Integrating Graphical Support with Reasoning in a Methodology for Ontology Evolution

Germán Braun* and **Laura Cecchi**
Universidad Nacional del Comahue
*Consejo Nacional de Investigaciones
Científicas y Técnicas (CONICET)

Pablo Fillottrani
Universidad Nacional del Sur
Comisión de Investigaciones Científicas
de la provincia de Buenos Aires (CIC)

Abstract

Ontology development and maintenance are complex tasks, so automatic tools are essential for a successful integration between the modeller's intention and the formal semantics in an ontology. In this paper we present a methodology for ontology evolution specifically designed for being used in ontology design tools. It exploits the ontology graphical representation, which is closer to the modeller's intention, and selected user queries, that reflect the formal semantics of the ontology, both generated within the context of a given user experience in a tool. In addition, a set of modular extension rules is supplied to capture ontology elements by abstracting the user from the whole model without losing consistency. The main benefit of this approach is to offer a trade-off between the inherent rigidity to the logic systems and the intuitive characteristics of the ontological modelling.

1 Introduction

Ontology development and maintenance are complex tasks. Domain experts capture the knowledge in the universe of discourse but they have a limited understanding of the semantics of ontology representation languages. Moreover, a good comprehension of the implicit knowledge in a middle-size ontology formalisation is difficult even for IT experts. Thus, automatic tools are essential for a successful integration between the modeller's intention and the formal semantics in an ontology.

Two of the most important features for any ontology development tool are support for graphical representation, and a consistent integration with a back-end reasoner for helping in the management of implicit knowledge. The first one is a primary source for abstraction, and provides an intuitive navigation of the ontology. Usually the graphical notation of a tool is based on EER (Gogolla 1994), UML (Booch, Rumbaugh, and Jacobson 2005), or ORM (Halpin and Morgan 2008), and therefore promotes usability of the tool and interoperability of the generated ontology. The second one helps in exploring, composing, and checking the ontology by making explicit to the user the overall semantic of the ontology. It is generally done by first translating the graphical ontology into a some formal representation, and later handling it to an external reasoner that can be then queried about ontology properties.

In this paper we present *Query-driven Ontology Development (QDOD)*, a methodology for ontology evolution specifically designed for being used in tools supporting these two features. *QDOD* takes advantage of the ontology graphical representation, which is closer to the modeller's intention, and of selected user queries, that reflect the formal semantics of the ontology, both generated within the context of a given user experience. The benefit of this approach is to offer a trade-off between the inherent rigidity to the logic systems and the intuitive characteristics of the ontological modelling. The objective is to decrease the complexity of the evolution process and the cognitive load for the user by reducing the number of possible ontology extensions.

QDOD approach starts by identifying possible new graphical elements from users queries. It continues suggesting further evolutions of the ontology by employing a built-in set of extension rules and complete logical reasoning to analyse the properties of the proposals. Finally, the user decides to accept or reject these suggestions. Information extracted from queries provided by users is key for understanding their needs and a way for managing the complexity during modelling. The data from these queries can result in an evolution of the model due to changes in expert perceptions about the domain.

QDOD focuses on graphical evolution of an ontology schema (intentional knowledge, TBox) by increasing the ontology alphabet, adding possible missing concepts, roles and axioms to the ontology. Concepts and roles can be extracted from user queries to extend the ontology. Axioms such as subsumption, equivalence and disjointness can be inferred by the back-end reasoner or by the extension rules.

In order to show that the *QDOD* aligns to the above principles of ontology design tools, we also introduce a theoretical framework for consistent mappings between graphical representation of ontologies The framework states that graphical representation of ontologies should be preserved through automatic transformations, giving to their explicit elements a more important rational status than to the implicit ones. We claim this framework is a requisite for any tool-based ontology transformation process, like *QDOD* .

The paper is structured as follows. Section 2 describes motivations of the work. Section 3 introduces a theoretical framework that any methodology to be integrated to a tool should follow and shows how *QDOD* satisfies this framework. Section 4 presents the *QDOD* methodology, a description of the rules it uses and the algorithm to generate ontology extensions. A case of study is described in section 5. The approach is compared with some related works in section 6, and section 7 contains final conclusions and future works.

2 Motivation

The main motivation of the work is to close the gap between domain experts, its understanding about ontologies, and the modelling tools to satisfy the ontology reusability challenge. The usage-based evolution is a partial solution for this problem but it also requires of graphical tools together with inference services for facilitating the ontology maintenance and reuse.

Despite most popular tools provide support of both graphical representation and reasoning services, they are not graphical-centric tools and evolution is not endorsed. Protégé (Knublauch et al. 2004) and NeOn (Hasse et al. 2008) provide graphical support but only Protégé offers a partial integration with automatic reasoning. The evolution support is mainly manual because changes should be committed by users. In NeOn toolkit, reasoning is enabled and evolution is supported by a non-graphical plug-in Evolva (Zablith 2009). ICOM (Fillottrani, Franconi, and Tessaris 2012) and OntoUML tool (Guizzardi and Wagner 2012) do provide a comprehensive graphical interface. However, the OntoUML reasoning support, which is based in Alloy (Jackson 2002), is not integrated to its graphical language. These features are available in ICOM which uses $\mathcal{ALCQI}$ (Tobies 2001) as its base logic, but its ontology evolution support is limited. The integration of graphical languages and reasoning is still incipient as it requires development environments with these capabilities. These systems are needed to give support to the evolution process as well as cover all the ontology engineering tasks.

Our proposal is based on a back-end reasoning system that provides the methodology to the complete set of extensional and intentional elements in the ontology, and a set of rules which guide the extension mechanism. Users will see the ontology graphically completed and evolved with all the deductions, and expressed in the graphical language itself. This rules-based approach can be considered as complementary to the ontology design patterns (Gangemi and Presutti 2009) since both provide a way to reduce the complexity of designing and understanding ontologies. Furthermore, our alternative offers flexibility and a fine-grained level where the focus is on a subset of graphical elements to analyse and thus introduce new ones, in contrast to design patterns which are not orientated towards ontology evolution. Similar to the Gangemi's approach, the rules allow to describe views where the evolution suggestions are consistent.

Finally, our motivation also arises from other field as the Software Engineering and, in particular, the Test-Driven Development (TDD) (Beck 2002). TDD is a programming technique for building software that guides the development by writing failing tests and refactoring. The TDD philosophy is extrapolated into ontology engineering. However, unlike TDD, *QDOD* also generates automatic suggestions that users can accept or reject to continue refactoring.

3 Theoretical Model

In order to integrate reasoning to graphical environments, we present a theoretical model that is to be independent of any Description Logic (DL) (Baader et al. 2003) and graphical methodology. The main advantage of this model is to coordinate the relationship between the ontology graphical representation and its formal description generated by the tool. Indeed, it has been conceived as cross-methodological approach which arises from the need to define baselines and thus enable the development of other methodologies based on this framework.

Definition 1 (Graphical Elements)**.** *x is a* graphical element *if x is a concept, an association, a role, an is-a relationship or an axiom with graphical representation.*

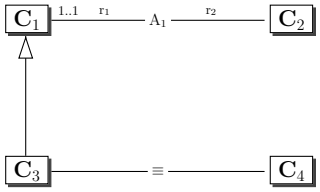


Figure 1: Ontology including the graphical elements detailed in the example 1.

Example 1. *The ontology in Fig. 1 contains the following graphical elements. C_1, C_2, C_3 and C_4 are classes, A_1 an association, and r_1, r_2 are roles with cardinality* $1..1$ *and* $0..N$*, respectively. In addition, the model includes an IsA link between C_1 and C_3, and an equivalence axiom between C_3 and C_4.*

The use of graphical elements for modelling purposes is desirable for users but may lead to formal consequences that could not be easily recognised by designers in complex models. For example, the inconsistency of the model or some of its elements. Hence, the integration of reasoning in tools is needed in order to identify these properties and notify the user about them. This is achieved by representing the semantics of each graphical element and the whole graphical ontology in terms of DL, and then by rendering the reasoning results in graphical notation. This process is formalised in the following definitions.

Definition 2 (Element Graphical-Logic Mapping)**.** *Let Δ be a set of graphical elements and let $x \in \Delta$. An* element graphical-logic mapping *is a function Ψ from Δ to a decidable DL fragment, where $\Psi(x)$ is the logical representation of x in the target DL.*

Example 2. *Let us consider $\mathcal{ALCQI}$ as a target logic, a possible mapping Ψ for the graphical elements of the example 1 is shown in Table 1.*

Classes and associations (or n-ary relations in EER) are graphical elements that will be represented as concepts in DL, and roles will be represented as roles in DL.

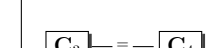
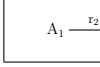
x	$\Psi(x)$	x	$\Psi(x)$
	C_1	A_1	A_1
	$C_3 \sqsubseteq C_1$		$C_3 \equiv C_4$
x	$\Psi(x)$		
	$C_1 \sqsubseteq (\leq 1\, r_1^-.A_1)$ $C_1 \sqsubseteq \exists r_1^-.A_1 \quad A_1 \sqsubseteq \exists r_1.C_1$ $C_1\_A_1\_min \sqsubseteq C_1 \sqcap (\geq 1\, r_1^-.A_1)$ $C_1\_A_1\_max \sqsubseteq C_1 \sqcap (\leq 1\, r_1^-.A_1)$		
	$A_1 \sqsubseteq \exists r_2.C_2$ $C_2\_A_1\_min \sqsubseteq C_2 \sqcap (\geq 1\, r_2^-.A_1)$ $C_2\_A_1\_max \sqsubseteq C_2 \sqcap (\leq 1\, r_2^-.A_1)$		

Table 1: Ψ function for the ontology depicted in Fig. 1.

Definition 3 (General Graphical-Logic Mapping)**.** *Let Δ be a set of graphical elements and let Ψ an element graphical-logic mapping. A* general graphical-logic mapping *is a function Θ from Δ to a decidable DL fragment, where $\Theta(\Delta) = \bigcup_{x \in \Delta} \Psi(x)$ in the target logic.*

Definition 4 (Graphical Ontology)**.** *Let Δ be a set of graphical elements and let Ψ be an element graphical-logic mapping. Ω is a* graphical ontology *generated from Δ if and only if $\Theta(\Delta)$ is an ontology in the target logic.*

We assume that there exists only one graphical ontology that can be generated from a set of graphical elements.

Example 3. *Considering the example 1 and supposing $\mathcal{ALCQI}$ as target logic, the ontology $\Theta(\Delta)$ where classes and associations are translated to DL concepts, is defined as follows.*

$$C_1 \sqsubseteq (\leq 1\, r_1^-.A_1) \qquad C_3 \equiv C_4$$
$$C_1 \sqsubseteq \exists r_1^-.A_1 \qquad C_1\_A_1\_min \sqsubseteq C_1 \sqcap (\geq 1\, r_1^-.A_1)$$
$$A_1 \sqsubseteq \exists r_1.C_1 \qquad C_1\_A_1\_max \sqsubseteq C_1 \sqcap (\leq 1\, r_1^-.A_1)$$
$$A_1 \sqsubseteq \exists r_2.C_2 \qquad C_2\_A_1\_min \sqsubseteq C_2 \sqcap (\geq 1\, r_2^-.A_1)$$
$$C_3 \sqsubseteq C_1 \qquad C_2\_A_1\_max \sqsubseteq C_2 \sqcap (\leq 1\, r_2^-.A_1)$$

Definition 5 (Consistent Graphical Ontology)**.** *Let Δ be a set of graphical elements and let Ψ be an element graphical-logic mapping. Ω is a* consistent graphical ontology *generated from Δ if and only if $\Theta(\Delta)$ admits a model in which at least one element has a non-empty extension in the target logic.*

In addition, we will introduce another desirable property of ontologies and thus also of graphical ones to complete the proposed formalisation.

Definition 6 (Consistent Graphical Element)**.** *Let Δ be a set of graphical elements and let Ψ be graphical-logic mapping, and let Ω be a graphical ontology with a mapping $\Theta(\Delta)$. An element $x \in \Delta$ is a* consistent graphical element *if and only if $\Theta(\Delta)$ admits a model in which the $\Psi(x)$ has a non-empty extension.*

Any methodology integrated in a graphical tool should ensure this mapping in every step of the process. Otherwise, possible mismatches between the ontology graphical and formal representations may occur.

4 Methodology

Query-driven Ontology Development (QDOD) is a methodology that generates from a consistent ontology Ω induced by a design tool from a set of graphical elements Δ, a possible extension Ω' that preserves the elements in Δ. The ontology Ω is supposed not to be complete, so evolution is required. The user has been previously inquired through the tool about properties of Ω, some of these queries may have led to add new elements in it.

QDOD process starts by taking one user query Q at a time and by extracting from it its graphical elements, so to start the evolution. The algorithm analyses the *answerability* of Q and it gets the first *reachable* graphical elements when Q is answerable. A query is *answerable* if and only if its graphical elements are already included in Ω. In contrast, the methodology will suggest the query elements to be added to the ontology when Q is not answerable, and will also generate reachable elements. A *reachable* element is a graphical element in Ω that is involved in the query Q or in user-accepted suggestions made by the tool.

```
SELECT ?WinterGame ?WinterSport
WHERE
 {?WinterGame ol:contains ?hasWinterSport .
  ?hasWinterSport ol:from ?WinterSport .
 }
```

Figure 2: SPARQL query for *Olympics* ontology.

Let us suppose the query Q, which is shown in Fig. 2, for the *Olympics* ontology, which will be explained in section 5. We identify $WinterGame$, $contains$, $hasWinterSport$, $from$ and $WinterSport$ as graphical elements. Thus, if the triple $hasWinterSport$, $from$ and $WinterSport$ does not belong to Ω then *QDOD* will suggest adding them to the ontology as concepts and roles, respectively, and the modeller will be required to accept or reject this suggestion. This behaviour is similar for other elements in Q. In this case, the reasoner will be invoked in checking consistency and inquiring about graphical elements in Ω.

The process continues looking for more evolution suggestions that can be generated by means of two ways. The first one is by inquiring to the back-end reasoner to check implicit axioms in Ω. Implicit axioms are those non-explicit deductions inferred by querying a reasoning system. The second one is by testing extension rules on reachable elements in order to generate new suggestions until the list of reachable elements becomes empty.

The general form of an extension rule $\mathcal{R}$ is the following:

$$\{RA_1, RA_2, ..., RA_n\} \Rrightarrow \{CR_1, CR_2, ..., CR_m\}$$

such that RA_i and CR_j are TBox formulae from DL of the underlying reasoner, $1 \leq i \leq n$ and $1 \leq j \leq m$.

In order to apply a rule, we must check if its antecedent is satisfied in the logical representation of the graphical ontology Ω, denoted by $\Theta(\Delta)$, and if its consequences are consistent with this representation. Thus, a back-end reasoner should be inquired about the satisfiability of the following properties of Ω, $\{RA_1, RA_2, ..., RA_n\}$, and about consistency of every resulting ontology after applying the rule consequent. Each RC_i represents a different possible ontology extension and it is the user who is required to select which $RC_i, 1 \leq i \leq m$ will be applied, if any. Therefore, the back-end reasoner would be inquired to verify if $\Theta(\Delta) \cup RC_i$ is consistent for any $i, 1 \leq i \leq m$.

These rules analyse all the relationships of the elements and offer locality to recommend extensions by identifying recurrent design patterns. In each case, the user is required to accept or reject extensions. The rejected ones will not be proposed again during the algorithm execution.

Every suggestion in the model needs to be checked for consistency so that each graphical element and the complete Ω are mapped to the target description logic. The leverage of automatic reasoning is enabled by a semantic definition of each element of the models and their user queries. This logical support is required to define when extension rules can be applied, display the implications of the suggestions derived from these rules, and assert the new intentional knowledge in the underlying knowledge base.

The output of this methodology is a consistent graphical ontology Ω' evolved from user queries, a set of extension rules and reasoning support. The pseudo-code of the *QDOD* algorithm is shown in Algorithm 1.

Next, we formally define the notion of graphical extension and show the complexity analysis of the *QDOD* algorithm.

Definition 7 (Graphical Extension)**.** *Let Ω be a consistent graphical ontology generated from a set of graphical elements Δ. A graphical extension of Ω w.r.t. a set of queries $Q_1, ..., Q_n$ is an ontology Ω' generated from a set of graphical elements Δ' such that:*
1. $\Delta \subseteq \Delta'$,
2. Ω' is consistent,
3. $\forall x \in \Delta'$, x is consistent,
4. Q_i is answerable in Ω', for all $1 \leq i \leq n$.

Theorem 1. *The* QDOD *algorithm for graphical ontology evolution terminates, and it runs in $O(l \times (m + f(n))$, for a maximum number of suggestions m generated from l reachable graphical elements, n graphical elements of the knowledge base, and $f(n)$ is the time of a query to a decidable back-end reasoner.*

Proof. Let Ω be a consistent ontology generated from the set of graphical elements Δ and let Q be a query with the associated set of graphical elements Δ_Q.

1. (line 1) Ω is generated from a finite set of graphical elements Δ,
2. (line 5) The query Q is composed by a finite set of elements in Δ_Q,

Algorithm 1 Query-driven Ontology Development

```
1: input: an user query Q and a consistent ontology Ω represented by a finite graphical elements set Δ.
2: output: an evolved ontology Ω' with a finite graphical elements set Δ'
3: re ← ∅
4: Ω' ← Ω
5: extract concepts and roles and associations from Q
6: if Q is answerable then
7:     add to re all the reachable elements from Q
8: else
9:     inquire back-reasoner for suggesting elements of Q which do not belong to Ω
10:    if suggestions are accepted then
11:        commit suggestions to Ω'
12:        add to re all the reachable elements from Q
13:    end if
14: end if
15: inquire back-end reasoner for implicit axioms in Ω
16: if implicit axioms exist then
17:    suggest implicit axioms to be added to Ω
18:    for all accepted evolution suggestion do
19:        commit suggestions to Ω'
20:        add to re all the reachable elements involved in the evolution
21:    end for
22: end if
23: while (re <> ∅) do
24:    select a concept or an association e from re
25:    remove e from re
26:    for all extension rule R on e do
27:        inquire back-end reasoner about the R antecedent and consequent
28:        if R antecedent is satisfied and R consequent is consistent in Ω then
29:            suggest R consequent as possible consistent evolutions
30:        end if
31:    end for
32:    for all accepted evolution suggestion do
33:        commit suggestions to Ω'
34:        add to re all the reachable elements related to e
35:    end for
36:    if any suggestion has been accepted then
37:        inquire back-end reasoner for implicit axioms in Ω
38:        if implicit axioms exist then
39:            suggest implicit axioms to be added to Ω
40:            for all accepted evolution suggestion do
41:                commit suggestions to Ω'
42:                add to re all the reachable elements involved in the evolution
43:            end for
44:            for all rejected evolution suggestion do
45:                suggestion is not generated anymore
46:            end for
47:        end if
48:    end if
49: end while
    return Ω'
```

3. (line 9) The suggestions generated from Q are finite since the query Q is finite,
4. (line 15-16) Back-end reasoner is decidable by hypothesis. Moreover, the amount of queries to the reasoner and the suggestion generated are finite because Δ is finite,
5. (line 7, 12, 20, 34 and 42) The list of the reachable elements re is finite due to it only includes elements from Δ_Q or Δ,
6. (line 23-49) The `while` block does not add new concepts

nor associations. Only roles or cardinalities are added to Ω when a rule is applicable. The `while` terminates since the list of the reachable elements re becomes empty after applying each possible extension rules and their evolution suggestions. Moreover, the checking for implicit axioms terminates as indicated in item (4) and rejected suggestions are not generated anymore in order to avoid repetitions.

Hence, for any consistent Ω, query Q and sequence of extension rule application, the *QDOD* algorithm terminates.

Now we demonstrate the complexity of the *QDOD* algorithm. Consider us a graphical ontology Ω generated from a set of graphical elements Δ as input.

1. (line 5-14) The complexity for the `if` is $\mathcal{O}(m_1)$ where m_1 are the suggestions generated from the query Q,
2. (line 15-21) The time of a query to the back-end reasoner is $f(n)$, for a decidable reasoner and n graphical elements,
3. (line 23-49) The complexity of the `while` is:
   3.1. (line 24-35) $O(m_2)$ where m_2 are the suggestions generated from applying rules,
   3.2. (line 36-48) $f(n) + m_3$ for checking of implicit axioms, and m_3 suggestions.

Hence, if we consider l reachable graphical elements, the *QDOD* algorithm upper bound is:

$$\mathcal{O}(m_1 + f(n) + l \times (m_2 + f(n) + m_3)) = \mathcal{O}(l \times (m_2 + f(n) + m_3))$$

and if consider the total amount of suggestions $m = max(m_2, m_3)$,

$$\mathcal{O}(l \times m + f(n))$$

□

Supposing the reasoner solves efficiently the queries, and since the number and the size of rules is bounded in general then we can say *QDOD* can be safely added to any design tool. Also, we show that *QDOD* follows the theoretical model presented in section 3.

Theorem 2. *Every extension generated from a consistent user query Q by applying an extension rule is a graphical extension.*

Proof. Let Ω be a consistent graphical ontology generated from the set of graphical elements Δ and from the mapping $\Theta(\Delta)$. Let $Q_1, ..., Q_{n-1}$ be a set of answerable queries in Ω and Q_n a new user query with Δ_{Q_n} graphical elements. We will demonstrate that this extension is also a graphical extension by considering the query Q_n and the definition 7. The ontology Ω' is obtained from Ω and Q_n by applying an extension rule $\mathcal{R}$.

1. $\Delta \subseteq \Delta'$ due to no graphical element is removed from Δ when a rule is applied, for any rule $\mathcal{R}$,
2. Ω' generated from Δ' and $\Theta(\Delta')$ is consistent by extension rules applicability,
3. From (2), Ω' is consistent. Then, $\Theta(\Delta')$ is also consistent (by def. 5) iff is consistent $\Theta(\Delta') = \bigcup_{x \in \Delta'} \Psi(x)$ (by def. 3) iff $\Psi(x)$ is consistent iff x is consistent (by def. 6),
4. $Q_1, ..., Q_{n-1}$ are answerable queries in Ω. If $\Delta_{Q_n} \subseteq \Delta'$ then by definition Q_n is also answerable. By contrast, if $\Delta_{Q_n} \not\subseteq \Delta'$, then the missing elements of Q_n are added to Ω in order to start the evolution. Thus Δ_{Q_n} becomes a subset or equal to Δ'. Therefore, Q_n turns out to be answerable.

Hence, every extension generated by applying an extension rule is a graphical extension.

□

Extension Rules

We present some examples of the extension rules in order to understand their role in the *QDOD* methodology. We next show some of them, but the list is not exhaustive. The rule family is available in (GILIA 2015).

The following rule proposes an IsA relationship between entities.

$\sqsubseteq$-rule: $\{A_2 \sqsubseteq A_1,\ A_1 \sqsubseteq \exists r_1.C_1,\ A_2 \sqsubseteq \exists r_1.C_2\} \Rrightarrow \{C_2 \sqsubseteq C_1, C_2 \equiv C_1\}$

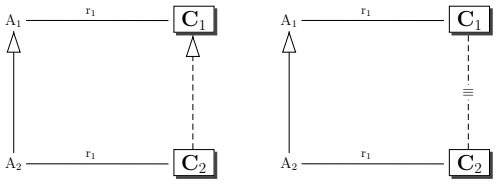


Figure 3: Graphical representation of $\sqsubseteq$-rule.

The $\sqsubseteq$-rule, which is graphically rendered as shown in Fig. 3, can be legitimised by analysing the involved classes and their relationships. By definition, if $A_2 \sqsubseteq A_1$ and $A_1 \sqsubseteq \exists r_1.C_1$ then $A_2 \sqsubseteq \exists r_1.C_1$. Moreover, if any explicit inequality exists in the ontology then we can suppose both roles r_1 in $A_1 \sqsubseteq \exists r_1.C_1$ and $A_2 \sqsubseteq \exists r_1.C_2$ represent the same role and they are identically defined. From these arguments, we could propose the rule consequent $\{C_2 \sqsubseteq C_1, C_2 \equiv C_1\}$ as possible extensions. Neither of these consequences is entailed by the ontology but they are considered anyway since their intuitive characteristics as expressed in the motivation of the present work.

For example, let us suppose the following partial and textual ontology Ω:

$$hasWinterSport \sqsubseteq hasSport$$
$$hasSport \sqsubseteq \exists from.OlympicSport$$
$$hasWinterSport \sqsubseteq \exists from.WinterSport$$

If we match these ontology elements to the $\sqsubseteq$-rule, we can obtain a consistent ontology Ω', which defines a new type for the $OlympicSport$ element by means of $WinterSport \sqsubseteq OlympicSport$, or a Ω'' where $WinterSport \equiv OlympicSport$. For this purpose, *QDOD* algorithm should inquire about properties $hasWinterSport \sqsubseteq hasSport$, $hasSport \sqsubseteq \exists from.OlympicSport$ and $hasWinterSport \sqsubseteq \exists from.WinterSport$, and about consistency of resulting ontology after applying $WinterSport \sqsubseteq OlympicSport$ and $WinterSport \equiv OlympicSport$. In fact, both suggestions of the example are consistent so that any of them could be a possible extension.

Other examples are the c-rule and r-rule which recommend cardinality and roles suggestions, respectively. Below

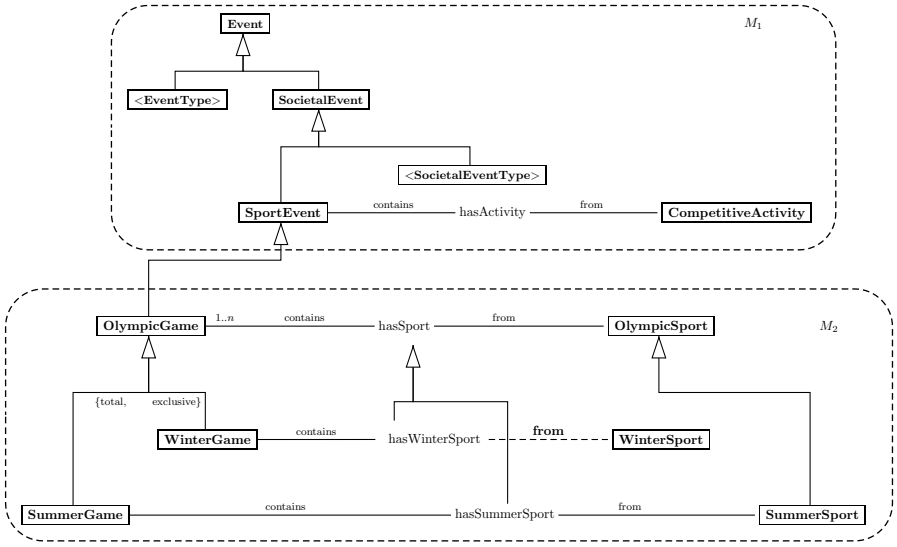


Figure 4: Dashed line indicates the query elements suggestion.

they are only presented due to space limitation, but their explanation can be found in (GILIA 2015).

c-**rule**: $\{A_1 \sqsubseteq \exists r_1.C_1, C_2 \sqsubseteq C_1, A_2 \sqsubseteq A_1, A_2 \sqsubseteq \exists r_1.C_2, C_1 \sqsubseteq \exists r_1^-.A_1\} \Rightarrow \{C_2 \sqsubseteq \exists r_1^-.A_2\}$

r-**rule**: $\{C_2 \sqsubseteq C_1, A_1 \sqsubseteq \exists r_1.C_1, A_2 \sqsubseteq \exists r_2.C_2, A_2 \sqsubseteq A_1\} \Rightarrow \{r_2 \sqsubseteq r_1\}$

One of the advantage of these rules is that can be included, modified and removed without affecting the host methodology. This approach allows to discover knowledge which could not be inferred from a logical point of view, but it could be suggested to users as a modelling option weaker than the logic one.

5 Case of Study

We will illustrate the *QDOD* methodology with an *Olympics* ontology graphical representation, which is partially integrated to an *Event* ontology. Let us consider the non-answerable SPARQL query Q shown in section 4, which inquires about every winter game and their associated sports. Fig. 4 shows the ontology after committing the suggestions (drawn in dashed line) extracted from Q.

The methodology considers Q and it evaluates if it is answerable. As Q is not answerable (Algorithm 1:line 9) since the role $from$ between the $hasWinterSport$ association and the $WinterSport$ class are missing in the graphical ontology, it is suggested and added to the ontology after the user accepts it. The classes $WinterGame$ and $WinterSport$ and the association $hasWinterSport$ involved in the query are added to the list of reachable elements (re).

The algorithm invokes the back-end reasoner for deducing implicit axioms (Algorithm 1:line 15). The system shows $hasSummerSport \sqsubseteq \neg hasWinterSport$ as an evolution suggestion. In our case, the user rejects it because it is not his intended model.

Next, it gets the first reachable element $WinterGame$ and applies it to the extension rules (Algorithm 1:line 26). The c-rule suggests $WinterGame \sqsubseteq \exists contains^-.hasWinterSport$, which it is accepted by the user and $WinterGame$ is again added to re. As no consequence appears after this extension (Algorithm 1:line 37), a new element $hasWinterSport$ is selected from re. The $\sqsubseteq$-rule suggests $WinterSport \sqsubseteq OlympicSport$ and $WinterSport \equiv OlympicSport$. If the user considers that any of the suggestions is an evolution, then selects one of them. In the example, the user selects $WinterSport \sqsubseteq OlympicSport$ (Fig. 5). The system adds $OlympicSport$ to re and inquires reasoner about possible implications of the evolved model. As no other new evolution suggestion appears after checking the rules on the remaining reachable elements, the algorithm terminates.

Even though the model is partially integrated to another one, *QDOD* allows to work on graphical evolution by isolating the subset of ontological elements identified by the user according to the intended model. *QDOD* offers two abstraction levels for modellers. The first one is provided by means of user queries to place emphasis on one of the integrated models and the involved concepts. The second one is supplied by extension rules which focus on elements previously identified and their relationships and contexts. Furthermore, *QDOD* keeps consistency in the whole model, which is supported by the back-end reasoning system, after evolving.

6 Related Work

While the frameworks described by Zablith et .al in (Zablith et al. 2014) try to cover all the evolution process, we are interested in approaches that detect needs of evolution from user behaviours because they represent a *genuine* and authentic real source of evolutions. The literature presents approaches involving application usage (Alani, Harris, and

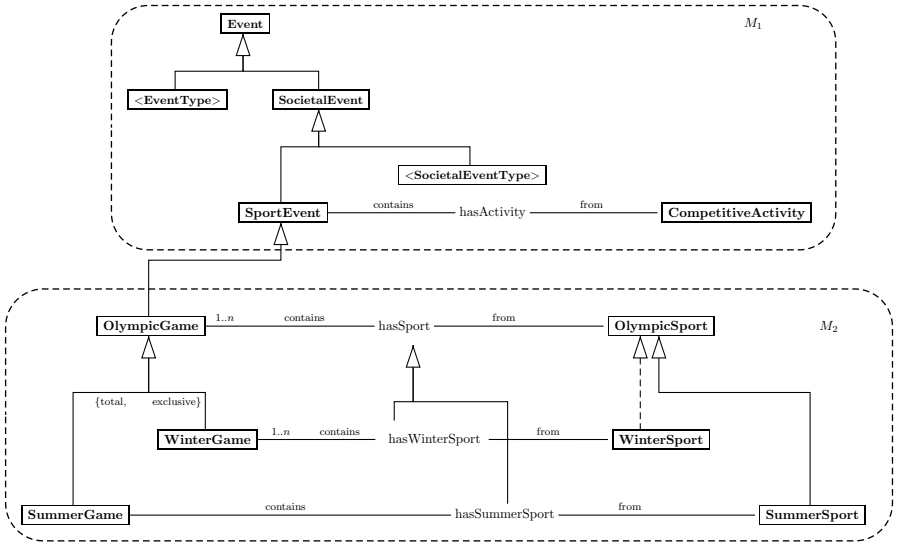


Figure 5: Evolved ontology with last subsumption suggestion in dashed line.

O'Neil 2006), ontology change log (Javed, Abgaz, and Pahl 2011), user feedback (Luczak-Rösch 2009), log of queries, search history (Bloehdorn et al. 2006) and context and domain information (Guelfi, Pruski, and Reynaud 2011). Nevertheless, evolution in these approaches is detected after the ontology has been designed. Although we do not cover the complete ontology evolution process, our work proposes to start filling this gap by supporting graphical evolution from user queries in a predictive way, at modelling time.

Similar to our approach, Guizzardi in (Guizzardi, das Graças, and Guizzardi 2011) proposes to use rules in pattern-based design. However, the automatic reasoning is missing and users queries are not considered in evolution. Baader et al. in (Baader et al. 2007) discusses a similar approach which considers to complete a DL knowledge base by using expert information and the DL base itself by means of the technique of formal concept analysis. Neither graphical support nor end-user queries is involved in this work. The query-driven extensions are also partially tried by Cuenca Grau et al. in (Grau et al. 2013) although in a more conservative context and without involving reasoning or graphical integration.

With respect to graphical tools, ICOM (Fillottrani, Franconi, and Tessaris 2012) and OntoUML (Guizzardi and Wagner 2012) are graphical-based tools but the evolution support is limited, while only ICOM integrates both graphical languages and automatic reasoning. Protégé (Knublauch et al. 2004) and TopBraid Composer (TopQuadrant 2011) do allow this integration but inferences are only restricted to is-a hierarchies. Their evolution support is partial: ontology differences (Protégé) and versioning and collaboration (TopBraid Composer). NeOn toolkit (Hasse et al. 2008), Kaon2 (Motik and Studer 2005) and SWOOP (Kalyanput et al. 2005) provide access to reasoners but they are not graphical-based tools. In evolution respect, NeOn incorporates a framework to support it from external sources. Kaon2 and SWOOP simply offer operations as redo and undo (Kaon2) or imports and versioning (SWOOP). Other tools as GrOWL (Krivov, Williams, and Villa 2007), OWLGrEd (Cerans et al. 2012), Graphol (Console et al.), "model outline" framework (do Amaral 2010) and VOWL (Lohmann, Negru, and Bold 2014) are also graphical tools. They define a graphical syntax and semantics to provide users with a visual representation of their models avoiding any complex textual syntax. The reasoning support is not provided in any of these tools as ICOM does and the ontology evolution is not properly supported.

These related works show that evolution from a graphical point of view is missing. As far as we know, our approach is the only one that proposes to integrate both graphical and reasoning support in an evolution methodology.

7 Conclusions and Future Works

This work introduces the *QDOD* methodology, an approach for ontology evolution based on queries and integrated within a graphical design tool. Its main advantage is to provide a trade-off between the logical rigidity from formal representations and the intuitive characteristics inherent to the ontology modelling. The methodology works by proposing ontology extensions based both on previous user queries and a set of modular extension rules. We have developed an algorithm to calculate evolutions and we have demonstrated complexity and computability properties. Moreover, we have defined a theoretical framework that supports this methodology and that is independent from *QDOD*. As far as we know, our approach is the first that integrates both graphical and reasoning support in an ontology evolution methodology.

As future works, we propose to implement the integration of *QDOD* methodology into an existing tool and evalu-

ate our proposal. This evaluation will be done by means of two techniques such as a behaviour-based one, which will allow us to register the number of suggestions accepted by user, and an opinion-based one, which will enable us to elicit users opinions about the use of the methodology (Gediga and Hamborg 2001).

We plan to provide support to user-defined rules and we are also working on extension rules involving instances so that we could provide evolution at extensional level (ABox) and possibly use it to enhance the intentional knowledge. Furthermore, new methodologies to complement *QDOD* could be developed by following the proposed theoretical framework.

References

Alani, H.; Harris, S.; and O'Neil, B. 2006. Winnowing ontologies based on application use. In *Proceeding of the ESWC'06*.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press.

Baader, F.; Ganter, B.; Sertkaya, B.; and Sattler, U. 2007. Completing description logic knowledge bases using formal concept analysis. In *In Proc. of IJCAI 2007*. AAAI Press.

Beck, K. 2002. *Test Driven Development: By Example*. Addison-Wesley Longman Publishing Co., Inc.

Bloehdorn, S.; Haase, P.; Sure, Y.; and Völker, J. 2006. Ontology evolution. In *Semantic Web Technologies — Trends and Research in Ontology-Based Systems*. chapter 4.

Booch, G.; Rumbaugh, J.; and Jacobson, I. 2005. *Unified Modeling Language User Guide, The (2Nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional.

Cerans, K.; Ovcinnikova, J.; Liepins, R.; and Sprogis, A. 2012. Advanced owl 2.0 ontology visualization in owlgred. In *DB&IS*, Frontiers in Artificial Intelligence and Applications. IOS Press.

Console, M.; Lembo, D.; Santarelli, V.; and Savo, D. F. Graphol: Ontology representation through diagrams. In *Informal Proceedings of the 27th International DL'14*.

do Amaral, F. N. 2010. Usability of a visual language for dl concept descriptions. In *RR*, Lecture Notes in Computer Science. Springer.

Fillottrani, P.; Franconi, E.; and Tessaris, S. 2012. The icom 3.0 intelligent conceptual modelling tool and methodology. *Semantic Web*.

Gangemi, A., and Presutti, V. 2009. Ontology design patterns. In *Handbook of Ontologies*. 2nd edition.

Gediga, G., and Hamborg, K.-C. 2001. Evaluation of software systems. *Encyclopedia of Computer Science and Technology*.

GILIA. 2015. Integrating graphical support with reasoning in a methodology for ontology evolution. Technical report. available at `http://tinyurl.com/nm4nos2` from Jun 7, 2015.

Gogolla, M. 1994. *Extended Entity-Relationship Model: Fundamentals and Pragmatics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Grau, B. C.; Giese, M.; Horrocks, I.; Hubauer, T.; Jiménez-Ruiz, E.; Kharlamov, E.; Schmidt, M.; Soylu, A.; and Zheleznyakov, D. 2013. Towards query formulation, query-driven ontology extensions in obda systems. In *OWLED*, CEUR Workshop Proceedings. CEUR-WS.org.

Guelfi, N.; Pruski, C.; and Reynaud, C. 2011. Adaptive ontology-based web information retrieval: The target framework. *Int. J. Web Portals* (3).

Guizzardi, G., and Wagner, G. 2012. Conceptual simulation modeling with onto-uml. In *Proceedings of the WSC'12*.

Guizzardi, G.; das Graças, A.; and Guizzardi, R. 2011. Design patterns and inductive modeling rules to support the construction of ontologically well-founded conceptual models in ontouml. In *CAiSE Workshops*.

Halpin, T., and Morgan, T. 2008. *Information Modeling and Relational Databases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2 edition.

Hasse, P.; Lewen, H.; Studer, R.; and Erdmann, M. 2008. The NeOn Ontology Engineering Toolkit.

Jackson, D. 2002. Alloy: A lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.* 11(2).

Javed, M.; Abgaz, Y.; and Pahl, C. 2011. Graph-based discovery of ontology change patterns. In *ISWC Workshops: EvoDyn'11*.

Kalyanput, A.; Parsia, B.; Sirin, E.; Grau, B.; and Hendler, J. 2005. Swoop: A 'web' ontology editing browser. *Journal of Web Semantics*.

Knublauch, H.; Fergerson, R.; Noy, N.; and Musen, M. 2004. The protégé owl plugin: An open development environment for semantic web applications.

Krivov, S.; Williams, R.; and Villa, F. 2007. Growl: A tool for visualization and editing of owl ontologies. *J. Web Sem.*

Lohmann, S.; Negru, S.; and Bold, D. 2014. The protégévowl plugin: Ontology visualization for everyone. In *The Semantic Web: ESWC 2014 Satellite Events*.

Luczak-Rösch, M. 2009. Towards agile ontology maintenance. In *ISWC'09*, Lecture Notes in Computer Science.

Motik, B., and Studer, R. 2005. KAON2–A Scalable Reasoning Tool for the Semantic Web. In *Proceedings of the 2nd ESWC'05*.

Tobies, S. 2001. Complexity results and practical algorithms for logics in knowledge representation. *CoRR* cs.LO/0106031.

TopQuadrant. 2011. TopQuadrant — Products — TopBraid Composer.

Zablith, F.; Antoniou, G.; d'Aquin, M.; Flouris, G.; Kondylakis, H.; Motta, E.; Plexousakis, D.; and Sabou, M. 2014. Ontology evolution: a process-centric survey. *The Knowledge Engineering Review* FirstView.

Zablith, F. 2009. Evolva: A comprehensive approach to ontology evolution. In *Proceedings of the 6th ESWC'09*.