OWL-Ontology Visualization Tool

Pavel Smirnov, Dmitriy Mouromtsev
SPB National Research University of
Information Technologies, Mechanics and Optics
Saint Petersburg, Russia
{smirnp, d.muromtsev}@gmail.com

Abstract

This paper describes an idea of visual modeling tool for knowledge bases. The tool is planned to be an interactive multimedia application, which intuitively demonstrates complicated ontological structures and makes cognition process more effective and interesting. This work is a part of project concerning the optics ontology development and building an educational portal to introduce it into Optics Museum of ITMO University. The tool visualizes ontology in form of a graph and is planned to be used on tablet PC-s or info-stands installed in the museum.

Index Terms: Visualization, Ontology Engineering, Flex, ActionScript.

I. Introduction

What role does visualization play in everyday life? It's not a secret, that the most progressive visual technologies are used in fields of entertainment: computer games and cinematography. But in spite of it, a visual modeling also is widely used in fields of science, education, medicine, architecture, building, machinery and so on.

How visualization can impact on educational or scientific processes? From this point of view visual technologies can be applied for modeling structures, processes, making work with data more demonstrative and user friendly. Semantic data and ontologies, to be basis of many scientific or educational portals, can be a good source for visualization applications.

II. Ontology as a platform for knowledge bases

Knowledge base is a kind of database aimed to operate with knowledge or metadata [1]. KB consists of structured data concerning with some field of science and supposed to be used in a reasoning process by some device or human with a concrete goal. Any hierarchical structure of items, objects, definitions, properties and relations is called ontology. In [2] it is shown that ontologies are useful structuring tools. Formally ontological model can be described by RDF-format (Resource Description Framework)[1] or OWL (Web Ontology Language)[2]. OWL is a language for making ontological statements, developed as follow-on from RDF. OWL is intended to be used over the World Wide Web, and all its elements (classes, properties and individuals) are defined as RDF resources, and identified by URIs. Both RDF and OWL can be presented as XML-nodes, which describe data objects and relations between them.

Development of ontology modeling and interfaces to knowledge bases provides an access to reliable information of very specific fields of knowledge, which traditionally

[1] http://www.w3.org/RDF/

[2] http://www.w3.org/OWL/

were a task to be solved by experts only. Usually ontology is formed by analysts according to information from protocols, documents or expert experience in concrete thematic [3]. KB consists of static facts and a set of rules (relations), which is used by a reasoner to suggest automatic decisions making regarding to new facts stored to it. KB can be considered as a core of an artificial intelligent system, because it make it possible to logically processing information and inferencing results according to a set of initial parameters. A part of science about artificial intelligence concerning of KB development is called knowledge engineering and learns as well methods of ontology forming. Combining ontologies and wiki-systems could be a easy and powerful web-platform for KB. Wiki-systems has became popular during last 5-7 years and widely used by in scientific, research and educational fields to publish knowledge online, exchanging information and organizing communities [4].

III. Practical application

The idea to implement some graphic tool to visualize ontological structures came to authors during a work on knowledge portal project [5]. Ontological structures visually can be represented in two ways: mind map [6] or concept-map [7]. The last one is more preferable to use for optics KB, because it does not require to build a diagram around one central object. It also allows to introduce different types of relations between individuals. Relations can be labeled with linking phrases such as "gives rise to", "results in", "is required by," or "contributes to". Objects, definitions, properties and relations altogether form a complicated structure, i.e. ontology. Technically concept map can be generated like a graph with named relations between objects. Visual objects, especially graphs, have a specific cognitive strength and can perform a valuable tool of cognitive graphics for structuring information. As a result, visual appearance in case of semantic ontologies gives a strong instrument or way of cognition [2].

As a basic platform for application we decided to use SpringGraph – an open-source flex (actionscript) library [8]. It provides a reach multimedia functionality - interactive browsing through graph nodes.

Comparison SpringGraph with other graph visualization engines showed it’s distinguished advantages. The greatest one is web-oriented implementation. Flash-application does not require any OpenGL-abilities from device, can be easily deployed on website and run in client browser, which supports flash. For example, Protégé offers a number of built-in visualization plugins like OntoViz or IsaViz [9]. The weak point of the first one is the fact that it is not able to produce good (understandable) layouts for graphs having more than 10 nodes. The second one uses GraphViz package for the graph layout. Both of them perform functionality to edit items, but have not web-implementation. UbiGraph [10] is an more powerful visualization tool, but it is server-side and requires installation of special client to be shown on client machine, so it’s not acceptable to use on website. Another strong advantage of SpringGraph is interactivity and UI-fluidity. Graphviz [11] library makes diagrams in formats, such as images and SVG for web pages, PDF or Postscript to be included in other documents. GraphLight [12] can offer WPF and Silverlight implementation. But all the tools below (except UbiGraph) provide only static visualization, without any run-time node positions calculations, which perform an amazing behavior. It’s actual because features like this implements one of key ideas to make cognition process more exciting and effective. Practically it would be actual, for example, studying museum’s exposition by children.

SpringGraph presents the radial layout tree with a node-link representation. In this layout, nodes are placed on concentric circles according to their depth in the tree (see figure 1). Space-efficiency comparison of existing visualization methods is described in [13].

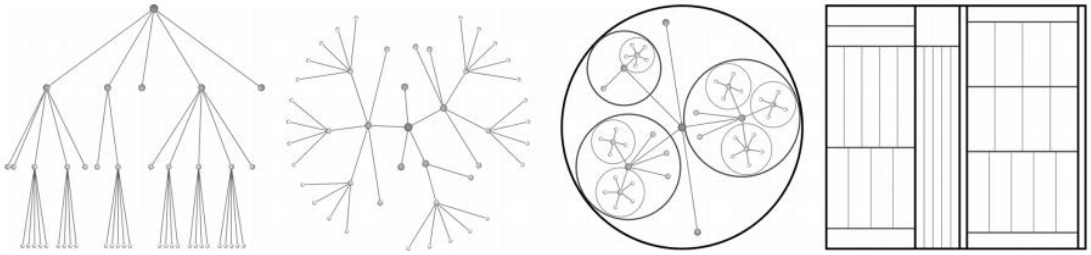

Fig. 1. Common tree visualization techniques
From left-to-right: rooted tree, radial tree, balloon tree, treemap layout

Standard SpringGraph features include processing of automatic nodes positioning, edges length calculations, expanding and collapsing nodes. The library is fully open-source and some new functionality like new events, actions, edge names, arrow directions, types of nodes, search through ontology objects was implemented during this work.

IV. The program architecture and algorithm

Technically flash-application consists of various classes, objects and interfaces. The most valuable of them are presented in this section. Interaction between these objects during data binding is displayed on sequence diagram at Fig. 2.

- *Application* – the standard actionscript object responsible for preparing application environment (width, height, background). It is a parent object of the whole application and contains all other children objects.
- *HTTPService* – the standard actionscript object aimed to send http-requests to specified URL and return a response.
- *Roamer* – the custom object inheriting and overriding *SpringGraph*-class. Roamer is responsible for a user applying interface functionality regarding of amount of displayed nodes, a length of edges, a graph depth and surfing history. Roamer detects changes of parameters and calls built-event from SpringGraph-class.
- *SpringGraph* – the custom class providing a set of properties and graphic methods outputs nodes and edges to the screen. It initializes Drawing Surface - graphical area for nodes and edges, which detects mouse moves, click-events and operates with size and scrollbar according to screen resolution.
- *Graph* – the custom object operating with an array of nodes and edges like a graph structure. It contains a set of methods like: add/remove, link/unlink, return neighbors and etc. Graph also serves as a data provider for *SpringGraph*.
- *Item* - the custom object to present a graph node. Used as an elementary object in *Graph*. Have appropriate attributes like: id, name, url, description and some custom (ex. shape or color).

.

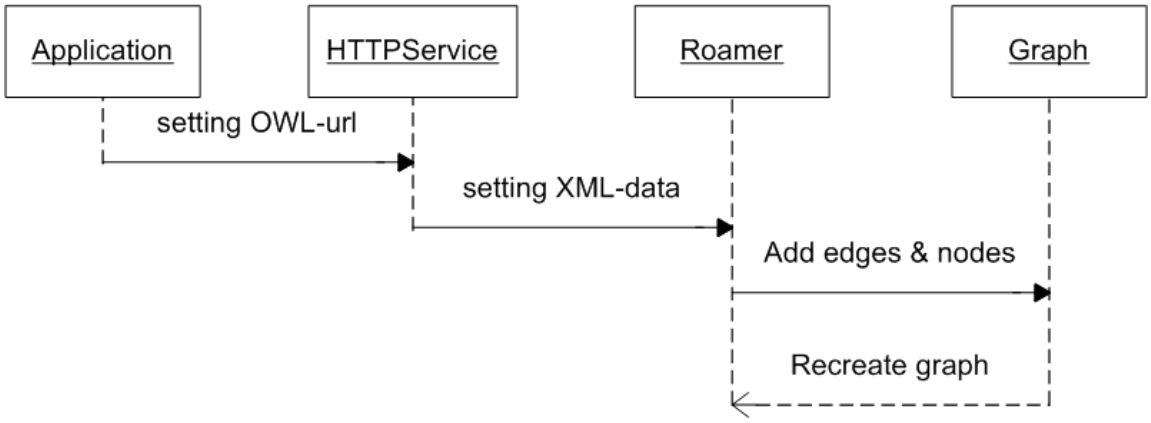

Fig. 2.Interaction between main objects in flash-application

After *Application* loading is completed, an URL-address is setting to HTTP-service. The URL is declared in JavaScript parameters of the flash-application. URL returns a special formatted Xml-file with an OWL-ontology structure. This file is generated by special Semantic Media Wiki extension, written in Java during previous work [5]. *HTTP-service* sends request and returns set of XML-objects, which is set to *Roamer* as a data provider. After the *Roamer*'s data provider changed, a new instance of *Graph* is created. It is filled by nodes (*Item*) extracted from XML had been received earlier. The next *SpringGraph* calls a draw-method to put elements into drawing surface and to render edges. Then a recreate-method is called iteratively until an optimal node positions and edge lengths would be found.

V. Basic use case

Surfing through ontology items could be done via clicking on nodes. Selected node is being centered on the screen and his neighbors would be shown. Application remembers surfing history and highlights visited items with a shadow. Double click on node calls an upload of detailed information about selected node to special html-frame placed above flash-application. User interface also contains some preferences controls. User can change an amount of items shown on the screen, an average length of edges and a graph depth. Ontology name and amount of nodes showed in UI as statistic information. User can also use a search through nodes, presented as drop-down list. If there would be too much items, it can be by auto-complete input field.

VI. Implementation

The result of technical part of this work is a flash-application, which is integrated into wiki-portal and represents the ontology of knowledge base. A special extension for wiki-engine was coded to automatically convert a description of nodes and relations from ontology to a specially formatted XML-file, being further used by application. A graph is building according to a page information declared through JavaScript-variables in html-code. This allows to select necessary node programmatically and makes possible an application uploading, if user surfs website traditionally (not through application). Application got an original UI design, which improves process or remembering/reproducing content, makes process more interesting and pleasant (see fig. 2). This tool is planned to be used not only on website, but also on mobile gadgets,

electronic informational stands in museums, libraries or other scientific or educational areas.

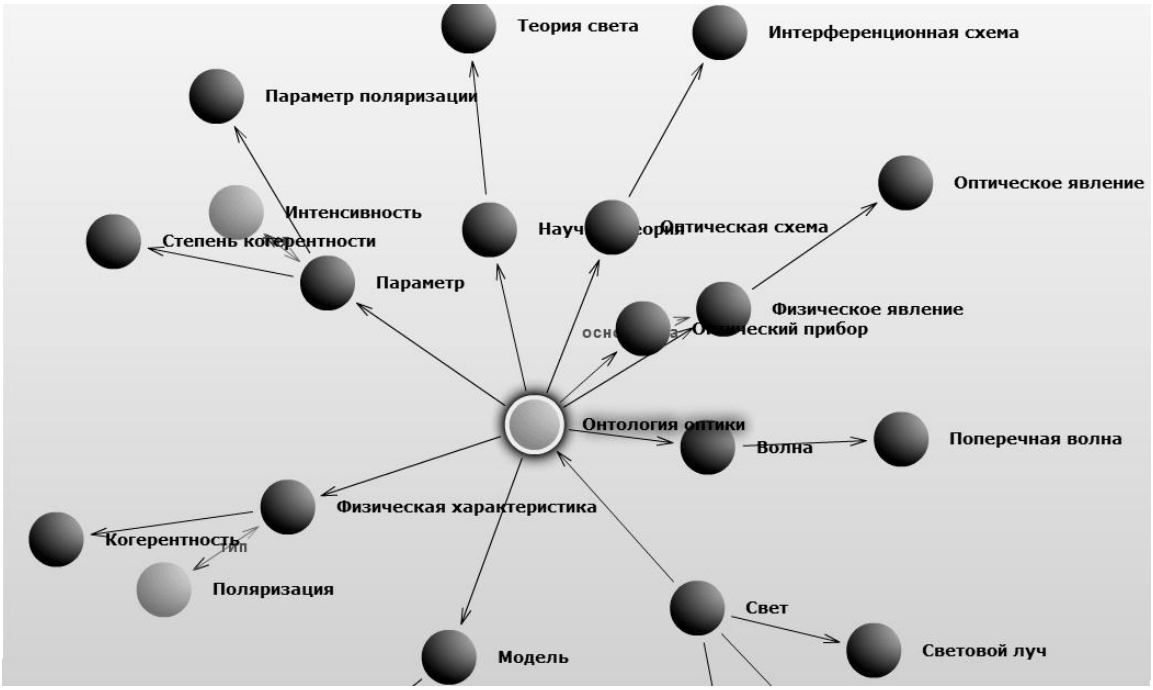

Fig. 2. Ontology of optics visualized by our application

VII. Conclusion and future work

The presented paper contains a short overview of the ontology engineering theory and knowledge base building aspects. The paper also describes the benefits, which visual modeling brings to educational and science areas. Some technical details of created application were overviewed here. Now application works with our ontology of optics and provides relations between physical phenomenon and persons, who discovered or declared it. The ontology was built by another part of our team and for the future we are planning to introduce it to Optics Museum of ITMO University [9] and provide it with electronic guides based on developed application.

References

[1] Knowledge base definition http://en.wikipedia.org/wiki/Knowledge_base.
[2] Gavrilova T. Knowledge Mapping for Teaching and Learning // Int. Journal ““The 21st Century: a scientific quarterly””, Nr 2(24), Warsaw, Poland, 2007.
[3] Gavrilova T., Muromtsev D. Artificial technologies in management, SpbSU, 2008.
[4] Examples of educational wikis, http://educationalwikis.wikispaces.com/Examples+of+educational+wikis.
[5] Zlobin A., Katkov Y., Mouromtsev D., Pochinok I. "Development knowledge base on optics for educational web applications". Artificial intelligence and desigioning 3/2011.
[6] Mind map definition http://en.wikipedia.org/wiki/Mind_Map.
[7] Concept map definition http://en.wikipedia.org/wiki/Concept_Map.
[8] SpringGraph homepage http://www.chaosreigns.com/code/springgraph/.
[9] F.Frasincar, A.Telea, G. Houben «Adapting graph visualization techniques for the visualization of RDF data» http://people.few.eur.nl/frasincar/papers/VSW2005/vsw2005.pdf.
[10] Graphviz homepage http://www.graphviz.org.
[11] UBIGraph homepage http://ubietylab.net/ubigraph/.
[12] GraphLight homepage http://graphlight.codeplex.com/.

[13] Danny Holten, Hierarchical Edge Bundles:Visualization of Adjacency Relations in Hierarchical Data http://www.win.tue.nl/~dholten/papers/bundles_infovis.pdf.
[14] Optics Museum of ITMO University http://optimus.edu.ru.