Scientific Visualization, 2023, volume 15, number 5, pages 73 - 88, DOI: 10.26583/sv.15.5.07

Application of an Agent-Based Model to Develop Ontological Data Visual Management Tool

D. S. Razdyakonov[1], D. I. Muromcev[2]
Faculty of Software Engineering and Computer Technologies, ITMO University, St. Petersburg, Russia

[1] ORCID: 0009-0009-9668-207X, ladone3@gmail.com
[2] ORCID: 0000-0002-0644-9242, d.muromtsev@gmail.com

Abstract

The paper presents an agent-based model of interactive visualization and proposes a method for its application in the development of the ontological data visual management tool. The process of user interaction with the visualization is represented as a state graph, where each node is a separate visualization. The initial state is known, and the final state is formed in the process of user interaction with the ontological data visual management tool. Each intermediate visualization from the state graph is transferred into the multidimensional Euclidean data space formed on the basis of the visualized ontology, which allows to calculate weighting coefficients on the state graph edges and to search the graph using known algorithms. Application of the model allows for a reduction in the labor intensity of executing user scripts due to the reduction of step-by-step visualization creation to the task of searching in the state space.

Keywords: ontologies, ontology management, ontology visualization methods, visualization models, interactive visualization.

1. Introduction

The efficiency of the ontology data management (ODM) process directly depends on the software tools that provide this process. Efficiency is measured both in man-hours for business tasks and in SPARQL/UPDATE query processing time for inter-service communication tasks.

Depending on the management objectives, various software tools are applied to improve the efficiency of the ODM process. For developing small ontologies represented as a dozen of related files, data engineers prefer to use Protégé [12] or WebVOWL [21], but for managing ontologies that belong to LOD (Linked Open Data), they use web platforms such as Metaphactory [10] or TopBraidComposer [25] web interfaces. Such platforms are able to perform lazy queries on data stored in public knowledge bases (KBs).

An important component of commercial ontology development tools is the visual management feature of ontology data (OD), which significantly improves the efficiency of ODM processes by automating the compilation of SPARQL queries and providing transparent manipulation of ontology data schema.

Visual management of ontology ***data*** *is a process of OD management performed with the use of software tools that combine the functions of human-machine and machine-human interfaces*. In other words, these are tools that allow you to edit and visualize OD at the same time.

At the same time, it is not enough that one and the same ODM tool performs two features at once (see Fig. 1). For example, a program that displays an ontology graph and provides a form for creating new ontology entities will not be considered a visual management tool if the two graphical interfaces are not linked together, i.e., do not share a common state.

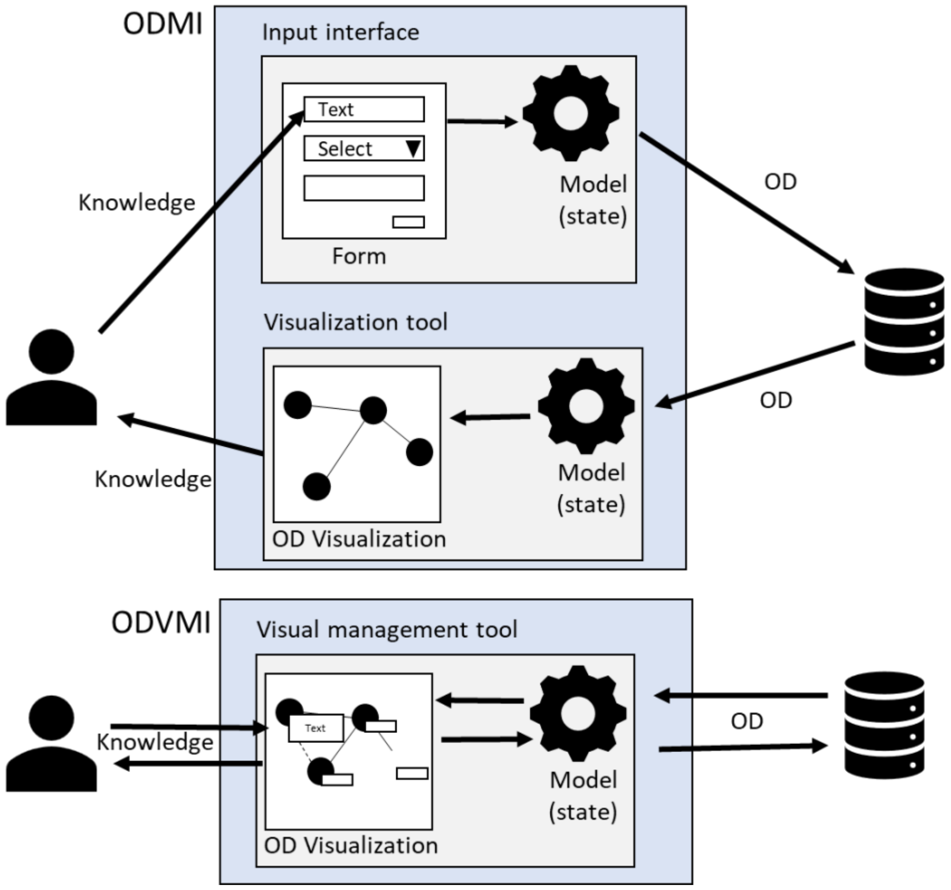


Figure 1 — OD management tool (ODMT) and OD visual management tool (ODVMT)

A challenge in the development of OD visual management tools (ODVMT) is that it is necessary to solve not only the tasks assigned to OD editing tools, but also the tasks assigned to OD visualization tools. For example, if there is a need to work with ODs that contain blank nodes, the tool should not only be able to edit or create structures containing blank nodes (SCBN), but should also be able to provide the appropriate visualization of SCBN.

When developing ODVMT, the order of user interaction with controls is important. If for ODMT the order of interaction can be freely changed without reference to the visualization, then for ODVMT the development is much more complicated because the same action can be performed in several ways, and the sequence of interaction with the controls directly affects the legibility of the visualization and the efficiency of interaction.

In ODVMT, part of the visualization inevitably becomes interactive as its current state depends on the actions performed by the user, so as part of our paper, we propose to use an interactive visualization model to optimize the ODVMT development process. This approach to the IV model allows us to formally describe the objective function of the IV process and ensure the purposefulness and acceleration of ODVM processes.

Interactive visualization (IV) *is a visualization that has a changeable state, where the state of the visualization can change either over time or in response to external events (input operations or modifications to visualization parameters).*

2. Related literature analysis

Existing ontology data management tools, in one way or another, implement the ODM methods, which are divided into development, visualization, and storage methods.

Currently, there are many works such as Ontology Visualization Methods — a Survey [14], Ontology Visualization Methods and Tools: A Survey of the State of the Art [7] and Ontology Visualization Protégé Tools — a Review [18], which provide detailed and comprehensive reviews on OD visualization tools and methods.

On the other hand, there are works such as Ontology Development Methods [9], Methodologies for Ontology Development [13], A Review On Ontology Development Methodologies for Developing Ontological Knowledge Representation Systems for Various Domains [2], and Methods for Ontology Development [5], which review and compare ontology development practices. In addition to the listed methodologies there are works Managing Ontologies: A Comparative Study of Ontology Servers [1], Hybrid Method for Storing and Querying Ontologies in Databases [20], Creating Knowledge Databases for Storing and Sharing People Knowledge Automatically Using Group Decision Making and Fuzzy Ontologies [15], and On Storing Ontologies Including Fuzzy Datatypes in Relational Databases [3], which address the issues of efficient storage of OD in KBs and other available formats.

While there are quite a few papers devoted to tools supporting the ODM process, the development of such tools is a separate and large area of knowledge that is rarely related to the field of semantic technologies and OD. From the point of view of user interface development, articles such as Integrating Human-Centered and Model-Driven Methods in Agile UI Development [8] and Paprika: Rapid UI Development of Scientific Dataset Editors for High Performance Computing" [16] may be of interest, and for immersion in the area of developing tools for interactive visualization of multidimensional data, Developing a System for Interactive Visual Analysis of Multidimensional Data [23] may be useful.

3. Agent-based interactive visualization model

In this paper, we formulate an approach to the development of ODVMT. We base our method on an interactive visualization model. Let's describe the model.

The state of the interactive visualization changes in response to user actions, where user actions can be either to edit the OD or to explore the OD.

The interactive visualization model is based on the process of interactive search, the most demanded "sub-process" of management when manipulating Big Data in a lazy visualization framework. Interactive search is a process in which the user changes the state of a visualization so that the visualization responds to a given search query. This topic is discussed in detail in An Ontology-Driven Visual Question-Answering Framework [4] and VQASTO: Visual Question Answering System for Action Surveillance Based on Task Ontology [19] papers.

The basis of the model states is an ontology graph G, which is a set of subject-predicate-object triples $\langle s, p, o\rangle$ (1). The ontology graph generates a multidimensional data space whose dimensions correspond to the domain of values of predicates from the ontology, i.e. a dimension is the set of possible objects occurring in triplets with the corresponding predicate.

U — set of URIs, where each URI defines a logical or physical resource.

B — set of blank nodes.

L — set of literal values such as string, integer, or boolean.

G — ontology graph.

$$s \in B \cup U; p \in U; o \in B \cup L \cup U \tag{1}$$

The data generate the space D (2), which consists of the dimensions D_i (Fig. 2).

$$D = \langle D_1, D_2, D_3, \dots D_i\rangle \mid D_i = o \in (B \cup L \cup U) : \langle s, p, o\rangle \in G \mid i = p. \tag{2}$$

The user observes the data space or a part of it on the screen, where the data space is projected onto the user's screen using a visualization function that takes as input the position and size of the search frame in the data space, and outputs a two-dimensional static visualization.

Search frame (SF) is a section of the data space that is available for the user to observe or the agent's position in the data space. During visualization, the data from the section enclosed by the search frame is converted by the visualization function $V_D(SF)$ into a set of parameters that are input to the visualization tool.

Search point (SP) is a position of the user's attention focus in the data space when using the interactive visualization tool.

User is a person who communicates with interactive visualization through human-machine interface (monitor/VR-glasses + visualization tool).

Agent is a user's projection inside the visualization tool, i.e., an abstraction describing the state of the visualization tool at each step of interactive visualization. In the system *Agent = SF*.

$IVis_D$ — interactive visualization of data space D (Fig. 3) is a tuple of two elements: data visualization function $V_D(SF)$ and tool function (visualization control function) $T_D(SF, INPUT)$. Where SF is the search frame, $INPUT$ is the set of commands from the visualization tool interface, and SF' is the new position of the search frame. v_D is the set of parameters that are input to the visualization tool.

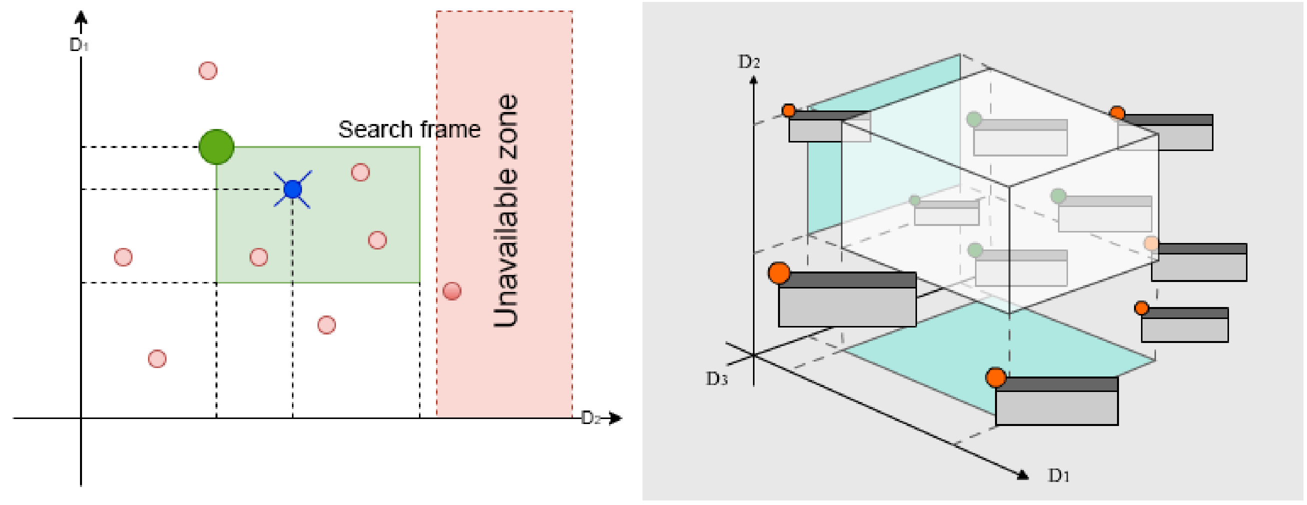


Figure 2 — 2D/3D data spaces

In our case v_D is a set of graphical primitives, as well as camera parameters and lighting settings that are input to the graphics engine for processing.

Performing a given number of simple actions (T_{sg}), the user by means of tools generates commands to the interactive visualization, which change the position of the search frame, thus changing the visualization state.

$$\begin{tikzcd}
& IVis_D = \langle V_D(SF), T_D(SF, Input)\rangle \arrow[dl] \arrow[dr] & \\
V_D(SF) \rightarrow v_D & & T_D(SF, Input) \rightarrow SF'
\end{tikzcd}$$

Figure 3 — Interactive data space D visualization

While exploring the visualization results, the user changes the position of the attention focus.

If we take the number of input operations (T_s) as time and the distance in the data (d_i) space (S_s) as the path, we can calculate the agent's movement speed in the data space (3).

$$V_s = \frac{\sqrt{(d_{1_a} - d_{1_b})^2 + (d_{2_a} - d_{2_b})^2 + \cdots + (d_{n_a} - d_{n_b})^2}}{\sum_{g=1}^{m} T_{s_g}} = \frac{S_s}{T_s} \tag{3}$$

where

a, b — initial and final position in the data space,

d_n — position coordinate in multidimensional space,

m — number of search steps.

The main advantage of our model is that it reduces the step-by-step creation of visualization to the task of searching in the state space generated from multidimensional ontological data and primitive actions of visualization tools. The model describes the interactive visualization process function in terms of speed, labor intensity, and distance, which allows for targeted and accelerated interactive search over ontological knowledge bases under conditions of limited observability of the data space.

Thus, we reduce the interactive visualization, i.e., the step-by-step creation of the visualization, to the task of searching in the state space. Accordingly, the target state is a set of values of the visualization parameters.

In terms of the model, the final and initial states, i.e., final and initial visualizations, differ in the position of the search frame in the data space, where the position is described by a point in the multidimensional data space and by the width of the frame in each dimension. It follows that optimization can proceed along two dimensions: the width and the position of the search frame in the multidimensional space. The optimization goal is to minimize the distance function between the target search frame position and the current search frame position in the data space, and to minimize the difference between the desired search frame width for each dimension and the current search frame width for those dimensions. If we consider the search frame width in the calculation of the objective function as a set of additional dimensions, we obtain a space of *2n* dimensions and a distance function as the objective function.

The objective function, which is used to optimize the search process in the state space, is the Euclidean distance in the *2n*-dimensional data space between two vectors *(p, q)* describing the target and current states.

$$f(\vec{p}, \vec{q}) = \sqrt{\sum_{i=1}^{2n} (p_i - q_i)^2} \tag{4}$$

Each vector has n values of dimension coordinates in the data space and n values of widths for the same dimensions, i.e., *2n* values. Since the process we reduce to the search task in the state space is a step-by-step visualization process, each state vector describes the visualization state at one step, i.e., it describes a set of parameter values that were used to create the visualization, where a visualization is a set of graphical primitives representing a part of the data space. At the same time, the target visualization may be unknown at the beginning of the search and will be formed during the user's interaction with the visualization tools. If we describe the process in terms of state space, we can say that the path to the target state at the time of search start is unknown and is computed during the heuristic search.

The state space is represented by a quartet [*N, A, S, GD*], where

N — set of consecutive visualizations or set of search frame positions in the data space,

A — set of input steps (changes of visualization parameters) in the process of creating the target visualization.

S — non-empty set of initial states, i.e., possible starting positions of the search frame.

GD — non-empty set of target states, i.e. a set of final visualizations that can be described in one of the following ways:

- Measurable properties of generated visualizations encountered in the search process. (For example, the number of visualization elements).
- The sequence and nature of the elementary input actions by which the target visualization was obtained.

Valid path is a path from a state of the set *S* to a state from the set *GD*. That is, it is a set of input steps and visualization states (positions of the search frame) that were sequentially performed in the process of searching for the target visualization.

Transitions in the state space correspond to the steps of the visualization creation process and describe the possibility of transition from one visualization state to another.

Visualization parameters are the position and width of the search frame in the data space for each data dimension (the visualization state is projected into a set of graphical primitives).

Table 1 — The interactive visualization states and their parameters

Visualization state	**Coordinates (hasColor, hasSugar)**	**Width of the search frame (hasColor, hasSugar)**	**Distance to target**
Visualization state 1	(1, 1)	(3, 4)	4.123
Visualization state 2	(1, 1)	(1, 4)	3.605
Visualization state 3	(1, 3)	(1, 1)	0
Visualization state 1.1	(1, 1)	(2, 4)	3.741
Visualization state 2.1	(1, 4)	(1, 1)	1

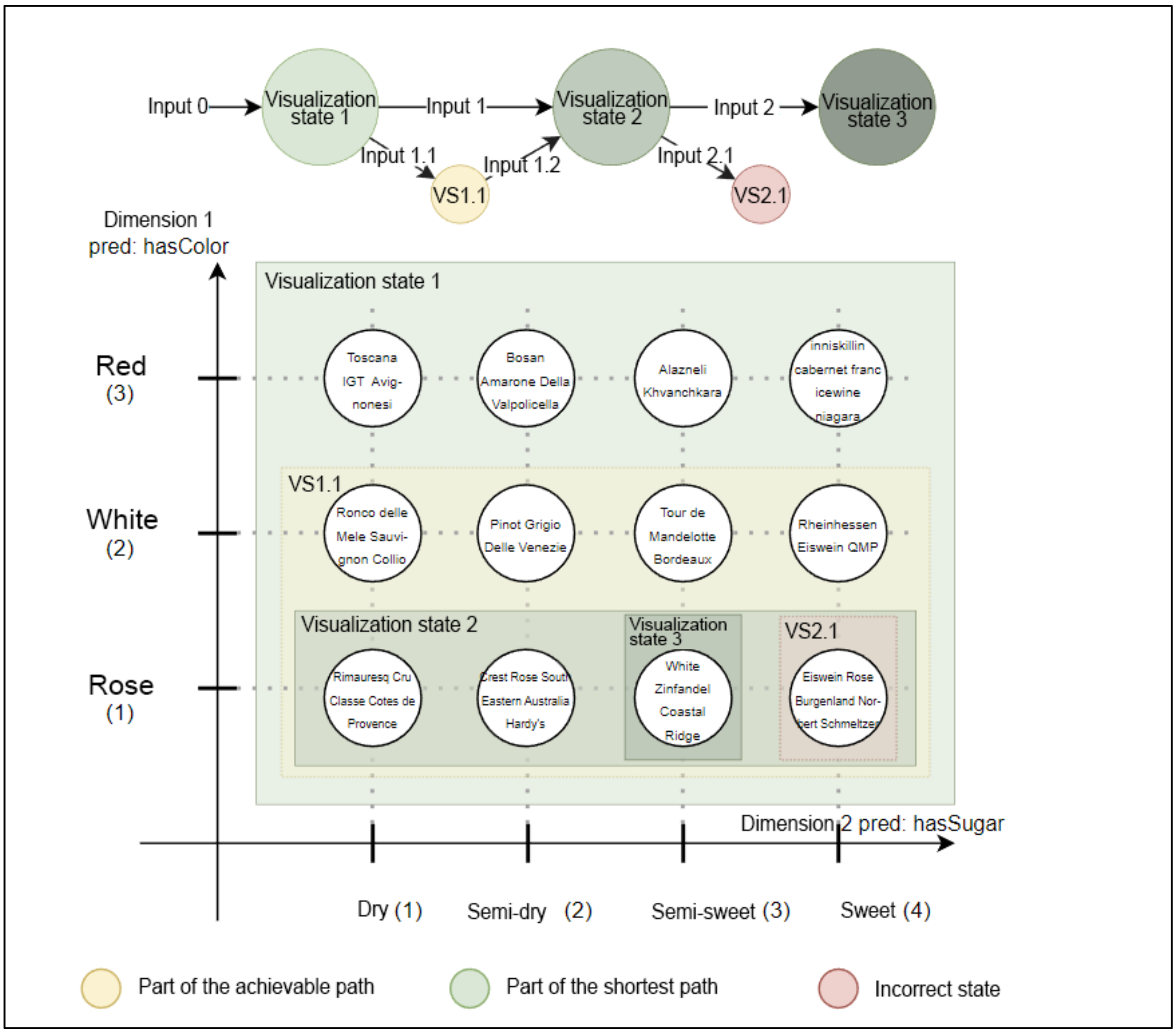
Table 2 — Calculation of path lengths taken step by step

Path	**Total labor intensity**	**Path length**
1(1) → C2(2) → 3	3 + 3 = 6	2 + 3.605 = 5.605
1(1.1) → 1.1 → (1.2) → C2(2) → 3	4 + 3 + 3 = 10	1 + 1 + 3.605 = 5.605
1(1) → C2(2.1) → 2.1	3 + 3 = 6	2 + 4.242 = 6.242

The model describes the state space and data space in which the state graph is located, as well as the speed at which it is possible to move from one state (graph node) to another using the available user interface controls. This reduces the visualization tool development to searching for the optimal path in a weighted state graph, where the weighting coefficients on the edges are the labor intensity of the transition from one state to another, which is calculated based on the speed and distance in the data space. Various search algorithms and optimization functions can be used to solve the search task.

4. Example of path calculation in a two-dimensional state space

Here is an example of interactive searching in terms of our model. The target ontology is a truncated wine ontology containing two attributes for each wine (color and sugar content).


Input 0
Visualization state 1
Input 1
Visualization state 2
Input 2
Visualization state 3
Input 1.1
VS1.1
Input 1.2
Input 2.1
VS2.1
Dimension 1 pred: hasColor
Red (3)
White (2)
Rose (1)
Visualization state 1
Toscana IGT Avignonesi
Bosan Amarone Della Valpolicella
Alazneli Khvanchkara
Inniskillin cabernet franc icewine niagara
VS1.1
Ronco delle Mele Sauvignon Collio
Pinot Grigio Delle Venezie
Tour de Mandelotte Bordeaux
Rheinhessen Eiswein QMP
Visualization state 2
Rimauresq Cru Classe Cotes de Provence
Crest Rose South Eastern Australia Hardy's
Visualization state 3
White Zinfandel Coastal Ridge
VS2.1
Eiswein Rose Burgenland Nor-ert Schmeltzer
Dimension 2 pred: hasSugar
Dry (1)
Semi-dry (2)
Semi-sweet (3)
Sweet (4)
Part of the achievable path
Part of the shortest path
Incorrect state
Figure 4 — Two-dimensional data space and visualization state graph

The search target is a wine with the following parameters: {color} - pink, {sugar content} - semi-sweet. The user should change the interactive visualization state so that only those wines that satisfy the search conditions are on the screen. The data space and the visualization state graph are shown in Fig. 4. The space contains 12 different instances of the Wine class. The main parameters of the interactive visualization state are presented in Tables 1, 2, and the visualization of the search state using the Ontodia3d tool [6] is presented in Fig. 5. The animated process of interacting with the Ontodia3d tool in VR is shown in Fig. 6.

This example shows how our model describes the data space and state graph of the interactive visualization, which allows us to reduce the step-by-step construction of the visualization to the task of searching the state space, which is formed on the basis of two parameters (in this example) describing all classes of the ontology. Such a description allows us to discard suboptimal ways of generating the target visualization and to identify bottlenecks in the visualization tool interface.

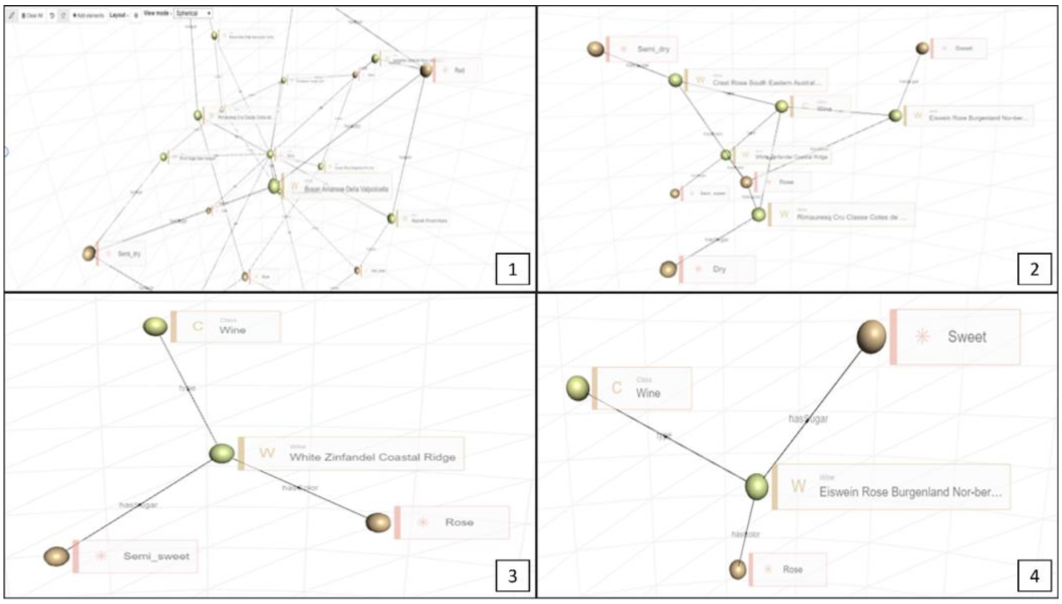

Semi_dry
Red
Sweet
Eiswein Rose Burgenland Nor-ber...
Rose
Dry
Wine
White Zinfandel Coastal Ridge
Semi_sweet
hasSugar
hasColor
type
1
2
3
4

Figure 5 — Visualization state 1–4

Figure 6 — The process of interacting with Ontodia3d in VR

5. Method for visual management of ontology data

Previously, we defined that a method is a technical way of developing ODVMT. We base our method on an interactive visualization model and propose to actively apply it in the interface development phase of ODVMT.

An important feature of our model is that the model describes the state graph of the interactive visualization and allows us to generate the objective function of the interactive visualization process (4).

Method sequence:

1. Generation of the target dataset for testing the selected scripts and compilation of the data space based on it. The target dataset, according to the model, is the graph $G = \langle s, p, o \rangle$.

   a. Extraction of all possible properties of ontology elements that can be used for interactive search. That is, the extraction of dimensions D_i from the graph G, where each dimension $D_i = o \in (B \cup L \cup U)$: $\langle s, p, o \rangle \in G \mid i = p$.

   b. Generation of a multidimensional ontology data space, where the number of dimensions of the space is equal to the number of properties $D = \langle D_1, D_2, D_3, \ldots D_i \rangle$.

   The generation of space on each dimension for different usage scripts is a separate big topic for research, but in the scope of our visualization, where visualization is generated as a three-dimensional graph (the tool is described in [6]) with semantic distribution of nodes in the visualization space, we propose to use Word Embedding algorithms, which are described in detail in the articles Using Word Embeddings for Visual Data Exploration with Ontodia and Wikidata [22] and Word Embeddings as Metric Recovery in Semantic Spaces [11].

2. Assembling a script set (C) for using the ODM $C = \langle C_1, C_2, \ldots C_i \rangle$.
3. Compiling the state graph SG of the interactive visualization. At this step, the main states of the interface of the ODM tool are extracted, and the state graph is compiled, where the nodes of the graph are the states of the interactive visualization, which have a fixed position in the data space, and the edges denote the possibilities of transition between the states $SG(V, E) = \langle V, E \rangle$ (V is the set of nodes, and E is the set of edges). The Euclidean distance function (4) is used to compute the weighting coefficients on the edges of the graph.
4. Generation of sets of valid paths in the state graph for each script $P = \langle P_{c1}, P_{c2} \ldots P_{ci} \rangle \mid P_{ci} = \langle (v_1, v_2), (v_2, v_3) \ldots (v_{i-1}, v_i) \rangle$, as well as the definition of the initial $S = \langle \ldots S_{ci} \ldots \rangle$ and final states for each script $GD = \langle \ldots GD_{ci} \ldots \rangle$. At this stage, standard algorithms for finding valid paths in a graph, such as Dijkstra's algorithm or A* algorithm, can be used.
5. Computation of the labor intensity of possible paths taking into account the weighting coefficients $L(P_{ci}) = Difficulty(v_1, v_2) + \ldots + Difficulty(v_{i-1}, v_i)$.
6. Sorting paths by length and removing redundant paths $RemoveDuplicates(Sort(L)) \mid L = \langle L(P_{c1}) \ldots L(P_{ci}) \rangle$.

Thus, it is possible to eliminate redundant interface elements and optimize the user's interaction with the ODMT interface.

In cases where it is impossible to define a fixed set of visualization states, for example, when the camera position in the state space has thousands of possible positions, you can evaluate visualization tools by the speed of movement in the state space. This approach requires the following steps:

1. Make a list of controls.
2. Calculate the speed of movement in the state space for each control and determine the zone of the data space in which the tool can be used.
3. Eliminate duplicate tools that allow you to move around in the same zone of the data space.

The possibility of using mathematical optimization to identify potential transitions between states when the interface controls necessary for such transitions between states have not yet been implemented is also of interest. In such cases, the optimization can show the optimal path between states so that programmers can then implement the interface controls that allow these transitions to occur.

As a result of applying the method to the OD visualization tool in 3D and VR spaces [6], the average labor intensity was reduced by 21.8% (Fig. 7). Where labor intensity is considered as the number of the elementary actions (clicking the mouse button, scrolling with the mouse wheel, entering a symbol using the keyboard) that the user has to perform in order to execute the script. The graph shows the average values of labor intensity obtained during the tests, so the numbers in Fig. 7 are fractional.

The reduction of labor intensity was achieved by removing suboptimal paths from the state graph of the interactive visualization according to the proposed method. The interaction process with the filter panel (Fig. 8) was optimized by ordering the input steps by means of the user interface, which reduced the path in the data space. In the first script, users were allowed to apply filters in random order, and after the modification, the application of filters was brought to a strict order. Fig. 9 shows the charts of convergence to the objective (target state). The charts are plotted only for the steps related to the application of filters.

The second part that was subjected to optimization was the control linkage panel (Fig. 10).

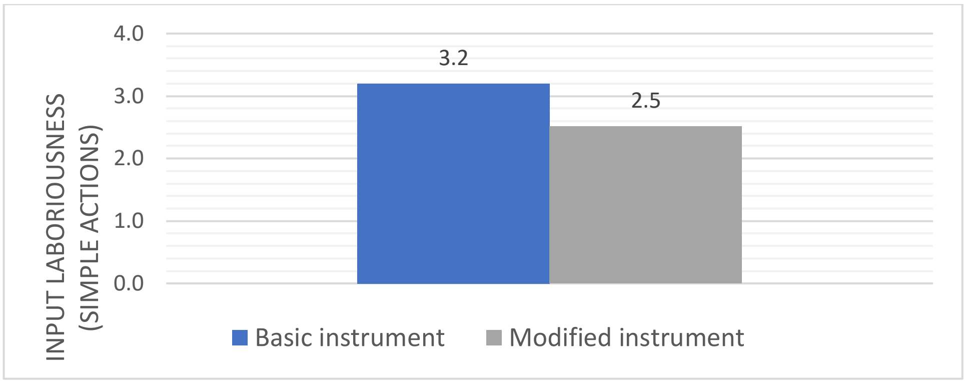


Figure 7 — Labor intensity of the ODM tool before and after applying the method

Initially, the panel gave a complete list of all linked controls, displaying all dimensions in which the control has a position other than zero, which, in terms of the position of the search frame in the data space, looked like displaying the entire space around the target control and then focusing on a certain position in that subspace. We optimized the interface so that the user would first select the type of linkage he is interested in, i.e., the dimension, and then the value in that dimension.

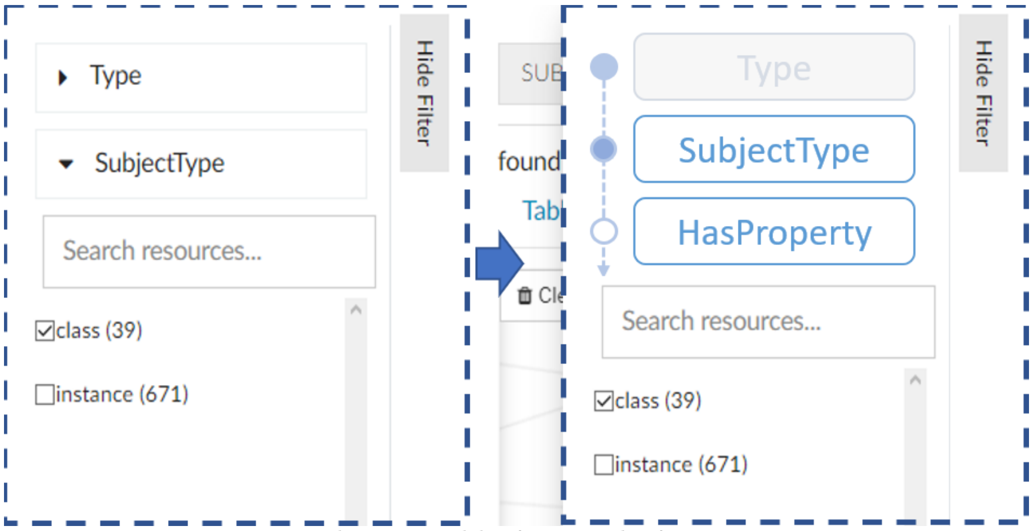


Figure 8 — Filter panel before and after optimization

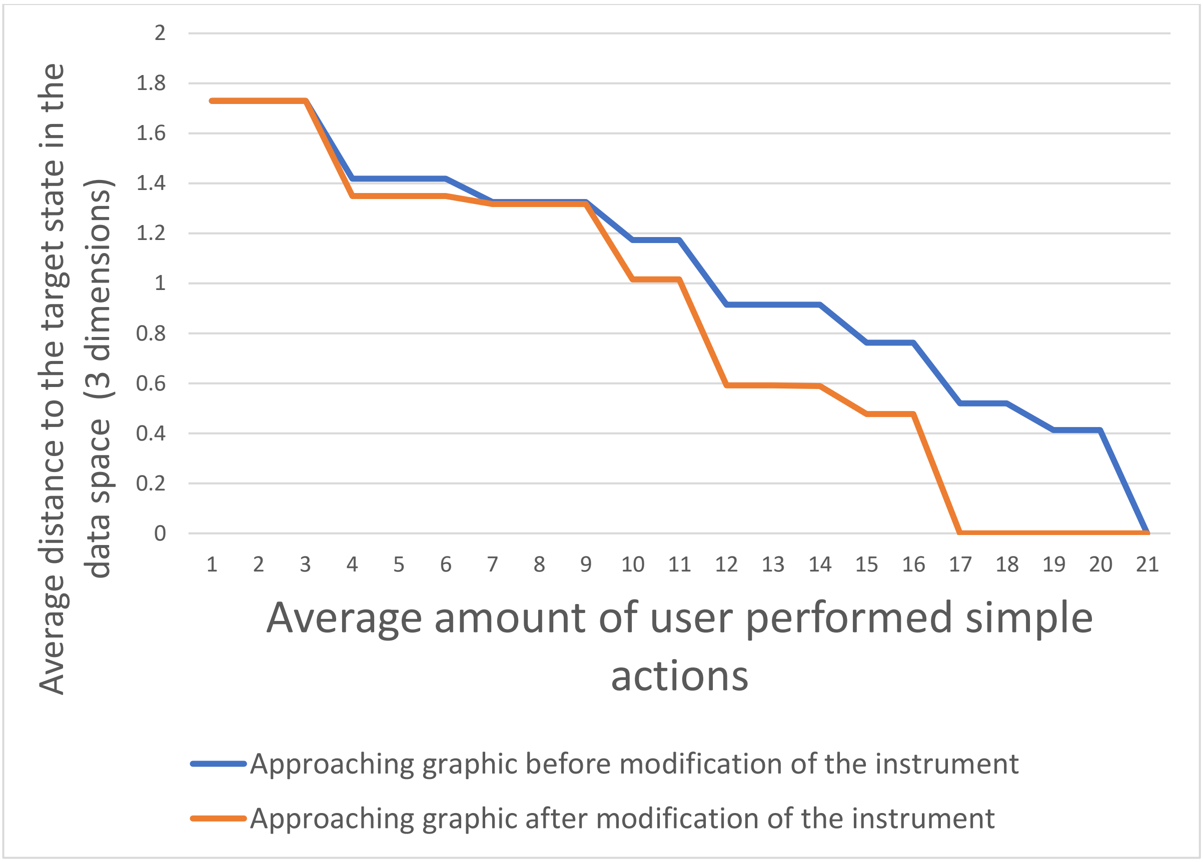


Figure 9 — Charts of convergence to the objective (target state)

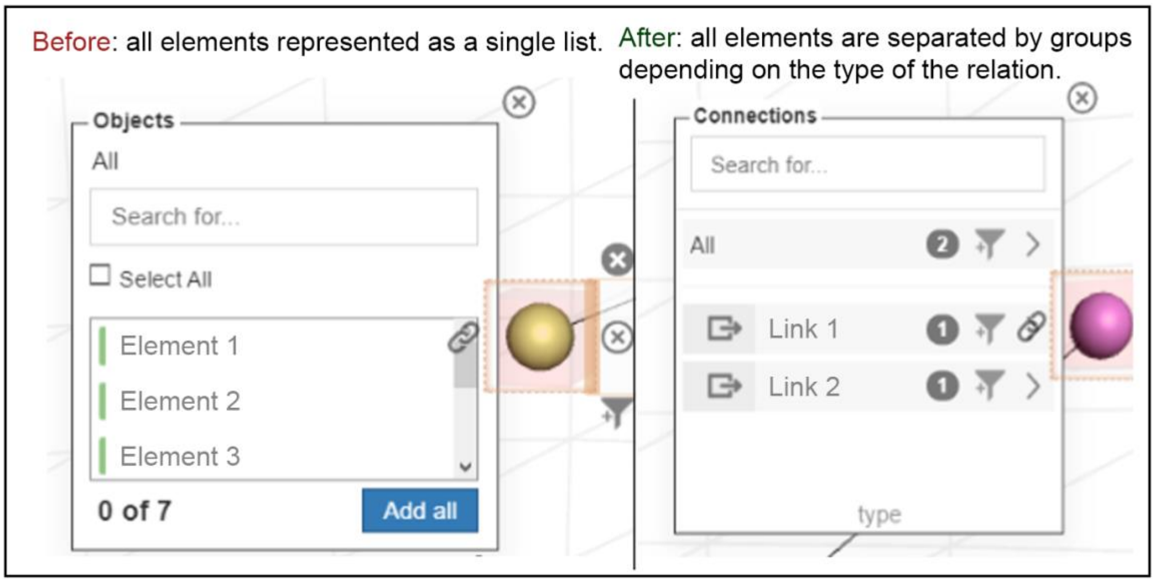

Before: all elements represented as a single list.
After: all elements are separated by groups depending on the type of the relation.
Objects
All
Search for...
Select All
Element 1
Element 2
Element 3
0 of 7
Add all
Connections
Search for...
All
2
Link 1
1
Link 2
1
type

Figure 10 — Link panel of the control before and after optimization

After computing the paths for the first and second interface variants, the new path in the state space appeared to be shorter. Figure 11 shows the weighted state graph with weighing coefficients denoting the distance in the state space between nodes (visualizations) (the graph was built using the service https://graphonline.ru).

6. Tools

The development and application of the method was performed using a commercial product Metaphactory [10] with a closed license (https://metaphacts.com/metaphactory-software-license-agreement). The Metaphactory platform provides an opportunity to modify the user interface of the web-application in real time. The described method was used to develop an auxiliary component that gathers and processes statistical data and allows evaluation of the configuration of the user interface in real time. To date, the component is in prototype status and is not included in the release build.

The main component of Metaphactory ontology graph visualization is the Ontodia component mentioned in [17, 24]. The component visualizes OD in the form of a two-dimensional graph, but for hypothesis testing a prototype of the component that visualizes in the form of a three-dimensional graph was developed. The public version of the Ontodia component (GNU General Public License) is available at https://github.com/metaphacts/ontodia. The Ontodia3d implementation uses the Metaphactory functionality, but the basic part of the component is placed in the open repository https://github.com/metaphacts/ontodia.

In the context of this study, we measured the parameters of labor intensity and speed of movement in data space using Ontodia and Ontodia3d components, based on which we formed appropriate layout-algorithms that minimize the labor intensity of creating visualizations.

A different use case assumes that the method can be applied to the development of ODVMT from scratch within a cyclic development model, where at each development cycle the method steps are applied and the interface is modified.

7. Conclusion

In this paper, we describe the concepts and definitions underlying OD and interactive visualization management processes and describe an agent-based model of interactive visualization. The main area for further research is the development of interactive visualization metrics for evaluating ODVMT.

In the A New Tool for Linked Data Visualization and Exploration in 3D/VR Space paper [6], based on the described interactive visualization model, we build a tool for visualizing OD in 3D and VR spaces to perform the process of interactive search (data exploration) and use this tool for visual editing of OD. The interactive visualization model and the visual management tool built on it are the basis of the OD visual management method described in this paper.

For this method, an approach to blank node visualization was also developed, which is described in Approach to Blank Node Processing in Incremental Data Visualization by the Example of Ontodia [17] and Approach to Blank Node Processing in Incremental Data Visualization by the Example of Ontodia [24]. This visualization approach eliminates the bottlenecks of visual management methods, i.e., it allows visual management (visualization and editing) of ODs containing blank nodes.

Identifiers of blank nodes cannot be used in SPARQL and UPDATE queries, which make it impossible to automatically build visualizations of graph fragments containing blank nodes as well as step-by-step editing of structures containing blank nodes. However, the approach described in [17, 24] assumes the use of context-dependent identifiers for blank nodes, which can be used for automated compilation of SPARQL and UPDATE queries as well as for step-by-step visualization of SCBN.

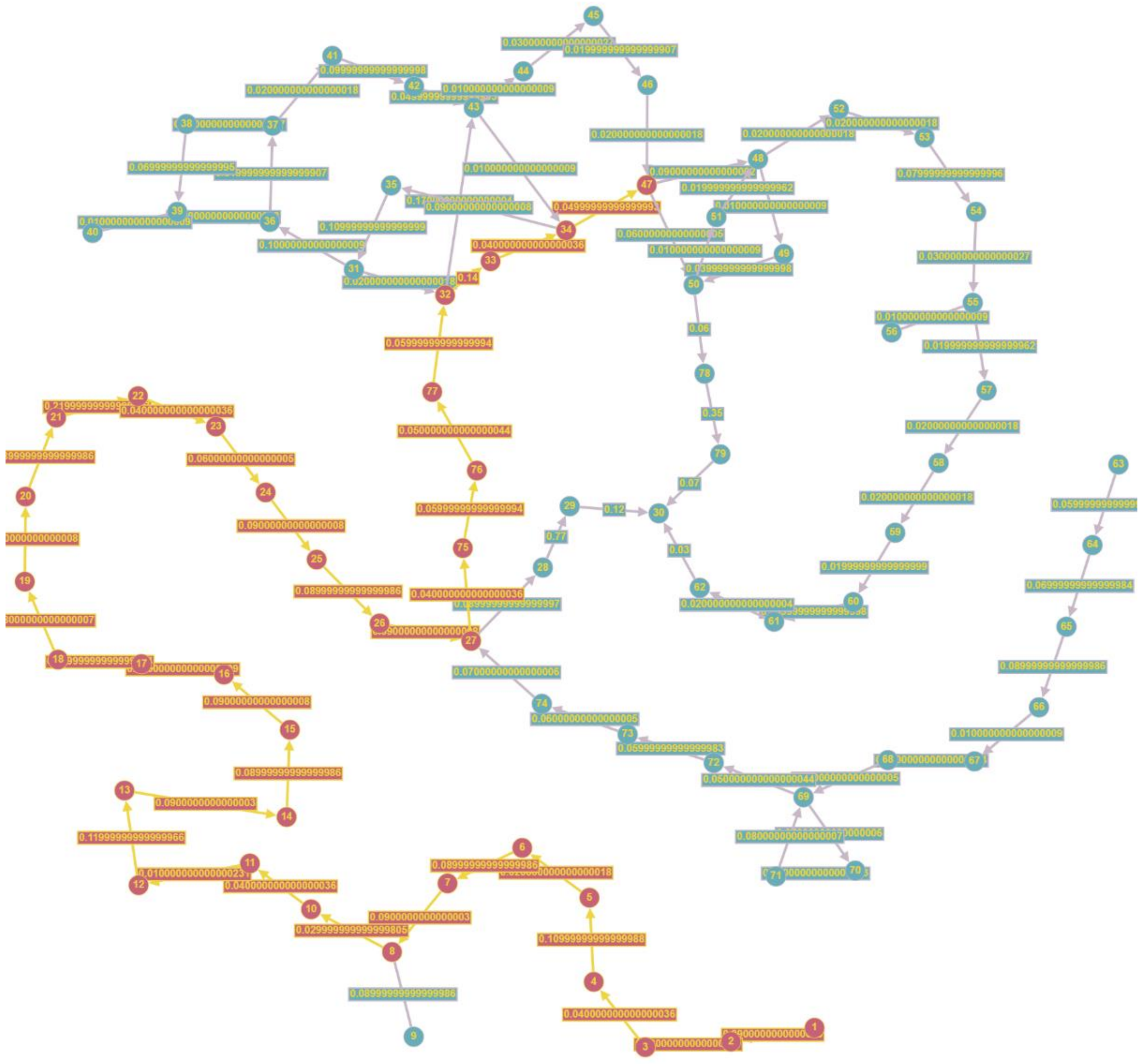


Figure 11 — Path in a weighted state graph

Another natural direction for continuing research on this topic is to investigate the possibilities of applying language models with attention mechanisms to compute the most relevant data for the user. In other words, the application of neural networks to automate the computation of the search frame size. The size of the search frame directly affects the amount of data that gets on the user's screen. Data coherence is lost if the search frame is too small, and data may be perceived for noise if frame is too large. The size of the search frame is the bottleneck of OD visualization and visual management tools, and automated calculation can improve the efficiency of OD management processes.

8. Terms and abbreviations

Data Management (DM) is the process of synchronizing data between multiple data sources with human involvement.

Ontological Data Management (ODM) is the management of data that is represented in an ontological format.

Interactive visualization (IV) is a visualization that has a changeable state, where the state of the visualization can change either over time or in response to external events (input operations or modifications to visualization parameters). As part of our study, we propose to use an interactive visualization model to optimize the ODVMT development.

OD visual management (ODVM) is a management process that is performed using ODM software tools that combine the functions of human-machine and machine-human interfaces, i.e., those where the OD visualization tool is simultaneously the OD editing tool.

OD — Ontological data.

ODMT — Ontology data management tool.

ODVMT — OD visual management tool.

SCBN — Structures containing blank nodes.

References

1. Ahmad, M. N., & Colomb, R. M. (2007, January). Managing ontologies: a comparative study of ontology servers. In ADC (Vol. 7, pp. 13-22).
2. Aminu, E. F., Oyefolahan, I. O., Abdullahi, M. B., & Salaudeen, M. T. (2020). A review on ontology development methodologies for developing ontological knowledge representation systems for various domains.
3. Barranco, C. D., Campaña, J. R., Medina, J. M., & Pons, O. (2007, July). On storing ontologies including fuzzy datatypes in relational databases. In 2007 IEEE International Fuzzy Systems Conference (pp. 1-6). IEEE.
4. Besbes, G., Baazaoui-Zghal, H., & Ghezela, H. B. (2015, July). An ontology-driven visual question-answering framework. In 2015 19th International Conference on Information Visualisation (pp. 127-132). IEEE.
5. Breitman, K. K., Casanova, M. A., & Truszkowski, W. (2007). Methods for ontology development. Semantic Web: Concepts, Technologies and Applications, 155-173.
6. Daniil, R., Wohlgenannt, G., Pavlov, D., Emelyanov, Y., & Mouromtsev, D. (2019). A new tool for linked data visualization and exploration in 3D/VR space. In The Semantic Web: ESWC 2019 Satellite Events: ESWC 2019 Satellite Events, Portorož, Slovenia, June 2–6, 2019, Revised Selected Papers 16 (pp. 167-171). Springer International Publishing.
7. Dudáš, M., Lohmann, S., Svátek, V., & Pavlov, D. (2018). Ontology visualization methods and tools: a survey of the state of the art. The Knowledge Engineering Review, 33, e10.
8. Fischer, H., Yigitbas, E., & Sauer, S. (2015, September). Integrating human-centered and model-driven methods in Agile UI development. In Proceedings of 15th IFIP TC. 13 International Conference on Human-Computer Interaction (INTERACT), S (pp. 215-221).
9. Gokhale, P., Deokattey, S., & Bhanumurthy, K. (2011). Ontology development methods. DESIDOC Journal of Library & Information Technology, 31(2).
10. Haase, P., Herzig, D. M., Kozlov, A., Nikolov, A., & Trame, J. (2019). metaphactory: A platform for knowledge graph management. Semantic Web, 10(6), 1109-1125.
11. Hashimoto, T. B., Alvarez-Melis, D., & Jaakkola, T. S. (2016). Word embeddings as metric recovery in semantic spaces. Transactions of the Association for Computational Linguistics, 4, 273-286.
12. Kamdar, M. R., Horridge, M., Wogulis, L., Fitzgerald, C., Hardi, J., Anderson, D., ... & Gonçalves, R. S. Interactive Exploration and Collaborative Curation of an Industry-Scale Healthcare Knowledge Graph Using the WebProtégé Cloud-Based Editor.
13. Jones, D., Bench-Capon, T., & Visser, P. (1998). Methodologies for ontology development.
14. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., & Giannopoulou, E. (2007). Ontology visualization methods—a survey. ACM Computing Surveys (CSUR), 39(4), 10-es.

15. Morente-Molinera, J. A., Pérez, I. J., Ureña, M. R., & Herrera-Viedma, E. (2016). Creating knowledge databases for storing and sharing people knowledge automatically using group decision making and fuzzy ontologies. Information Sciences, 328, 418-434.

16. Nassiet, D., Livet, Y., Palyart, M., & Lugato, D. (2011, July). Paprika: Rapid UI development of scientific dataset editors for high performance computing. In International SDL Forum (pp. 69-78). Berlin, Heidelberg: Springer Berlin Heidelberg.

17. Razd'yakonov, D. S., Morozov, A. V., Pavlov, D. S., & Muromtsev, D. I. (2020). Approach to Blank Node Processing in Incremental Data Visualization by the Example of Ontodia. Programming and Computer Software, 46, 384-396.

18. Sivakumar, R., & Arivoli, P. V. (2011). Ontology visualization PROTÉGÉ tools–a review. International Journal of Advanced Information Technology (IJAIT) Vol, 1.

19. Vo, H. Q., Phung, T. H., & Ly, N. Q. (2020, November). VQASTO: Visual question answering system for action surveillance based on task ontology. In 2020 7th NAFOSTED Conference on Information and Computer Science (NICS) (pp. 273-279). IEEE.

20. Vysniauskas, E., Nemuraite, L., & Paradauskas, B. (2011). Hybrid method for storing and querying ontologies in databases. Elektronika ir Elektrotechnika, 115(9), 67-72.

21. Wiens, V., Lohmann, S., & Auer, S. (2018, August). WebVOWL Editor: Device-Independent Visual Ontology Modeling. In ISWC (P&D/Industry/BlueSky).

22. Wohlgenannt, G., Klimov, N., Mouromtsev, D., Razdyakonov, D., Pavlov, D., & Emelyanov, Y. (2019). Using word embeddings for visual data exploration with ontodia and wikidata. arXiv preprint arXiv:1903.01275.

23. Maslennikov O.P., Milman I.E., Safiulin A.E., Bondarev A.E., Nizametdinov Sh.U.1, Pilyugin V.V. (2014). Development of a system for analyzing of multidimensional data. Scientific Visualization, 6(4), 30-49.

24. Razdyakonov D.S., Morozov, A. V., Pavlov, D. S., & Muromtsev, D. I. (2020) Approach to Empty Node Processing in Portionalization of Data Example Ontodia. Programming, (6), 16-29.

25. Alatrish, E. S. (2013). Comparison some of ontology. Journal of Management Information Systems, 8(2), 018-024.