GizMO – A Customizable Representation Model for Graph-Based Visualizations of Ontologies

Vitalis Wiens
TIB Leibniz Information Centre for Science and Technology and Fraunhofer IAIS
Hannover and St. Augustin, Germany
vitalis.wiens@gmail.com

Steffen Lohmann
Fraunhofer IAIS
St. Augustin, Germany
steffen.lohmann@iais.fraunhofer.de

Sören Auer
TIB Leibniz Information Centre for Science and Technology and L3S Research Center, Leibniz University
Hannover, Germany
auer@l3s.de

ABSTRACT

Visualizations can support the development, exploration, communication, and sense-making of ontologies. Suitable visualizations, however, are highly dependent on individual use cases and targeted user groups. In this article, we present a methodology that enables customizable definitions for the visual representation of ontologies.

The methodology describes visual representations using the OWL annotation mechanisms and separates the visual abstraction into two information layers. The first layer describes the graphical appearance of OWL constructs. The second layer addresses visual properties for conceptual elements from the ontology. Annotation ontologies and a modular architecture enable separation of concerns for individual information layers. Furthermore, the methodology ensures the separation between the ontology and its visualization.

We showcase the applicability of the methodology by introducing GizMO, a representation model for graph-based visualizations in the form of node-link diagrams. The graph visualization meta ontology (GizMO) provides five annotation object types that address various aspects of the visualization (e.g., spatial positions, viewport zoom factor, and canvas background color). The practical use of the methodology and GizMO is shown using two applications that indicate the variety of achievable ontology visualizations.

CCS CONCEPTS

• **Human-centered computing → Visualization**; • **Information systems → Web Ontology Language (OWL)**.

KEYWORDS

Ontology visualization; annotation ontology; customization; visual representation; visualization framework; visual notation.

ACM Reference Format:
Vitalis Wiens, Steffen Lohmann, and Sören Auer. 2019. GizMO – A Customizable Representation Model for Graph-Based Visualizations of Ontologies. In *10th International Conference on Knowledge Capture (K-CAP'19), November 19–21, 2019, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3360901.3364431

Preprint

The following article has been accepted for **K-Cap 2019**.
After it is published, it will be found at https://doi.org/10.1145/3360901.3364431

K-CAP '19, November 19–21, 2019, Marina Del Rey, CA, USA
ACM ISBN 978-1-4503-7008-0/19/11…$15.00
https://doi.org/10.1145/3360901.3364431

1 INTRODUCTION

Ontologies provide formal machine-readable representations for conceptualizations of information in various domains. The development, exploration, communication, and sense-making of ontologies can be facilitated using visual representations [1]. Various ontology visualization methods and tools are available, and new ones are being developed every year. The applied methods range from indented trees and chord diagrams to treemaps and Euler diagrams. According to a recent survey [4], most methods and tools visualize the content of ontologies using two-dimensional graph-based representations in the form of node-link diagrams.

The challenge with most approaches, however, is grounded in their design. On the one hand, visualization methods are created with a particular definition for the representation model. On the other hand, users perceive the provided visualization and build a mental model for the interpretation of the content [16]. Ideally, the visual representation model corresponds to the user's mental model. However, these match typically *only* in some aspects and diverge from the user's expectations and previous experiences with other visualization tools. In order to satisfy the varying demands of users and use cases, suitable visualizations require customizable representation models.

In this article, we introduce a methodology for customizable visualizations of ontologies. The methodology defines visual representation models using the OWL annotation mechanisms. A set of annotation ontologies addresses different aspects of the visualization and enables separation of concerns. The use of `owl:imports` statements enables to enrich ontologies with visual definitions for their depiction.

We showcase the applicability of the methodology by introducing GizMO, a representation model for graph-based ontology visualizations in the form of node-link diagrams. Two applications show the utilization of GizMO and the variety of achievable ontology visualizations. The applications are designed to reduce textual crafting of annotation ontologies and showcase the interoperability of the methodology.

The remainder of this article is structured as follows: After introducing core requirements for the visual representation of ontologies and further deepening the research challenge in Section 2, we summarize related work in Section 3. Details of the methodology and the GizMO representation model are provided in Sections 4 and 5, followed by the presentation of the two applications in Section 6. Finally, we conclude with an outlook on future work in Section 7.

2 MOTIVATION AND REQUIREMENTS

Visualizations provide an abstraction of information that reinforces human cognition for the sense-making of ontologies. However, a visual representation model has to correspond to the user's mental model in order to provide a suitable visualization. Thus, only flexible and customizable visualization approaches for ontologies can fulfill the demands of different use cases and user groups.

Without any loss of generality for the methodology, the remainder of this article focuses on *domain* ontologies and their visual representations in the form of node-link diagrams. As illustrated in Figure 1, the two main visual characteristics of different methods and tools are *visual appearance* and *spatial arrangement* (i.e., the graphical notation used to draw the ontology graph and the layout of nodes and edges). Thus, a customizable visual representation model needs to address the following requirements: i) provide the customizable visual appearance of rendering elements in order to coincide with the user's *mental model*; ii) provide spatial information and visibility status of rendering elements in order to coincide with the user's *mental map*; iii) provide the means to represent and share the definition of visualizations.

Visual Appearance – Visual appearance of elements is described by a visual notation that formally defines the graphical depiction of ontology elements, such as `owl:Class`. Examples of visual notations for ontologies are VOWL [8] and Graffoo [5], but also UML is often used to represent ontologies [3], or different tool-specific notations. Although UML has a standard visual notation, various styles exist, such as the visual representation of ontologies with TopBraid Composer [7], a UML version of the VOWL notation [9], or the UML mapping of the NeOn Toolkit [6].

Graph-based visualizations can be categorized into name-label-only and nested visualizations [4]: Name-label-only visualizations depict the elements of the ontology as individual labeled nodes and links. Nested visualizations (e.g., UML) aggregate information (e.g., the data properties of a class) and visualize them as a list of attributes inside the corresponding node. Thus, in order to coincide with the user's *mental model*, a representation model has to provide a customizable notation that defines visual characteristics and also encodes the depiction of aggregated elements.

Spatial Information – Spatial information is essentially necessary in order to preserve the *mental map* of users [16]. The spatial arrangement of elements in different layouts (e.g., hierarchical trees or circular layouts) can facilitate the organization of information and sense-making. The preservation of a user-defined layout is only obtainable when spatial information is attached to the domain ontology. The spatial information has to correspond to the used notation in order to reconstruct the visualization correctly. For example, nested visualizations typically do not encode spatial information for aggregated elements. Therefore, using spatial information of a nested visual notation in a different one can result in an invalid spatial assignment of elements. Consequently, a visual representation model has to provide spatial information that corresponds to the used notation and the domain ontology for which it has been created.

Glyph Specific Information – A glyph is a collection of rendering primitives. For example, a glyph could be a composition of a shape (e.g., circle) and a label text. Glyph specific information has to

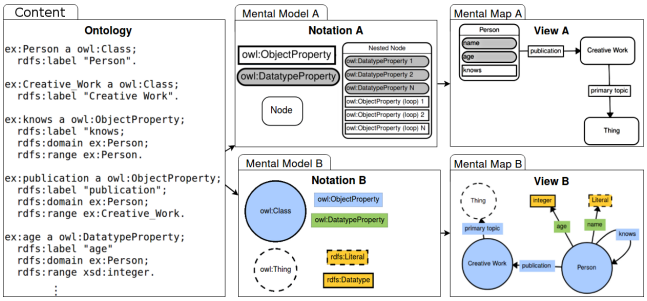

Figure 1: Separation of concerns into notations and views. Notations define how OWL constructs are depicted. Views provide spatial position assignment and visibility status.

address additional properties, such as the visibility status of a glyph. With growing size and complexity of an ontology, visualizations become harder to read due to visual clutter and information overload. Various ontology visualization tools address this by providing filtering mechanisms in order to reduce the information load for human's limited cognitive capacity [2]. Thus, balancing the cognitive load requires strategies for determining parts of the ontology to be visualized and to be hidden.

Suitable ontology visualizations may require additional customization for individual glyphs. An often requested feature is the modification of visual characteristics for some elements of the domain ontology. These could be modifications of shapes and colors in order to highlight or group these elements in the visualization. Also, the replacement of selected shapes and labels with icons or images is sometimes requested. Thus, a representation model requires to provide customizable visual definitions for elements of the domain ontology.

Representation, Governance, and Management – Some of the available tools, such as Cytoscape [13], already partially fulfill the described requirements. However, the proposed solutions typically are restricted to a specific tool and visualization method. A methodology that defines visual representations in the form of OWL annotation ontologies can overcome such boundaries. Nevertheless, such a methodology has to ensure the following criteria:

1) **Reusability:** The methodology should provide the means to recreate and modify existing notations or even design new ones in such a way that these can be shared and reused. The methodology should provide the means to define spatial positions for elements of the domain ontology. Additionally, it should provide the means to customize glyph specific information (e.g., visibility status, shape, and color).
2) **Preserving the originality of the domain ontology:** Our methodology uses OWL annotation mechanisms in order to enrich elements with information for their visual depiction. However, directly attaching visual descriptions to the domain ontology or its elements results in an unwanted solution: the modification of OWL constructs and elements of the domain ontology. Thus, the methodology should provide the means to bind meta information to elements of the domain ontology without the need to directly modify them.

3) **Separation of concerns:** The separation of concerns plays an essential role in fostering flexible visual representations. The methodology should provide the means for the flexible exchange of specific visual properties. Our methodology addresses different aspects of the visualization on two information layers that are represented as annotation ontologies. Using import statements domain ontologies are enriched with visual definitions for their depiction.

In summary, suitable visualizations require to be in correspondence with the user's mental model in order to facilitate the interpretation of the visualized content. In this article, we present a methodology for customizable visual representations and apply it for the design of GizMO – a representation model for graph-based visualizations of ontologies.

3 RELATED WORK

Dudáš et al. [4] provide a comprehensive recent survey of ontology visualization methods and tools. In the following, we discuss related work that targets the presentation of ontology information and its customization to user needs.

Early work by Pietriga et al. [12] develops the concept of Fresnel Lenses, a presentation vocabulary for RDF. Lenses, formats, and CSS classes are responsible for the visualization of RDF data. The objective of the lenses is to select the content and apply custom orderings of the data. The formats and the CSS classes define how the information is presented. IsaViz [10] is a related approach that enables the definition of visual representations for ontologies based on Graph Style Sheets (GSS) [11]. GSS are similar to CSS and use a selector to which attributes are assigned. Cytoscape [13] is a visualization tool that applies a similar approach in order to enable customizable visualizations of node-link diagrams.

GSS and CSS enable the definition of styles for rendering primitives. However, they do not address the spatial positioning as required in graph-based visualization methods. Furthermore, GSS and CSS do not operate on OWL constructs, but apply styles on elements in the DOM tree. Also, specific requirements, such as the distinction between name-label-only and nested visualizations, are not supported by these languages. Thus, GSS and CSS lack capabilities required for the comprehensive representation of graph-based ontology visualizations and are not sufficient in this context.

The Web Annotation Data Model [17] defines a model for describing associations between resources. In this model, annotations comprise of a target and a body, where the body contains additional data that should be associated with the target resource. It provides a standard description method for annotations to be shared between systems. While allowing for the definition of style information as annotations, it is designed for the general annotation of resources and not for the visual representation of ontologies.

Most approaches define an underlying visual representation model. However, these models are not sharable because they are created for a specific system, tool, or visualization method. In contrast, our methodology defines visual representation models as OWL ontologies. Furthermore, it separates the visual representation into two layers: a *global* one for general notations, and a *local* one for individual customizations. These annotation ontologies for each layer can be shared and reused between users and tools.

4 METHODOLOGY

Numerous visual representation models have been developed. However, these are typically restricted to the corresponding visualization method and tool. In this section, we introduce a methodology that can overcome method and tool-specific boundaries and enables customizations for ontology visualizations. Furthermore, we provide a more detailed discussion for individual parts of the methodology.

Ontologies are not designed with the focus of information presentation to humans [14]. They are created and shared as file representations in various serializations (e.g., Turtle, N3, etc.), supporting the semantics-aware exchange and processing of information. Visualization methods and tools *parse* the textual definition and depict the ontology accordingly to a specific visualization method. Thus, *any* ontology visualization tool has a parsing mechanism and is capable of interpreting ontologies.

Our methodology exploits the above mentioned fact and defines visual representations in annotation *ontologies* that can be used with arbitrary ontologies using `owl:imports` statements. Thus, a separation of concerns is realized for the visual abstraction layer. The visual abstraction is divided into two information layers. The first layer provides *global* visual descriptions for OWL constructs. Accordingly, this layer is independent of the visualized ontology. Conceptual elements will be depicted based on their type (`rdf:type`) assertion to OWL constructs. The second layer provides *local* visual descriptions for elements from the visualized ontology. This layer is designed to describe additional information, such as spatial position and visibility status. Accordingly, this layer is bound to a particular ontology, and its visual descriptions are only valid *locally* for the individual conceptual elements of the visualized ontology. In this article, the terms *mental model*, *global layer*, and *notation* refer to the definition of visual properties for OWL constructs. The terms *mental map*, *local layer*, and *view* correspond to visual properties for conceptual elements of the visualized ontology.

Inspired by the Web Annotation Data Model [17], the methodology uses targeting properties to link representational definitions with OWL constructs and individual elements from the visualized ontology. Visual properties of distinct elements are organized and instantiated in instances of type `owl:NamedIndividual`. All properties used in the methodology are of type `owl:AnnotationProperty`. As illustrated in Figure 2, these instances link visual description to corresponding elements. This conceptualization attaches meta information to resources without modifying them. Thus, it preserves the originality of OWL constructs and the ontology.

Notations – Notations are designed to describe the visual depiction of OWL constructs on the global layer. Named individuals provide the grouped instantiation of visual properties which are linked to individual OWL constructs using the targeting properties. Since annotation ontologies define visual notations, these can be easily exchanged using adjustments of the import statements.

While *domain* ontologies and their graph-based representations are in focus of this work, the methodology in itself is not restricted to only those. Conceptualization form upper-level ontologies can be enriched with visual descriptions in the same fashion. Various visualization methods, such as treemaps and Euler diagrams, could be defined when adequate notations are created and visualization frameworks support that kind of visualization.

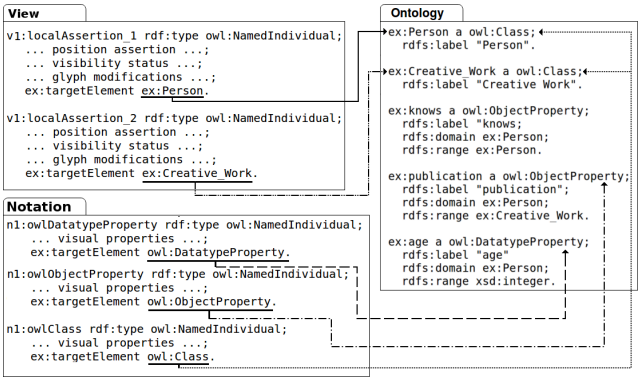

Figure 2: Organization of visual properties in instances and indication of the target property linking conceptualization.

Views – Views are annotation ontologies that hold a set of named individuals that target conceptual elements of the visualized ontology. Views are designed to provide additional information, such as the spatial position and visibility status for the targeted elements. However, the used notation is also essential to obtain valid assertions for the position of the elements. Thus, in our methodology, we enforce views to provide the information for which notation they were created.

Furthermore, views provide *optional* modification of glyph specific information. A glyph is collection of rendering primitives (e.g., shape and label text) for OWL constructs. Since views encode information about the used notations, modified glyphs can be created by applying the visual description of the corresponding OWL construct and then overwriting visual properties in correspondence with the provided glyph modification. For example, elements from the domain ontology can be visually highlighted and grouped by adjustments of their visual attributes (e.g., colors, shapes, and sizes).

Containers – The design of the methodology enables the orchestration of notations and views. Domain ontologies can be enriched with metadata through the use of `owl:imports` (if the annotation ontologies are exposed under a dereferenceable URI). Annotation ontologies for visual representations can be exchanged, shared, reused, and adjusted for the current need of users.

Containers are ontologies that consist only of a set of imported ontologies that correspond to the input ontology for the visualization, notations, and views. Thus, the visual representations can be injected into ontologies without modifying them. Disambiguation between conceptual and visual elements in the combined model can be realized through different namespaces and the conceptualization of the methodology that encapsulates visual properties in instances solely of type `owl:NamedIndividual` and annotation properties.

4.1 Discussion

The methodology is an abstract conceptualization for the description of customizable visual representations of ontologies. The central aspect of the methodology is its utilization of OWL for definitions of visual representation models. Furthermore, it provides conceptualizations for separation of concern in terms of annotation ontologies.

The organization of visual properties in `owl:NamedIndividuals` and the use of linking properties provide the means to enrich any resource with description for its depiction. Thus, the full spectrum of resources, such as OWL constructs and conceptual elements from ontologies can be addressed.

The methodology provides no restrictions concerning which annotation properties are grouped and instantiated in named individuals. This aspect is governed by an implementation of the methodology, such as GizMO. Thus, the expressivity is not restricted by the methodology, but rather by the implementation and the corresponding visualization framework.

The aspect of usability should be governed by such applications and solely be their responsibility. We argue that the corresponding frameworks should address the variety in technological stacks. Visualization frameworks have to interpret the representation models and create the depictions in their specific programming languages. Since the methodology has no restrictions, it could also describe user interactions, such as visual modifications on mouse hovering or even descriptions for mouse click events. Nonetheless, the methodology is only as viable as its realization and implementation in visualization frameworks.

The constraints of the methodology are the use of annotation properties for assignment of visual property values, and the use of `owl:NamedIndividuals` for the grouped assignment. The identification of domain restrictions can realize the disambiguation between conceptual and visual annotation properties. Conceptual instances are of type `owl:NamedIndividual`, however, these are additionally assigned to other OWL constructs. Thus, due to multiple type assertions, these instances belong conceptually to the element defined in the ontology. For example:

```
ex:Author_1   a  owl:NamedIndividual, foaf:Person .
ex:Example_1  a  owl:NamedIndividual .
```

Author_1 becomes an instance of the class `Person`. In contrast, Example_1 is an instance that belongs to `owl:Thing`. We argue that instances of `owl:Thing` and their asserted annotation properties provide a reasonable disambiguation mechanism. Such abstract information does not provide value for the conceptualization of ontologies. Thus, the methodology provides full coverage and disambiguation conceptualizations, while usability and expressivity are aspects that are addressed by corresponding implementations.

5 GIZMO

GizMO is a representation model for the definition of customizable graph-based ontology visualizations. Based on the methodology, GizMO defines visual representations as OWL ontologies. In this section, we provide an overview of our design decisions, our technical realization, and discuss coverage and limitation of GizMO.

5.1 Preliminaries

GizMO builds on on the following concepts:
OWL Constructs – The language constructs of RDF(S) and OWL, e.g., `owl:Class`, `owl:objectProperty`, `rdfs:subClassOf`.
GizMO Core Ontology – Set of defined annotation properties, along with value restrictions. Besides visual properties (e.g., shape, color, and position), it defines further annotation properties used by the representation model.

Annotation Object – Based on the methodology, annotation properties are grouped by instances of type `owl:NamedIndividual`. They provide targeting properties that link visual definitions to corresponding elements. This conceptualization ensures the disjunction for the *domain* ontology and its visualization. Furthermore, the use of targeting properties prevents unnecessary manipulations of OWL constructs and conceptual elements. These named individuals are extended with an additional annotation property which defines their annotation object type. The annotation object types enable the disambiguation of the named individuals for different parts of the representation model. The GizMO representation model uses five annotation object types:

- **Glyph Annotation Object:** Organizes the set of annotation properties that address the visual appearance of OWL constructs.
- **Visualization Annotation Object:** Organizes the set of annotation properties addressing the visual representation of conceptual elements from the domain ontology, including their spatial position assignment, visibility status, and optional modified glyph information.
- **Triple Annotation Object:** Identifies triples from the domain ontology and asserts the visual annotation objects for the corresponding subject, predicate, and object element.
- **Notation Annotation Object:** Holds additional information for the notation, such as a canvas background color.
- **View Annotation Object:** Holds additional information for the view, such as the used notation and viewport configuration (e.g., zoom factor).

Notation – Consists of a *notation annotation object* and a collection of *glyph annotation objects*.
View – Consists of a *view annotation object*, a set of *triple annotation objects*, and a set of *visualization annotation objects*.

5.2 Visual Graph Mapping

Ontology visualization methods and tools apply a mapping that provides a formal definition for the visual representation of OWL constructs. Graph-based visualizations, such as node-link diagrams, represent the concepts of an ontology by a graph $G(N, E)$, where classes typically map to the set of nodes N and their interrelations are described by the set of edges E using the terms of the ontology. These nodes and edges have a visual component for the graphical depiction based on the mapping. However, even for two-dimensional graph-based visualizations, these mappings provide deviations for nested and non-nested representations in the final depiction of the ontology (cf. Figure 1, views A and B).

Furthermore, OWL provides constructs that can be mapped differently (e.g., `a:ClassA owl:equivalentClass b:ClassB.`). One representation could map `a:ClassA` and `b:ClassB` to nodes in the graph, having `owl:equivalentClass` as an edge between the two nodes. Another representation could merge the two classes into one node, thus the OWL construct `owl:equivalentClass` does not have a single visual component, but rather a mapping description for the involved elements.

Some visual notations provide for OWL constructs a visual component and a mapping description. For example, `rdfs:Literal` is a resource that maps to a node that has a visual component. Thus, all datatype properties with this range restriction will provide an edge to this *single* node. Depending on the domain ontology and its complexity, such *simple* mapping could result in an overcrowded and cluttered visualization. Correspondingly, node splitting or glyph multiplications are performed by some notations.

Table 1: OWL constructs and corresponding mappings currently supported by GizMO.

OWL Construct	Mapping
Nodes	
owl:Class	Visual-component-only
owl:Thing	Combination (glyph multiplication)
Links	
owl:ObjectProperty	Visual-component-only
owl:DatatypeProperty	Visual-component-only
rdfs:subClassOf	Combination (glyph multiplication)
Datatypes	
rdfs:Literal	Combination (glyph multiplication)
rdfs:Datatype	Combination (glyph multiplication)
Other	
rdfs:label	Combination (label in domain node)
rdfs:domain	Mapping-description-only
rdfs:range	Mapping-description-only

Ontology mappings can be categorized into three groups: i) visual-component-only creates for an OWL construct a node or an edge; ii) mapping-description-only defines how axioms such as `owl:equivalentClass` are handled; iii) combination of i) and ii).

The methodology does not employ definitions for the mapping of OWL constructs. This design decision transfers the responsibility of the mapping to the corresponding notation and maintains the flexibility of the methodology for various notations.

The current conceptualization of GizMO does not support such customizable mapping definitions. GizMO is limited to a subset of OWL constructs and a fixed implicit mapping. Table 1 provides an overview of support OWL constructs and their mappings.

While GizMO supports only a limited set of OWL constructs and corresponding mappings, its conceptualization provides five additional customizable visual elements that mitigate current constraints. Default elements for nodes and properties describe a visual-component-only mapping for OWL constructs that are not supported by GizMO. Unsupported datatypes are described by a corresponding default element that defines their visual component and enforces a glyph multiplication. Furthermore, GizMO provides descriptions for collapse/expand mechanisms of multiple links between two nodes. The visual representation of collapsed links is provided by one of the additional elements. The mappings for nested visualizations (e.g., UML), are implicitly defined in GizMO and are represented by a nested node description. The use of nested visual representations is defined in the *notation annotation object*.

GizMO has been designed to showcase the applicability of the methodology. Regardless of the set of supported OWL constructs and the implicit mappings, GizMO provides already customizable visualizations of ontologies that can be used to recreate existing notations, such as UML and VOWL.

5.3 Technical Realization and Design Decisions

The technical realization is conceived in correspondence with the methodology. An annotation ontology, the GizMO core ontology[1], defines annotation properties for the GizMO representation model. This ontology provides value restrictions and comments, addressing purpose and usage for each annotation property. Furthermore, it defines five annotation object types (i.e., glyph, visualization, triple, notation, and view annotation object) for the disambiguation of `owl:NamedIndividuals`. GizMO defines additionally rudimentary interaction descriptions for hovering on glyphs. A subset of annotation properties indicating their purpose is shown in Table 2.

The methodology employs separation of concerns for the representation model. Thus, visual depictions are combinations of the associated annotation ontologies (notations and views). The methodology creates these associations using the import mechanisms of OWL. However, defining visual representations for ontologies requires the *authorship* for the ontology. Whereas authors of ontologies can explicitly import notations and views into the ontology and publish it with the desired visualization, containers enable to associate the domain ontology with its visual representation if no authorship is available using the import statements for the individually required ontologies (i.e., domain ontology, notation(s), and view(s)).

Our technical realization builds upon the assumption that any ontology visualization tool has a parsing mechanism. Ontology parsers, such as the OWL-API, load an ontology and its import statements into a combined model. However, in this combined model the identification of elements from the domain ontology (also optional other imported domain ontologies) and the resources corresponding to the visualization is required. Since definitions for visual depiction are grouped in named individuals that use the annotation properties of the GizMO core ontology, our implementation uses its namespace to distinguish between conceptual and visual elements. The disambiguation between named individuals of the different annotation ontologies is provided by their namespaces and the annotation object types, e.g., the glyph annotation object type corresponds to a named individual of a notation. Figures 3 and 4 show examples of instantiated named individuals for notations and views in the GizMO representation model.

[1]https://github.com/gizmo-vis/gizmo/blob/master/coreOntology/gizmoCore.ttl

Table 2: Subset of GizMO annotation properties.

GizMO property	Description
Linking Properties	
`targetElement`	Linking visual properties to OWL constructs.
`subjectElement`, `predicateElement`, `objectElement`	Graph triple pattern definition.
`subjectDescription`, `predicateDescription`, `objectDescription`	Definition of visual properties for corresponding rendering primitives in views.
Visual Properties	
`renderingType`	Specification of geometric shapes.
`width, height, radius`	Specification of geometric shape characteristics.
`bgColor, strokeElement, strokeWidth`	Specification of visual property characteristics.
`position_x, position_y, visible`	Spatial information and visibility status.
Annotation Objects	
`annotationObjectDescription`	Description for its purpose.
`isTypeOf`	Specification into glyph, visualization, triple, notation, and view annotation object.

```
########## owl:Class ##########
vowl:owlClass rdf:type owl:NamedIndividual;
    gizmo:renderingType "circle"^^xsd:string;
    gizmo:bgColor       "#aaccff"^^xsd:string;
    gizmo:radius        "50"^^xsd:positiveInteger;
    gizmo:strokeElement "true"^^xsd:boolean;
    gizmo:strokeStyle   "solid"^^xsd:string;
    gizmo:strokeWidth   "2"^^xsd:positiveInteger;
    gizmo:strokeColor   "#000000"^^xsd:string;
    gizmo:fontFamily    "Helvetica,Arial,sans-serif"^^xsd:string;
    gizmo:fontColor     "#000000"^^xsd:string;
    gizmo:fontSize      "12px"^^xsd:string;
    gizmo:hoverInCursor "pointer"^^xsd:string;
    gizmo:hoverInColor  "#ff0000"^^xsd:string;
    gizmo:isTypeOf      gizmo:GlyphAnnotationObject;
    gizmo:targetElement owl:Class .
```

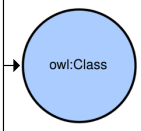

Figure 3: Named individual for visualization of `owl:Class`.

Since OWL does not provide a conceptualization of an "annotation object" that groups annotation properties, our realization of the GizMO uses a simplified assertion of visual properties, such as `xsd:` datatypes. Visual properties are annotation properties that are grouped in instances solely of the type `owl:NamedIndividual`. Thus, instances for the visualizations become automatically instances of `owl:Thing`, which are on a higher abstraction layer in comparison to elements from the domain ontology. Such simplified assertion has a practical advantage: GizMO does not introduce a single class. This has the benefit that our representation model will not conflict with other visualization tools. For example, the class tree of Protégé will not be cluttered with elements that correspond to the visualization.

Views – Views require more refined considerations of the linking property approach. Definitions of domain ontologies use OWL constructs, such as `rdfs:Literal`, `rdfs:subClassOf`, etc., multiple times. As discussed in Section 5.2, these are typically represented as multiple glyphs. Since multiple glyphs correspond to a single OWL construct, a bijective mapping using *only* a single linking property is not possible (cf. Figure 4).

GizMO solves this challenge through the use of multiple linking properties in the *triple annotation object*. Any ontology can be serialized in a triple format (e.g., N3), such that it consists of a set of subject, predicate, and object triples. Since OWL uses URIs for resource descriptions, a triple itself is unique. A triple annotation object provides three distinct linking properties for subject, predicate, and object. Three additional linking properties (e.g., `gizmo:subjectDescription`) point to *visualization annotation objects*. These provide the assigned values for their spatial position, visibility status, and optional glyph modification information. Thus, a unique assertion is ensured, even for multiple uses of identical OWL constructs. Figure 4 shows the definition of two *triple annotation objects* and their corresponding *visualization annotation objects* that define positions and glyph modifications.

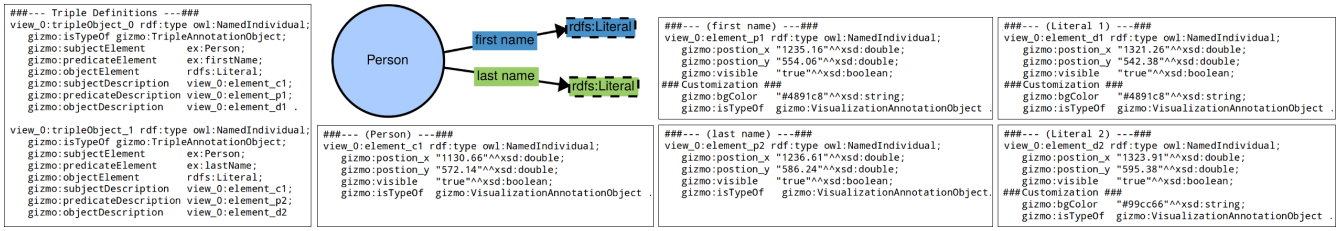

Figure 4: Disambiguation for glyph specific information with additional glyph modifications for some objects (e.g., first name).

6 IMPLEMENTATION

The utilization of the methodology, however, can only be achieved when tools and frameworks are extended towards the interpretation of annotation ontologies that define visual representation models. In the following, we briefly describe two applications[2] that are operating on the GizMO representation model.

Both applications use the same rendering engine and *partially* implement the semantic zooming approach for ontology graphs [15]. These applications use an implicit mapping for a defined subset of OWL constructs and default elements for not mapped elements. Furthermore, the collapse/expand mechanisms for datatypes, datatype properties, and object properties are used to describe nested node visualizations. Whereas the semantic zooming approach removes collapsed elements from the visualization, our frameworks render the collapsed elements with their visual descriptions inside the corresponding node. Notations specify the collapse/expand state *globally* in the notation annotation object.

Notation Editor – The notation editor[3] is designed to remove the textual crafting of notations. It enables users to visually create definitions for the representations of OWL constructs in a WYSIWYG manner. Implemented as a proof of concept prototype, the usability, richness of features, and user experience are not in focus. Created notations are exported as annotations ontologies and can be shared, reused, and integrated using `owl:imports` statements.

Visualization Framework – The framework[4] is designed for the visualization of ontologies with the GizMO representation model. Additionally, it provides the means to create views and containers. Domain ontologies (also containers) and notations can be loaded independently. The visualization framework has a data processing pipeline and a visualization pipeline. The data processing pipeline reads an ontology and organizes the comprised information into domain ontology, notations, and views (if available). In the case of missing notations and views, the default notation is used, and the spatial arrangement is created automatically.

The visualization pipeline receives the processed data and creates customizable glyph objects for the elements from the domain ontology. These glyph objects are initialized with a default notation. Based on the loaded notation, these are overwritten with the definition asserted in the glyph annotation objects. The visualization pipeline then continues with view definitions. Based on the loaded view, the information for the position, visibility status, and optional glyph modifications are updated in the corresponding glyphs. Additionally, the view annotation object asserts the viewport configuration (e.g., zoom factor).

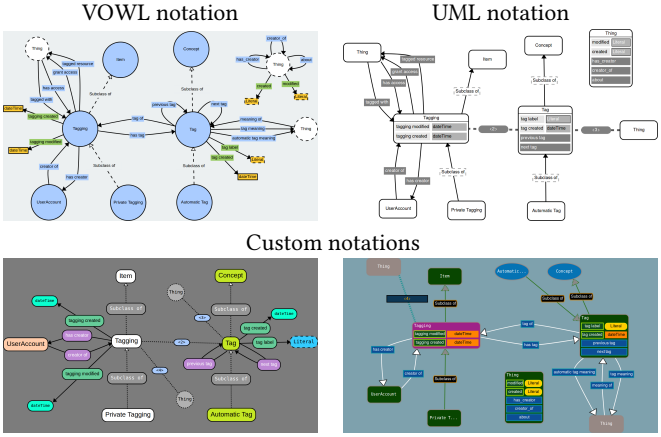

Figure 5: Examples created with GizMO, realizing VOWL, UML, and custom notations.

The current implementation employs only a force-directed layout algorithm. Other layout algorithms pose an implementation effort and are not contributing to the general aspect of this work. However, users have the option to pause the force-directed layout process, manually align the layout and save it as a view. The pause/play state of the force-directed layout is saved in the view annotation object. Descriptions for the usage and features of the framework can be found on the corresponding landing page[2].

7 CONCLUSION

Visualizations of ontologies can support the development, exploration, communication, and sense-making. However, suitable visualizations are highly dependent on individual use cases and targeted user groups. Flexible and customizable approaches are required in order to allow users to adjust visual representations to their needs.

In this work, we have presented a methodology for customizable visual representations of ontologies. The central aspect of the methodology is its utilization of OWL for definitions of visual representation models. The methodology separates the visual abstraction into two layers: The *global* layer reflects the mental model of users and addresses the customizable visual representation of OWL constructs. The *local* layer addresses the mental map of users and provides the means to customize the spatial arrangement, visibility status, and optional glyph modifications.

[2]Landing Page: https://gizmo-vis.github.io/gizmo/
[3]Notation Editor: https://gizmo-vis.github.io/gizmo/notationEditor/
[4]Visualization Framework: https://gizmo-vis.github.io/gizmo/visualizationFramework/

The applicability of the methodology is demonstrated through GizMO, a representation model for graph-based visualizations of ontologies. The GizMO core ontology defines annotation properties for visual attributes (e.g., shapes, colors, positions, etc.). Annotation objects provide grouped instantiations of values that are linked to OWL constructs and elements of the domain ontology. Different types of annotation objects target various aspects of the visualization. These provide a conceptual separation between the *global* and *local* layers for the visual representation.

Through the use case of GizMO, we show that the customizable visual mapping of OWL constructs poses a particular challenge. GizMO implements a fixed implicit mapping for OWL constructs in the context of a minimal viable product (MVP) prototype. However, customizable visual representation models have to provide definitions for the mapping of OWL constructs. For example, the merging of nodes that are linked via the `owl:equivalentClass` property or the multiplication of glyphs. Future work and refinement of GizMO will address the customizable mapping of OWL constructs.

GizMO indicates additional requirements for the disambiguation of multiplied glyphs. OWL constructs, such as `rdfs:Literal`, are used multiple times. This poses the challenge for identifying corresponding glyphs in the visualization. GizMO addresses this challenge through the use of triple annotation objects. These identify the corresponding triples and assert for subject, predicate, and object elements from the domain ontology the corresponding visualization annotation objects. Visualization annotations objects define the spatial information, visibility status, and optional glyph modifications. The visibility flag fosters the reduction of cognitive load by excluding elements from the visualization. The optional glyph modifications enable the customization of visual attributes (e.g., colors and shapes) for elements of the domain ontology.

The design decisions for the methodology and the technical realization of GizMO are conceived to facilitate the customizable definitions for the visual representation of ontologies. The success of the methodology and GizMO depends on the integration into other frameworks and tools. GizMO is currently limited in its coverage of OWL constructs and the implicit mapping. Some visual representations cannot merely be described by visual appearance and spatial arrangement of glyphs. We describe the UML-based notations using collapsing and expanding mechanisms, whereas other notations may require additional behavioral descriptions (e.g., edge bundling). Regardless of its limitations, Figure 5 illustrates the variety of already possible visualizations.

Future work will address the refinement of the GizMO core ontology and its coverage of OWL constructs. The customizable mapping for OWL constructs, such as `owl:equivalentClass`, will be extended in order to enable the definition for the merging of classes into one node. GizMO is one use case of the methodology and applies only to graph-based visualizations. The definition of other notation types, such as treemaps or chord diagrams, using our methodology is another topic for future work. However, only the implementation of frameworks that can operate on the methodology for visual representation models will result in meaningful and suitable visualizations of ontologies. Additionally, we plan to add further layout algorithms and interaction mechanisms to the framework so that it can serve as a comprehensive reference implementation for GizMO in the future.

The abstract nature of the methodology and the provided discussion in Section 4.1 indicate the induced responsibility for realizations, such as GizMO. One the one hand, GizMO has been designed in the context of an MVP, and its limitations are prevalent towards the customizable mappings. On the other hand, the evaluation of the usability and richness of features will result in an assessment of the implemented visualization frameworks and not GizMO nor the methodology. Thus, this work postpones the evaluation of GizMO for future work.

In conclusion, we hope that the methodology will foster the creation of custom visual representations for ontologies and be useful to researchers, ontology engineers, and domain experts.

Acknowledgement. This work is co-funded by the European Research Council project ScienceGRAPH (Grant agreement #819536). In addition, parts of it evolved in the context of the Fraunhofer Cluster of Excellence "Cognitive Internet Technologies". Additionally we would like to thank Maria-Esther Vidal, Mikhail Galkin, and Christian Mader for valuable discussions, comments, suggestions, and guidance.

REFERENCES

[1] Chaomei Chen. 2002. *Visualizing the Semantic Web: XML-Based Internet and Information Visualization.* Springer.
[2] Nelson Cowan. 2000. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences* 24 (2000), 87–185.
[3] Stephen Cranefield and Martin K. Purvis. 1999. UML as an Ontology Modelling Language. In *Intelligent Information Integration (CEUR Workshop Proceedings)*, Vol. 23. CEUR-WS.org. http://ceur-ws.org/Vol-23/
[4] Marek Dudáš, Steffen Lohmann, Vojtěch Svátek, and Dmitry Pavlov. 2018. Ontology visualization methods and tools: a survey of the state of the art. *Knowledge Eng. Review* 33 (2018), e10.
[5] Riccardo Falco, Aldo Gangemi, Silvio Peroni, David Shotton, and Fabio Vitali. 2014. Modelling OWL Ontologies with Graffoo. In *The Semantic Web: ESWC 2014 Satellite Events (LNCS)*, Vol. 8798. Springer, 320–325.
[6] Peter Haase, Saartje Brockmans, Raul Palma, Jérôme Euzenat, and Mathieu d'Aquin. [n.d.]. The NeOn UML Profile for Networked Ontologies. http://neon-project.org/deliverables/WP1/NeOn_2007_D1.1.2.pdf.
[7] Holger Knublauch. [n.d.]. Graphical Ontology Editing with TopBraid Composer's Diagram Tab. https://www.topquadrant.com/2012/06/29/graphical-ontology-editing-with-topbraid-composers-diagram-tab/.
[8] Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. 2016. Visualizing Ontologies with VOWL. *Semantic Web* 7, 4 (2016), 399–419.
[9] Stefan Negru, Florian Haag, and Steffen Lohmann. 2013. Towards a unified visual notation for OWL ontologies: insights from a comparative user study. In *I-SEMANTICS 2013 - 9th International Conference on Semantic Systems, ISEM '13, Graz, Austria, September 4-6, 2013.* 73–80.
[10] Emmanuel Pietriga. 2003. IsaViz: A visual authoring tool for RDF. http://www.w3.org/2001/11/IsaViz.
[11] Emmanuel Pietriga. 2006. Semantic Web Data Visualization with Graph Style Sheets. In *Proceedings of the 2006 ACM Symposium on Software Visualization (SoftVis '06).* ACM, New York, NY, USA, 177–178.
[12] Emmanuel Pietriga, Christian Bizer, David R. Karger, and Ryan Lee. 2006. Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In *5th International Semantic Web Conference, ISWC 2006 (LNCS)*, Vol. 4273. Springer, 158–171.
[13] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. 2003. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research* 13, 11 (2003), 2498–2504.
[14] Frank van Harmelen and Deborah McGuinness. 2004. *OWL Web Ontology Language Overview.* W3C Recommendation. W3C. http://www.w3.org/TR/2004/REC-owl-features-20040210/.
[15] Vitalis Wiens, Steffen Lohmann, and Sören Auer. 2017. Semantic Zooming for Ontology Graph Visualizations. In *Proceedings of the Knowledge Capture Conference, K-CAP 2017.* ACM, 4:1–4:8.
[16] John R. Wilson and Andrew Rutherford. 1989. Mental models: Theory and application in human factors. *Human Factors* 31, 6 (1989), 617–634.
[17] Benjamin Young, Robert Sanderson, and Paolo Ciccarese. 2017. *Web Annotation Data Model.* W3C Recommendation. W3C.