# Extension Rules for Ontology Evolution within a Conceptual Modelling Tool

Germán Braun[1,2] and Laura Cecchi[1]

[1] *Grupo de Investigación en Lenguajes e Inteligencia Artificial*
Departamento de Teoría de la Computación - Facultad de Informática
Universidad Nacional del Comahue
[2] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)*

**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. Nevertheless, tools need to provide a way to capture the intuitive structures inherent to the conceptual modelling and to focus on ontology elements currently being refactored by abstracting the user from the whole ontology without losing consistency. This can be done by means of a set of extension rules that identify elements from an ontology and suggest possible consistent evolutions. Rules guide the development of ontologies by taking source elements and refactoring. In this paper, we present a small catalogue of extension rules to cover these identified requirements and thus to be integrated into a tool for ontological modelling as built-in reasoning services. Each rule is defined and analysed by considering different theories of design patterns.

## 1 Introduction and Motivation

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. 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. Nevertheless, tools also need to provide a way to capture the modeller's intentions or the intuitive structures inherent to the conceptual modelling. The first ones are sources for abstractions and for exploring, composing and checking the ontology by making explicit to the user its overall semantic. The last one provides a way to focus on ontology elements currently being refactored by abstracting the user from the whole ontology maintaining consistency and giving support

to the ontological evolution. This can be done by means of a set of extension rules that identify elements from an ontology and suggest possible consistent evolutions. Rules guide the development of ontologies by taking source elements and refactoring together with the underlying reasoning services.

In order to offer rules-based support, some efforts have been undertaken. In particular, Guizzardi et al. [1] propose an automatic model-checking editor that takes advantage of a well-behaved and predefined set of design patterns. Unlike our approach, the modelling rules are derived from this set of patterns and are implemented by means of an interactive dialogue between the modeller and an automated tool running these rule sets. Interfaces with reasoning systems have not been considered in this work.

In this paper, we present a set of extension rules which work at intentional level (TBox) of an ontology. An extension rule is a Description Logic (DL) structure with an antecedent which must be satisfied in the model to be applied, and a consequent which contains the new knowledge to be asserted. Extension rules are to be used together with other artefacts as user queries and reasoning systems, where the former identifies relevant parts of an ontology and the latter inquires the model to check rules applicability, their possible consequences and side effects. A rule is applicable iff its antecedent is satisfied and its consequent maintains the consistency of the model. Our rules-based approach and the ontology design patterns [2] can be considered as complementary since both provide a foundation for modularity to maintain and reduce the complexity of designing and understanding ontologies. Our alternative offers flexibility and a fine-grained level where the focus is on a reduced set of graphical elements to analyse so as to introduce new ones, as opposed to design patterns which are not orientated towards ontology evolution since they involve more elements in their definitions than rules. Similar to the design patterns, the rules allow to describe views in which the evolution suggestions are consistent. As a consequence, they have the effect of changing only the ontology elements involved in the refactoring, which can be intra- or inter-ontology elements. In addition, rules present modularity as a key property so that they could be modified and extended without affecting the host methodology.

This rule-based approach is to be implemented on the methodology underlying to ICOM tool [3, 4, 5] extending the reasoning services provided by the tool. In this context, we will consider its graphical primitives and its translation to the underlying DL $\mathcal{ALCQI}$ as our initial development framework. ICOM is an advanced conceptual modelling tool, which allows the user to design multiple diagrams adopting as neutral language a hybrid of an EER and UML languages with inter- and intra-schema constraints. Complete logical reasoning is employed by the tool to verify the specification, infer implicit axioms, i.e. new intentional knowledge, devise stricter constraints and manifest any inconsistency. The leverage of automated reasoning to support the domain modelling is enabled by a precise semantic definition of all the elements of the class diagrams. ICOM graphical language can be represented in $\mathcal{ALCQI}$, although the graphical language cannot express inverse roles. $\mathcal{ALCHIQ}$ is necessary for the full

ICOM language, including non-graphical extensions of role hierarchies and the view language. Moreover, ICOM allows partial graphical evolution of an ontology schema (intentional knowledge) by adding axioms which are inferred through defined reasoning services. Users will see the ontology graphically completed and evolved with all the deductions and expressed in the graphical language itself.

This work is structured as follows. Section 2 details the set of rules with a brief explanation and an example of each one of them. Discussions and related works are presented in section 3. To conclude the paper, section 4 elaborates on final considerations and directions for future works.

## 2 Extension Rules

The objective of the extension rules is to reduce the search space of ontology elements, and thus decrease the complexity of designing and understanding ontologies. They also focus on identifying relevant parts of models, and facilitate ontology verification, maintenance and integration. This approach allows to discover knowledge which is not inferred from a logical point of view but it is suggested from an intuitive point of view. Rules can be inter-modules in order to reuse, combine and share modules, which are central issues in ontology engineering branches as modularity.

In comparison with ontology design patterns [2], extension rules work with elements of ontologies in a greater level of detail than patterns. While Gangemi's approach assumes that there exist problems that can be solved by applying common solutions and define small, task-oriented ontologies with explicit documentation, extension rules handle a limited set of ontology elements so as to introduce new ones by affecting the overall shape of the ontology and maintaining its consistency.

The aim is to integrate this set of rules to a graphical tool so that the graphical ontology and rules are mapped to the ICOM DL. This logical support enables defining when an extension rule can be applied, deducing implicit axioms to display the implications of the suggestions derived from extension rules and asserting the new intentional knowledge in the underlying knowledge base.

The general form of the rules is the following:

$$\{A_1, A_2, ..., A_n\} \Rrightarrow \{B_1, B_2, ..., B_m\}$$

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

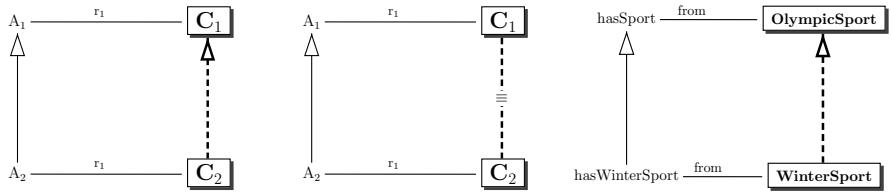
In order to apply a rule, we must check if its antecedent is satisfied in the logical representation of the graphical ontology $\Omega$, denoted by $\Theta$, and if its consequences are consistent with this representation. Thus, a back-end reasoner should be inquired about the satisfiability of the following properties of $\Omega$, $\{A_1, A_2, ..., A_n\}$, and about consistency of every resulting ontology after applying the rule consequent. Each $B_i$ represents a different possible ontology extension and it is the user who is required to select which $B_i, 1 \leq i \leq m$ will be applied,

if any. Therefore, the back-end reasoner would be inquired if $\Theta \cup B_i$ is consistent for any $i, 1 \leq i \leq m$

We present the extension rules catalogue by means of a brief explanation about each rule with some comparisons with other approaches such as design patterns or OntoUML [6], which is beyond the scope of the tool. Moreover, rules are formalised as DL formulae since they work on the ICOM underlying DL and a graphical description about how they are interpreted is depicted. Examples about how the rule is used are shown in the context of a domain ontology, where rule suggestions are depicted in dashed line. According to ICOM's graphical syntax, the primitives $C_i$ and $A_i$ are ICOM's classes and associations respectively, which will be translated as $\mathcal{ALCIQ}$ concepts, and $r_i$ are ICOM's roles which will be also considered $\mathcal{ALCIQ}$ roles.

**$\sqsubseteq$-rule # 1**

$$\{A_2 \sqsubseteq A_1,\ A_1 \sqsubseteq \exists r_1.C_1,\ A_2 \sqsubseteq \exists r_1.C_2\} \ \Rightarrow\ \{C_2 \sqsubseteq C_1,\ C_2 \equiv C_1\}$$




(a) (b) (c)

Figure 1: (a)(b) Graphical representation of $\sqsubseteq$-rule # 1. (c) $\sqsubseteq$-rule # 1 applied in *Olympics*

The aim of this rule is to identify missing IsA relationships between classes (and associations) and suggest them. It is intended to capture those scenarios whose concepts could be related by means of possibly the same role, and thus suggest a missing relationship as a subsumption axiom obtaining a more precise model or an equivalence axiom in which case the involved classes could be factored.

The rule, which is graphically rendered as shown in Fig. 1a and 1b, 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. Considering these arguments, the rule consequent $\{C_2 \sqsubseteq C_1, C_2 \equiv C_1\}$ could be proposed as possible extensions. Similar to Gangemi's design patterns [2], our rule can be a way to complete some patterns as *Classification* whose aim is to represent

the relations between concepts and entities and *Type of entities*, which allows to identify the type of any element of the knowledge base.

Another way of validating this rule is using the Guizzardi's definitions in [6] although in this case we should consider what object types are being represented. A type is rigid iff for every instance $x$ of that type, $x$ is necessarily an instance of that type. As these types can be related in a chain of taxonomic relations and if in the domain under modelling $C_1$ and $C_2$ are defined as rigid types then the *Subkind* pattern in [1] is completed by means of the rule suggestion $C_2 \sqsubseteq C_1$. Also, the suggested subsumption can be justified if $C_2$ is modelled as a phase type, which are anti-rigid types whose instances can move in and out of the extension of these types without affecting their identity. Consequently, the *Phase* Pattern could be also completed by $C_2 \sqsubseteq C_1$.

*Example 1.* Let us suppose the following partial and textual ontology $\Omega$:

$$hasSport \sqsubseteq \exists from.OlympicSport$$
$$OlympicSport\_hasSport\_min \sqsubseteq OlympicSport \sqcap (\geq 1\ from^{-}.hasSport)$$
$$OlympicSport\_hasSport\_max \sqsubseteq OlympicSport \sqcap (\leq 1\ from^{-}.hasSport)$$
$$hasWinterSport \sqsubseteq \exists from.WinterSport$$
$$WinterSport\_hasWinterSport\_min \sqsubseteq WinterSport \sqcap (\geq 1\ from^{-}.hasWinterSport)$$
$$WinterSport\_hasWinterSport\_max \sqsubseteq WinterSport \sqcap (\leq 1\ from^{-}.hasWinterSport)$$
$$hasWinterSport \sqsubseteq hasSport$$

If we match these ontology elements to the $\sqsubseteq$-rule # 1, we can obtain a consistent ontology $\Omega'$, which defines a new type for the *OlympicSport* by means of $WinterSport \sqsubseteq OlympicSport$ as depicted in Fig. 1c, or a $\Omega''$ where $WinterSport \equiv OlympicSport$.

**$\sqsubseteq$-rule # 2**

$$\{C_2 \sqcup C_3 \sqcup ... \sqcup C_n \sqsubseteq C_1,\ A_1 \sqsubseteq \exists r_1.C_2,...,\ A_1 \sqsubseteq \exists r_1.C_n\} \Rrightarrow \{A_1 \sqsubseteq \exists r_1.C_1\}$$

This rule, which is graphically represented as depicted in Fig. 2a, intends to identify redundant relationships between concepts within an IsA relationship. The same roles involving each subclass of an IsA could be defined by relating it to the superclass of the hierarchy.

Although it is not caught by the current definition of the rule, if a totality constraint is defined then the rule suggestion is a strong suggestion since each instance of the superclass belongs to one of its subclasses. Consequently the validation can be obtained from a logical point of view: if $\forall i, 2 \leq i \leq n, C_i \sqsubseteq C_1$ and $A_1 \sqsubseteq \exists r_1.C_i$ then $A_1 \sqsubseteq \exists r_1.C_1$. Nevertheless, if any other constraint (exclusive or partial) is defined then the rule suggestion is weaker than the previous one.

Despite being $\sqsubseteq$-rule # 2 a common-sense rule, in our methodological context it enables the user to optimise relationships in possibly big-size ontologies without removing existing ones in order to show each consistent modelling scenarios.

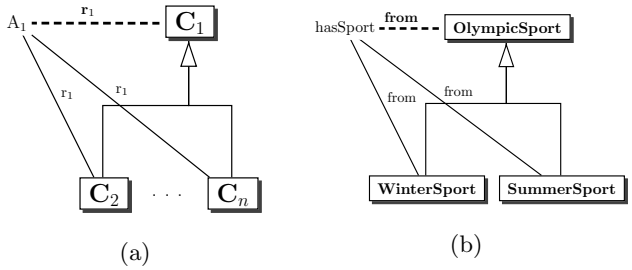



Figure 2: (a) Graphical representation of $\sqsubseteq$-rule # 2. (b) $\sqsubseteq$-rule # 2 applied in *Olympics* ontology

*Example 2.* Let us consider the ontology shown in Fig. 2b where an `olympicSport` can be a `winterSport` or a `summerSport` and both related to other classes by means of the association *hasSport* and the role $from$. The DL representation of this ontology is not included here but it is similar to the one shown in the example associated to $\sqsubseteq$-rule # 1.

The new relationship $hasSport \sqsubseteq \exists from.OlympicSport$ will be suggested by the rule, as depicted in Fig. 2b in dashed line, and thus it will allow to optimise the ontology and reduce the underlying representation as follows (as long as user decides to remove the redundant existing relationships).

$$WinterSport \sqcup SummerSport \sqsubseteq OlympicSport$$
$$hasSport \sqsubseteq \exists from.OlympicSport$$
$$OlympicSport\_hasSport\_min \sqsubseteq OlympicSport \sqcap (\geq 1\ from^{-}.hasSport)$$
$$OlympicSport\_hasSport\_max \sqsubseteq OlympicSport \sqcap (\leq 1\ from^{-}.hasSport)$$

***c*-rule**

$$\{A_1 \sqsubseteq \exists r_1.C_1,\ C_2 \sqsubseteq C_1,\ A_2 \sqsubseteq A_1,\ A_2 \sqsubseteq \exists r_1.C_2,\ C_1 \sqsubseteq \exists r_1^{-}.A_1\} \Rrightarrow \{C_2 \sqsubseteq \exists r_1^{-}.A_2\}$$

The *c*-rule intends to capture those cardinalities that cannot be logically deduced but they can be considered as graphically intuitive in models such as the ontology shown in Fig. 3a. At first, nothing can be concluded about the minimum participation of $r1$ relating $C_2$ with $A_2$, since the $A_2$ relationship may not contain all instances of $C_2$ in the $A_1$ relationship. Nevertheless, from an intuitive and graphical point of view, we consider that this minimum participation could be inherited since both $C_2$ and $A_2$ are subclasses of $C_1$ and $A_1$ respectively, then they have the same properties than their parents. In certain contexts, the *c*-rule can be considered as an instance of the Gangemi's pattern named *Role task* where $C_2$ is a role, which is represented in the pattern as a concept that classifies an object, and $A_2$ is a task. While the pattern does not explicit any cardinality, we conclude that roles make sense only if they have at least one task associated.

According to Guizzardi's definitions, this rule can be also considered as an instance of the *Role Modeling Design* pattern [1], where $C_2$ must be modelled

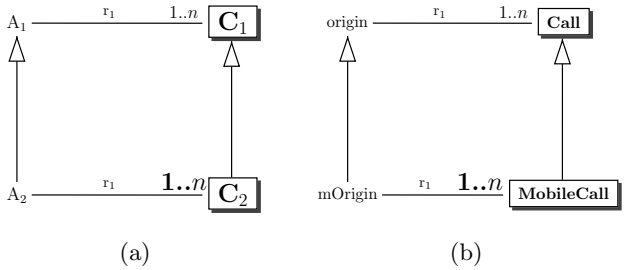



Figure 3: (a) Graphical representation of *c*-rule. (b) *c*-rule applied in *Phone* ontology

as a role, i.e. an anti-rigid type similar to phase type but the changes among subtypes occur due to changes in their relational properties. As a consequence, there exists a relational dependence between $C_2$ and $A_2$ which requires minimum cardinality $\geq 1$ in order to justify the existence of $A_2$ in the model.

*Example 3.* Let us suppose the following partial ontology $\Omega$ from Phone domain, which is depicted in Fig. 3b and translated to DL as follows. $\Omega$ models calls and mobile calls which could have different origins.

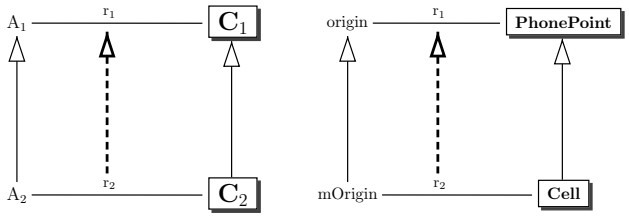
$$Call \sqsubseteq (\leq 1\ r_1^-.origin)$$
$$Call \sqsubseteq \exists r_1^-.origin$$
$$origin \sqsubseteq \exists r_1.Call$$
$$Call\_origin\_min \sqsubseteq Call \sqcap (\geq 1\ r_1^-.origin)$$
$$Call\_origin\_max \sqsubseteq Call \sqcap (\leq 1\ r_1^-.origin)$$
$$mOrigin \sqsubseteq \exists r_1.MobileCall$$
$$MobileCall\_mOrigin\_min \sqsubseteq MobileCall \sqcap (\geq 1\ r_1^-.mOrigin)$$
$$MobileCall\_mOrigin\_max \sqsubseteq MobileCall \sqcap (\leq 1\ r_1^-.mOrigin)$$
$$MobileCall \sqsubseteq Call$$
$$mOrigin \sqsubseteq origin$$

According to rule description, $MobileCall \sqsubseteq \exists r_1^-.mOrigin$ could be proposed to extend $\Omega$. Notice that intuitively only one `origin` is possible for any `call` so that this could be considered for a `mobileCall` since it is also a `call`.

**$r$-rule**

$$\{C_2 \sqsubseteq C_1,\ A_1 \sqsubseteq \exists r_1.C_1,\ A_2 \sqsubseteq \exists r_2.C_2,\ A_2 \sqsubseteq A_1\} \Rrightarrow \{r_2 \sqsubseteq r_1\}$$

The aim of this rule, depicted in Fig. 4a, is to find recurrent ontological structures where a hierarchy of roles can be defined and suggested. From a logical point of view, 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 $C_2 \sqsubseteq C_1$ and $A_1 \sqsubseteq \exists r_1.C_1$ then $A_1 \sqsubseteq \exists r_1.C_2$. Consequently, both $A_2$ and $C_2$ concepts are related to $C_1$ and $A_1$ respectively through $r_1$ so that there could exist unexpected relationships between $A_2$ and $C_1$ or $C_2$ and $A_1$. This scenario can be identified as the *Relation Specialization* anti-pattern in [7] where one of the proposed solutions matches the consequent of our $r$-rule.




(a) (b)

Figure 4: (a) Graphical representation of $r$-rule. (b) $r$-rule applied in *Phone* ontology

*Example 4.* Fig. 4b shows a model extracted from ontology `Phone` and how the rule can be applied to this model. Let us suppose that two instances `PhonePoint`, for example `phonePoint1` and `phonePoint2`, are related to `origin1` and `origin2` respectively. Notice that without asserting $r_2 \sqsubseteq r_1$ and if both phone points are considered as instances of `Cell` then it could not be guaranteed that these cells belongs to the same origin.

## 3 Discussion and Related Works

All the logical systems suppose ideal objects but they do not consider the information about the reality or intuition. This explains that only those axioms implied by the current models will be suggested as possible ontology evolutions. Nevertheless, if we consider the modelling as an approach based on empirical facts, we need something more than only the formal logic. We need observations, experiments and pattern analysis to confirm our hypothesis. The benefit of a rules-based approach is to offer a trade-off between the inherent rigidity to the logic systems and the intuitive characteristics of the ontological modelling.

Similar to our approach, Guizzardi in [1] proposes to use rules in pattern-based design. However, the proposed inductive rules are extracted from a subset of the design patterns underlying to OntoUML and its application requires of an intensive interaction between users and the tool so as to specify each possible element and its relationships in the ontology under development. As a consequence, reasoning services are not invoked during this modelling process. Respecting to ontology evolution, it is partially supported by reducing the possible choices of modelling primitives to be adopted and by offering a step-by-step modelling activity. Finally, this feature has not been developed in the last OntoUML version.

In order to analyse the current graphical tools, we consider the following key factors: the graphical, automatic reasoning and evolution support and their integration in the tool. OntoUML [6] is a graphical tool but the reasoning and evolution support is limited. In spite of offering an insufficient graphical interface, Protégé [8] and TopBraid Composer [9] allow this integration. The inferences are shown in the graphical editor, but these are only restricted to IsAs

hierarchies. Their evolution support is also partial: manual, ontology differences (Protégé) and versioning and collaboration (TopBraid Composer). Finally, NeOn toolkit [10], Kaon2 [11] and SWOOP [12] provide access to reasoners but they are not graphical-based tools. Respecting to evolution, only NeOn incorporates a specific framework to support it from external sources. Kaon2 and SWOOP simply offer some operations as redo and undo (Kaon2) or imports and versioning (SWOOP). Other tools as GrOWL [13], OWLGrEd [14], Graphol [15], "model outline" framework [16] and VOWL [17], which is a Protégé plugin, are also graphical-centered tool. They define a graphical syntax and semantics providing users with a visual representation of their models and thus avoid any complex textual syntax. Nevertheless, the reasoning support is not provided in any of these tools as ICOM does and the ontology evolution is not properly supported.

The intention behind rules is to reduce the complexity of the evolution process and they are defined to be implemented within a graphical tool. Nevertheless, they cannot be isolated from other complementary artefacts such as user queries which allow to identify relevant parts of an ontology and back-end reasoning systems which allow to inquire the model to check rules applicability and their possible consequences. In any case, it is the user who decides which of the suggestions (rule consequences) are appropriate, if any, according to his intended model.

## 4 Conclusions and Future Works

This work introduces a small catalogue of extension rules which intends to capture the intuitive characteristics inherent to the ontology modelling. Each rule has been defined and analysed by considering different theories of design patterns and by showing the intended meaning through illustrations where the intuition is presented. The aim is to place emphasis on a reduced search space of possible ontology extensions, and thus also reduce the complexity when trying with large ontologies. This alternative offers flexibility and a fine-grained level where the focus is on a subset of ontology elements to analyse in order to introduce new ones, in contrast to design patterns which are not orientated towards ontology evolution since its smallest unit of work is an ontology when in rules it is an ontological element. Then they have the effect of changing only the ontology elements involved in the refactoring, which can be intra- or inter-ontology ones. Rules must be used together with other artefacts such as user queries and back-end reasoning systems to identify relevant parts of an ontology and check the consequences of application on the whole model. Furthermore, the modularity is a key property in this approach so that some changes on the set of rules are independent of the underlying methodology. An example about the usage of these rules in a methodology has been published in [18].

As future works, we propose to identify new extension rules and define a logical formalism as an application framework for these rules. The rules will be also evaluated 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 them [19]. We plan to provide support to user-defined rules and to rules involving instances so that we could supply evolution at extensional level (ABox) and possibly use it to extend the intentional knowledge.

## References

1. Guizzardi, G., das Graças, A., Guizzardi, R.: Design patterns and inductive modeling rules to support the construction of ontologically well-founded conceptual models in ontouml. In: CAiSE Workshops. (2011)
2. Gangemi, A., Presutti, V.: Ontology design patterns. In: Handbook of Ontologies. 2nd edn. (2009)
3. Franconi, E., Ng, G.: The i.com tool for intelligent conceptual modelling. In: Proc. of 7th KRDB'2000. (2000)
4. Fillottrani, P., Franconi, E., Tessaris, S.: The new icom ontology editor. In: Description Logics. CEUR Workshop Proceedings, CEUR-WS.org (2006)
5. Fillottrani, P., Franconi, E., Tessaris, S.: The icom 3.0 intelligent conceptual modelling tool and methodology. Semantic Web (2012)
6. Guizzardi, G.: Ontological foundations for structural conceptual models. PhD thesis, University of Twente, Enschede, The Netherlands, Enschede (October 2005)
7. Guizzardi, G., Sales, T.P.: Detection, simulation and elimination of semantic antipatterns in ontology-driven conceptual models. In: Conceptual Modeling - 33rd International Conference, ER 2014. Proceedings. (2014)
8. Knublauch, H., Fergerson, R., Noy, N., Musen, M.: The protégé owl plugin: An open development environment for semantic web applications. (2004)
9. TopQuadrant: TopQuadrant — Products — TopBraid Composer (2011)
10. Hasse, P., Lewen, H., Studer, R., Erdmann, M.: The NeOn Ontology Engineering Toolkit. (2008)
11. Motik, B., Studer, R.: KAON2–A Scalable Reasoning Tool for the Semantic Web. In: Proceedings of the 2nd ESWC'05. (2005)
12. Kalyanput, A., Parsia, B., Sirin, E., Grau, B., Hendler, J.: Swoop: A 'web' ontology editing browser. Journal of Web Semantics (June 2005)
13. Krivov, S., Williams, R., Villa, F.: Growl: A tool for visualization and editing of owl ontologies. J. Web Sem. (2007)
14. Cerans, K., Ovcinnikova, J., Liepins, R., Sprogis, A.: Advanced owl 2.0 ontology visualization in owlgred. In: DB&IS. Frontiers in Artificial Intelligence and Applications, IOS Press (2012)
15. Console, M., Lembo, D., Santarelli, V., Savo, D.F.: Graphol: Ontology representation through diagrams. In: Informal Proceedings of the 27th International Workshop on Description Logics. (2014)
16. do Amaral, F.N.: Usability of a visual language for dl concept descriptions. In: RR. Lecture Notes in Computer Science, Springer (2010)
17. Lohmann, S., Negru, S., Bold, D.: The protégévowl plugin: Ontology visualization for everyone. In: The Semantic Web: ESWC 2014 Satellite Events. Lecture Notes in Computer Science. (2014)
18. Braun, G., Cecchi, L., Fillottrani, P.: Integrating Graphical Support with Reasoning in a Methodology for Ontology Evolution. In: Proc. of the 9th Int. Workshop on Modular Ontologies WoMO 15 IJCAI 15. CEUR Workshop Proceedings (2015)
19. Gediga, G., Hamborg, K.C.: Evaluation of software systems. Encyclopedia of Computer Science and Technology (2001)