Requirements Driven Data Warehouse Design: We Can Go Further

Selma Khouri, Ladjel Bellatreche, Stéphane Jean, Yamine Aït-Ameur

Requirements Driven Data Warehouse Design: We Can Go Further

Selma Khouri[1,2], Ladjel Bellatreche[1], Stéphane Jean[1], and Yamine Ait-Ameur[3]

[1] LIAS/ISAE-ENSMA – Poitiers University, France
{selma.khouri,bellatreche,jean}@ensma.fr
[2] National High School for Computer Science (ESI), Algiers, Algeria
s_khouri@esi.dz
[3] ENSEEIHT/IRIT, Toulouse, France
yamine@enseeiht.fr

Abstract. Data warehouses ($\mathcal{DW}$) are defined as data integration systems constructed from a set of heterogeneous sources and user's requirements. Heterogeneity is due to syntactic and semantic conflicts occurring between used concepts. Existing $\mathcal{DW}$ design methods associate heterogeneity only to data sources. We claim in this paper that heterogeneity is also associated to users' requirements. Actually, requirements are collected from heterogeneous target users, which can cause semantic conflicts between concepts expressed. Besides, requirements can be analyzed by heterogeneous designers having different design skills, which can cause formalism heterogeneity. Integration is the process that manages heterogeneity in $\mathcal{DW}$ design. Ontologies are recognized as the key solution for ensuring an automatic integration process. We propose to extend the use of ontologies to resolve conflicts between requirements. A pivot model is proposed for integrating requirements schemas expressed in different formalisms. A $\mathcal{DW}$ design method is proposed for providing the target $\mathcal{DW}$ schema (star or snowflake schema) that meets a uniformed and consistent set of requirements.

Keywords: Data warehouse, semantic heterogeneity, formalism heterogeneity, integration, ontology-based design.

1 Introduction

Data warehouses are defined as data integration systems which data are extracted from a set of heterogeneous sources and materialized in a unified view, in order to answer business and analysis requirements collected from users and decision makers. Concept heterogeneity is one of the most critical issues in $\mathcal{DW}$ system design. Heterogeneity is often associated to data. It is caused by syntactic and semantic conflicts occurring between data stored in different sources. However, same conflicts can occur when gathering requirements from users and decision makers. For example *naming conflicts* occur when concept naming differs between users (eg. synonymy and homonymy conflicts). *Scaling conflicts* arise when different value measures are used when expressing requirements

(for example the price of a product can be given in dollar or in euro). *Confounding conflicts* occur when concepts used by users seem to have the same meaning, but differ in reality due to different measuring contexts. For example, a property price is applied only to new products for user 1, but is applied to all products for user 2. *Representation conflicts* arise when designers describe the same concept in different ways. For example, customer's name is represented by two attributes *FirstName and LastName* for designer 1, and only by one attribute *Name* for Designer 2. The projection of requirements on the ontology helps to identify these conflicts and inconsistencies in order to resolve them.

A second type of heterogeneity concerns formalisms used to define the requirements model. Different formalisms can be used by designers for defining user's requirements. This situation is particularly observed with the development of global enterprises having various corporations that can spread over different states, countries or even continents, where the number of designers may increase and become strongly heterogeneous. Each designer has its own design habits. Consequently, they may use different vocabularies and formalisms to represent their requirements. This brings several challenging issues related to requirements integration: (i) how to integrate vocabularies, (ii) how to integrate the formalisms and (iii) how to identify conflicts and inconsistencies between requirements in an efficient way.

Unfortunately, most $\mathcal{DW}$ design methods focus on data integration and omit requirements integration. This can be explained by the slow evolution of $\mathcal{DW}$ design methods. In the *first generation* of $\mathcal{DW}$ design, dedicated studies have mainly concerned three phases [9]: logical design phase that defines a unified view of data, the ETL (Extract-Transform-Load) phase that extracts data from sources, transforms data if necessary and loads data into the target schema, the physical design phase that implements the final $\mathcal{DW}$ schema and defines some relevant optimization structures. Issues related to integration were managed in the ETL phase. In the *second generation* of $\mathcal{DW}$ design, two additional phases were added: requirements definition and conceptual design phase. Kimball's studies [12] introduced requirements definition in $\mathcal{DW}$ design. Different requirements driven design methods followed proposing to define $\mathcal{DW}$s from a set of users' requirements. Other hybrid methods proposed to define $\mathcal{DW}$s from both sources and requirements. The conceptual design phase has completed the design cycle in order to provide a $\mathcal{DW}$ schema independent of all implementation issues, which facilitates its validation by users However, integration issues were not studied for these additional phases, and concerned exclusively managing conflicts occurring between data sources. There is now a consensus on a typical $\mathcal{DW}$ life cycle that includes the following phases [8]: requirements definition, conceptual design, logical design, ETL phase and physical design.

Several studies were proposed in the literature related to the problem of data integration. The major progress toward automatic integration resulted from some levels of explicit representation of data meaning through ontologies [4]. Ontologies are defined as consensual and explicit representations of conceptualization. Some $\mathcal{DW}$ design methods have proposed the use of domain ontologies in order

to manage conflicts between sources. The main proposition of this paper is to extend the use of ontologies in order to manage requirements conflicts. Furthermore, we take advantage of an important ontological skill: *reasoning* in order to identify different relationships between requirements, and to obtain a design schema of better quality. Three main contributions are proposed in this paper:

- Concerning formalism heterogeneity: two feasible scenarios may be offered to designers: (i) they use a generic formalism (pivot model) to express their requirements. Pivot model is a solution that reduces the complexity of exchanging different models. (ii) Designers keep using their favorite formalism and a mapping between their model and the pivot one is established. This scenario is better than the first one since it provides more autonomy to designers. We propose as a first contribution, a pivot model between three requirements formalisms usually used in $\mathcal{DW}$ design (process driven formalism, use case formalism and goal formalism).
- Concerning vocabulary heterogeneity: conflicts occurring between data stored in sources or between requirements collected from users can be solved if the meaning of each object (term) used is defined precisely and explicitly. We thus propose the use of a shared ontology integrating data sources. The second contribution consists of connecting the requirements pivot model to this ontology in order to eliminate all semantic conflicts, and unify the heterogeneous vocabularies used during requirements collection.
- Reasoning: Ontological reasoning mechanisms are then used to identify different relationships between requirements, which constitutes the third contribution. Only a set of consistent requirements is recorded to identify the target $\mathcal{DW}$ schema.

In order to realize these contributions, we propose a design method for $\mathcal{DW}$ schema definition, covering the following phases: requirements definition and analysis, conceptual design, logical design and physical design. This method takes as inputs: a shared ontology integrating sources and a set of users' requirements (collected from heterogeneous users and defined using different formalisms). It provides a $\mathcal{DW}$ schema (star or snowflake) covering a set of consistent requirements. We used *Lehigh University Benchmark*[1] (LUBM) ontology to illustrate our proposal. Figure 1 illustrates the proposed approach.

The rest of the paper is organized as follows: section 2 presents related works. Section 3 presents the proposed design method. The pivot and the ontology models are first described. Ontological reasoning mechanisms are used in order to analyze the given set of requirements. The four design phases are then presented. Section 4 presents the implementation of our approach. We illustrate how the analysis of requirements allows obtaining a $\mathcal{DW}$ schema of better quality. Section 5 concludes the paper.

[1] `swat.cse.lehigh.edu/projects/lubm/`

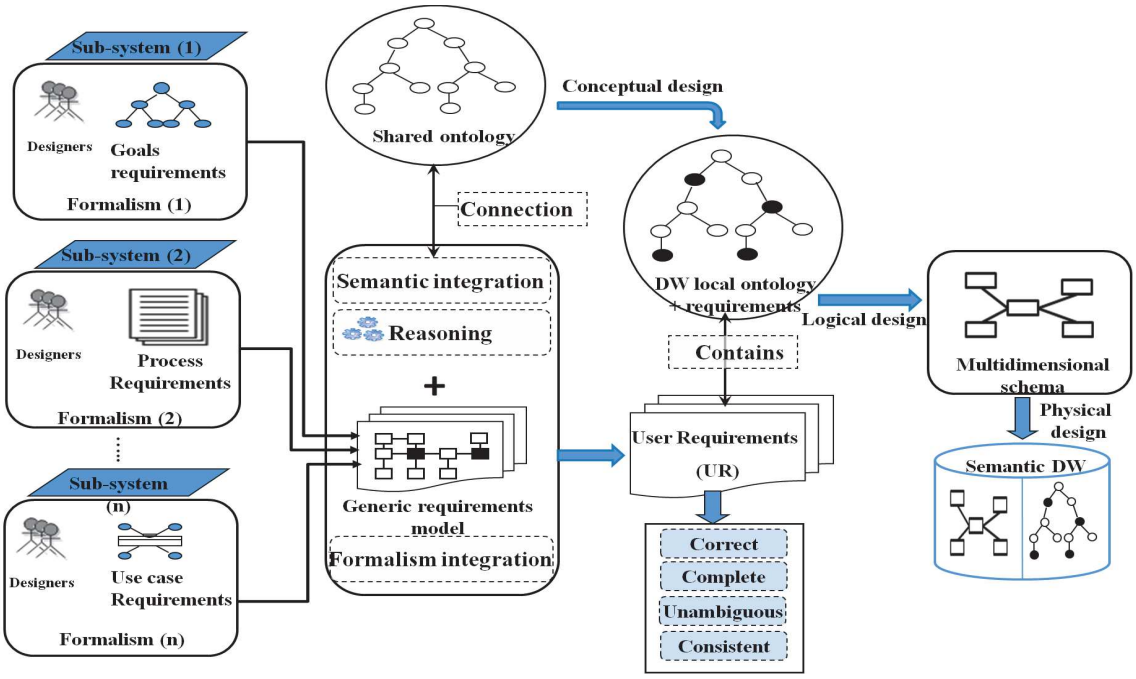

Fig. 1. Approach overview

2 Related Work

This section presents different studies related to $\mathcal{DW}$ design, we focus on efforts proposing requirements driven approaches. Then, we present studies using ontologies to define a $\mathcal{DW}$ schema. Finally, we present studies using ontologies for analyzing requirements and reasoning on them.

2.1 $\mathcal{DW}$ Design: A Requirements Driven Perspective

The purpose of $\mathcal{DW}$ design schema is to define a target schema providing a unified view of data and answering a set of requirements. This schema must handle multidimensional concepts (facts, dimension, measures, dimensions attributes and hierarchies). The $\mathcal{DW}$ schema can be defined at different abstraction levels: conceptual, logical or physical. Different methods have been proposed to define this design schema. The instability of $\mathcal{DW}$ life cycle makes most research efforts concentrate on one or two design phases. Proposed studies usually deal with the definition of $\mathcal{DW}$ schema or the ETL phase populating the schema. The ETL phase is out of the scope of this study. The definition of the requirements phase in $\mathcal{DW}$ design emerged from different studies proposing: *supply* driven, *demand* driven and *hybrid* approaches.

Requirements definition plays a crucial role in $\mathcal{DW}$ design and determines its functional behavior and all needed enterprise information. The requirement engineering process can be divided into four activities: requirements elicitation and analysis, specification, validation and management. Requirements elicitation and analysis in $\mathcal{DW}$ design literature differ according to the object analyzed.

We distinguish: (1) *Process* driven analysis [22] that analyzes requirements by identifying business processes of the organization, (2) *User* driven analysis that identifies requirements of target users and unifies them in a global model like [3,14] that develops use case models to define $\mathcal{DW}$ requirements, and (3) *Goal* driven analysis [6] that identifies goals and objectives that guide decisions of the organization at different levels. These requirements can be functional or non functional.

Lopez et al. [15] classify requirements specification techniques into three categories: (i) *informal* techniques using natural language, sometimes with structuring rules, (ii) *semi-formal* techniques generally based on graphic notations with a specified syntax like IStar or UML diagrams [14] and (iii) *formal* techniques based on mathematical or logical notations providing a precise and non-ambiguous framework for requirements modeling. For example, [11] propose to use description logic formalism to define requirements.

Several research efforts were proposed to deal with formalism heterogeneity problem. The work of [21] is an example of these studies, where the authors propose solutions to integrate semi-formal formalisms (that use diagram and tabular techniques) and formal formalisms (that use mathematics, logic or algebra). However, this effort has not been made for $\mathcal{DW}$ design.

2.2 Ontologies for Designing $\mathcal{DW}$s

Ontologies have been introduced in $\mathcal{DW}$ design for integrating heterogeneous sources. In these studies, a domain ontology is assumed existent. The set of sources reference this ontology. These references can be defined a priori during the source design, or a posteriori using matching algorithms that align sources to the ontology. Bellatreche et al. [1] provide an overview of different integration scenarios based on ontologies. Ontological methods for designing $\mathcal{DW}$s emerged recently, following both supply driven and demand driven approaches. The first two methods are mainly supply-driven, where a domain ontology is used as a schema integrating data sources: [17] defines the $\mathcal{DW}$ multidimensional model (facts and dimensions) from an OWL ontology by identifying functional dependencies (*Functional ObjectProperties*) between ontological concepts. Nebot et al. [16] define a semi-automatic method to build multidimensional tables from semantic data guided by the user requirements. We proposed in [11] a hybrid design method that extends the use of ontologies for resolving two issues: integration of sources and for the specification of the requirements model. However, formalism heterogeneity issue is not studied in this work. Romero et al. proposed in [18] a hybrid method producing a multidimensional model from an OWL ontology describing sources. Requirements are then used to identify the ETL operations needed for mapping sources to target data stores.

2.3 Ontologies for Requirements Engineering

Requirements engineering field has used ontologies since the 80's and still in recent works to support analysis and reasoning on requirements. Proposed studies

provide solutions dedicated for transactional systems (not decisional ones). As instance, [10] proposed an ontological method for analyzing requirements, where a mapping between specified requirements and ontological elements is established. This ontology consists of a thesaurus and inference rules. [13] proposed an approach to improve requirements specified in natural language by the use of linguistic ontologies. [19] studied the problem of requirements expression and their refinement. The authors propose the use of goal-oriented analysis language to describe each requirement that can be refined into sub-goals. The majority of these studies manage heterogeneity of vocabularies, but they ignore the heterogeneity of the used modeling languages.

Other studies used ontologies for reasoning about requirements. As instance, Siegemund et al.[20] used ontologies for structuring concepts, requirements and relationships captured during requirements elicitation. The approach provides : an ontology-based requirements *meta model* describing meta data and requirements relationships, and a set of *consistency and completeness rules* for validating the requirement Specification. Goknil et al.[7] propose a metamodel supporting the common concepts of some requirements modeling approaches. Four types of requirements relationships are identified: Refines, Requires, Conflicts, and Contains. Based on this formalization, analysts can perform reasoning on requirements to detect implicit relations and inconsistencies. The entered requirements and their relations are stored in an OWL ontology.

We notice however that these studies are dedicated for transactional requirements. Surprisingly, no effort has been made for exploiting ontological specification and its reasoning capabilities for analyzing $\mathcal{DW}$ user's requirements in order to enhance the $\mathcal{DW}$ schema defined. Besides, the main limitation of these ontological proposals is about the consensuality of their ontologies. The ontology presented in these studies is not consensual, it is only defined to store a set of relevant requirements. This limits designers that aim to share and exchange their models with other project groups referencing the same requirements ontologies. We assume in our approach the existence of a *consensual* ontology defined by *domain experts*.

3 Preliminaries : Ontology Formalism

OWL is the ontology definition language endorsed by the World Wide Web Consortium (W3C). OWL language is based on description logic formalism (a first order logic). DL formalism is defined as the formalism used to define logics specifically designed to represent structured knowledge and to reason upon. We used DL concepts definition to formalize the ontology model. The ontology model is formally defined as follows OM: <C, R, Ref (C), Formalism>

- C: denotes *Concepts* of the model (atomic concepts and concept descriptions).
- R: denotes *Roles* (relationships) of the model. Roles can be relationships relating concepts to other concepts, or relationships relating concepts to data-values (like Integers, Floats, etc).

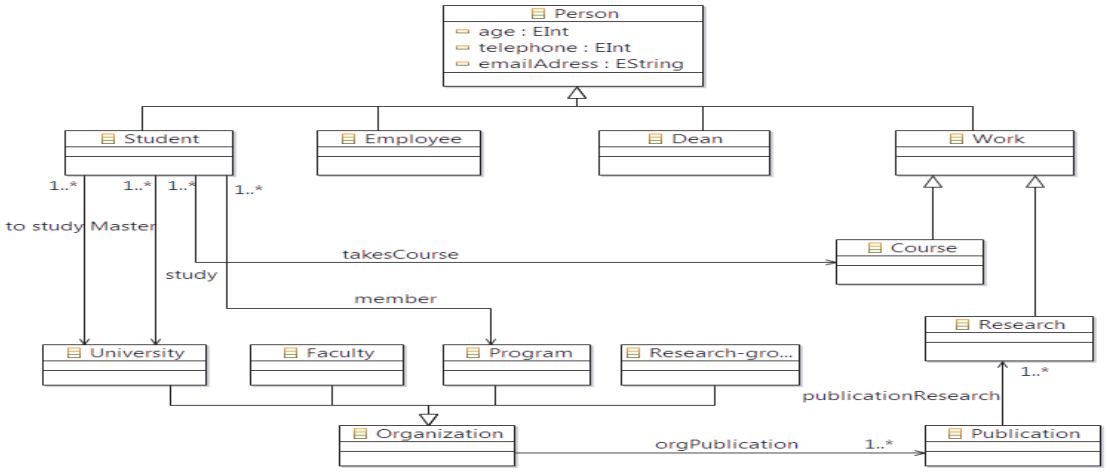

Fig. 2. A partial view of LUBM ontology schema

- $Ref : C \cup R \rightarrow$ (Operator, Exp(C,R)). Ref is a *function* defining terminological axioms of a DL TBOX. Operators can be inclusion ($\sqsubseteq$) or equality ($\equiv$). Exp(C,R) is an expression over concepts and roles of OM using constructors of description logics such as union, intersection, restriction, etc. (e.g., Ref(Student)$\rightarrow$($\sqsubseteq$, Person $\sqcap$ $\forall$takesCourse(Person, Course))).
- Formalism is the *formalism* followed by the global ontology model like RDF, OWL, etc.

In our context, we assume the existence of a shared global ontology. An ontology is shared when the sources are committed to using its ontological definitions, which are accepted and eventually standardized. Each contributor of a project shall reference that ontology "as much as possible" (i.e. each local class must reference its smallest subsuming class in the shared ontology) Locally, designers may extend it by other concepts and properties to fitful his local requirements. As consequence, each designer will have his own ontology (called local ontology). The designers may communicate through the common used concepts defined in the shared ontology.

4 Proposed Method

We present in this section the method we propose to design $\mathcal{DW}$ schemas. We describe our proposal following the design steps: requirements definition, conceptual design, logical design and physical design.

4.1 Requirements Definition

This phase includes three steps: (1) definition of the pivot model, (2) connection of the pivot model to the ontology model and (3) ontological analysis of requirements.

Definition of the Pivot Model: in order to identify the different components of our pivot model, we deeply studied three important formalisms used in requirements-driven $\mathcal{DW}$ design : Goal-Oriented formalism, Process-Oriented language (we studied MCT model, a process model of MERISE, a french modeling methodology), and UML use case formalism. We proposed in [2] the proposed pivot model. Let's take the following requirement example to illustrate the model concepts: "the system should analyze attendance of students to courses".

Three main components are identified: Actions, Results and Criteria (Fig3.(b)). Each requirement is designated by one action to accomplish (*Analyze Attendance*). If a requirement includes more than one action, it can be decomposed in multiple requirements (one for each action). Each requirements is influenced by one or many criteria (*Student*, *Course*). Each requirement have a result to fulfill (*Attendance*) that can be measured by a formal or semi formal metric (*the number of students attending the course*). Each requirement involves one or many actors that interact with the system to achieve the requirement. Two types of requirement are distinguished: *functional* and *non-functional*. Requirements can be related with each other through one of the following relationships: (*Requires*, *Refines*, *Contains* and *Conflicts*). These relationships will be populated by using reasoning rules on requirements.

Formally, we define a requirement as follows:
Requirement:$< \mathcal{A}, \mathcal{R}, \mathcal{M}, \mathcal{C} >$, in which:

- $\mathcal{A}$: the action that a system performs to yield an observable result.
- $\mathcal{R}$: the results realized by the system.
- $\mathcal{M} = \{m_1, m_2, ..., m_n\}$, a set of metrics quantifying the result.
- $\mathcal{C} = \{c_1, c_2, ..., c_n\}$, a set of sequence of criteria influencing the requirement's result.

Connection of Ontology Model to Requirements Model: the domain ontology is used in our approach as a formal and consensual domain dictionary, from which the designer can choose the most relevant concepts to express collected requirements. Requirements are structured using the proposed pivot model. They are afterwards expressed at the ontological level. In order to achieve this, we defined a mapping between coordinates of each requirement (Action and Criteria) and the resources (concepts) of the domain ontology. The connection between the ontology and requirement pivot model is presented in figure 3, where part (a) presents a fragment of the ontology metamodel connected to the pivot model (part (b)). The merged meta-model, called *OntoPivot* is defined as follows:

$\mathcal{O}ntoPivot :< \mathcal{GO}, \mathcal{P}ivot_{model} >$, Ontological Pivot, such that:

- $\mathcal{GO} : < C, R, Ref(C), Formalism >$ is the global shared ontology
- $\mathcal{P}ivot_{model}$: $< \mathcal{A}ctor, \mathcal{R}equirement, \mathcal{R}elationship >$, such that:
  - $\mathcal{R}equirement$:$< \mathcal{A}, \mathcal{R}, \mathcal{M}, \mathcal{C} >$, such that:
    - $\mathcal{A} = \{a_1, a_2, ..., a_n\}$, set of actions. For each $a \in \mathcal{A}$, $a \in 2^C U 2^R U 2^{Ref}$ (ontological domain).
    - $\mathcal{C} = \{c_1, c_2, ..., c_n\}$, set of criteria. For each $c \in \mathcal{C}$, $c \in 2^C U 2^R U 2^{Ref}$.
  - $\mathcal{R}elationships = \{Contains, Refines, Conflicts, Requires\}$, set of relations between requirements. For each $relation \in \mathcal{R}elationships$, $relation \in 2^R$.

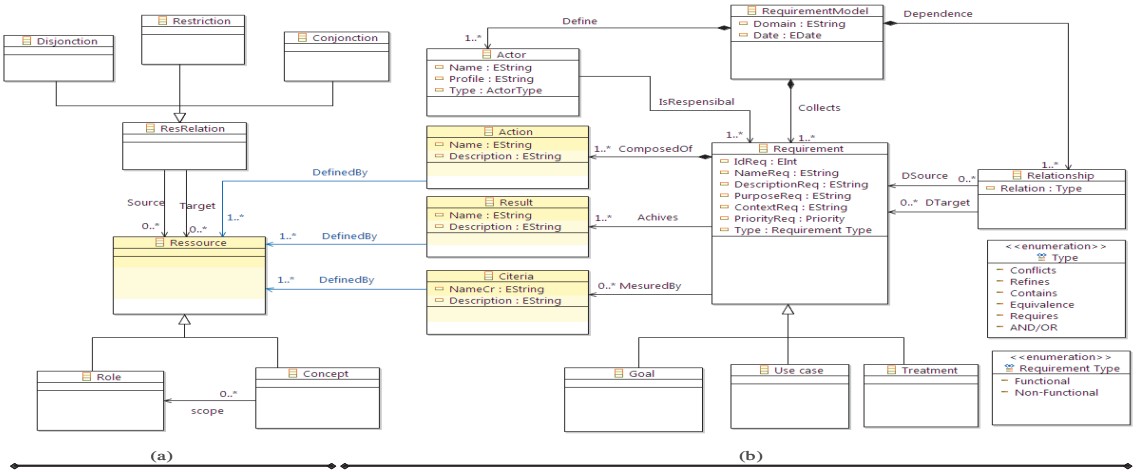

Fig. 3. Pivot metamodel connected to the ontology metamodel [2]

Note that each requirement can introduce new concepts to the ontology. For example, the requirement "the system should analyze attendance of students to courses" will be defined as a new concept in the ontology (having action, result, criteria and metric *properties*). This concept is defined as an instance of a meta concept "owl:Requirement" extending OWL meta model (see figure 4). This requirement concept can introduce a new concept "AnalyzeAttendance", whereas students and courses are already defined in the shared ontology. Consistency reasoning mechanism is used to identify incoherences and correct them. This process allows the definition of an application ontology, which combines a domain ontology and task (requirements) ontology.

Ontological Analysis of Requirements: once requirements are structured using the pivot model and expressed formally using the ontology model, they are analyzed to discover hidden relationships between them. As stated in the literature, four main relationships between requirements can be defined: contains, refines, requires and conflicts. Some reasoning mechanisms are already supported by the ontology like the equivalence between requirements concepts, others must be defined as new rules. Let's assume that: Subclass(C) is the set of subclasses of

each class c ∈ C, Role(c) is the set of roles having class c as domain, Action(R) denotes the action class of requirement R, Criteria(R) denotes the set of criteria classes of Requirement R. If a role is used to define a requirement's action, its domain class is returned. If an expression (using Ref function) is used to define requirement's action or result, it is considered as a defined class. We formally defined the following reasoning rules to identify the four relationships between requirements:

- Refinement relationship: A requirement R refines a requirement R' if R is derived from R' by adding more details to its properties [7]. Formally, refines relation is defined as follows:

R refines R' if Action(R) $\sqsubseteq$ Action(R') AND (Criteria(R) $\subset$ Subclass(Criteria(R')) OR Criteria(R) $\subset$ Role(Criteria(R')))

  Example 1. R: The system shall analyze messages sent to individuals, teams, or all course participants at once.
  R': The system shall analyze messages sent.
  where: Action(R) and Action(R'): AnayzeMessagesSent,
  Criteria (R): {Individual, Team, Participant}, Criteria (R'): {$\top$}
  We observe that : Action(R) $\equiv$ Action(R') and Criteria(R) $\subset$ Subclass (Criteria(R'))

- Containment relationship: A requirement R contains a requirement R' if R' are parts of the whole R1 (part-whole hierarchy) [7]. Formally, containment relation is defined as follows:

R contains R' if Action(R) $\sqsubseteq$ Action(R') And Criteria(R) $\subset$ Criteria(R')

  Example 2. R: The system shall allow lecturers to analyze enrollment policies based on grade, first-come first-serve and department.
  R': The system shall allow lecturers to analyze enrollment policies based on grade.
  where: Action(R) and Action(R'): AnalyzeEnrollment,
  Criteria (R): {Grade, Position,Department}, Criteria (R'): {Grade}
  We observe that : Action(R) $\equiv$ Action(R') and Criteria(R) $\subset$ Criteria(R')

- Conflict relationship: A requirement R conflicts with a requirement R2 if the fulfillment of R1 excludes the fulfillment of R2 and vice versa [7]. Formally, conflicts relation is defined as follows:

R refines R' if Action(R) $owl : disjointWith$ Action(R') And Criteria(R) $\subseteq$ Subclass(Criteria(R')) OR Criteria(R) $\subseteq$ Criteria(R')

  Example 3. R: The system shall allow lecturers to limit the number of students subscribing to a course.
  R': the system shall have no maximum limit on the number of course participant ever.

where: Action(R): LimitNbStudent and Action(R'): NotLimitNbStudent,
Criteria (R): {Student, Course}, Criteria (R'): {Participant, Course}
We observe that : Action(R) $owl : disjointWith$ Action(R')
and Criteria(R) $\subset$ Subclass(Criteria(R'))

– Require relationship: A requirement R requires a requirement R2 if R1 is fulfilled only when R2 is fulfilled [7]. We introduce for this relation a new relation 'owl:Require' between OWL entities (E1 owl:Require E2) extending OWL meta model, that denotes that entity E1 is a precondition for entity E2. Formally, requires relation is defined as follows:

R requires R' if Action(R) $owl : Require$ Action(R') And Criteria(R) $\subseteq$ Subclass(Criteria(R')) OR Criteria(R) $\subseteq$ Criteria(R')

Example 4. R: The system shall allow analyze students notification.
R': the system shall provide messaging facilities.
where: Action(R): AnalyzeNotification and Action(R'): ProvideMessaging,
Criteria (R): {Student}, Criteria (R'): {$\top$}
We observe that : Action(R) $owl : Require$ Action(R')
and Criteria(R) $\subset$ Subclass(Criteria(R'))

4.2 Conceptual Design

A $\mathcal{DW}$ ontology (DWO) viewed as a conceptual abstraction of the $\mathcal{DW}$, is defined from the global domain ontology (GO) by extracting all concepts and properties used by user requirements. Three scenarios are possible:

1. $DWO = GO$: the GO corresponds exactly to users' requirements,
2. $DWO \subset GO$: the DWO is extracted from the GO,
3. $DWO \supset GO$: the GO does not fulfill all users' requirements.

We defined in [11] different reasoning mechanisms for checking the consistency of the ontology and for identifying multidimensional concepts. We also proposed an algorithm that analyses users' requirements in order to identify the multidimensional role of concepts and properties and store them as ontological annotations. The multidimensional annotation of DWO is based on user requirement. Following Kimball's definition, we consider that each requirement Result is the fact to analyze, each of its metrics is a measure candidate for this fact, and each of its criteria is a candidate dimension. Facts are linked to dimensions by looking for one-to-many relationships between corresponding ontological concepts. Dimensions hierarchies are formed by looking for many-to-one relationships between dimensions linked to the same fact. This annotation is validated by the designer.

DWO definition extends DO formalization as follows: <C, R, Ref (C), Formalism, Multidim> where $Multidim : C \cup R \rightarrow$ Role. $Multidim$ is a function that denotes the multidimensional role (fact, dimension, measure, attribute dimension) of concepts and roles.

4.3 Logical Design

The logical model of the $\mathcal{DW}$ is generated by translating the annotated DWO into a relational model. Several works in the literature proposed methods for translating ontologies described in a given formalism (PLIB, OWL, RDF) to a relational or object-relational representation. This translation can follow three possible relational representations: *vertical*, *binary* and *horizontal*. Vertical representation is used for RDF ontologies, and stores data in a unique table of three columns (subject, predicate, object). In a binary representation, classes and properties are stored in tables of different structures. Horizontal representation translates each class as a table having a column for each property of the class. We proposed in [5] a set of translation rules for representing PLIB and OWL ontology (classes, properties and restrictions) in a relational schema following the binary and horizontal representations.

4.4 Physical Design

This last phase implements the final $\mathcal{DW}$ schema using a chosen DBMS. Both conventional or semantic data repositories can be used to implement the $\mathcal{DW}$ schema. Semantic data repository stores both the logical and conceptual schema (in the form of a local ontology). As we extended the ontology with the requirements model, even requirements can be stored in the repository. We implemented the obtained $\mathcal{DW}$ schema using two semantic repositories: OntoBD (academic database) and Oracle semantic database. OntoDB supports a horizontal storage layout, whereas Oracle supports a vertical storage layout.

5 Implementation

In order to implement our approach, we used LUBM ontology related to university domain, and the CMS (course management system) requirements document[2]. CMS provides a set of 60 requirements related to teaching and management of courses including interactions with students taking the course. Requirements have been adapted to a decisional application. We modified actions of requirements to analysis actions, which are more suitable for $\mathcal{DW}$ applications.

The implementation of LUBM ontology is made using *Protege* framework, defined as free, open-source ontology editor and framework for building intelligent systems (http://protege.stanford.edu/). The ontology is defined using *OWL2* language. The definition of the pivot requirements model at the ontological level is defined by the extension of OWL ontology meta model. Figure 4 illustrates this extension. Requirement meta class is defined as a new class instantiating meta class 'class', which defines all classes of the ontology. The whole pivot model is defined. Action, Criteria and Result are defined as properties of Requirement class, they have owl:Class or owl:Property as a range. Each CMS requirement is defined as an instance of this Requirement class. Relationships between requirements are defined as roles.

[2] The full requirements document is available at `http://www.home.cs.utwente.nl/~goknila/sosym/`

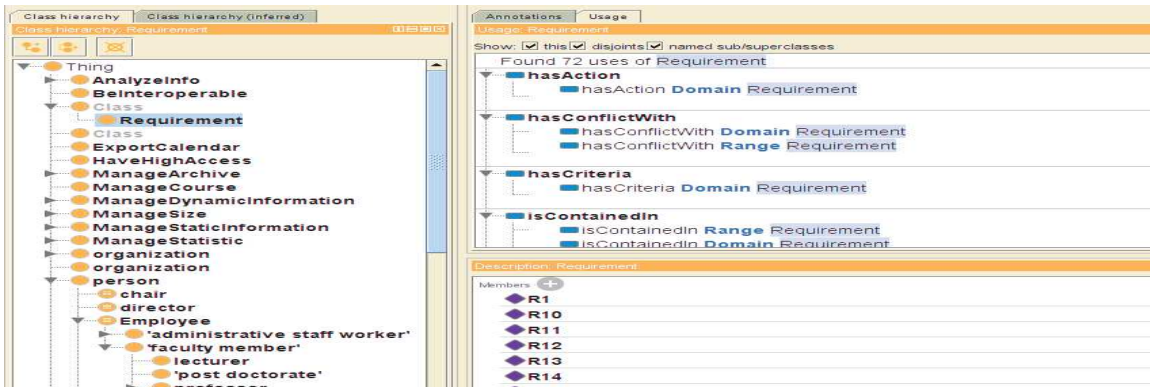

Fig. 4. OWL meta model extended with the requirement pivot model using Protege Editor

The set of reasoning rules, identifying relationships between requirements, are implemented in a java program accessing the ontology using *OWL API* (owlapi.sourceforge.net/). Each relationship inferred is stored in the ontology for the corresponding requirements. The program identified a set of relationships between defined requirements: 20 refinement relationships, 10 containment relationships, 12 require relationships and 4 conflict relationships. Figure 5 presents a set of requirements and discovered relationships between them. The schema is obtained using Protege plugin *OntoGraf*[3].

The *DWO* is defined from LUBM ontology using the modularity method *OWLExtractor*. This method is chosen because it is dedicated for OWL ontologies and it provides a Protege plugin implementing the method. The method takes as inputs the domain ontology and a signature (set of terms which will be extracted in the local ontology). In our approach, the signature corresponds to the set of requirements. We identified a subset of relevant requirements by analyzing the relationships between them. For example, refinement and containment relationships allow to eliminate some redundant requirements. When a requirement contains other requirements, the first requirement is kept, the contained requirement can be ignored. When a requirement refines another requirement, the first one gives more details (usually more criteria) to the second one. The second requirement can thus be ignored. Require relationship allow to identify the set of requirements that must be included in the final schema as they present necessary prerequisites to other requirements. Conflict relationships allow to identify requirements that cannot be fulfilled together, and cause inconsistencies. The designer must choose one of these conflictual requirements. Each requirement has a priority attribute, which can be used to eliminate requirements having the lowest priority. Require relationship can also be used. If a requirement is required by other requirements, it is more careful to not reject it.

[3] `http://protegewiki.stanford.edu/wiki/OntoGraf`

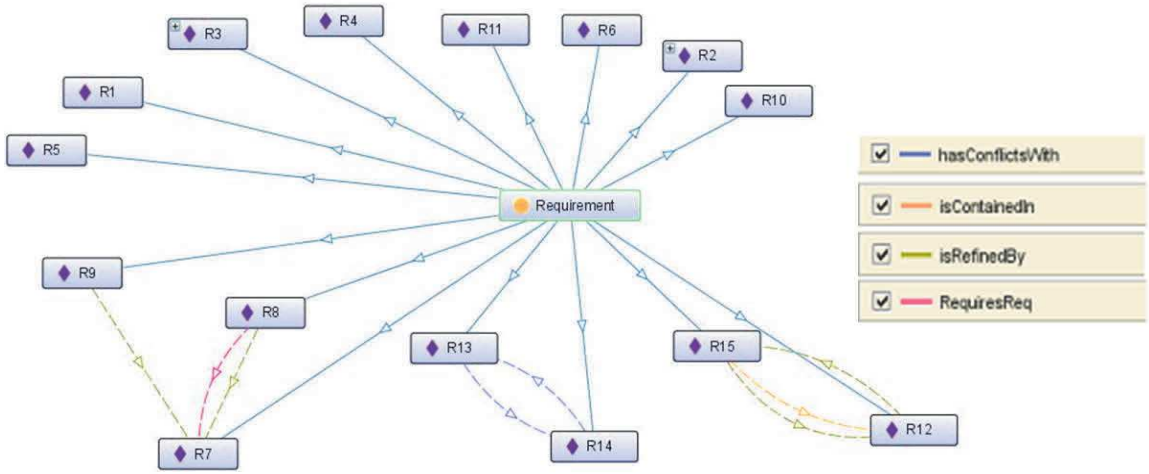

Fig. 5. Discovered relationships between requirements

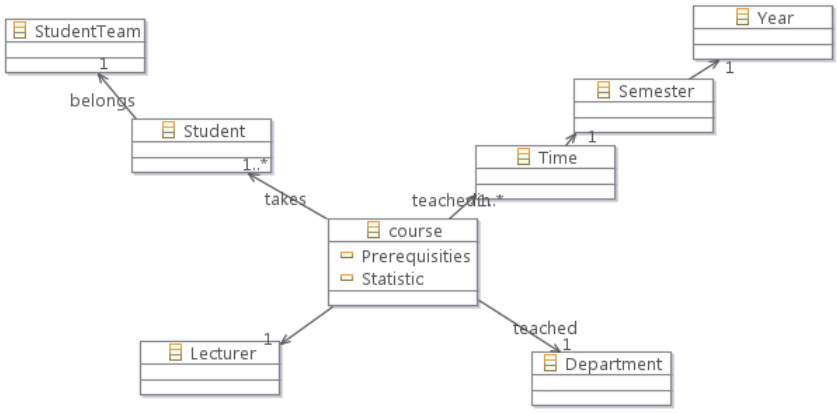

Fig. 6. $\mathcal{DW}$ Multidimensional schema obtained

The annotation algorithm is executed to annotate the extracted ontology by multidimensional annotations. Figure 6 illustrates the obtained multidimensional schema.

The reasoning rules help us to obtain a $\mathcal{DW}$ schema of better quality. Suppose that step 2 (analysis of requirements) is ignored. This would provide a schema containing conflict requirements. For example, the schema cannot answer the non functional requirement stating that the system should "limit space of storage for courses", and another requirement stating that the system should "maximize space of specific courses ". The schema would include redundant concepts due to the presence of containment and refinement relations between requirements. In fact, instead of managing and validating 60 requirements, we just have to manage 40 requirements. The validation of this schema is easier since it has to be validated by a consistent subset of requirements.

6 Conclusion

Various $\mathcal{DW}$ design methods have been proposed covering different design phases: conceptual, logical, physical and ETL design phases. Most of these methods consider integration issues related to data, but ignore requirements integration. User's requirements are collected from heterogeneous users, which usually causes semantic conflicts. Requirements are analyzed and formalized by different designers, which can cause schematic and formalisms heterogeneity. We propose in this paper to manage requirements integration for $\mathcal{DW}$ definition through an ontology-based design method. The method takes as inputs a set of requirements, and a shared ontology integrating sources. For handling formalisms heterogeneity, we defined a pivot model between three formalisms usually used for $\mathcal{DW}$ requirements models (process, use case and goal formalisms). The pivot model is connected to the shared ontology. This connection allows expressing requirements using ontological concepts which eliminates semantic conflicts. It also allows reasoning on requirements in order to identify semantic relationships between requirements (refine, contain, require and conflict relationships). The $\mathcal{DW}$ schema is then defined by following three design stages: conceptual, logical and physical design. The target $\mathcal{DW}$ schema is defined from a set of coherent and consistent requirements. We illustrated the proposed approach using LUBM ontology and requirements defined in the CMS requirements document.

There are different open issues that we are currently working on like: the management of requirements evolution, the completion of the approach with the ETL process for loading data, and the evaluation of the approach in a large scale case study in which we evaluate $\mathcal{DW}$ quality metrics and get designers feedback.

References

1. Bellatreche, L., Dung, N.X., Pierra, G., Hondjack, D.: Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. Computers in Industry 57(8), 711–724 (2006)
2. Boukhari, I., Bellatreche, L., Khouri, S.: Efficient, unified, and intelligent user requirement collection and analysis in global enterprises. In: Proceedings of International Conference on Information Integration and Web-based Applications & Services, p. 686. ACM (2013)
3. Bruckner, R., List, B., Schiefer, J.: Developing requirements for data warehouse systems with use cases. In: Proc. 7th Americas Conf. on Information Systems, pp. 329–335 (2001)
4. Doan, A., Halevy, A.Y., Ives, Z.G.: Principles of Data Integration. Morgan Kaufmann (2012)
5. Fankam, C.: OntoDB2 : Un systeme flexible et efficient de Base de Donnees á Base Ontologique pour le Web semantique et les donnees techniques. PhD thesis, ENSMA (December 2009)
6. Giorgini, P., Rizzi, S., Garzetti, M.: Goal-oriented requirement analysis for data warehouse design. In: Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP, pp. 47–56. ACM (2005)

7. Goknil, A., Kurtev, I., Berg, K., Veldhuis, J.-W.: Semantics of trace relations in requirements models for consistency checking and inferencing. Softw. Syst. Model. 10, 31–54 (2011)
8. Golfarelli, M.: From user requirements to conceptual design in data warehouse design a survey. In: Data Warehousing Design and Advanced Engineering Applications Methods for Complex Construction, pp. 1–16 (2010)
9. Inmon, W.H.: Building the data warehouse. J. Wiley (2002)
10. Kaiya, H., Saeki, M.: Ontology based requirements analysis: Lightweight semantic processing approach. In: Proceedings of the Fifth International Conference on Quality Software, pp. 223–230. IEEE Computer Society (2005)
11. Khouri, S., Boukhari, I., Bellatreche, L., Jean, S., Sardet, E., Baron, M.: Ontology-based structured web data warehouses for sustainable interoperability: Requirement modeling, design methodology and tool. Computers in Industry, 799–812 (2012)
12. Kimball, R., Reeves, L., Thornthwaite, W., Ross, M., Thornwaite, W.: The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses, 1st edn. John Wiley & Sons, Inc., New York (1998)
13. Körner, J.S., Torben, B.: Natural language specification improvement with ontologies. Int. J. Semantic Computing 3, 445–470 (2009)
14. List, B., Schiefer, J., Tjoa, A.M.: Process-oriented requirement analysis supporting the data warehouse design process a use case driven approach. In: Ibrahim, M., Küng, J., Revell, N. (eds.) DEXA 2000. LNCS, vol. 1873, pp. 593–603. Springer, Heidelberg (2000)
15. López, O., Laguna, M.A., García, F.J.: Metamodeling for requirements reuse. In: Anais do WER02-Workshop em Engenharia de Requisitos, Valencia, Spain (2002)
16. Nebot, V., Berlanga, R.: Building data warehouses with semantic web data. Decision Support Systems (2011)
17. Romero, O., Abelló, A.: Automating multidimensional design from ontologies. In: Proceedings of the ACM Tenth International Workshop on Data Warehousing and OLAP, pp. 1–8. ACM (2007)
18. Romero, O., Simitsis, A., Abelló, A.: Gem: Requirement-driven generation of etl and multidimensional conceptual designs. In: Data Warehousing and Knowledge Discovery, pp. 80–95 (2011)
19. Saeki, M., Hayashi, S., Kaiya, H.: A tool for attributed goal-oriented requirements analysis. In: 24th IEEE/ACM International Conference on Automated Software Engineering, pp. 674–676 (2009)
20. Siegemund, K., Edward, J., Thomas, Y., Yuting, Z., Pan, J., Assmann, U.: Towards ontology-driven requirements engineering. In: 7th International Workshop on Semantic Web Enabled Software Engineering (October 2011)
21. Wieringa, R., Dubois, E.: Integrating semi-formal and formal software specification techniques. Information Systems 23(3-4), 159–178 (1998)
22. Winter, R., Strauch, B.: A method for demand-driven information requirements analysis in data warehousing projects. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003, pp. 9–19. IEEE (2003)