# Ontology evolution for an experimental data integration system

Liliana Ibanescu, Patrice Buche, Stephane Dervaux, Rim Touhami, Juliette Dibie-Barthelemy

# Ontology Evolution for an Experimental Data Integration System

**Abstract:** This paper describes an ontology evolution activity designed for a data integration system called ONDINE (Ontology based Data INtegration) which proposes a complete workflow to acquire, annotate and query experimental data extracted from scientific documents. The core element of the ONDINE system is an ontology which allows experimental data annotation and querying. In order to adapt to domain changes, new usages and new annotated data, the ontology may change. This paper presents our *a priori* ontology evolution activity, which takes as input an ontology in a consistent state, defines and applies some changes and manages all the consequences of those changes by producing an ontology in a consistent state. The implementation and evaluation of the evolution activity are presented. Our work is illustrated through an ONDINE system's use case, the annotation of experimental data in the domain of matter transfer.

**Keywords:** ontology evolution, data integration system, web semantic language

## 1 Introduction

Ontologies are one of the fundamental layers of the Semantic Web and are designed to represent the knowledge from a domain in terms of concepts (or classes), relations between these concepts and instances of these concepts (Guarino et al., 2009). An ontology, defined as a formal, explicit specification of a shared conceptualisation (Studer et al., 1998) may change whenever the domain changes or when domain experts need to add or to restructure the knowledge. When an ontology is used as a system component (i.e. the knowledge backbone) of an advanced information system its evolution is a complex process and raises many challenges as, for example, the formal representation of ontology changes, the verification of ontology consistency after applying ontology changes, and the propagation of those changes to ontology related entities (e.g. data sets annotated with the ontology).

In Buche et al. (2013), a complete data integration system, called ONDINE (ONtology-based Data INtEgration) is presented. It is designed to annotate experimental data extracted from scientific documents, to store them into an annotation knowledge base and to allow afterward a flexible querying of the annotation knowledge base. The backbone of ONDINE system is an Ontological and Terminological Resource (OTR), composed of a conceptual component and a domain one, called naRyQ (n-ary Relations between Quantitative experimental data) (Touhami et al., 2011), dedicated to the representation of experiments. Each experiment involves a studied object, some control parameters and a result. It is represented using a n-ary relation without distinguished arguments as recommended by the W3C (World Wide Web Consortium) in Noy et al. (2006). The conceptual component of naRyQ is divided into two subcomponents: a *core conceptual component* to represent n-ary relations between experimental data and a *domain conceptual component* to represent specific concepts of a given application domain. naRyQ can therefore be reused for different application domains: only the specific part of the OTR, which is at the heart of the scientific experimental data integration, must be redefined to use naRyQ for another domain.

The first version of the ONDINE system, containing the annotation and querying subsystems, were presented in Buche et al. (2013) with results in three distinct domains: microbial risk, chemical risk and aeronautics. Relying on these successful experiences of using ONDINE for different application domains, three new projects were developed. The first one concerns matter transfer, the second one is in biorefinery, and the third one is about the sustainability of the durum wheat chain. For each new project, the domain component of naRyQ was adapted by adding, updating and deleting concepts and/or relations. The management of this evolution task is not easy in general (Zablith et al., 2015) and is clearly time consuming and error prone when the ontology is complex with many relations between concepts, as it is the case in our experimental domains. There is therefore a need to assist domain experts in the management of the naRyQ evolution.

Current research activities in ontology evolution can be grouped in two main categories (Zablith et al. (2015)). In a first category, research activities propose *a posteriori* approaches to maintain the coherence of an ontology during its evolution (see Haase and Stojanovic (2005); Djedidi (2009); Khattak et al. (2013)). This type of approach allows the detection of inconsistencies, then suggests how to repair them. In a second category, research activities propose *a priori* approaches, where the coherence checking is made before the application of changes. These approaches are favored from an engineering point of view when the ontology's building is concerned, avoiding backtracking after modification, resulting in a loss of time and resources. The work of Stojanovic (2004) is the first to propose an ontology

evolution process defined for KAON ontologies and using strategies for the task of managing changes. Jaziri et al. (2010) define kits of changes in order to manages the inconsistencies generated by each change. Tissaoui et al. (2011) present a system of evolution of an OTR dedicated to the semantic annotation of text documents. Mahfoudh et al. (2015) propose a framework based on graph rewriting rules that maintains a set of constraints. In our work, we have combined, adapted and extented these existing approaches on ontology evolution in order to propose a naRyQ *a priori* evolution activity taking into account its main originality that is its inter dependent concepts to manage experimental data. Our approach was implemented in a plug-in, called DynarOnto, designed to assist domain experts in the naRyQ evolution activity.

The extended version of ONDINE presented in this paper where the OTR evolution is taken into account is given in Figure 1. ONDINE system is composed of three subsystems: (1) the annotation subsystem of experimental data extracted from scientific documents using concepts from naRyQ; (2) the querying subsystem designed to query the annotation knowledge base using naRyQ, (3) the new subsystem, detailled in this paper, designed to manage naRyQ evolution in a consistent way.

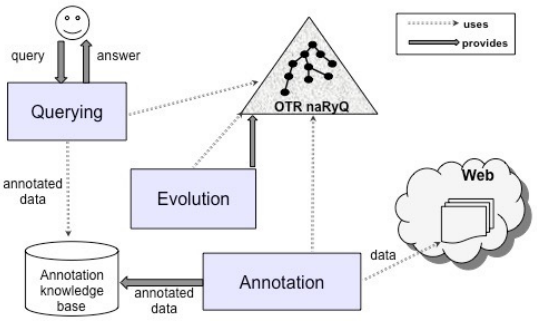


**Figure 1** ONDINE system architecture.

Our paper is organized as follows. In Section 2, naRyQ is first briefly recall and then we present our definition of its coherence. In Section 3, its evolution activity is detailed. Section 4 present the implementation of our evolution approach, called DynarOnto, and its evaluation. The presentation of DynarOnto is illustrated in Section 4 through an ONDINE system's use case, the annotation of experimental data in the domain of matter transfer. Finally, we conclude in Section 5.

## 2 naRyQ and its Coherence

In this section, naRyQ presented in Touhami et al. (2011) is first recalled, then the coherence constraints it must respect are presented.

### *2.1 The Ontological and Terminological Resource naRyQ*

naRyQ (Touhami et al., 2011) is an Ontological and Terminological Resource (OTR) designed to annotate and query experimental data. Figure 3 gives an excerpt of naRyQ in the matter transfer domain, called TRANSMAT (for MATter TRANSfer) available at `http://doi.org/10.15454/1.430814846357855E9`.

**Example 1:** Let us consider the experiment with the result, permeability, the studied object, packaging, and the following control parameters: packaging thickness, temperature and differential partial pressure. This experiment can be represented by the 5-ary relation *Permeability_Relation* from Figure 2.

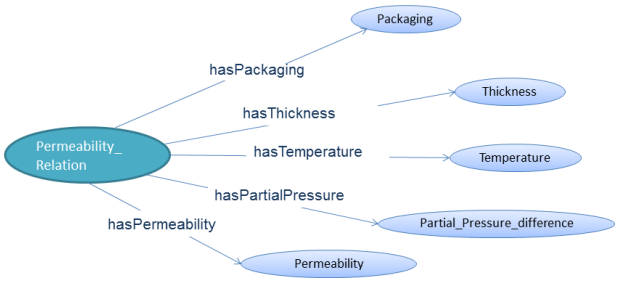


**Figure 2** n-ary relation Permeability_Relation.

We briefly remind in the following the conceptual component (i.e. its ontology) and the terminological component of naRyQ and present its composition in three application domains.

#### *2.1.1 The conceptual component of naRyQ*

The conceptual component of naRyQ is composed of a *core conceptual component* to represent n-ary relations between experimental data and a *domain conceptual component* to represent specific concepts of a given application domain.

The core conceptual component has two layers: an upper layer, called *up core conceptual component*, to represent n-ary relations between any arguments, and a lower layer, called *down core conceptual component*, to represent n-ary relations between experimental data. The representation of n-ary relations between experimental data requires a particular focus on the management of quantities.

In the up core conceptual component, generic concepts *Relation_Concept* and *Argument* represent respectively n-ary relations and their arguments. In the down core conceptual component, generic concepts *Dimension*, *UM_Concept*, *Unit_Concept* and *Quantity* allows the management of quantities and their associated units of measure. The subconcepts of the generic concept *Symbolic_Concept* represent the non numerical arguments of n-ary relations between experimental data.



Comment citer ce document :
Ibanescu, L., Buche, P., Dervaux, S., Touhami, R., Dibie, J. (2016). Ontology evolution for an experimental data integration system. International Journal of Metadata, Semantics and Ontologies, 11 (4), 231-242. , DOI : 10.1504/IJMSO.2016.10004259

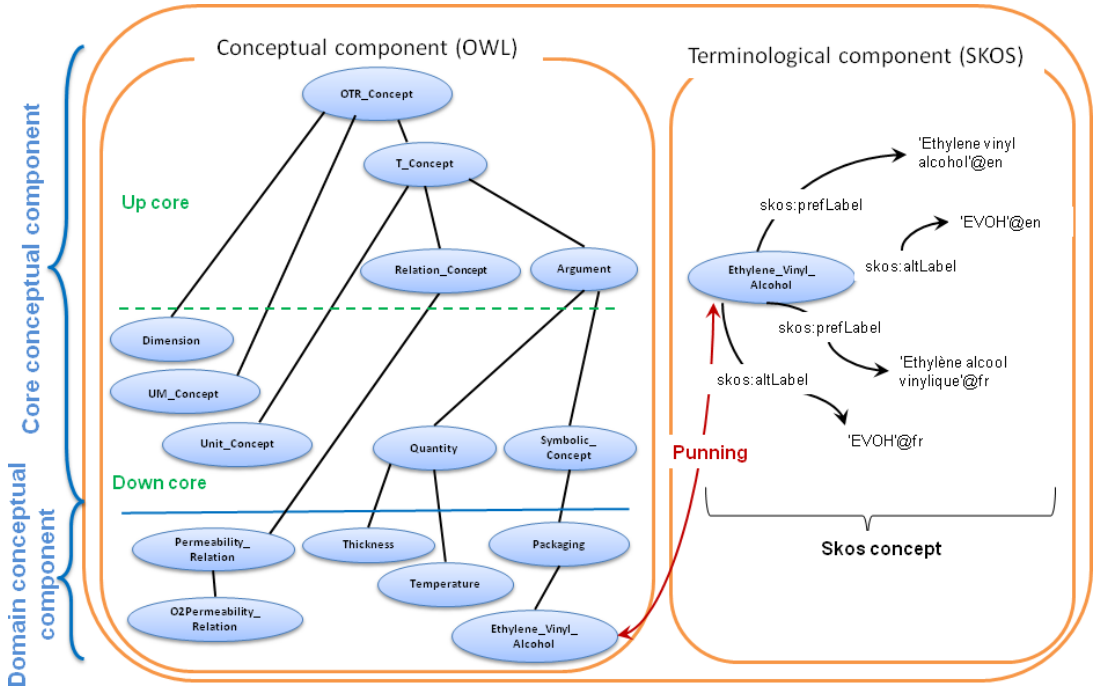


**Figure 3** An excerpt of naRyQ in the matter transfert domain.

All concepts are represented as OWL classes (see `http://www.w3.org/TR/owl-ref` for more details), hierarchically organized by the subsumption relation *subClassOf* and pairwise disjoints.

### *2.1.2 The terminological component of naRyQ*

The terminological component of naRyQ contains the set of terms describing the studied domain and are used to annotate experimental data. Sub concepts of the generic concepts *Relation_Concept*, *Symbolic_Concept* and *Quantity*, as well as instances of the generic concept *Unit_Concept*, are all denoted by at least one term of the terminological component. Each of those subconcepts or instances are, in a given language, denoted by a preferred label and optionally by a set of alternative labels, which correspond to synonyms or abbreviations. Labels are associated with a concept or an instance thanks to SKOS (Simple Knowledge Organization Scheme) labeling properties recommended by W3C (see `http://www.w3.org/TR/skos-reference/` for more details).

**Example 2:** In Figure 3, English terms *Ethylene vinyl alcohol* and *EVOH* denote the symbolic concept *Ethylene_Vinyl_Alcohol*.

### *2.1.3 Three application domains of naRyQ*

Three different domain components of naRyQ were developed. The first one concerns matter transfer, the second one is about biorefinery, and the third one is about the sustainability of the durum wheat chain. Table 2.1.3 presents the number of concepts for each application domain.

| Application domain | Concepts | Symbolic concept | Quantity | N-ary relation |
|---|---|---|---|---|
| TRANSMAT (matter transfer) | | 930 | 62 | 52 |
| Biorefinery | | 188 | 54 | 32 |
| Wheat_durum | | 104 | 8 | 4 |

**Table 1** Number of concepts in the three domain components of naRyQ.

Let us notice that the concepts for the management of units of measure are managed in a transversal way in a single OTR currently composed of 268 units. Measure units can therefore be shared by all the domain conceptual components of naRyQ.

The three versions of naRyQ are available at `http://pfl.grignon.inra.fr/atWeb/`.

### *2.2 naRyQ $\mathcal{CC}$-coherence*

In Haase and Stojanovic (2005) the coherence of an ontology is classified in three categories: i) structural coherence which is related to constraints of the ontology's representation languages, ii) logical coherence where the semantic correctness of the ontology's entities are checked and iii) user-defined coherence which refers to specific user requirements and constraints related to the ontology's context of use. Inspired by Stojanovic (2004), a set of conditions that the ontology must respect to be coherent for each category of coherence is defined. These conditions are called Coherence Constraints, denoted by $\mathcal{CC}$-constraints. **An ontology is $\mathcal{CC}$-coherent if it respects a set of defined $\mathcal{CC}$-constraints**, which are model dependent.

The $\mathcal{CC}$-constraints defined for naRyQ in order to take into account its specificity that is its inter



Comment citer ce document :
Ibanescu, L., Buche, P., Dervaux, S., Touhami, R., Dibie, J. (2016). Ontology evolution for an experimental data integration system. International Journal of Metadata, Semantics and Ontologies, 11 (4), 231-242. , DOI : 10.1504/IJMSO.2016.10004259

dependent concepts and the SKOS representation of its terminological component are presented below. A full list of $\mathcal{CC}$-constraints defined for naRyQ can be found in Touhami (2015).

### *2.2.1 Structural $\mathcal{CC}$-constraints*

naRyQ was modeled using a subset of OWL2-DL entities (e.g. classes, properties, constructors of classes) and axioms. The constraint defined by the W3C group which states that every OWL axiom must be well defined (see `http://www.w3.org/TR/owl-ref/#OWLDL`) was therefore applied to each axiom and constructor used in the naRyQ modeling. Two cases are distinguished:

- if an entity $e$ is an anonymous concept (e.g. a restriction), then its definition relies on one or more other entities, which must already be defined in the OTR;
- if an entity $e$ is a named concept, then its definition relies on another entity, which must already be defined in the OTR.

Thirteen structural $\mathcal{CC}$-constraints were extracted from the literature, denoted by $\mathcal{CC}$s. Each one was defined for each named or anonymous concept, which refers to other entities.

Two examples of $\mathcal{CC}$s are presented below.

- $\mathcal{CC}s_1$ Each anonymous class defined by a value restriction, `owl:allValuesFrom` or `owl:someValueFrom` links a pair property-class or property-data type. The property and the class used in the definition of the anonymous class must be defined in the OTR.
- $\mathcal{CC}s_2$ A concept (or relation) that subsumes another concept (or relation) should be defined in the OTR.

We also defined **two new structural constraints** linked to terms, the terminological part of naRyQ represented with SKOS concepts being specific to our OTR according to the literature on ontology evolution (see Tissaoui et al. (2011)).

- $\mathcal{CC}s_3$ Each instance of `skos:concept` must have at least a preferred label.
- $\mathcal{CC}s_4$ Labels denoting a concept or a given instance should contain exactly one preferred label for each language.

### *2.2.2 Logical $\mathcal{CC}$-constraints*

Six logical $\mathcal{CC}$-constraints, denoted by $\mathcal{CC}$l, were extracted from the literature. Two examples of $\mathcal{CC}$l are presented below.

- $\mathcal{CC}l_1$ A class cannot be disjoint with its superclass.
- $\mathcal{CC}l_2$ If two values restrictions `owl:allValuesFrom` which connect a pair property-concept are associated with the same concept $c$, then the concepts defining the restrictions cannot be disjoint.

### *2.2.3 User-defined $\mathcal{CC}$-constraints*

The user-defined $\mathcal{CC}$-constraints, denoted by $\mathcal{CC}$u, correspond either to quality criteria modeling (Stojanovic, 2004) or to specific criteria corresponding to the modeled task.

Nine generic $\mathcal{CC}$u which refer to quality criteria modeling and **twenty new $\mathcal{CC}$u which are specific to the task of experimental data integration** (i.e. annotation and querying) were defined. These twenty new $\mathcal{CC}$u are divided into two groups: nine $\mathcal{CC}$u correspond to the annotation of generic n-ary relations and eleven $\mathcal{CC}$u correspond to the annotation of n-ary relations between experimental data. Eight examples of new $\mathcal{CC}$u specific to the task of experimental data integration are presented below.

Six new $\mathcal{CC}$u specific to the annotation of generic n-ary relations:

- $\mathcal{CC}u_1$ Each domain concept must be linked by a specialization relation to at least another named concept of the ontology which belongs to the core ontology.
- $\mathcal{CC}u_2$ N-ary relations and arguments are mutually disjoint.
- $\mathcal{CC}u_3$ A n-ary relation has at least two arguments.
- $\mathcal{CC}u_4$ An instance of a n-ary relation is linked to at least two instances of arguments.
- $\mathcal{CC}u_5$ A n-ary relation belonging to a hierarchy of relations inherits from all the arguments of its parent. This n-ary relation can also add new arguments and/or override the inherited arguments by more specific ones.
- $\mathcal{CC}u_6$ Each n-ary relation and each argument is associated with a terminological part.

Two new $\mathcal{CC}$u specific to the annotation of n-ary relations between experimental quantitative data:

- $\mathcal{CC}u_7$ Each quantity must be associated with only one dimension through the value restriction `hasValue` and the property `hasDimension`. The dimension must be defined as an instance of the concept `Dimension`.
- $\mathcal{CC}u_8$ Each quantity must be associated with its units of measurement (at least one) through the `owl:allValuesFrom` restriction and the property `hasUnitConcept`. The units of measurement are defined by enumeration using the `owl:oneOf` constructor.



Comment citer ce document :
Ibanescu, L., Buche, P., Dervaux, S., Touhami, R., Dibie, J. (2016). Ontology evolution for an experimental data integration system. International Journal of Metadata, Semantics and Ontologies, 11 (4), 231-242. , DOI : 10.1504/IJMSO.2016.10004259

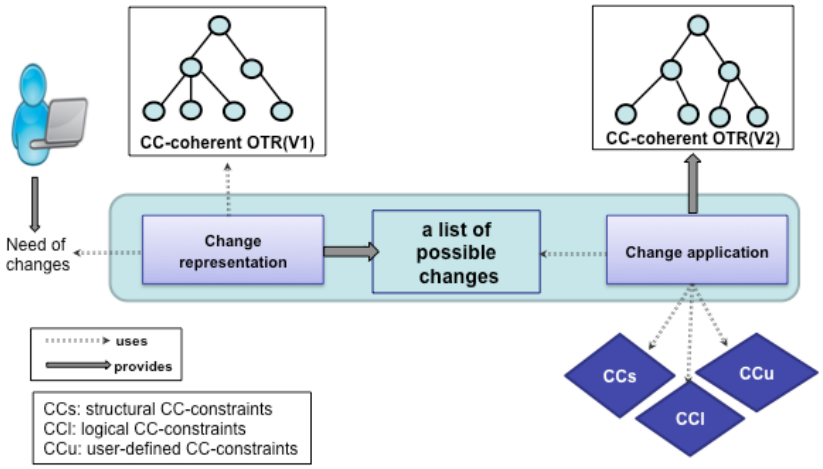


**Figure 4** The ontology evolution activity.

We present, in the following section, our *a priori* ontology evolution activity where the $\mathcal{CC}$-coherence checking is made before the application of changes. Let us notice that with an *a posteriori* approach, most of the structural and logical $\mathcal{CC}$-constraints presented above can be checked by generic OWL reasonners. Nevertheless, this is not the case for the user-defined $\mathcal{CC}$-constraints that are specific to the modeled task and at the heart of our ontology evolution activity.

## 3 Coherent Evolution of naRyQ

The ontology evolution activity presented in this paper is defined as in the NeOn Glossary of ontology engineering tasks which states that ontology evolution is 'the activity of facilitating the modification of an ontology by preserving its consistency' (see Palma et al. (2012)). Our ontology evolution activity proposes an *a priori* approach to help domain experts of the ONDINE system to manage the naRyQ evolution while preserving its $\mathcal{CC}$-coherence. It is represented in the workflow given in Figure 4 (Touhami et al., 2015), based on the methodological guidelines given in Palma et al. (2012) and Haase and Stojanovic (2005).

The first step of the evolution activity is detailed in Subsection 3.1 and consists in presenting all the possible changes for naRyQ evolution to the domain expert. From this list of changes, the domain expert chooses the ones to be applied. Subsection 3.2 presents the second step which consists in applying changes while preserving the $\mathcal{CC}$-coherence of naRyQ.

### *3.1 Change Representation*

The first step of the ontology evolution activity from Figure 4 consists in presenting to the domain expert all the possible changes identified as usefull by the domain expert to naRyQ evolution.

In order to generate all needed changes in our application context, all entities and axioms used to model naRyQ in OWL2-DL and SKOS were first identified. Fifty five changes were selected from the literature and **twenty six new changes** were defined to take into account the specificity of naRyQ (i.e. its inter dependent concepts and the SKOS representation of its terminological component) in our application context. Among these changes, some changes are not accessible to the domain expert but are rather used in the kits of changes, presented in the next subsection. Table 3.1 contains a subset of changes required for the evolution of naRyQ, where changes in bold are new ones compared to the literature.

| Change | Element | Role |
|---|---|---|
| addClass | NamedClass | add an OWL class to an ontology |
| addSubClassOf | SubClass | add a subsumption link |
| updateDomain | Domain | update the domain of a relation |
| updateSomeValues FromRestriction | SomeValuesFrom | update an existential restriction |
| **addDataTypeRestriction** | DataTypeRestriction | add a value range |
| **updatePrefLabel** | SkosPrefLabelAssertion | update a preferred label of a SKOS concept |

**Table 2** A subset of changes for naRyQ evolution.

### *3.2 Change Application*

After applying a change, one or more $\mathcal{CC}$-constraint may be violated (e.g. adding a new concept which represents a n-ary relation violates $\mathcal{CC}u_3$ constraint). This is particularly the case in naRyQ where there exist several relations between concepts and many concepts are associated with a terminological part. Hence, a second step is necessary to restore the $\mathcal{CC}$-coherence of

naRyQ while applying a change. To achieve this goal, the notions of kit of changes from Jaziri et al. (2010) and of strategy from Stojanovic (2004) were adapted.

### *3.2.1 Kit of changes*

To maintain *a priori* the $\mathcal{CC}$-coherence of naRyQ during its evolution, a kit of changes was associated with each change that violates one or more $\mathcal{CC}$-constraints and which is accessible to the domain expert. Its definition takes into account all the $\mathcal{CC}$-constraints as defined in Subsection 2.2 which can be violated by the requested changes. Definition 3.1 of a kit of change inspired from Jaziri et al. (2010) is given below.

**Definition 3.1:** *Given a $\mathcal{CC}$-coherent ontology $O$ and a change $c$, if the application of $c$ to $O$ doesn't preserve the $\mathcal{CC}$-coherence of $O$, then* **a kit of changes** *is associated with $c$. It is composed of:*

- *preconditions: a set of assertions which must be true in order to apply $c$ to $O$;*
- *the change $c$;*
- *mandatory additional changes: a set of changes which are attached to $c$ in order to restore the $\mathcal{CC}$-coherence of $O$ when $c$ is applied to it;*
- *optional additional changes: a set of changes which can be applied in addition to mandatory additional changes.*
- *post-conditions: a set of assertions which must be true after the application of $c$ to $O$.*

A change can therefore be 'safely' applyed to naRyQ (i.e. while preserving its $\mathcal{CC}$-coherence) if it is applied with its corresponding kit of changes that is automatically applied.

Sixty three kits were defined for naRyQ according to the possible changes identified as usefull by the domain experts in our application context.

Let us give an example of kit of changes where the complexity of the naRyQ evolution management is stressed by (i) the management of its inter dependent concepts: n-ary relations with their arguments (see below mandatory additional changes M3, M4 and optional additional change O2), quantities with their units of measurement (see mandatory additional change M5 and optional additional change O3) and (ii) the management of the naRyQ terminological component (see below mandatory additional change M1 and optional additional change O1).

**Example 3:** The *addClass* change allows a new class *newc* to be added to naRyQ. The kit of changes associated with the *addClass* change is defined in the following.
**Precondition**: the class *newc* must not exist in the set of concepts $C$ of naRyQ.
**Change**: the creation of the new class *newc*.
The application of this change violates several $\mathcal{CC}$-constraints of the OTR. The application of a set of mandatory additional changes is then required.
**Mandatory additional changes**:

M1. The additional changes `addSkosConcept` and `addPrefLabel` are triggered to respect $\mathcal{CC}u_6$.

M2. The creation of a disjunction relationship between the new class and all its sibling classes is necessary to respect $\mathcal{CC}u_2$.

M3. If the new class *newc* is a n-ary relation, then at least two arguments must be associated with *newc* in order to respect $\mathcal{CC}u_3$. Additional changes `addAllValuesFromRestrictionToClass`, `addSomeValuesFromRestrictionToClass` or `addHasValueRestrictionToClass` are applied.

M4. If the new class *newc* is a n-ary relation, then cardinality restrictions (greater than or equal to 1) must be associated with the mandatory arguments of *newc* to ensure their existence at the instance level in order to respect $\mathcal{CC}u_4$.

M5. If the new class *newc* is a quantity, then a dimension and a unit of measurement must be associated with it in order to respect $\mathcal{CC}u_7$ and $\mathcal{CC}u_8$.

M6. Adding a specialization relation between *newc* and its superclass $c$ is necessary to respect $\mathcal{CC}u_1$.

Besides these mandatory additional changes, it is possible to apply other optional additional changes which are not necessary to maintain the $\mathcal{CC}$-coherence of naRyQ but which may be requested by the domain expert. Below are examples of optional additional changes.
**Optional additional changes**:

O1. adding other prefered and alternative labels to the new class,

O2. adding other arguments to the new class if it is a n-ary relation,

O3. adding other units of measurement if it is a quantity.

**Post-condition**: the class *newc* exists in the set of concepts $C$ of naRyQ.

It is important to notice that each mandatory or optional additional change can violate other $\mathcal{CC}$-constraints and can be resolved by triggering the required kits of changes. Therefore each additional change can also call a kit of changes.



Comment citer ce document :
Ibanescu, L., Buche, P., Dervaux, S., Touhami, R., Dibie, J. (2016). Ontology evolution for an experimental data integration system. International Journal of Metadata, Semantics and Ontologies, 11 (4), 231-242. , DOI : 10.1504/IJMSO.2016.10004259

### *3.2.2 Evolution Strategies*

The kit of changes defined in Definition 3.1 can be used when there is only one possible solution to solve an incoherence. But sometimes there can be several possible solutions to correct violated constraints. In this case, evolution strategies (Stojanovic, 2004) have to be used in order to represent the different alternatives.

**Example 4:** In order to delete a concept, there are several solutions to deal with its subconcepts and there are also several solutions to deal with its terms:

- To deal with orphaned concepts (subconcepts of deleted concept), the possible solutions are 1) delete them, 2) link them to super concepts or 3) link them to the root.
- To deal with terms of deleted concept, the possible solutions are: 1) delete them or 2) associate them with super concepts.

**Definition 3.2:** *Given a $\mathcal{CC}$-coherent ontology $O$ and a change $c$, if the application of $c$ to $O$ leads to several possible alternatives to maintain the $\mathcal{CC}$-coherence of $O$, then the choice made between these alternatives is called* **evolution strategy**.

There exist two types of kits of changes: those with evolution strategies and those without. The kit of changes associated with the change *addClass*, presented in Example 3, is without evolution strategies. Kits with evolution strategies can be considered as a variant of the first type since they need to fix the strategy (i.e. the choice) before applying the kit.

## 4 DynarOnto

The ontology evolution activity presented above is implemented as a Protégé plug-in (Protégé is a free open source ontology editor available at `http://protege.stanford.edu/`) called DynarOnto. It aims to implement the ONDINE evolution subsystem and assists domain experts in the naRyQ evolution activity. Fifteen kits of changes that was identified as the most usefull by the domain experts on the sixty three ones were implemented in DynarOnto.

This section deals with the presentation of DynarOnto and its illustration through a use case in ONDINE system: the annotation of experimental data which needs an OTR evolution. We first briefly present the annotation subsystem of ONDINE, then DynarOnto through a use case and finally the evaluation of DynarOnto.

### *4.1 The annotation subsystem of ONDINE*

The annotation subsystem of ONDINE from Figure 1 is implemented in the @Web software available at `http://www6.inra.fr/cati-icat-atweb/Web-platform`. The @Web annotation module implements a complete workflow to extract experimental data tables from scientific documents and semantically annotate them with n-ary relation concepts from naRyQ. The five steps are presented in Figure 5.

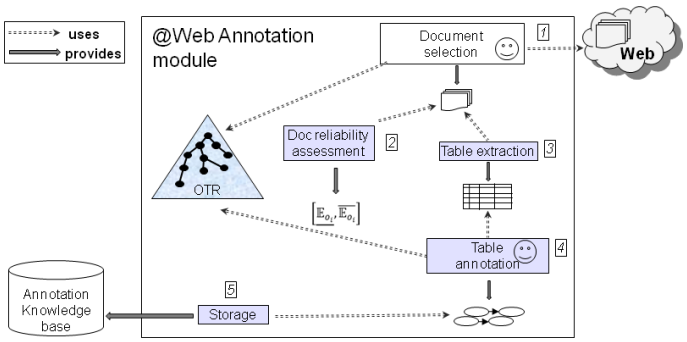


**Figure 5** The annotation module of @Web software.

In the first step, relevant documents for the studied application domain described in naRyQ are retrieved from the Web and manually selected by an expert of the domain. Then, in the second step, each document is associated with a reliability assessment score (see Destercke et al. (2013) for more details). The third step deals with data tables extraction from the selected documents. Those data tables are then presented to the domain expert for validation allowing him/her to select those that are relevant for the studied domain. In the fourth step, taking into account the content of the data table, the domain expert selects from the n-ary relation concepts defined in the studied domain conceptual component of naRyQ those relevant to semantically annotate the data table. In the fifth and last step of @Web annotation module, the annotated data tables are stored in a RDF triple store and could therefore be queried in the ONDINE querying subsystem (see Figure 1).

### *4.2 The use case*

Let us focus on the fourth step, the table annotation, of @Web annotation module from Figure 5 and consider the TRANSMAT OTR from Figure 3 and the data table from Figure 6. The domain expert have to select the relevant n-ary relation concept(s) of TRANSMAT to semantically annotate the data table. To do so, he/she is guided by the experimental result(s) of the target n-ary relation concept(s). Two experimental results can be identified in the data table of Figure 6 thanks to the considered domain application: *O2_Permeability* and *CO2_Permeability*.

Let us consider the n-ary relation concept *O2_Permeability_Relation* of TRANSMAT which is a subrelation of the n-ary relation *Permeability_Relation* of Figure 2. It is composed of the six following arguments: Packaging, Thickness, Temperature, Partial_pressure_difference, Relative_humidity and

O2_permeability that is its experimental result. The domain expert can select this n-ary relation concept to annotate the data table. Let us notice that the data table of Figure 6 does not necessarily contain all the arguments of the n-ary relation concept to be annotated with it. In our case, there are three missing arguments in the data table of Figure 6 which only contains the temperature (the first column T(°C)), the relative humidity (the second column RH %) and O2_permeability (the fourth column).

Unfortunatly, the domain expert cannot find in TRANSMAT the relevant n-ary relation concept having for experimental result the concept *CO2_Permeability* to annotate the data table of Figure 6 with. A new n-ary relation concept has therefore to be added to TRANSMAT using DynarOnto.

The two next subsections will detail this update step and the data table annotation's step of @Web annotation module with the new version of TRANSMAT.

**Table2. Central Composite DesignArrangement and Responses**

| Variable levels | responses | | | |
|---|---|---|---|---|
| T(°C) | RH (%) | $CO_2$ permeability ($amol^{-1}m^{-1}Pa^{-1}$) | $O_2$ permeability ($amol^{-1}m^{-1}Pa^{-1}$) | Select-ivity |
| 9 | 14.6 | 258 | 111 | 2.3 |
| 39 | 14.6 | 314 | 131 | 2.4 |
| 9 | 85.3 | 11475 | 1011 | 11.3 |
| 39 | 85.3 | 22353 | 863 | 25.9 |
| 3 | 50 | 317 | 181 | 1.7 |
| 45 | 50 | 1026 | 233 | 4.4 |
| 24 | 0 | 88 | 77 | 1.1 |

**Figure 6** A data table from a scientific document in matter transfer domain.

### *4.2.1 Presentation of DynarOnto through the use case*

In this section, we present DynarOnto through the evolution need identified in the studied use case.

A new n-ary relation concept *CO2_Permeability_Relation* has to be added to TRANSMAT in order to annotate the data table of Figure 6. This n-ary relation concept must be composed of the six following arguments: Packaging, Thickness, Temperature, Partial_pressure_difference, **Relative_humidity** and **CO2_permeability**. It is a subrelation of the existing n-ary relation *Permeability_Relation* of Figure 2 and is called *CO2_Permeability_Relation*. Arguments presented in bold above are specific to *CO2_Permeability_Relation*. The other arguments are inherited from its super n-ary relation concept *Permeability_Relation* to respect $\mathcal{CC}u_5$.

To add this new class in TRANSMAT, the *addClass* change and its associated kit of change presented in Example 3 is used. The screen shot from Figure 7 presents the corresponding interface. It shows the three different designed menus of DynarOnto:

- The *Relation Changes* menu presents the changes that can be applied to a n-ary relation concept (e.g. add a n-ary relation, delete a n-ary relation, update the arguments of a n-ary relation).
- The *Argument Changes* menu contains the changes that can be applied on arguments of a n-ary relation concept (i.e. Quantity and Symbolic_Concept).
- *Evolution Parameters* menu helps the domain expert to define its evolution strategy that will be taken into account during the OTR evolution. This menu consists in setting which choices will be made between the possible alternatives before applying a change (e.g. deleting orphaned concepts) by clicking on several checkboxes. These choices can be modified before each change application.

To add the new n-ary relation concept *CO2_Permeability_Relation* using DynarOnto, the domain expert starts by clicking on *Add relation* in the *Relation changes* menu. Using the panel of the displayed interface (i.e. panel 1), the domain expert enters the name of the new n-ary relation, *CO2_permeability_Relation*. Then, he/she chooses *Permeability_Relation* as its parent class in the hierarchy of n-ay relation concepts of the OTR displayed by clicking on "Choose" button.

Inherited arguments then appears in panel 3 of the displayed interface. The domain expert can both specialize inherited arguments and/or add new ones by specifying their numbers in panel 2. To define the new n-ary relation concept *CO2_permeability_Relation*, the domain expert should add the new argument *Relative_humidity* and specialize the inherited argument *Permeability* in *CO2_permeability*, using the panel 2.

To associate terminology to the new n-ary relation concept, the domain expert uses panel 4.

Finally, when the domain expert clicks on "OK" button to validate the *CO2_permeability_Relation* creation, DynarOnto automatically checks that the associated preconditions are respected (e.g. $\mathcal{CC}u_3$: *CO2_permeability_Relation* has at least two arguments). Let us notice that the interface eases the verification of some preconditions. For instance, when the domain expert chooses to specialize an argument, the plugin displays only the hierarchy of more specific concepts. Once the preconditions are checked, the plugin automatically applies the requested change and the set of additional mandatory changes, allowing violated $\mathcal{CC}$-constraints to be resolved.

### *4.2.2 Annotation of a data table through the use case*

After the update step of the TRANSMAT OTR presented in the previous section, the data table of Figure 6 can be annotated. The screenshot presented in Figure 8 corresponds to the selection of the relevant n-ary relation concepts in @Web annotation

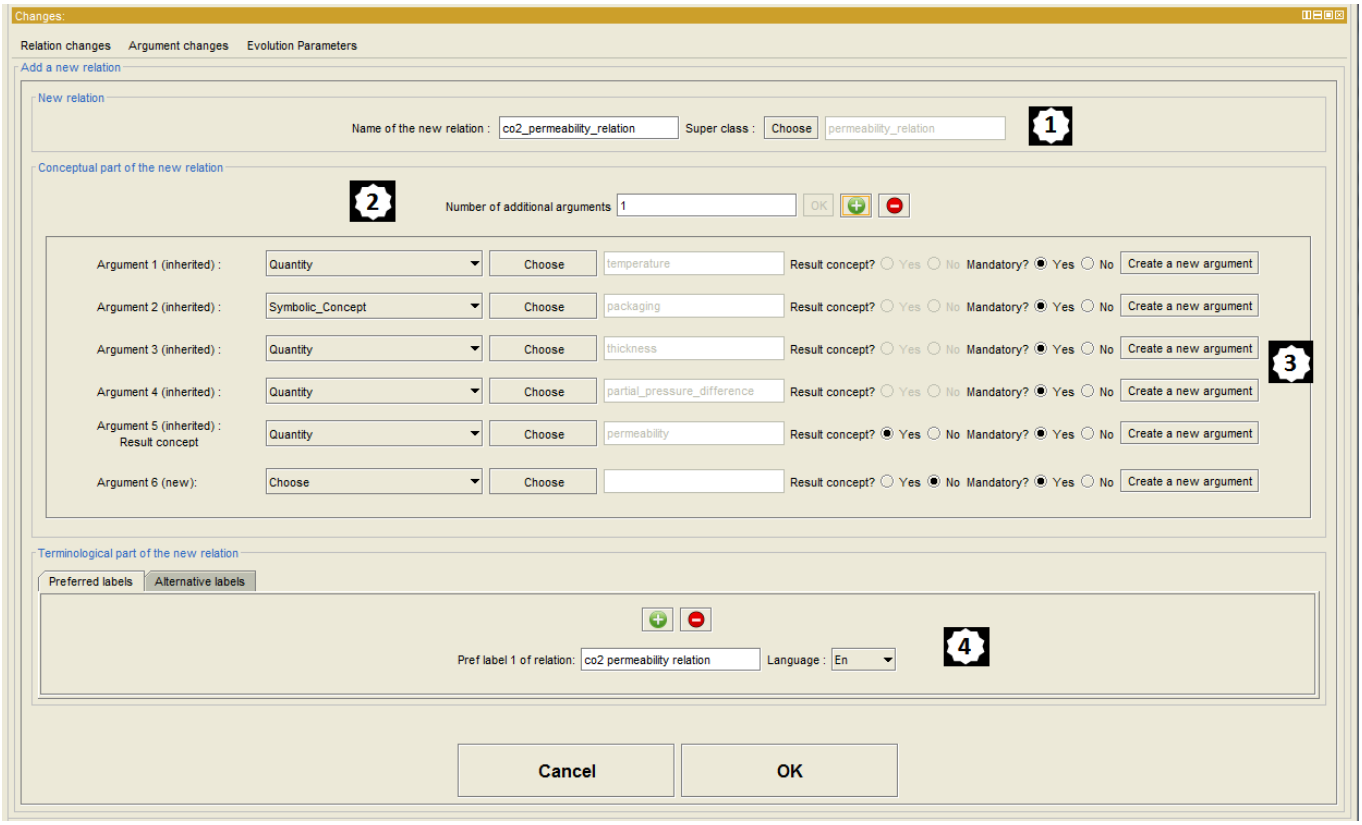


**Figure 7** Screen shot of the interface of DynarOnto to add a new n-ary relation.

module to annotate this data table, that are *O2_permeability_Relation* and *CO2_permeability_Relation*. The signature of both n-ary relation concepts (i.e. their set of arguments) are visualized in a table, one signature per row. This will guide the domain expert in his/her annotation task, allowing him/her not to forget to fulfill arguments of the selected n-ary relation concepts which guarantee the validity and reusability of data. Figure 9 shows the original data table and its annotation using @Web. Let us notice that the missing arguments packaging and thickness in the original data table have been manually extracted from the corresponding scientific textual document by the domain expert in order to complement the data table annotation.

### *4.3 DynarOnto Evaluation*

DynArOnto was evaluated in an incremental way by ten users with different backgrounds from domain experts to ontology engineers. This choice of different users with different backgrounds was guided by our intent to check that DynArOnto is also relevant for ontology engineers that are experts in ontology management using, in particular, Protégé. Three evaluation sessions were organized, DynArOnto interface being improved after each session. Users were asked to apply three changes (add n-ary relations, update n-ary relations and delete arguments). A quiz was created to collect their evaluations. Most participants declare that using DynArOnto to manage the evolution is easier than using Protégé. It is also reflected in the time spent by users to apply the changes: the add relation change (resp. update/suppress change) was done with Protégé in an average of 42 minutes (resp. 11 minutes). Using DynArOnto, user time was reduced by 20% (resp. 27%). Finally, DynArOnto helped the users to manage the evolution of an OTR dedicated to the annotation of experimental data while guaranteeing its $\mathcal{CC}$-coherence, which was not the case in Protégé where the users were not guided and make some errors (e.g. incorrect use of OWL restrictions used to link arguments to n-ary relations, forgetting to associate terms to concepts).

## 5 Conclusion

We have presented in this paper an ontology evolution activity for a data integration system called ONDINE which proposes a complete workflow to annotate and query experimental data extracted from scientific documents. The core element of the ONDINE system is an Ontological and Terminological Ressource (OTR), called naRyQ, designed for experimental data representation. The goal of our ontology evolution activity was to better manage the evolution of the OTR naRyQ when a new domain was considered. The ONDINE system described in Buche et al. (2013) was used in 6 different domains: microbial risk, chemical risk, aeronautics, matter transfer, biorefinary, and durum wheat. These six domain ontologies were build by

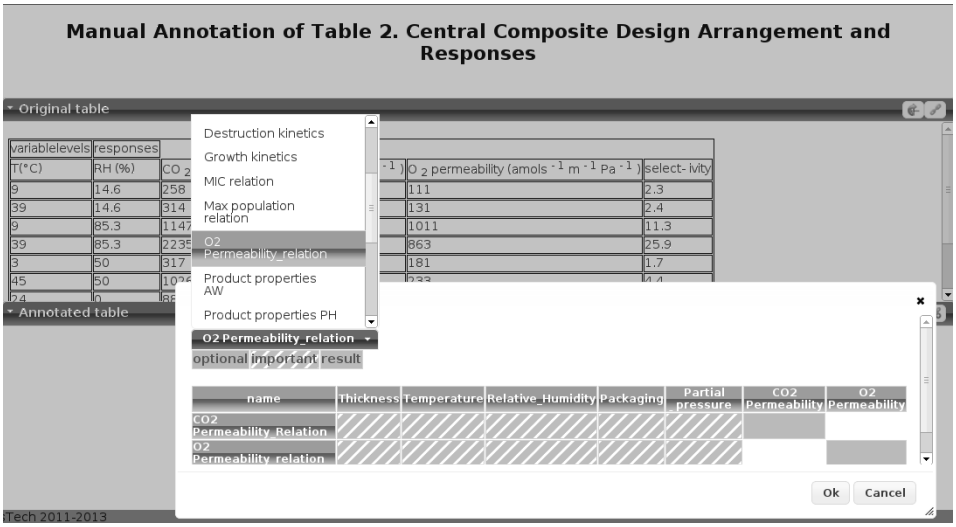


**Figure 8** Selection of the relevant n-ary relation concepts in @Web annotation module to annotate the data table that appears at the top of the screen

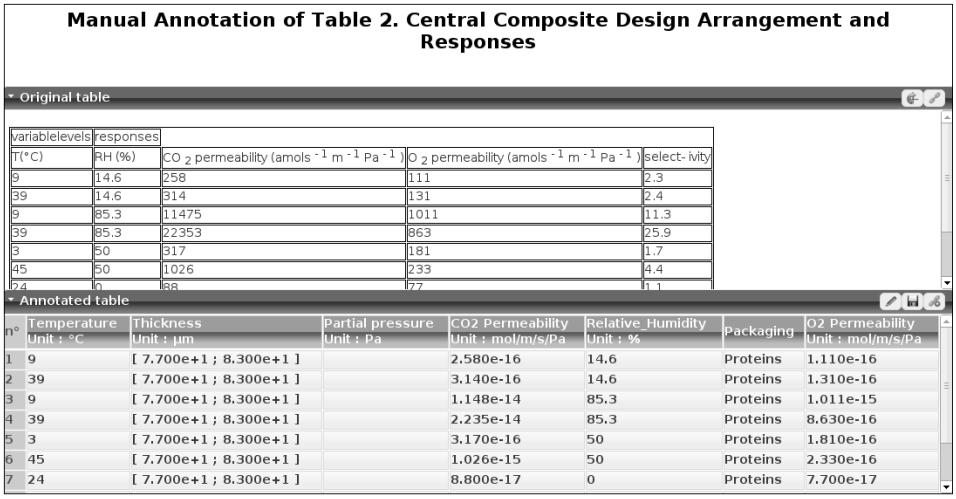


**Figure 9** The annotation of a data table with @Web annotation module

reengineering existing ressources using the Scenario 6 from the NeON methodology (see Suárez-Figueroa et al. (2012a)). During those building processes, ontology engineers identified the need to assist the domain experts in their evolution activity in order to avoid errors and to reduce time.

We therefore propose in this paper an *a priori* ontology evolution activity designed to assist the domain experts in the management of the OTR evolution while preserving its coherence.
The first step consists in identifying all the necessary constraints (i.e. structural, logical and user-defined) corresponding to the changes identified as useful in our application context by the domain experts. 15 structural constraints, 6 logical constraints and 29 user-defined constraints were defined. We mainly studied the user-defined constraints that are specific to the modeled task and therefore original according to the literature on ontology evolution activity. They deal with the main characteristics of our OTR that are a SKOS representation of its terminological component and its inter dependent concepts to manage quantitative data. Let us notice that for their completeness, we consider the following definition: a set of requirements can be considered complete if, and only if, users (i.e. domain experts) review the requirements and confirm that they are not aware of additional requirements (inspired by Wieringa (1996) and used also in Suárez-Figueroa and Gómez-Pérez (2012)).
The second step of our ontology evolution activity consists in applying the changes while preserving the coherence of naRyQ, using a set of kit of changes. A kit of changes allows the coherence of naRyQ to be preserved *a priori* by checking a set of preconditions, by applying a set of additional changes and/or by using evolution strategies. 63 kits of changes were defined which confirms the complexity of our evolution activity on an ontology with a rich structure.

Our ontology evolution activity was implemented as a Protégé plug-in, called DynarOnto, that assists

domain experts. It was tested and evaluated for the evolution of an OTR applied to the domain of matter transfer. The results of the evaluation indicate a gain in time, nevertheless the real significant benefit was in the assistance given during the evolution process allowing the domain experts to make less errors.

Further work will be to fully implement and to integrate our evolution activity in the @Web software using DynarOnto implementation choices. More research should be done on the topic concerning how to propagate changes to all the ontology related artifacts: individuals, mapping, applications, metadata.

## References


Buche, P., Dibie-Barthélemy, J., Ibanescu, L., and Soler, L. (2013). Fuzzy web data tables integration guided by an ontological and terminological resource. *IEEE Trans. Knowl. Data Eng.*, 25(4):805–819.

Destercke, S., Buche, P., and Charnomordic, B. (2013). Evaluating data reliability: An evidential answer with application to a web-enabled data warehouse. *IEEE Trans. Knowl. Data Eng.*, 25(1):92–105.

Djedidi, R. (2009). *Approche d'évolution d'ontologie guidée par des patrons de gestion de changement*. PhD thesis, Université Paris-Sud XI.

Guarino, N., Oberle, D., and Staab, S. (2009). What is an ontology? In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 1–17. Springer Berlin Heidelberg.

Haase, P. and Stojanovic, L. (2005). Consistent evolution of OWL ontologies. In *The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*, pages 182–197.

Jaziri, W., Sassi, N., and Gargouri, F. (2010). Approach and tool to evolve ontology and maintain its coherence. *Int. J. Metadata Semant. Ontologies*, 5(2):151–166.

Khattak, A. M., Latif, K., and Lee, S. (2013). Change management in evolving web ontologies. *Knowledge-Based Systems*, 37(0):1 – 18.

Mahfoudh, M., Forestier, G., Thiry, L., and Hassenforder, M. (2015). Algebraic graph transformations for formalizing ontology changes and evolving ontologies. *Knowl.-Based Syst.*, 73:212–226.

Noy, N., Rector, A., Hayes, P., and Welty, C. (2006). Defining n-ary relations on the semantic web. W3C working group note. http://www.w3.org/TR/swbp-n-aryRelations.

Palma, R., Zablith, F., Haase, P., and Corcho, Ó. (2012). Ontology evolution. In Suárez-Figueroa et al. (2012b), pages 235–255.

Stojanovic, L. (2004). *Methods and Tools for Ontology Evolution*. PhD thesis, University of Karlsruhe, Germany.

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197.

Suárez-Figueroa, M. C. and Gómez-Pérez, A. (2012). Ontology requirements specification. In Suárez-Figueroa et al. (2012b), pages 93–106.

Suárez-Figueroa, M. C., Gómez-Pérez, A., and Fernández-López, M. (2012a). The neon methodology for ontology engineering. In Suárez-Figueroa et al. (2012b), pages 9–34.

Suárez-Figueroa, M. C., Gómez-Pérez, A., Motta, E., and Gangemi, A., editors (2012b). *Ontology Engineering in a Networked World*. Springer.

Tissaoui, A., Aussenac-Gilles, N., Hernandez, N., and Laublet, P. (2011). EvOnto - joint evolution of ontologies and semantic annotations. In *KEOD 2011*, pages 226–231.

Touhami, R. (2015). Definition of naRyQ OTR CC-Constraints and kits of changes. Technical report. http://umr-iate.cirad.fr/content/download/4847/36913/version/1/file/TechnicalReport.pdf.

Touhami, R., Buche, P., Dibie-Barthélemy, J., and Ibanescu, L. (2015). Ontology evolution for experimental data in food. In Garoufallou, E., Hartley, R. J., and Gaitanou, P., editors, *Metadata and Semantics Research - 9th Research Conference, MTSR 2015, Manchester, UK, September 9-11, 2015, Proceedings*, volume 544 of *Communications in Computer and Information Science*, pages 393–404. Springer.

Touhami, R., Buche, P., Dibie-Barthélemy, J., and Ibănescu, L. (2011). An Ontological and Terminological Resource for n-ary Relation Annotation in Web Data Tables. In *OTM 2011 (2)*, volume 7045 of *LNCS*, pages 662–679. Springer.

Wieringa, R. J. (1996). *Requirements Engineering: Frameworks for Understanding*. John Wiley & Sons, Inc., New York, NY, USA.

Zablith, F., Antoniou, G., d'Aquin, M., Flouris, G., Kondylakis, H., Motta, E., Plexousakis, D., and Sabou, M. (2015). Ontology evolution: a process-centric survey. *Knowledge Eng. Review*, 30(1):45–75.