Visual Modeling of OWL Individuals and Classifier-Based Value Sets in OWLGrEd

Jūlija Ovčiņņikova[1,*], Kārlis Čerāns[1,*]

[1]Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia

Abstract

We present new functionality in the OWLGrEd web-based visual ontology editor for compact representation of OWL individuals and classifier-based controlled value sets. The approach provides three alternative representations of individuals and a unified classifier element supporting datatype enumerations, OWL individual enumerations, and SKOS vocabularies. The proposed notation maps the visual elements to corresponding OWL and SKOS semantics, enabling more compact ontology diagrams.

Keywords

OWL 2, visual ontology modeling, individuals, classifiers, SKOS

1. Introduction

Visual ontology editors make ontologies easier to understand and support communication between ontology engineers and domain experts. By presenting OWL axioms graphically, they provide an overview of ontology structure that is often difficult to obtain from textual serializations alone. However, diagrams become harder to read when they contain many individuals or controlled values. By compacting individual representations, diagram complexity can be reduced while maintaining the ontology's formal meaning.

Controlled value sets occur frequently in ontologies, for example, for gender, status, country, role, or type. Depending on their intended semantics, these values may be represented as datatype enumerations, OWL individuals, or SKOS concepts, with SKOS providing a standard model for publishing and sharing controlled vocabularies. We aim at a visual ontology editor that provides a compact notation together with import and export semantics for each representation.

Existing tools address these requirements partially. General OWL editors such as Protégé [1] and TopBraid Composer [2] support ontology authoring, import, and export. Graphical tools, including OWLGrEd [3, 4], Graphol [5], Chowlk [6], WebVOWL [7], Ontodia [8], Graffoo [9], and OWL2Graffoo [10], support visual modeling, exploration, and ontology exchange with varying support for individuals and enumerations. VocBench [11], Skosmos [12], and SKOS Play! [13] focus on controlled-vocabulary management and publication. To our knowledge, no existing tool combines compact individual-list representations, multiple classifier types, and import-aware graphical rendering.

In this paper, we extend the OWLGrEd notation and its web-based implementation [14] with configurable visual representations for OWL individuals and a unified classifier element for controlled value sets. The contribution is a common graphical notation for different levels of individual detail and different semantic interpretations of controlled value sets, with defined mappings to existing OWL and SKOS constructs. Individuals can be shown as separate nodes, compact lists within class nodes, or dedicated lists with selected property assertions, with the representation configurable during ontology import. Classifiers support datatype enumerations, OWL individual enumerations, SKOS vocabularies, and combined OWL/SKOS representations.

The 25th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2026) - Satellite Events (Workshops, Tutorials, Demos, Posters)

*Corresponding author.

julija.ovcinnikova@lumii.lv (J. Ovčiņņikova); karlis.cerans@lumii.lv (K. Čerāns)

0000-0002-5884-763X (J. Ovčiņņikova); 0000-0002-0154-5294 (K. Čerāns)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-26

CEUR-WS.org/Vol-4340/paper7.pdf

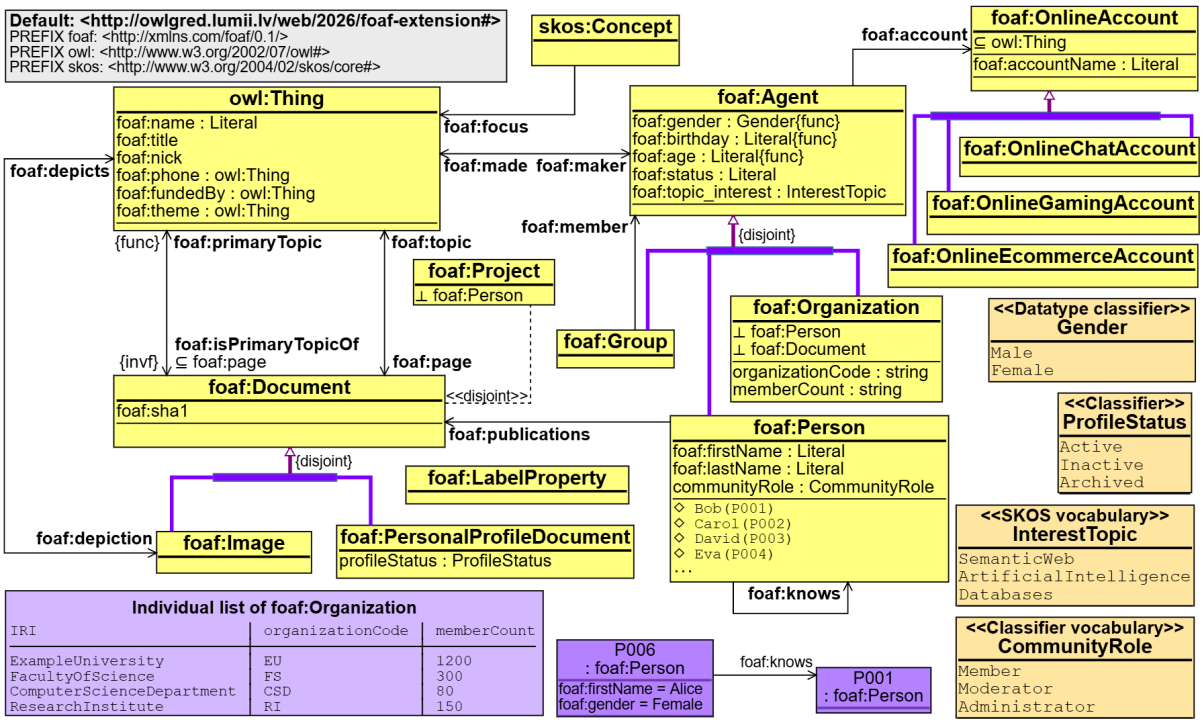


Figure 1: FOAF ontology example

2. OWLGrEd Editor and Notation

OWLGrEd[1] [14] is a web-based open-source visual ontology editor that uses a UML-like style graphical notation for OWL 2 ontologies. Classes are represented as class nodes, object properties as associations, and data properties as attributes. More complex OWL constructs, including restrictions, equivalence, cardinalities, and property characteristics, are represented using dedicated visual elements or Manchester syntax expressions. Individuals use UML object notation and may be connected through graphical property assertions. The namespace declarations are shown in a dedicated node, and the names from the default namespace are displayed without a prefix. The notation and features are illustrated using a reduced view of the Extended FOAF demo ontology in Figure 1, with less illustrative properties omitted for clarity. The complete ontology and OWLGrEd project are available online[2].

3. Representation of Individuals in OWLGrEd

OWL individuals can be visually represented in three ways, providing different levels of detail and compactness: as separate nodes, as a compact list inside a class node, or as a dedicated individual list node. The individual specification can be imported from and exported to standard OWL formats. During import, visualisation parameters control the number and representation of individuals shown.

3.1. Separate individual nodes

The first alternative represents each individual as a separate node. This is the classical OWLGrEd notation and provides the most detailed view, since class assertions and property assertions involving the individual can be shown directly in the diagram. In Figure 1, *P006* is shown as an instance of *foaf:Person*. The data property assertion *foaf:firstName = Alice* is displayed inside the individual node, while the object property assertion *foaf:knows* is shown as a graphical link from *P006*.

[1]Source code: https://github.com/LUMII-Syslab/owlgred; Online playground: https://owlgred.semtech.lv/
[2]Extended FOAF Demo Projecthttps://github.com/LUMII-Syslab/owlgred/wiki/Extended-FOAF-Demo-Project

3.2. Individual list inside a class node

The second alternative places individuals in a compact list inside the corresponding class node. Each individual is shown in the form *display label (IRI)*, where the display label may be taken from a selected property assertion. For example, in Figure 1, the *foaf:Person* class contains *Bob (P001)*, where *P001* is the individual IRI and *Bob* is the value of the selected *foaf:firstName* property (can be set in the property dialog). Other property assertions remain available for inspection and editing, but are not shown in the diagram. A *OneOf* marker can optionally define a class as exactly the set of listed individuals. This representation keeps the individuals visible without adding a separate node for each of them.

3.3. Dedicated individual list node

The third alternative uses a dedicated individual list node for individuals of a particular class. Unlike the compact list embedded in a class node, it is shown as a separate graphical element and presents selected property assertions in table-like form. This provides a compact view in which individuals and their values can be viewed directly. In Figure 1, the *Individual list of foaf:Organization* node shows the Organization individuals together with their *organizationCode* and *memberCount* values. A *OneOf* marker can be used to define the class as exactly the set of listed individuals.

For both list-based representations, the number of individuals shown in the diagram can be limited. A default limit can be set during import and changed separately for each class node or dedicated individual list node. Individuals beyond the limit remain in the ontology and are still available for inspection, editing, and export.

4. Representation of Classifiers in OWLGrEd

OWLGrEd introduces a dedicated classifier element for controlled value sets. Classifiers with different meanings are shown using the same basic visual structure, while their semantic mode determines how the classifier and its values are represented in the exported ontology. The values may be treated as literals, OWL individuals, SKOS concepts, or both OWL individuals and SKOS concepts.

4.1. Datatype classifier

The first mode is the Datatype classifier, which represents a controlled set of literal values and is exported as an OWL datatype enumeration. In Figure 1, this mode is illustrated by the *Gender* classifier with the values *Male* and *Female*. The data property *foaf:gender* in the *foaf:Agent* class has the range *Gender*, so its value is limited to one of these literals. This mode is suitable for simple controlled values that do not need to be modeled as OWL individuals. The corresponding OWL Functional Syntax is shown below.

```
Declaration(Datatype(:Gender))
DatatypeDefinition(:Gender DataOneOf("Male" "Female"))
DataPropertyRange(foaf:gender :Gender)
```

4.2. OWL individual classifier

The second mode is the OWL individual classifier, in which the classifier is exported as an OWL class and its values as named individuals. Compared with the OWL individual representations described in Section 3, where closure is optional, this mode defines the classifier as a closed set of distinct values by default. In Figure 1, this mode is illustrated by the *ProfileStatus* classifier with *Active*, *Inactive*, and *Archived* values. The object property *profileStatus* has the range *ProfileStatus*. The classifier is defined as exactly the set of listed individuals using *ObjectOneOf*, while *DifferentIndividuals* states that the values are distinct. This mode is suitable when controlled values need to be used as ontology individuals, for example in property assertions or reasoning.

```
Declaration(Class(:ProfileStatus))
Declaration(NamedIndividual(:Active))
Declaration(NamedIndividual(:Inactive))
Declaration(NamedIndividual(:Archived))

ObjectPropertyRange(:profileStatus :ProfileStatus)
EquivalentClasses(:ProfileStatus
  ObjectOneOf(:Active :Inactive :Archived))
DifferentIndividuals(:Active :Inactive :Archived)
```

4.3. SKOS vocabulary classifier

The third mode is the SKOS vocabulary classifier, in which the classifier is exported as a *skos:ConceptScheme* and its values as *skos:Concept* instances. The values are linked to the scheme using *skos:inScheme*. This mode is suitable for controlled vocabularies intended for publication, reuse, or exchange rather than for defining a closed OWL enumeration. In Figure 1, this mode is illustrated by the *InterestTopic* classifier with the values *SemanticWeb*, *ArtificialIntelligence*, and *Databases*. The object property *foaf:topic_interest* has the range *skos:Concept*, while an additional restriction limits its values to concepts belonging to the *InterestTopic* scheme. The *InterestTopic* scheme is used by the *foaf:Agent* class through the *foaf:topic_interest* property.

```
Declaration(NamedIndividual(:SemanticWeb))
ClassAssertion(skos:ConceptScheme :InterestTopic)
ClassAssertion(skos:Concept :SemanticWeb)
ObjectPropertyAssertion(
  skos:inScheme :SemanticWeb :InterestTopic)

ObjectPropertyRange(foaf:topic_interest skos:Concept)
SubClassOf(foaf:Agent
  ObjectAllValuesFrom(foaf:topic_interest
    ObjectIntersectionOf(skos:Concept
      ObjectHasValue(skos:inScheme :InterestTopic))))
```

4.4. Classifier vocabulary

The fourth mode is the Classifier vocabulary, which combines the OWL individual classifier and SKOS vocabulary modes. The classifier is exported as an OWL class and as a *skos:ConceptScheme*, while its values are exported both as named individuals and as *skos:Concept* instances. In Figure 1, this mode is illustrated by the *CommunityRole* classifier with the values *Member*, *Moderator*, and *Administrator*. The object property *communityRole* has the range *CommunityRole*. The same values can therefore participate in OWL modeling while also being published and annotated as SKOS concepts. Each value is linked to *CommunityRoleConceptScheme* using *skos:inScheme*, while *CommunityRole* is defined as exactly the set of the listed distinct individuals. The *CommunityRole* classifier is used by the *foaf:Person* class through the *communityRole* property.

```
Declaration(Class(:CommunityRole))
Declaration(
  NamedIndividual(:CommunityRoleConceptScheme))
Declaration(NamedIndividual(:Member))
Declaration(NamedIndividual(:Moderator))
Declaration(NamedIndividual(:Administrator))

ClassAssertion(skos:ConceptScheme
  :CommunityRoleConceptScheme)

ClassAssertion(:CommunityRole :Member)
ClassAssertion(skos:Concept :Member)
ObjectPropertyAssertion(skos:inScheme
  :Member :CommunityRoleConceptScheme)

ClassAssertion(:CommunityRole :Moderator)
ClassAssertion(skos:Concept :Moderator)
ObjectPropertyAssertion(skos:inScheme
  :Moderator :CommunityRoleConceptScheme)

ClassAssertion(:CommunityRole :Administrator)
ClassAssertion(skos:Concept :Administrator)
ObjectPropertyAssertion(skos:inScheme
  :Administrator :CommunityRoleConceptScheme)

EquivalentClasses(:CommunityRole
  ObjectOneOf(:Member :Moderator :Administrator))
DifferentIndividuals(:Member :Moderator :Administrator)
ObjectPropertyRange(:communityRole :CommunityRole)
```

5. Related Work

General-purpose OWL editors such as **Protégé** [1] and **TopBraid Composer** [2] support ontology authoring, import, export, individual editing, and enumerations, mainly through structured views and forms. TopBraid Composer additionally provides graphical ontology views. Neither offers configurable graphical alternatives for representing individuals in compact or detailed forms.

Graphical OWL tools provide richer visual notation, but differ in how explicitly and compactly they represent individuals and enumerations.

OWLGrEd desktop version [3, 4] supports the OWL 2 ontology life cycle. Its architecture allows the graphical notation to be extended, as demonstrated by earlier extensions such as support for enumerated classes. However, it is limited to Windows and uses an older technology stack.

Graphol [5] represents individuals, *oneOf* constructors, and related axioms as separate graphical elements. This makes the OWL structure visible, but may require several elements for a single controlled value set. **Graffoo** [9] and **Chowlk** [6] provide UML-based diagrammatic alternatives for OWL modeling. **Graffoo** represents individuals and class-membership axioms graphically, while restrictions use Manchester syntax. The Graffoo itself does not provide direct ontology import or export. **OWL2Graffoo** [10] adds ontology import using Graffoo notation, although support for individuals is experimental.

Chowlk uses UML-based notation and provides several graphical representations for individuals, including separate individual elements and compact lists associated with classes. It also supports enumerated classes and datatypes, but follows a diagram-to-OWL workflow and does not provide general OWL import. Across these approaches, controlled value sets are represented through ordinary OWL constructs rather than through a single classifier element supporting datatype, OWL-individual, SKOS, and combined OWL/SKOS semantics.

WebVOWL [7] and **Ontodia** [8] are primarily intended for ontology and RDF exploration rather than authoring. **WebVOWL** provides an interactive overview of the ontology schema, with individuals shown indirectly through instance counts and sidebar lists. **Ontodia** focuses on instance-level exploration, displaying individuals as connected nodes and supporting diagram arrangement and sharing, but not ontology modification or export.

Vocabulary-oriented tools address controlled values from a management and publication perspective. **VocBench** [11] supports OWL and SKOS editing and ontology exchange, mainly through structured views, with additional graphical exploration. **Skosmos** [12] focuses on browsing and publishing SKOS vocabularies, while **SKOS Play!** [13] supports vocabulary visualisation, validation, and conversion to SKOS. These tools do not integrate controlled value sets as classifier elements within a general graphical OWL model. The contribution addresses a missing intersection in existing tools: compact individual-list representations, multiple classifier semantics, and import-aware graphical rendering. OWLGrEd combines these capabilities (the classifier import remains under development).

6. Conclusions

This paper presented new OWLGrEd functionality for compact visual modelling of OWL individuals and controlled value sets. Individuals can be represented at different levels of detail, from separate nodes to class-embedded or dedicated lists, while the classifier element provides a common visual notation for datatype, OWL-individual, SKOS, and combined OWL/SKOS representations. The defined mappings preserve the corresponding ontology semantics while allowing diagrams to show only the information relevant to the modelling task. Future work will include empirical evaluation of the notation and completion of classifier import.

Acknowledgments

This work was supported by activity 1.1.1.9 Research application No 1.1.1.9/LZP/1/24/037 of the Activity "Post-doctoral Research" "*Visual methods and tools for ontology management*".

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (GPT-5.6, OpenAI) in order to: Paraphrase and reword; Improve writing style; Grammar and spelling check; Formatting assistance. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. A. Musen, Protégé Team, The Protégé project: A look back and a look forward, AI Matters 1 (2015) 4–12. URL: https://protege.stanford.edu/. doi:10.1145/2757001.2757003.

[2] TopQuadrant, Inc., TopBraid Composer: Documentation, Online documentation, 2026. URL: https://topbraidcomposer.org/html/, accessed: 2026-07-09.

[3] J. Bārzdiņš, G. Bārzdiņš, K. Čerāns, R. Liepiņš, A. Sproģis, Uml style graphical notation and editor for owl 2, in: P. Forbrig, H. Günther (Eds.), Perspectives in Business Informatics Research, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 102–114.

[4] K. Čerāns, J. Ovčiņņikova, R. Liepiņš, M. Grasmanis, Extensible visualizations of ontologies in owlgred, in: P. Hitzler, S. Kirrane, O. Hartig, V. de Boer, M.-E. Vidal, M. Maleshkova, S. Schlobach, K. Hammar, N. Lasierra, S. Stadtmüller, K. Hose, R. Verborgh (Eds.), The Semantic Web: ESWC 2019 Satellite Events, Springer International Publishing, Cham, 2019, pp. 191–196.

[5] D. Lembo, V. Santarelli, D. F. Savo, G. D. Giacomo, Graphol : A Graphical Language for Ontology Modeling Equivalent to OWL 2, Future Internet 14 (2022) 1–29. URL: https://ideas.repec.org/a/gam/jftint/v14y2022i3p78-d760974.html.

[6] S. Chávez-Feria, R. García-Castro, M. Poveda-Villalón, Chowlk: from uml-based ontology conceptualizations to owl, in: P. Groth, M.-E. Vidal, F. Suchanek, P. Szekley, P. Kapanipathi, C. Pesquita, H. Skaf-Molli, M. Tamper (Eds.), The Semantic Web, Springer International Publishing, Cham, 2022, pp. 338–352.

[7] V. Wiens, S. Lohmann, S. Auer, Webvowl editor: Device-independent visual ontology modeling, in: Proceedings of ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks, volume 2180 of *CEUR Workshop Proceedings*, 2018. URL: https://ceur-ws.org/Vol-2180/paper-75.pdf.

[8] D. Mouromtsev, D. Pavlov, Y. Emelyanov, A. Morozov, D. Razdyakonov, M. Galkin, The simple, web-based tool for visualization and sharing of semantic data and ontologies, in: Proceedings of the ISWC 2015 Posters & Demonstrations Track, volume 1486 of *CEUR Workshop Proceedings*, 2015. URL: https://ceur-ws.org/Vol-1486/paper_77.pdf.

[9] R. Falco, A. Gangemi, S. Peroni, D. Shotton, F. Vitali, Modelling OWL ontologies with Graffoo, in: The Semantic Web: ESWC 2014 Satellite Events, Lecture Notes in Computer Science, Springer, 2014, pp. 320–325. URL: https://link.springer.com/chapter/10.1007/978-3-319-11955-7_42. doi:10.1007/978-3-319-11955-7_42.

[10] J. Duchateau, OWL2Graffoo: A draw.io plugin graffoo importer, GitHub repository, 2026. URL: https://github.com/jduchateau/OWL2Graffoo, accessed: 2026-07-09.

[11] A. Stellato, A. Turbati, M. Fiorelli, T. Lorenzetti, E. Costetchi, C. Laaboudi, W. Van Gemert, J. Keizer, Towards VocBench 3: Pushing collaborative development of thesauri and ontologies further beyond, in: P. Mayr, D. Tudhope, K. Golub, C. Wartena, E. W. De Luca (Eds.), Proceedings of the 17th European Networked Knowledge Organization Systems Workshop (NKOS 2017), volume 1937 of *CEUR Workshop Proceedings*, CEUR-WS.org, Thessaloniki, Greece, 2017, pp. 39–52. URL: https://ceur-ws.org/Vol-1937/paper4.pdf.

[12] O. Suominen, H. Ylikotila, S. Pessala, M. Lappalainen, M. Frosterus, J. Tuominen, T. Baker, C. Caracciolo, A. Retterath, Publishing SKOS vocabularies with Skosmos, 2015. URL: https://skosmos.org/publishing-skos-vocabularies-with-skosmos.pdf, manuscript submitted for review.

[13] T. Francart, SKOS Play!: Thesaurus and taxonomy visualization tool, 2026. URL: https://skos-play.sparna.fr/play/about?lang=en, sparna; version 0.10.0.

[14] J. Ovčiņņikova, M. Grasmanis, L. Lāce, K. Čerāns, OWLGrEd: Web-based visual ontology management, in: The Semantic Web: ESWC 2026 Satellite Events, Lecture Notes in Computer Science, Springer, 2026. Forthcoming.