends on different kinds of schemas collected. (2) The repository contains selected concepts and relationships to be included in the requirements ontology.

2.3 Requirements Ontology Structure

The structure of this ontology includes: a) Concepts, b) Relationships, c) constraints d) Current versus Future Labels, described below:

a) Concepts (Terms) Description:

Each term has one or several attributes with one or several values and one or several synonyms and antonyms. We have chosen to define the terms of this ontology by using the analogy method which is a form of Case Based Reasoning (CBR) [11] with small modifications to adapt it to our case study. The analogy method is based on the use of historic data, i.e. past data and similar data to retrieve the information and knowledge in order to solve a problem. In our case, besides the terms that represent current requirements, we need to define the terms that represent potential future requirements. Therefore, terms are extracted from the schema repository as follows. either: 1) from schemas belonging to the same domain of the database to be modeled 2) from schemas belonging to a similar domain to the domain of the database to be modelled therefore it is referred as first domain in similarity 3) from schemas belonging to a similar domain of the similar domain of the domain of the database to be modelled. This domain is referred as second domain in similarity.

b) Relationships Description
Relations are between two concepts. There are six kinds of relations: hierarchic identified by the label "kind-of", which expresses the specialization of one concept regarding another and inherits attributes from this super concept; composition identified by the label "has", which expresses that a concept is a part of another concept; descriptive, when it is possible to define several types of relations and is identified by a verb form; reflexive allows self-loops in which an arc whose endpoints are the same concept and, finally synonym and antonym identified respectively by the labels "synon"and "antony". We consider the previous example of meeting domain to show some relationships among concepts: kind-of (meeting, conference) is a hierarchic relationship, has (meeting, utterances) is a composition relationship, lives (Person, Country), originates (Person, Country) and represents (Person, Country) are descriptive relationships and finally invites (Person, Person) is a reflexive relationship.

c) Constraints
Similar to the work presented in [5] and [6], we use four types of constraints which are respectively: 1) pre-requisite constraint, 2) mutually inclusive constraint, 3) mutual exclusive constraint and 4) temporal constraint.

d) Current versus future labels
The requirements ontology is a labeled graph: special labels are added and exploited in order to indicate whether a concept, respectively a relationship belongs to current or future requirements. A concept respectively, a relationship belongs to either current requirements or future requirements but not to both at the same time. This is main structure characteristic that distinguishes the requirements ontology from the remaining domain ontologies. This classification of concepts/relationships is needed in the evaluation of the evolution cost. The details are not presented of this work because of lack of space.

3 Multi-representation Strategy

The multi-representation strategy is well-known in the object-modeling field, as well as in the spatial databases. In [12], the multi-representation strategy based on stamping in geographic databases is presented. On the other hand, in the object modeling, the multi-representation is called semantic object views. It allows to make the object visible for certain applications and to hide it to others using the views mechanism. In this work, we focus on the multi-presentation based on stamping. This strategy consists in using stamps at the conceptual level in order to have different representations for the modeling of the same universe of discourse i.e. the modeling of the same real-world. A stamp S is defined as a vector S=<s1, s2, Sn> where each s¡ represents the ¡ representation of the real-world. For example, in the following simple example, we have defined a stamp S =<S1, S2> in which, according to the element S1 of the stamp S, the conceptual schema contains the entities E1, E1' and the relationships A1'. However, according to the element S2 of the stamp S, the conceptual schema contains the entities E1 and E2 and one relationship A1. This is illustrated in figure 5.

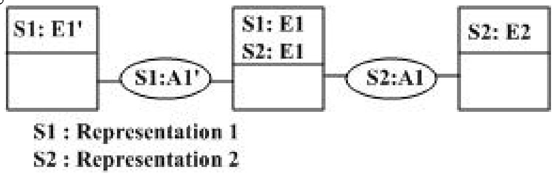

Fig.5. A Simple Example using Multi-Representation Based on Stamping

The stamping mechanism is not a simple mechanism as it may appear. For example, in the case of successive evolutions on the database schema, the stamp components and the constraints on the stamps should be studied carefully in order to avoid any potential contradiction among them.

4 Motivating Examples

In this section, we present examples of schema changes to illustrate how the predictive approach for evolution discussed in this paper works. The examples portray the three categories of changes presented previously, which are 1) the additive evolution, 2) the subtractive evolution and 3) the descriptive evolution.
Integrity constraints are not presented in the following examples because of lack of space; however, they play an important role in enhancing the use of the ontology and in avoiding redundancy and contradiction among concepts (entities) and relationships.

4.1 Additive Evolution

Adding an element on existing schema is not always obvious because there are two types of additive changes: simple and complex. For the simple additive change, the database administrator can use the functionalities of the DBMS (Database Management Systems) to add for example a table or an attribute. Whereas for the complex additive change, adding an element perturbs the dependency between existing elements and causes damaging effects on existing applications. Consequently, the logical schema is in inconsistent state and the associated applications do not work anymore. A case of complex additive change is illustrated in the side 1 of the figure 6 in which the addition of the entity E1 creates problems for existing applications. The way to resolve such a problem with the predictive consists in:

1 - At the ontological level: the database designer examines whether the requirements ontology reveals the existence of concepts/relationships that belong to the category of future requirements and represent potential simple and complex additive changes.
2- At the conceptual level, the database designer incorporates these concepts/ relationships using the multi- representation strategy based on stamping mechanism. The resulted conceptual schema represents consequently two universes of discourse (real-world). This is illustrated in the side 2 of the figure 6.

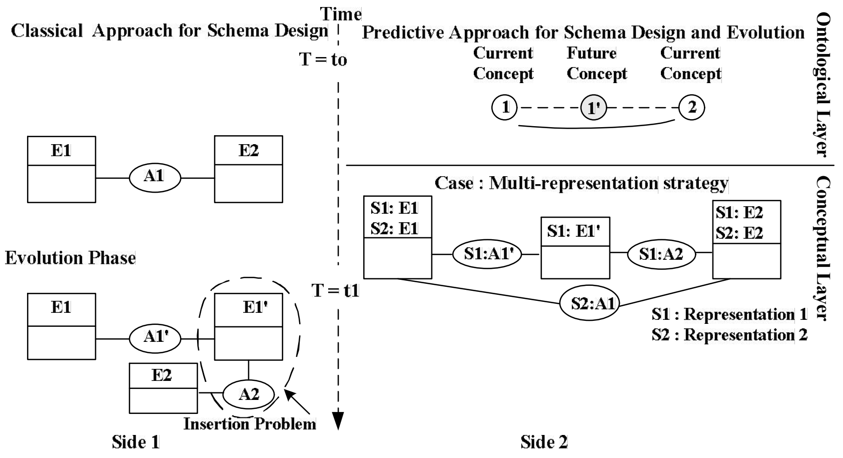

Fig.6. Presentation of additive Evolution on a Simple Example with both Classical and Predictive approaches

A case of complex and simple additive changes on the schema is illustrated in figure 7 where we consider the example that represents the modeling of a simple case of travel, in which the requirement is to determine the clients traveling around the world.

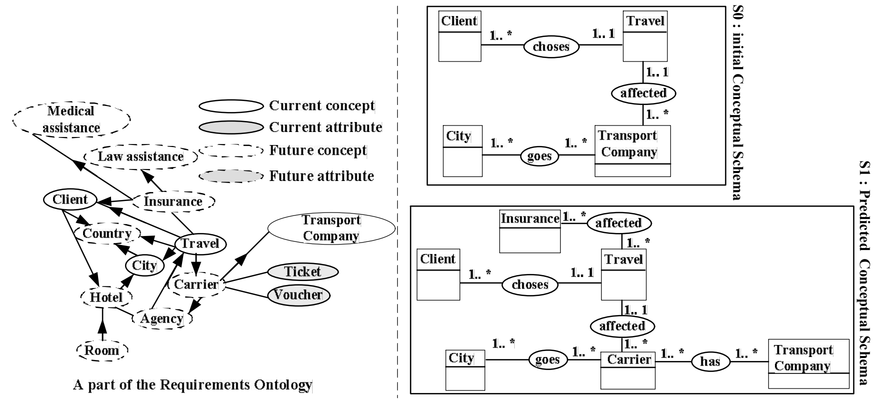

Fig.7. Modelling a simple example of travel with both classical and predictive approaches

In a classical design approach, the initial conceptual schema S0 contains four entities which are *Client*, *Travel*, *City* and *Transport Company*. However, in the predictive approach, the schema S1 that has been proposed by the requirements ontology contains six entities. The two additional entities *Insurance* and *Carrier* represent two potential future changes on the schema that belong to simple and complex additive changes respectively.

4.2 Subtractive Evolution

Subtractive evolution occurs when elements on the schema become negligible and are no longer required. However, deleting an element on existing schema is not always obvious because there are two types of subtractive changes: simple and complex. For the simple subtractive change, the database administrator can use the functionalities of the DBMS (Database Management Systems) to delete the non required elements. Whereas for the complex subtractive evolution, the DBMS does not offer any functions for it and the changes have direct and critical consequences on the schema and applications. A case of complex subtractive change is illustrated in the side 1 of the figure 8 in which the deletion of the entity E2 creates problems for the existing applications that need such entity. The way to resolve such a problem with the predictive consists in:
1 - At the ontological level: the database designer examines whether the requirements ontology reveals the existence of concepts/relationships that belong to the category of current requirements and represent potential simple and complex subtractive changes that need to be changed in the future.
2- At the conceptual level: the database designer incorporates these concepts/ relationships using the multi- representation strategy based on stamping mechanism. The resulted conceptual schema represents consequently two universes of discourse (real-world). The whole process is illustrated in the side 2 of the figure 8.

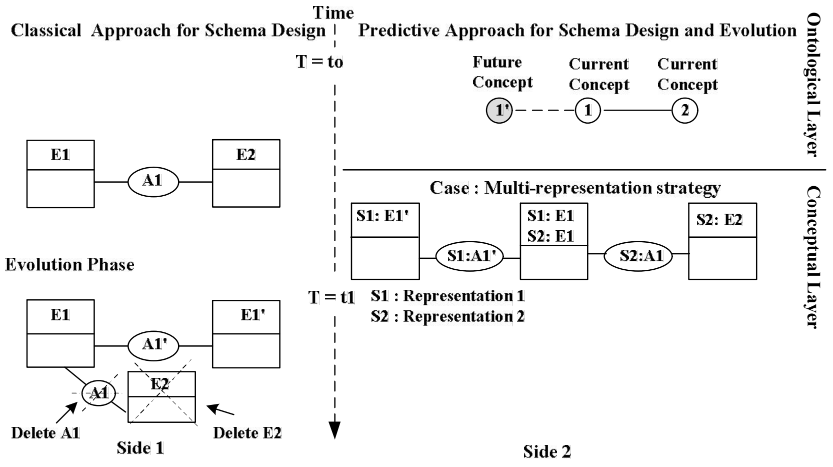

Fig.8. Presentation of subtractive evolution on a simple example with both Classical and Predictive approaches

4.3 Descriptive Evolution

Descriptive evolution is made for convenience or efficiency. It is the hardest to handle in traditional database systems because it implies more than one risky modification operation on the schema. The consequences of the changes on the schema are also critical, such as data loss. This is illustrated in the side 1 of the figure 9. Similarly to the previous, we follow the same steps for the resolution of this problem according to the predictive approach. The whole process is illustrated in the side 2 of the figure 9.

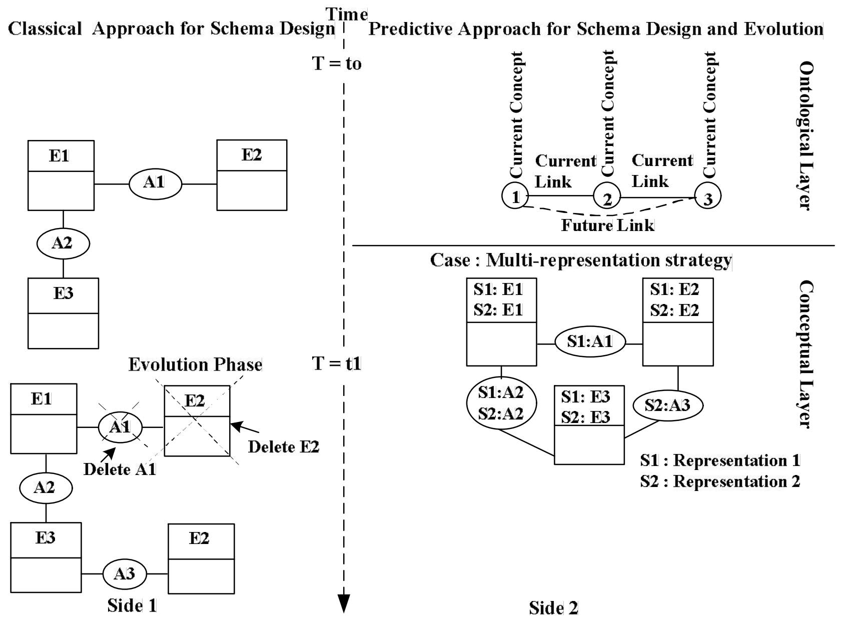

Fig.9. Presentation of Descriptive evolution on a simple example with both Classical and Predictive approaches

5 Conclusion and future Work

In this paper, we have presented another area where ontologies have a colossal potential of utilization and which is related to information systems. It concerns database schema evolution. The approach we propose belongs to a new tendency called the tendency of a priori approaches. It implies the investigation of potential future requirements besides the current requirements during the standard requirements analysis phase of schema design or redesign and their inclusion into the conceptual schema. Those requirements are determined with the help of a domain ontology called "a requirements ontology" using data mining techniques and schema repository. The advantages of this approach include: 1) new perspectives in the way requirements are inspected and integrated into the schema, 2) two categories of database designers were taken into consideration, the category of those who design a schema from scratch and the category of those who redesign the schema from existing schemas using reverse engineering and dependency graphs, 3) the reinforcement of the conceptual schemas, 4) and finally the compatibility of the approach with any data model. The effectiveness of this approach for evolution is limited by the amount and the quality of the knowledge accumulated inside the requirements ontology. Therefore, we have taken into consideration the problem of the evolution of the requirements ontology as well. For this purpose, we have adopted the multi-representation strategy based on stamping mechanism. In [13] a multi-representation solution for ontologies is presented. This solution develops a language based on description logic (DL) [14] to implement the stamping mechanism. Unfortunately, this new approach is not without problems. Predictions of potential future requirements might not turn into effect. The changes that have been included might never be used; however they can be removed completely without any alteration on the database schema. Future work will proceed in both theoretical and practical directions. The theory will focus on extending the idea behind the requirements ontology and the stamping mechanism. The practical work consists in testing this approach significantly through several case studies with the use of a prototype that is under development.

References

1. Connor, R. C. H., Q. I. Cutts, et al.: Using Persistence Technology to Control Schema Evolution. Proceedings of the ACM symposium on Applied computing Phoenix, Arizona, United States ACM Press New York, NY, United States, (1994)
2. Roddick, J. F :A survey of schema versioning issues for database systems. Information and Software Technology (37(7)): 383-393. (1995)
3. Borgida, A. and K. E. Williamson: Accommodating Exceptions in Databases, and Refining the Schema by Learning from them. 11th International Conference on Very Large Data Bases, Stockholm, Sweden,, Morgan Kaufmann, (1985)
4. Benatallah, B. A :Unified Framework for Supporting Dynamic Schema Evolution in Object Database. 18th International Conference on Conceptual Modeling (ER99) Springer-Verlag London, UK, (1999).
5. Sugumaran, V. and V. C. Storey: Ontologies for conceptual modeling: their creation, use, and management." Data Knowledge. Engineering 42(3): 251-271 (2002)
6. Sugumaran, V. and V. C. Storey: An Ontology-Based Framework for Generating and Improving Database Design. Natural Language Processing and Information Systems, 6th International Conference on Applications of Natural Language to Information Systems, NLDB, Stockholm, Sweden, Springer, (2002)
7. Kroenke, D. M.: Database Processing: Fundamentals, Design and Implementation. Database Processing: Fundamentals, Design and Implementation. G. S. d. Acevedo, Pearson Prentice Hall: 265-275. (2004)
8. WordNet: http://wordnet.princeton.edu/
9. Lacy, L. W.: OWL: Representing Information Using the Web Ontology Language. (2005)
10. Bounif H. , Pottinger R.: Schema Repository for Database Schema Evolution, 2nd international workshop on Data Management in Global Data Repositories (GREP) 2006 at International Conference on Database and Expert Systems Applications 06

11. Aamodt, A. and E. Plaza."Case-Based Reasoning: Foundational Issues, Methodological Variations, and Systems Approaches." AI Communications 7(1): 39-59. (1994)
12. Spaccapietra S. et al.:Supporting Multiple Representations in Spatio-Temporal databases. In *Proceedings of the 6th EC-GI & GIS Workshop*, Lyon, France, June 28-30 (2000),
13. Benslimane D. et al.: Multi-representation in ontologies. Proceedings of 7th East-European Conference on Advances in Databases and Information Systems, ADBIS 2003, Dresden, Germany, September 3-6, (2003).
14. Baader, F. and W. Nutt: Basic Description Logics. The Description Logic Handbook: theory, implementation and application. F. Baader: 43-95, Cambridge University Press. (2003).