Received: 16 November 2018 | Revised: 8 January 2019 | Accepted: 25 January 2019
DOI: 10.1111/exsy.12380

WILEY Expert Systems

SPECIAL ISSUE PAPER

Onyx: A new Canvas-based tool for visualizing biomedical and health ontologies

Övünç Öztürk | **Hasan Gökhan Açikgöz**

Department of Computer Engineering, Manisa Celal Bayar University, 45140, Yunusemre, Manisa, Turkey

Correspondence
Övünç Öztürk, Department of Computer Engineering, Manisa Celal Bayar University, 45140, Yunusemre, Manisa, Turkey.
Email: ovunc.ozturk@cbu.edu.tr

Abstract

Ontologies provide formal, machine-readable, and human-interpretable representations of domain knowledge. Therefore, ontologies have come into question with the development of Semantic Web technologies. People who want to use ontologies need an understanding of the ontology, but this understanding is very difficult to attain if the ontology user lacks the background knowledge necessary to comprehend the ontology or if the ontology is very large. Thus, software tools that facilitate the understanding of ontologies are needed. Ontology visualization is an important research area because visualization can help in the development, exploration, verification, and comprehension of ontologies. This paper introduces the design of a new ontology visualization tool, which differs from traditional visualization tools by providing important metrics and analytics about ontology concepts and warning the ontology developer about potential ontology design errors. The tool, called Onyx, also has advantages in terms of speed and readability. Thus, Onyx offers a suitable environment for the representation of large ontologies, especially those used in biomedical and health information systems and those that contain many terms. It is clear that these additional functionalities will increase the value of traditional ontology visualization tools during ontology exploration and evaluation.

KEYWORDS

biomedical and health ontologies, e-health, ontology visualization, OntoQA, OOPS! pitfall scanner, OWL, Semantic Web

1 | INTRODUCTION

The Semantic Web is an extension of the current Web in which information is given a well-defined meaning, which better enables computers and people to work in cooperation (Berners-Lee, Hendler, & Lassila, 2001). It is generally accepted that the integration of the Semantic Web, artificial intelligence, and machine learning technologies will lead to significant changes in the field of economics and knowledge management. Ontologies are important for Semantic Web vision, where they are used to annotate websites for machine interpretation. As a result, ontology engineering has recently become an important research area. However, the development and management of ontologies by only experienced people slows progress in this area, as it is also important for casual users to read, write, and modify ontologies. In this context, ontology visualization tools have become a solution. It has been observed that integrating ontology visualization tools into ontology development tools such as Protégé (Noy, Fergerson, & Musen, 2000) is not preferred by casual users because the tools require prior knowledge of description logic or syntactic details about ontology development. Therefore, stand-alone web-based ontology visualization tools such as WebVOWL (Lohmann, Link, Marbach, & Negru, 2015) are becoming popular. This type of tool enables users to understand the ontology model without prior knowledge, which can help expand and diversify the userbase.

Within this work, a new stand-alone web-based ontology development tool called Onyx is presented. Unlike traditional ontology visualization tools, Onyx analyses ontology concepts and reports ontology design pitfalls in addition to graphically visualizing the ontology. The tool has the

[1]This research was done while the second author was an undergraduate student at Manisa Celal Bayar University.

following additional components: (a) metric calculator: returns a summary of the metrics about an ontology concept; (b) pitfall scanner: returns pitfalls, which could represent or lead to ontology design errors. Onyx has advantages in terms of speed, analysis, error detection, and readability. Thus, it is envisioned that the tool will be adopted by both ontology engineers and casual users in the field. Additionally, Onyx is proposed to be a solution for displaying large ontologies, which are particularly common in the biomedical and health domains.

The remainder of this article is structured as follows. Section 2 introduces ontologies, ontology evaluation metrics, ontology design pitfalls, and the technical details and technologies regarding the implementation of the tool. Section 3 defines the visual elements, colour scheme of the Onyx specification, and the visualization and pitfalls of ontology analysis. Section 4 evaluates the proposed tool and compares it with existing ontology visualization tools. Finally, the last section concludes the paper with a summary and future directions.

2 | BACKGROUND KNOWLEDGE AND TECHNICAL DETAILS

In Studer, Benjamins, and Fensel (1998), the definitions of ontology of Gruber (1993) and Borst (1997) are merged as follows: An ontology is a formal, explicit specification of a shared conceptualization. This definition requires that the conceptualization should express a shared view between several parties, that is, a consensus rather than an individual view; this conceptualization should also be expressed in a machine-readable format. Ontologies play a key role in the vision of the Semantic Web, and there is a growing need for effective ontology visualization for design, management, and browsing (Katifori, Halatsis, Lepouras, Vassilakis, & Giannopoulou, 2007).

In this article, we present a new ontology visualization tool called Onyx. In the implementation phase, we used Meteor Framework (Meteor Development Group, first release: first release: 2012). Meteor is an open-source JavaScript web framework written using Node.js. Ontology is stored in the Cayley (Michener & Community, 2014), which is an open-source graph database. Cayley's goal is to be part of the developer's toolbox that handles linked data and graph data. The user interface was developed using the React.js library (Walke & Popov, 2011). Communication between Web applications and the Cayley Graph Database is implemented via Gremlin queries. Gremlin (TinkerPop, first release: first release: 2009; Rodriguez, 2015) is a domain specific language for traversing property graphs.

Figure 1 shows the architecture of the Onyx tool. The Web Application has two main components: the metric calculator and the pitfall scanner. These modules are JavaScript applications that are executed automatically after loading the ontology using Gremlin queries. After executing Gremlin queries, the query results are stored in a MongoDB database and visualized using the Cytoscape.js library.

In this work, we used Cytoscape.js (Franz et al., 2015; Shannon et al., 2003), which is a JavaScript library for developing complex interactive network visualizations. As a result, the tool can be used with both desktop browsers (e.g., Chrome) and mobile browsers. Additionally, our mobile application supports interactivity and actions such as zoom in/out, box selection, and panning.

Currently, most preferred web graphics technologies are SVG and Canvas. WebVOWL-style web-based visualization tools use older SVG technologies. The Cytoscape.js library supports Canvas, and according to our research, Onyx is the only ontology visualization tool that uses Canvas technology.

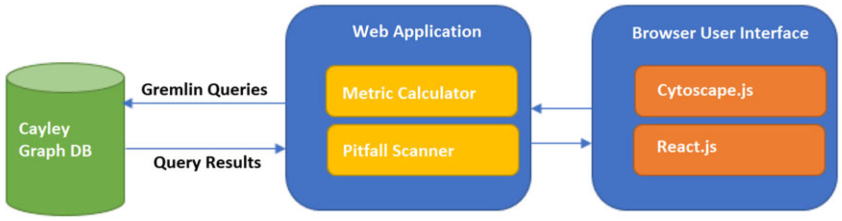


FIGURE 1 Onyx ontology visualization tool architecture

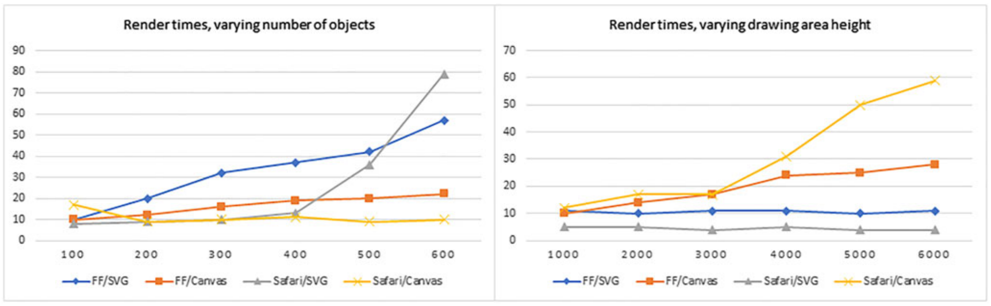
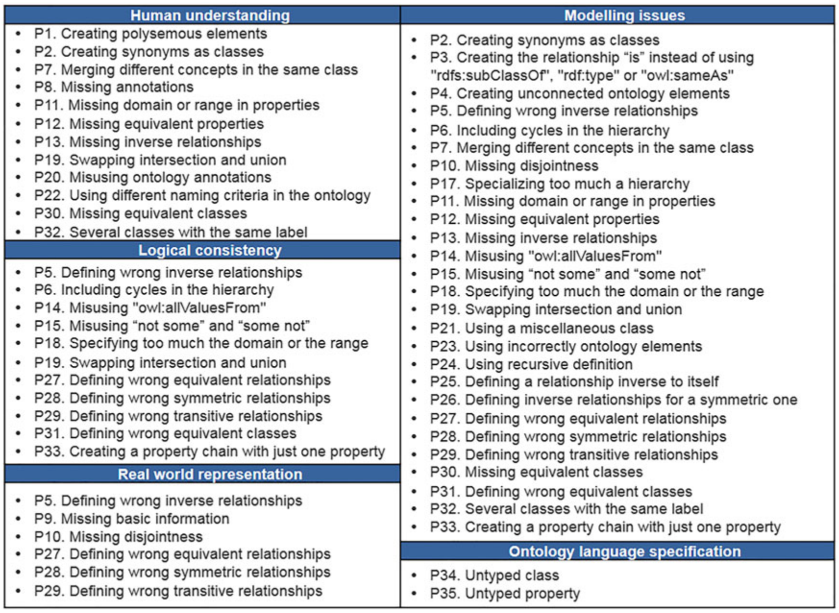


FIGURE 2 Comparing the performances of SVG and Canvas technologies



Human understanding

- P1. Creating polysemous elements
- P2. Creating synonyms as classes
- P7. Merging different concepts in the same class
- P8. Missing annotations
- P11. Missing domain or range in properties
- P12. Missing equivalent properties
- P13. Missing inverse relationships
- P19. Swapping intersection and union
- P20. Misusing ontology annotations
- P22. Using different naming criteria in the ontology
- P30. Missing equivalent classes
- P32. Several classes with the same label

Logical consistency

- P5. Defining wrong inverse relationships
- P6. Including cycles in the hierarchy
- P14. Misusing "owl:allValuesFrom"
- P15. Misusing "not some" and "some not"
- P18. Specifying too much the domain or the range
- P19. Swapping intersection and union
- P27. Defining wrong equivalent relationships
- P28. Defining wrong symmetric relationships
- P29. Defining wrong transitive relationships
- P31. Defining wrong equivalent classes
- P33. Creating a property chain with just one property

Real world representation

- P5. Defining wrong inverse relationships
- P9. Missing basic information
- P10. Missing disjointness
- P27. Defining wrong equivalent relationships
- P28. Defining wrong symmetric relationships
- P29. Defining wrong transitive relationships

Modelling issues

- P2. Creating synonyms as classes
- P3. Creating the relationship "is" instead of using "rdfs:subClassOf", "rdf:type" or "owl:sameAs"
- P4. Creating unconnected ontology elements
- P5. Defining wrong inverse relationships
- P6. Including cycles in the hierarchy
- P7. Merging different concepts in the same class
- P10. Missing disjointness
- P17. Specializing too much a hierarchy
- P11. Missing domain or range in properties
- P12. Missing equivalent properties
- P13. Missing inverse relationships
- P14. Misusing "owl:allValuesFrom"
- P15. Misusing "not some" and "some not"
- P18. Specifying too much the domain or the range
- P19. Swapping intersection and union
- P21. Using a miscellaneous class
- P23. Using incorrectly ontology elements
- P24. Using recursive definition
- P25. Defining a relationship inverse to itself
- P26. Defining inverse relationships for a symmetric one
- P27. Defining wrong equivalent relationships
- P28. Defining wrong symmetric relationships
- P29. Defining wrong transitive relationships
- P30. Missing equivalent classes
- P31. Defining wrong equivalent classes
- P32. Several classes with the same label
- P33. Creating a property chain with just one property

Ontology language specification

- P34. Untyped class
- P35. Untyped property

FIGURE 3 OOPS! Ontology pitfall catalogue

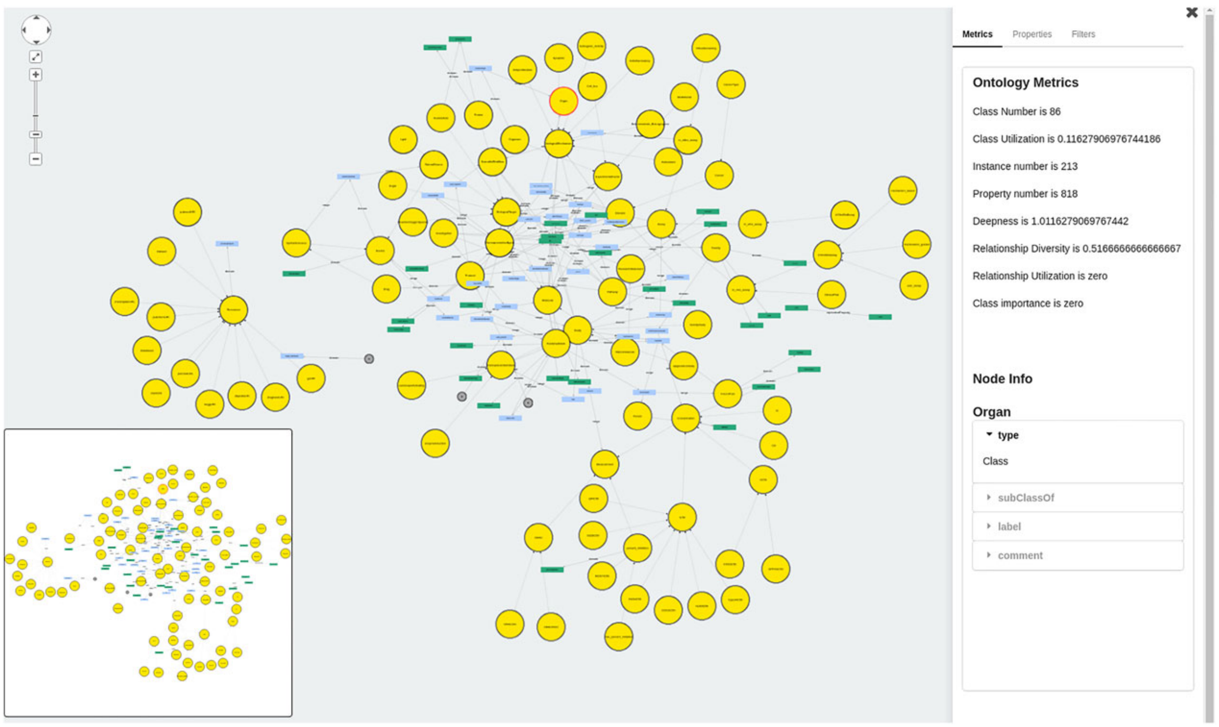
In Figure 2, the performances of the SVG and Canvas technologies are tested with a programme that draws lines with circles in a drawing area at a fixed size.[1] As seen, Canvas technology provides better results in terms of performance. Therefore, the performance of our ontology visualization tool is predicted to be better than those of tools based on SVG technology, especially as the number of objects in the drawing area increases.

Another useful aspect of the tool is that OntoQA ontology metrics are calculated and displayed to the user (on the right of the screen). OntoQA (Tartir, Arpinar, Moore, Sheth, & Aleman-meza, 2005) is a well-known and widely used metric-based ontology evaluation approach. The metric calculator component of Onyx calculates the following OntoQA metrics:

1. Schema metrics
   (a) Relationship diversity: the number of noninheritance relations divided by the total number of relationships defined in the schema
   (b) Deepness (SD): the average number of subclasses per class
2. Knowledgebase (KB) metrics
   (a) Class utilization (CU): the number of instantiated classes divided by the total number of classes defined in the schema
3. Class-specific metrics
   (a) Class importance (Imp(Ci)): the number of direct or indirect instances of the class divided by the total number of class instances defined in the schema.
   (b) Relationship utilization (RU): the number of attributes used by class instances divided by the number of attributes defined for the class.
4. Relationship-specific metrics
   (a) Relationship importance (Imp(Ri)): the number of relationship individuals divided by the total number of relationship individuals defined in the schema.

Another feature of the Onyx tool is the visualization of ontology pitfalls. OOPS! (Poveda-Villalón, Gómez-Pérez, & Suárez-Figueroa, 2014) proposes a catalogue of ontology pitfalls to improve the technical quality of the ontology. In the catalogue, 35 pitfalls are proposed to facilitate ontology evaluation; these faults are examined under five categories (Figure 3). The RESTful web service[2] is used to integrate the features of OOPS! into Onyx. This API can scan all pitfalls except P_1, P_9, P_{14}, P_{15}, P_{16}, P_{17}, and P_{18}. However, there is ongoing work to identify the remaining eight pitfalls.

[1]Test results are taken from http://smus.com/canvas-vs-svg-performance/.
[2]http://oops-ws.oeg-upm.net/rest.



Ontology Metrics

Class Number is 86

Class Utilization is 0.11627906976744186

Instance number is 213

Property number is 818

Deepness is 1.0116279069767442

Relationship Diversity is 0.5166666666666667

Relationship Utilization is zero

Class importance is zero

Node Info

Organ

type

Class

subClassOf

label

comment

FIGURE 4 Visualization of the Cancer Chemoprevention Ontology (CANCO) in Onyx

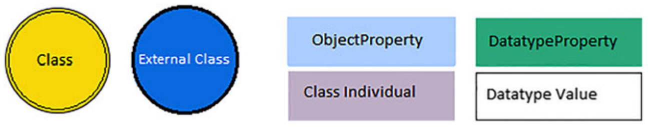


FIGURE 5 Visual elements from the Onyx specification

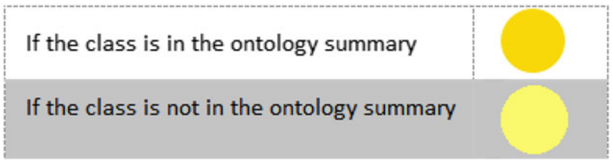

If the class is in the ontology summary
If the class is not in the ontology summary

FIGURE 6 Colour scheme for the classes

3 | ONYX ONTOLOGY VISUALIZATION TOOL

Onyx ontology visualization tool[3][4] focuses on the visualization of the ontology schema TBOX. In terms of scalability and readability, ABOX, which contains the instances of the ontology, is not visualized. This approach, namely, TBOX-based visualization, is adopted by most current ontology visualization tools. Concept instances, literals, and XML data types are visualized if and only if they are used in a concept definition. Figure 4 shows the Cancer Chemoprevention Ontology (CANCO) in Onyx. The CANCO constitutes a vocabulary that can describe and semantically interconnect the different paradigms of the cancer chemoprevention domain.[5] The mini-map of the ontology is shown on the bottom-left corner of the screen, and the zoom component is in the upper-left corner. The panel, which is on the right of the screen, allows metrics to be viewed and filters to be applied. When the Metrics Tab is selected, ontology-related metrics are shown on the top of the panel. The bottom of the panel shows the metrics of the selected node. The Filters Panel allows restrictions or union/intersection classes to be filtered from the graph. The user can also choose to view the ontology taxonomy using this tab. The Filters Panel also allows the pitfalls of the ontology to be listed. When a pitfall is selected from the list, nodes with this pitfall are shown in the graph area. Figure 5 shows the visual elements of the Onyx specification. An ontology class is represented by a circle, and a property is represented by a rectangle.

[3] Run the application from: http://swlab.cbu.edu.tr/onyx.
[4] Download the source code from: https://github.com/mcbu-semanticweb-lab/onyx.
[5] http://bioportal.bioontology.org/ontologies/CANCO.

Onyx visualizes the language constructs of the OWL Web Ontology language specification (Bechhofer et al., 2004) as follows:

a. Simple classes: Onyx visualizes classes as circles. The classes in the ontology are divided into two groups according to their importance, and classes with different importance groups are shown with different colours, as shown in Figure 6.
b. Simple properties: Properties in the schema are visualized as directed solid lines between two classes or between a class and a literal or XML datatype. If the property is an Object Property, then it relates two classes. We put a blue rectangular label, which shows the name and characteristics of the Object Property, on the property line. If the property is a Datatype Property, then it relates a class and a literal

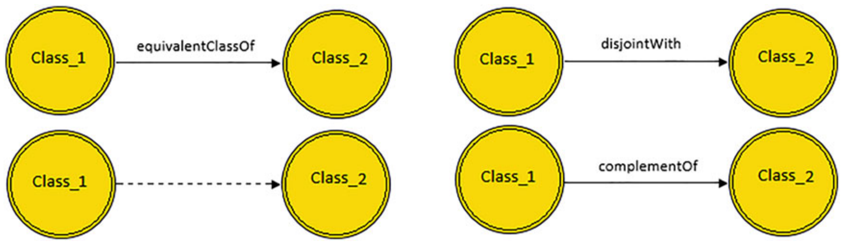


FIGURE 7 Visualization of some special properties from the OWL vocabulary

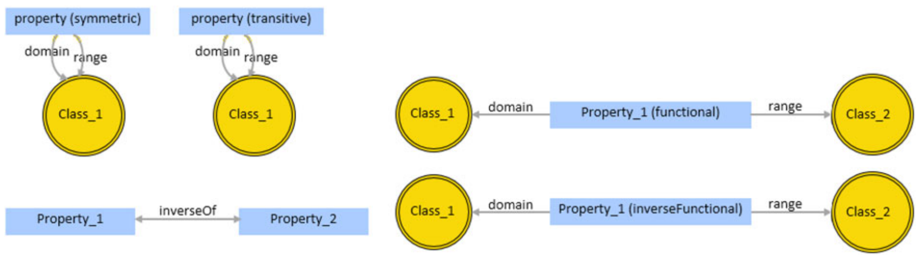


FIGURE 8 Visualization of OWL property characteristics

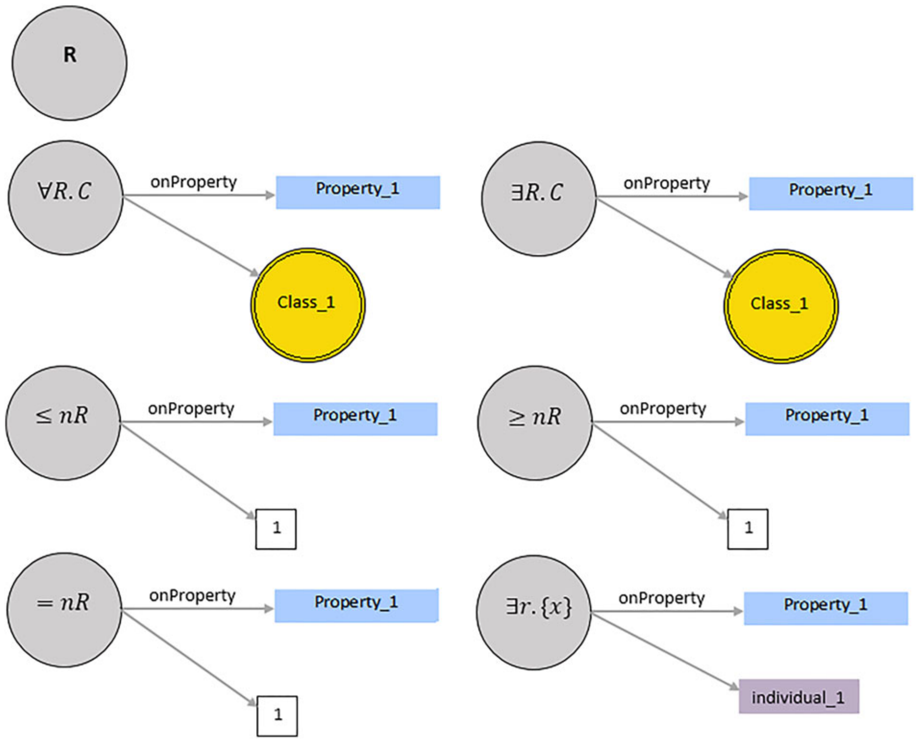


FIGURE 9 Visualization of property restrictions

or XML datatype. We put a green rectangular label, which shows the name and characteristics of the Datatype Property, on the property line. The rest of the properties from the OWL vocabulary are visualized, as in Figure 7. We do not use labels for these properties. Because the subclassOf relation has a different significance, it has a special notation. The subclassOf relation is shown with a dashed line.

c. Property characteristics: OWL language defines five property characteristics: TransitiveProperty, SymmetricProperty, FunctionalProperty, inverseOf, and inverseFunctionalProperty. Property characteristics are visualized, as in Figure 8.
d. Property restrictions: Different from existing tools, Onyx supports the visualization of the details of the restriction classes. OWL language defines four types of property restrictions: allValuesFrom, someValuesFrom, cardinality, and hasValue. All restrictions are represented by a grey circle labelled R. Whenever the grey circle is clicked, the definition of the restriction is expanded. Onyx visualizes the property restrictions, as in Figure 9. Showing restriction classes on the graph may cause the graph to become cluttered; therefore the user has an option to filter the restrictions.
e. Enumerated classes, intersection, union: A class may be defined by listing all of its instances in the OWL language. OWL also supports defining a class using intersection or union operators. These language constructs are used with the help of a collection construct, and they are visualized, as in Figure 10.

Onyx displays the OntoQA metrics in the window on the right of the screen. The window that shows the schema and knowledgebase metrics are always on top of the screen. However, class/attribute-specific metrics are shown when the corresponding class/attribute is clicked.

Onyx gets the pitfall report from OOPS! RESTful web service and lists the name, detailed description, and importance level (i.e., minor, important, critical) of each pitfall in the Filters Panel. When the user selects a pitfall from the list, if the pitfall is related to a class, then the class is highlighted in red. Similarly, if the pitfall is related to a property, then the line of the property is shown in red (Figure 11). A sample report of a pitfall from OOPS! RESTful web service is shown below:

1. Pitfall name: P_5 (Defining wrong inverse relationships.)
2. Pitfall description: Two relationships are defined as inverse relations when they are not necessarily inverse relations. For example, something is sold or something is bought; in this case, the relationships isSoldIn or isBoughtIn are not inverse.
3. Importance level: Critical

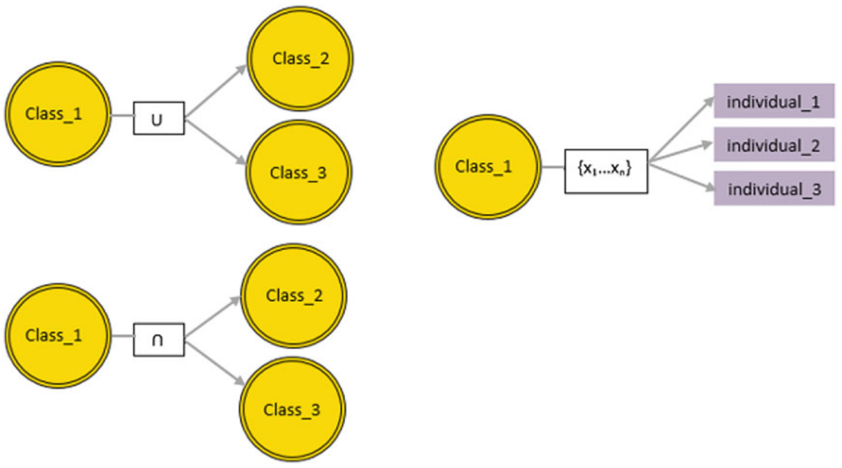


FIGURE 10 Visualization of enumerated classes, intersection, and union

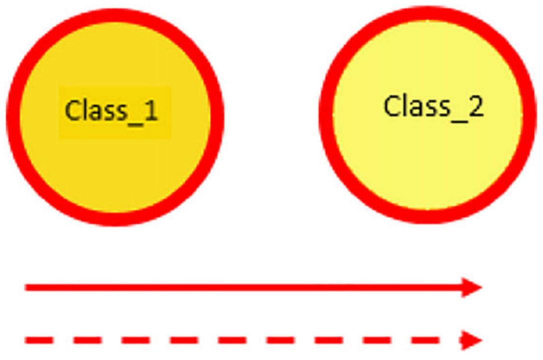


FIGURE 11 Visualization of ontology pitfalls

One of the strong features of the Onyx tool that it has the option of selecting the most important classes, which are called key concepts. This option is especially important for visualizing large ontologies. Onyx starts by asking the user if he/she would like to enable key concept extraction. The key concept extraction algorithm (Peroni, Motta, & d'Aquin, 2008) selects the key concepts in the ontology. The user starts with the most important part in the ontology and gradually expands the nodes to see the concepts that are not in the key concepts list. Users can also retract a node to avoid seeing concepts that he/she is not interested in. Key concepts and regular concepts are visualized with different colours (Figure 6). Enabling key concept extraction option prevents a cluttered view in the graph area and ensures better scaling with large ontologies. However, the user can disable this option when visualizing smaller ontologies. Disabling this option facilitates the comprehension of the overall view of the ontology.

The user can also focus on a selected entity using the Show Neighbourhood button on the Filters Panel. This option centres the view on the selected entity and its surroundings and hides other parts of the ontology.

4 | EVALUATION

In Saghafi (2016), ontology visualization tools are classified into three groups: (a) graph-based methods, which draw the ontology as a graph; (b) multimethod visualization techniques, which draw the ontology using blocks, lists, linked histograms, hierarchical connected circles, node-link diagrams, hyperbolic or radial trees, tree maps, and so forth; and (c) Euler diagram-based methods, which visualize the topological properties of ontologies. Onyx is a graph-based ontology visualization tool that can be used to gain a general overview of ontologies. We compared Onyx with existing visualization tools in the following subsections using three criteria that are preferred for evaluating ontology visualization tools.

4.1 | Functionality

In this section, we evaluate the general features of Onyx. Dudás, Zamazal, and Svátek (2014) identify the following 17 features implemented in ontology visualization tools:

1. Zoom-out overview (F1): zooming out to have a better overview of the ontology. Onyx uses a zoom scale bar to control the zoom factor.
2. Radar view (F2): displaying a mini-map of the displayed ontology. Onyx displays the mini-map on the bottom-left corner of the screen.
3. Graphical zoom (F3): zooming to extend the graphical element. Graphical zoom is adjusted using the zoom scale bar.
4. Focus on selected entity (F4): focusing the view on a selected entity and its surroundings. Onyx supports showing the direct neighbours of a class option. This option focuses on a selected class and shows only the direct neighbours of this class.
5. History (F5): tracking the actions performed by the user to support undo/redo actions.
6. Pop-up window (F6): displaying details of a selected entity. Onyx displays the details of the selected entity in the Metrics panel.
7. Incremental exploration (F7): starting with a small part of the ontology and gradually expanding the nodes selected by the user. Onyx has the "enable key concept extraction" option, which displays only the most important concepts in the ontology. The user can gradually expand the nodes to see the concepts that are not in the key concepts list.
8. Search (F8): facilitating finding specific elements. It may be difficult to find a term among hundreds of others. Onyx supports keyword search with autocomplete, which is a feature in which the application predicts the rest of the word the user is typing. Some visualization tools such as OWLViz (Horridge, 2005), KC-Viz (Motta, Peroni, Gómez-Pérez, d'Aquin, & Li, 2012), and GrOWL (Krivov, Williams, & Villa, 2007) do not support search. However, Jambalaya (Storey et al., 2002), NavigOWL (Hussain et al., 2014), OntoGraf (Falconer, 2010), and WebVOWL support keyword search option. MEMO GRAPH (Ghorbel et al., 2016) is a user-friendly ontology visualization tool especially designed to be used by everyone. It is the single tool that supports speech-to-text search.
9. Hide selected entity (F9): hiding the ontology concepts that are not of interest. The user clicks the Hide Entities button and selects the entities to be hidden.
10. Filter specific entity type (F10): hiding all object properties at once, for example. Jambalaya, KC-Viz, SOVA, TGVizTab (Alani, 2003), and WebVOWL support filtering specific entity types. Onyx supports the following filtering options (using the Filters tab):
    (a) Show class hierarchy: This option only displays the subClassOf relations between classes.
    (b) Hide restrictions: This option removes the classes labelled with the letter R.
    (c) Hide datatype properties: This option hides relations between a class or a literal/XML data type.
    (d) Hide object properties: this option hides the relation between classes (except subClassOf).
    (e) Hide intersectionOf/unionOf: This option hides the details about the intersection and union operations. This option shows the classes that are the results of the operations but hides the classes that are the operands.
    (f) Hide enumerated classes: This option hides the individuals of enumerated classes.
    (g) Hide property characteristics: This option hides the property characteristics including functional, inverse functional, transitive, symmetric, and inverseOf.

11. Fisheye distortion (F11): magnifying the part of the graph around the mouse while leaving the larger graph unaffected. Onyx does not support this feature.
12. Edge bundles (F12): grouping edges with similar paths. Onyx does not support edge bundles.
13. Drag and drop navigation (F13): moving a graph that does not fit the screen by dragging with the mouse.
14. Drag and drop user layout (F14): interactively reordering the nodes of the graph.
15. Clustering (F15): displaying a subgraph of important nodes. Onyx supports displaying classes based on their importance using the key concept extraction algorithm.
16. Integration with editing (F16): selecting a node or an edge and editing its properties in, for example, a pop-up window. Onyx does not support the editing of the visualized ontology.
17. Graphical editing (F17): creating new entities by, for example, drawing edges between displayed nodes. OWLGrEd (Barzdins, Barzdins, Cerans, Liepins, & Sprogis, 2010) and TopBraid (TopQuadrant, 2007) are visualization tools with graphical editing support. Onyx does not support the editing of the visualized ontology.

Dudás et al. (2014) analyses the features of existing ontology visualization tools. We extend this evaluation using the features of WebVOWL and Onyx in Table 1.

4.2 | Scalability

Scalability is a central concern when designing systems to handle large amounts of data gracefully (Munzner, 2014). Saghafi (2016) classifies the scalability of the ontology visualization tools as limited, relatively high, and high. According to this classification, most existing ontology visualization tools have limited scalability. Most of the remaining tools can handle large ontologies using visualization methods other than graphs. These visualization tools use tree maps, indented trees, or node-link diagrams to represent large ontologies. For example, Jambalaya uses tree maps; Multi-View Ontology Visualization (Kuhar & Podgorelec, 2012) uses hierarchical connected circles, indented trees, and node-link diagrams; and OntoVismod (García-Peñalvo, Palacios, Navarro, & Therón, 2012) incorporates tree maps, hierarchical trees, and radial layouts. The second group of tools can handle large ontologies by visualizing only the most important classes in the ontology (e.g., KC-Viz) or supporting a simplified view of the ontology; for example, Ontology Visualizer and OWLViz display only class hierarchy, and SOVA (Kunowski & Boinski, 2012) displays only classes in the ontology. In the same manner, Onyx uses the key concept extraction algorithm to handle large ontologies.

We chose to compare Onyx with WebVOWL, as it is a graph-based, stand-alone web-based ontology visualization tool such as Onyx. Saghafi (2016) classified WebVOWL as a relatively highly scalable visualization tool. The maximum ontology dimension that can be visualized with the current version of WebVOWL is limited. The visualization of large ontologies with WebVOWL is a major issue with the tool. The

TABLE 1 Features of existing ontology visualization tools (Dudás et al., 2014)

Tool	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17
Jambalaya	1		1	1		2	1	2	2	2	1			2		2	
KC-Viz	2		2		2	2	2	1	2	2			2	2	2		
OntoGraf			2	2		2	2	2		1			2	2		1	
OntoViz			2						2				2				
OWLGrEd	1	2	1		2	2						1		2		2	2
OWLViz	2		1	2		2	2	1	2							2	
SOVA	1		2					2		2			2	2			
TGVizTab	1		1	2			2	2	2	2	1		2				
TopBraid		2	1						2					2			2
WebVOWL	2		2			2		2		2			2	2			
Onyx	2	2	2	2	2	2	2	2	2	2			2	2	2		

TABLE 2 Time performances of (in seconds) of Onyx and WebVOWL

Ontology	Class count	Onyx	WebVOWL
Dengue Fever Ontology (IDODEN)	5019	22	215
Emergency care ontology (ONTOLURGENCES)	10031	212	-
Human Disease Ontology (DOID)	12694	287	-
Mammalian Phenotype Ontology (MP)	13927	309	-
Human Phenotype Ontology (HP)	17387	423	-

TABLE 3 Language constructs that can be visualized by existing tools (Dudás et al., 2014)

	Jambalaya	KC-Viz	OntoGraf	OntoViz	OWLGrEd	OWLViz	SOVA	TGVizTab	TopBraid	WebVOWL	Onyx
Classes	+	+	+	+	+	+	+	+	+	+	+
Object Prop.	+	+	+	+	+	-	+	+	+	+	+
Datatype Prop.	-	-	-	+	+	-	-	-	+	+	+
Instances	+	+	+	+	+	-	+	-	+	-	-
Annotations	-	-	-	-	+	-	-	-	+	-	-
Univ./Exist. Rest.	+	-	+	+	+	-	+	-	+	-	+
Cardinality	-	-	-	+	+	-	+	-	+	-	+
Enumeration	-	-	+	+	+	-	+	-	+	-	+
Intersection	-	+	+	+	+	-	+	-	+	+	+
Union	-	-	+	+	+	-	+	-	+	+	+
Complement	-	-	+	+	+	-	+	-	+	-	+
equivalentClass	+	-	+	+	+	-	+	-	+	-	+
disjointWith	-	-	-	-	+	-	+	-	+	+	+
subClassOf	+	+	+	-	+	-	+	+	+	+	+
Property Char.	-	-	-	-	+	-	+	-	+	+	+

selected SVG-based application has some limitations in this regard. Table 2 shows the ontologies used for comparison and the class counts of each ontology[6]. Table 2 gives the comparison results of the two ontology visualization tools. Tests are conducted to show the performance of Onyx using the key concept extraction algorithm. The tool displayed an ontology with 17,387 classes in 423 seconds. WebVOWL did not open ontologies with class counts greater than 10,000 in a reasonable time (<20 minutes). The tests are conducted on a laptop with an Intel i5 3.2 GHz processor and 16 GB memory. The performance can be improved on a computer with better hardware configuration.

4.3 | OWL language support

Dudás et al. (2014) analyse the language-level expressiveness of existing ontology visualization tools. We extend this evaluation using the expressiveness of WebVOWL and Onyx in Table 3. Onyx adopts TBOX-based visualization and supports all types of OWL language constructs.

Although Onyx supports most OWL language constructs, the main aim of the tool is to avoid unnecessarily confusing graphical representations. Therefore, we implemented the following abstractions to make the image more readable: (a) Restriction classes are represented with a grey circle labelled with the letter R. Definitions of the restrictions are expanded when the node is selected by the user. (b) In addition to these abstractions, the filters described in Subsection 4.1 can be applied to the ontology graph, thus avoiding a cluttered view.

5 | CONCLUSION AND FUTURE WORK

Ontology visualization tools are important because visualization can help develop, explore, verify, and comprehend ontologies. This paper introduces the design of a new ontology visualization tool called Onyx, which was implemented as a stand-alone web application. Unlike traditional visualization tools, Onyx provides important metrics and analytics about the ontology concepts and warns the ontology developer about potential ontology design errors. According to our research, Onyx is the only ontology visualization tool that uses Canvas technology, which is faster than its rival, SVG. Onyx also supports key concept extraction. Due to these features, Onyx offers a suitable environment for the representation of large ontologies, especially those used in biomedical and health information systems and those that contain many terms. A potential direction for future work is to implement Onyx as a plug-in of Protégé Ontology Editor. Protégé is a free, open-source platform that provides a growing user community with a suite of tools for constructing domain models and knowledge-based applications with ontologies. As an extension to Protégé, Onyx will benefit from a community with a large number of users (there are currently 316,741 registered users), a library of reusable components and a flexible architecture. Onyx could also be extended to cover a wide variety of OWL 2 language constructs.

ORCID

Övünç Öztürk https://orcid.org/0000-0001-7127-7902

[6]All of the ontologies can be downloaded from https://bioportal.bioontology.org/ontologies.

REFERENCES

Alani, H. (2003). TGVizTab: An ontology visualisation extension for Protégé. In *K-Cap'03 Workshop on Visualization Information in Knowledge Engineering*, 26–26 October 2003, Sanibel Island, Florida, USA, ACM Press.

Barzdins, J., Barzdins, G., Cerans, K., Liepins, R., & Sprogis, A. (2010). OWLGrEd: A UML style graphical editor for OWL.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2004). OWL web ontology language reference. (*Tech. Rep.*): W3C Retrieved from http://www.w3.org/TR/owl-ref/

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, *284*(5), 34–43.

Borst, W. N. (1997). Construction of engineering ontologies for knowledge sharing and reuse. (Ph.D. Thesis), University of Twente, Enschede, Netherlands.

Barak Michener and Community (2014). Cayley Graph Database. Retrieved from https://cayley.io/

Dudáš, M., Zamazal, O., & Svátek, V. (2014). Roadmapping and navigating in the ontology visualization landscape. In K. Janowicz, S. Schlobach, P. Lambrix & E. Hyvönen (eds) *Knowledge Engineering and Knowledge Management. EKAW 2014. Lecture Notes in Computer Science* (Vol. *284*). Springer, Cham.

Falconer, S. (2010). Ontograf Protégé plugin. Retrieved from http://protegewiki.stanford.edu/wiki/OntoGraf

Franz, M., Lopes, C. T., Huck, G., Dong, Y., Sumer, O., & Bader, G. D. (2015). Cytoscape. js: a graph theory library for visualisation and analysis. *Bioinformatics*, *32*(2), 309–311.

García-Peñalvo, F. J., Palacios, R. C., Navarro, J. F. G., & Therón, R. (2012). Towards an ontology modeling tool. A validation in software engineering scenarios. *Expert Systems with Applications*, *39*(13), 11468–11478.

Ghorbel, F., Ellouze, N., Métais, E., al Hamdi, F., Gargouri, F., & Herradi, N. (2016). MEMO GRAPH: An ontology visualization tool for everyone. In R. J. Howlett, L. C. Jain, B. Gabrys, C. Toro & C. P. Lim (eds) *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 20th International Conference KES-2016*, York, UK, 5–7, September 2016, Elsevier, Vol. *96*, pp. 265–274.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, *5*(2), 199–220.

Horridge, M. (2005). Owlviz—A visualisation plugin for the protégé owl plugin. Retrieved from http://www.co-ode.orgldownloads/owlviz

Hussain, A., Latif, K., Rextin, A. T., Hayat, A., & Alam, M. (2014). Scalable visualization of semantic nets using power-law graphs. *Applied Mathematics & Information Sciences*, *8*(1), 355–367.

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., & Giannopoulou, E. (2007). Ontology visualization methods: A survey. *ACM Computing Surveys*, *39*(4), 43.

Krivov, S., Williams, R., & Villa, F. (2007). GrOWL: A tool for visualization and editing of owl ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, *5*(2), 54–57. Retrieved from https://gesispanel.gesis.org/preprints/index.php/ps/article/view/111

Kuhar, S., & Podgorelec, V. (2012). Ontology visualization for domain experts: A new solution. In *International Conference on Information Visualisation*, Montpellier, France, pp. 363–369.

Kunowski, P., & Boinski, T. (2012). Sova: Simple ontology visualization API. Retrieved from http://protegewiki.stanford.edu/wiki/SOVA

Lohmann, S., Link, V., Marbach, E., & Negru, S. (2015). WebVOWL: Web-based visualization of ontologies, *Knowledge engineering and knowledge management* (pp. 154–158). Cham: Springer International Publishing.

Meteor Development Group (first release: 2012). Meteor homepage. Retrieved from github.com/meteor/meteor

Motta, E., Peroni, S., Gómez-Pérez, J. M., d'Aquin, M., & Li, N. (2012). Visualizing and navigating ontologies with KC-VIz, *Ontology engineering in a networked world* (pp. 343–362). Berlin: Springer.

Munzner, T. (2014). *Visualization analysis and design*. Boca Raton, FL: CRC Press.

Noy, N. F., Fergerson, R. W., & Musen, M. A. (2000). The knowledge model of Protégé-2000: Combining interoperability and flexibility, *Knowledge engineering and knowledge management methods, models, and tools: 12th international conference, EKAW 2000 Juan-Les-Pins, France, October 2–6, 2000 proceedings* (pp. 17–32). Berlin, Heidelberg: Springer Berlin Heidelberg.

Peroni, S., Motta, E., & d'Aquin, M. (2008). Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures, *The semantic web* (pp. 242–256). Berlin, Heidelberg: Springer Berlin Heidelberg.

Poveda-Villalón, M., Gómez-Pérez, A., & Suárez-Figueroa, M. C. (2014). Oops! (ontology pitfall scanner!): An on-line tool for ontology evaluation. *International Journal on Semantic Web and Information Systems*, *10*(2), 7–34.

Rodriguez, M. A. (2015). The gremlin graph traversal machine and language (invited talk). In *Proceedings of the 15th Symposium on Database Programming Languages*, ACM, pp. 1–10.

Saghafi, A. (2016). Visualizing ontologies – a literature survey, *Graph-based representation and reasoning: 22nd International Conference on Conceptual Structures, ICCS 2016, Annecy, France, July 5-7, 2016, Proceedings* (pp. 204–221). Cham: Springer International Publishing.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., … Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, *13*(11), 2498–2504.

Storey, M. D., Noy, N. F., Musen, M. A., Best, C., Fergerson, R. W., & Ernst, N. A. (2002). Jambalaya: An interactive environment for exploring ontologies. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, San Francisco, California United States, pp. 239–239.

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

Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P., & Aleman-meza, B. (2005). Ontoqa: Metric-based ontology quality analysis. In *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, Houston, TX, pp. 45–53.

TinkerPop, A. (first release: 2009). Gremlin homepage. Retrieved from https://github.com/tinkerpop/gremlin/wiki

TopQuadrant (2007). Topbraid composer. http://www.topbraidcomposer.com/

Walke, J., & Popov, D. (2011). Reactjs homepage. Retrieved from Releases-facebook/react,GitHub

AUTHOR BIOGRAPHIES

Övünç Öztürk is an assistant professor at Manisa Celal Bayar University. He received BS degree in Computer Engineering from Middle East Technical University and MS and PhD degrees in Computer Engineering from Ege University. His research interests include semantic web technologies, database management systems, embedded systems, web programming, and internet of things.

Hasan Gökhan Açikgöz is a computer engineering undergraduate student at Manisa Celal Bayar University. He has been conducting his undergraduate thesis under the supervision of Dr. Övünç Öztürk. His skills include web development and mobile application development.

How to cite this article: Öztürk O, Açikgöz HG. Onyx: A new Canvas-based tool for visualizing biomedical and health ontologies. *Expert Systems*. 2019;e12380. https://doi.org/10.1111/exsy.12380