A Method of Construction of Automated Basic Ontology

Vasyl Lytvyn[1], Victoria Vysotska[2], Waldemar Wojcik[3], Dmytro Dosyn[4]

[1,2]Information Systems and Network Department, Lviv Polytechnic National University, Bandery str., 12, Lviv, Ukraine, 79013

vasyl.v.lytvyn@lpnu.ua[1], victoria.a.vysotska@lpnu.ua[2]

[3]Institute of Electronics and Information Technology, Lublin University of Technology, Nadbystrzycka str., 38A, Lublin, Poland, 20618

waldemar.wojcik@pollub.pl[3]

[4]Systems Analysis Laboratory, Karpenko Physico-Mechanical Institute of the NAS of Ukraine, Naukova str., 5, Lviv, Ukraine, 79060

dmytro.dosyn@gmail.com[4]

Abstract. The paper describes an approach to development of a computer system that automatically constructs an ontology base. Basic modules of the system and its operation are described, as well as the choice of software tools for implementation. Application of the proposed system allows to fill the domain ontology in an automatic mode. Therefore, this paper introduces an approach to development of an automated basic ontology composition. An architecture of synthesis of the ontology system is created using CROCUS (Cognition Relations or Concepts Using Semantics) software model. The main system modules and their functions are described. A decision of SDK for system realization is justified. Application of the proposed system can fill an ontology of subject area automatically.

Keywords: computer system, ontology, knowledge base, database, text document, machine learning, intelligent agent, utility, semantic, logic of predicate.

1 Introduction and review of current research on the topic

A study of results and developments in the area of intellectual information systems and Internet services [1-8] led us to suggest the following soft/hardware decisions:

- Realization of the ontology synthesis system as a subsystem of the Internet portal system;
- Using OWL as a knowledge presentation language;

- Using HTN and OWL-S as structures of the automated knowledge base planning language;
- Java API for Protégé OWL as the API and the library processing classes, in particular for the machine learning (reinforcement learning) of the OWL-ontology and knowledge bases;
- Link Grammar Parser as an instrument of the grammatically-semantic analysis of English text documents [2];
- Apache-PHP-MySQL as a software tool to build a web-portal based user interface;
- Wget as a web service for automated access to search engines with a query, formed from the keywords;
- SWRL as a logical new knowledge output language with deductive and inductive methods;
- WordNet as the basic glossary of English.

An ontology in OWL language contains a high-level meaning automation of the subject area [3]. A high-level ontology provides [4]:

- A logical output of new knowledge with the addition of new messages with the context [5];
- The verification of the validity of obtained statements [6];
- The evaluation of the probability of the message sources [7];
- The ensuring of the knowledge base logical integrity [8].

The machine learning is provided with the means of Java API Protégé-OWL. These means contain libraries of classes, which realize methods to work with OWL-structures like reading and addition. Therefore, the tools of machine learning work in addition to the OWL-ontology. Java API Protégé-OWL takes templates of grammatically semantic structures to recognize statements (the first order predicates) into the research and/or educated texts including new elements as the result of such recognition [9]. Link Grammar Parcer [2] divides a grammatically correct definition of sentence into interconnected pairs of words. LGB contains a table that has all the conformities between grammar constructions of English and syntax-semantically links between words (intellections). LGP API allows to link this table to OWL-ontology, so the table can adapt in the process of learning the given object area dynamically. Java API Protege-OWL based means of machinery education contain a generalized description of the semantic link, which serves as the template for generating new types of semantic link during studying. In addition, it forms appropriate vectors and indications of these links to form and identify semantic links in a text. Herewith, properly classes of links and their properties adding to the OBP. Exemplars of those classes are for the description of existing and new classes of ontology by their use as first rank predicates.

ZMN ontologies make sense only as the part some intellectual system. Optimal solutions in our opinion are that, where such an intellectual system in the information search system for which an adaptive ontology is an instrument for information research, analysis and classification on the one hand, which uses search instrumentalities to provide data for its filling, new predicates and rules synthesis, learning new means and semantic links on the other [10-11]. An intellectual system of information searching based on the adaptive ontology, material science knowledge base, a database of scientific publications became such a decision [12-13].

A developed architecture of the ontology synthesis system was realized with the usage of selected and decrypted means and program-technically solutions as a CROCUS (Cognition Relations or Concepts Using Semantics) [14-17].

2 Main modules of CROCUS

The overall concept of CROCUS at the fig. 1 is introduced. A subsystem of the ontology learning uses training texts of annotations of scientific publications from article DB. The system forms a plural of key words to fill the DB in. It chooses the main metadata about the publications in the defined subject area in Internet (ScienceDirect, CiteSeer, Wiley Online Library, and Springer) including their annotations, which become the core of analysis and ontology learning.

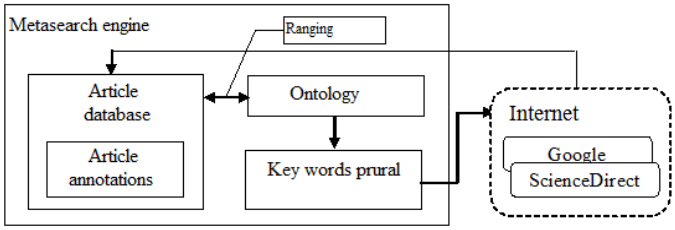

Fig. 1. CROCUS concept schema

The essence of the knowledge extraction method from the natural text document is into building of the intellectual agent activity strategy – an informational model of the recognition subject of its specification based on dedicated from recognized text document data. A plan is considered to be a specific optimal strategy realization of some task, which has an intellectual agent within the subject area.

The plan is built with the same with informational model formal knowledge representation language – a database of an intellectual agent. Considering that, such a knowledge base is already an overall plan of intellectual agent functioning, build basing on the natural text recognition is a sub-plan. It means that it is a specification of an overall plan and it bases on it. A value of information, received as a result of recognition the context of a text document is determined as increasing of the updated intellectual agent functioning plan expected utility. Scientific publications range for the relevance to the users informational demands, for the conformity to ontology, which displays these demands. An analysis of each annotation as natural text is made, builds its image in the terms of ontology as predicates and rules in this purpose. These predicates and rules are added in the knowledge base of the system and the expected utility of an intellectual agent is calculated again. A system puts those publications nearer to the beginning of the list, which data including leads to the greater reliability change with such a type of ranging.

A system can adapt to the users requirements by saving his preference system in the DB. Each user can perform an education of his ontology. The system saves the data about this process, leads the session statistics, and provides the possibility to correct the education errors and does backtracking to previous versions of ontology.

CROCUS system modules are shown at the picture 2. A client has a possibility to control the priority of document ranging, to correct their order in the list of the most important (relevant to the client informational requirements) document and classify them with the help of graphic interface. The most important documents are used for ontology learning and building of the efficient sets of key schemas and new, received from Internet, articles (their metadata including annotation) insert into the DB of publications with the link with preferences of the user and other prerequisites of the document receiving.

An annotation processing after its previous processing, conversion into the massive of predicates as a result of grammatically-syntax analysis of Link Grammar Parseris executed. Formed annotation models are supplemented with semantically near ontology predicates – the context of this annotation. Supplemented annotation models are compared between themselves to calculate the semantic length between their semantic weight midpoints and so the nearest by content documents are chosen with their further ranging and classification. Two main functions of CROCUS:

1. An interactive automatic building of the problem area ontology;
2. Searching, saving and classification (ranging) of scientific publications as in interactive semiautomatic as in automatic mode.

Each of these functions is realized with its base set of functionality modules but a part of them was a double appointment. CROCUS is realized as an object oriented paradigm by using Java as a hierarchy of code classes, which copies call each other with determined at that moment parameters or they interact through throw events and/or handlers. Most of them have a Swing graphical interface and AWT libraries. All the connected libraries have an open source status. Project has a full functionality and has all the necessary means for its development (evolution). A functional assignment of the main CROCUS system modules is shown at the fig.2.

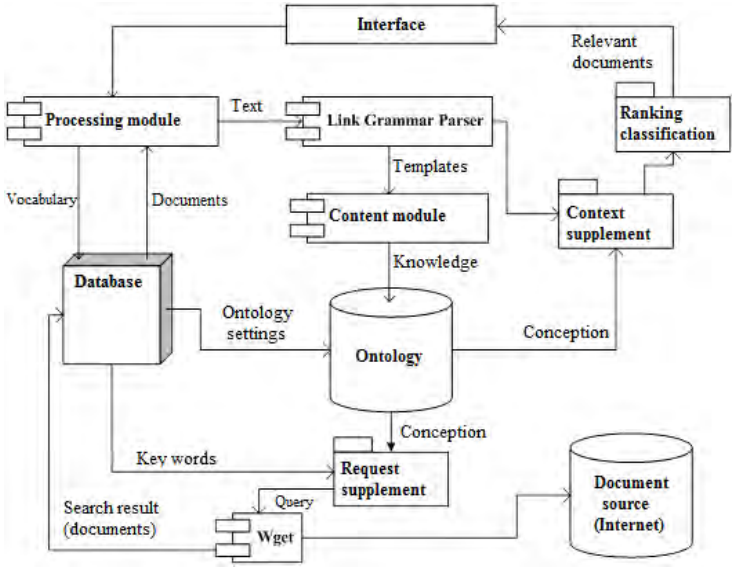

Fig. 2. CROCUS modules

Detailed analysis of similar systems is made in [14-17].

3 Functionality of CROCUS modules

Functionality of the main CROCUS modules is described in table 1.

Table 1. Functional purpose of the main CROCUS modules

№	Modules	Functionality
1	Attribute.java	A subprogram to detect attributive links in sentences, or rather verbal links. Each adjectival link is a form of verbal, actually: –A tomato is green = a tomato**'s color is** green". Herewith an ontology has to contain a binary predicate –**eolor is**" (<domain> = <tomato>, <range of acceptable values> = <a green color>)
2	BinaryLink.java	Determination of the horizontal binary link weight
3	CGrammarObject.java	Breaks sentences into words-tokens which are accompanied by a letter, which means a type of word (n-noun, v-verb etc.), so it is possible to get words and its language type using methods **getName**() or **getType**(). Type is determined by LinkGrammarParser
4	CGrammarObjectArr.java	A copy constructor of this class forms the massive of grammatical objects from the text line _sentence'
5	CGrammarObjectType.java	Class of elements type massive CGrammarObjectArr
6	CLinkType.java	Class of methods of breaking link type arrays between pairs of words in sentence at the main type of link **getType**() and its subtypes **getSpecification**()
7	ControlGUI.java	The main program control window
8	CSOLink.java	Breakes the results of Link Parcer work – determines the conformity of specified by Parser types of link to the words, marked by brackets into the previous string
9	CSOLinksArr.java	Puts exemplars of CSOLink type into a dynamic array
10	Descriptor.java	Forms a prural of patterns into one array
11	DParsedData.java	Forms text lines with the results of LGP work
12	FunctionGUI.java	A graphical interface template
13	GSL.java	Forms a pattern of semantic link type to recognize semantic links by their patterns
14	Is_a.java	Procedure of adding subclasses to existing classes
15	MainProc.java	The main procedure in _non-graphical variant' – it is not supported since 02.05.2011 12:38. 1. Creates constant parameters to initialize an external for this package Link Grammar Parses; 2. Opens 3 threads: to enter Parser, to output its data and errors; 3. The online URL-address of an ontology is indicated; 4. The name of text file, which will extend an

№	Modules	Functionality
		ontology, is read from the string of program initialization (1st parameter in string); 5. An ontology is read from the specified address; 6. Its contents is put into standard output channel; 7. Contents is put into the resulting.owl file using the DumpOWLModel procedure; 8. Searching Is-a links; 9. Searching Consist-of links; 10. Executing an Attribute procedure.
16	MetaObject.java	Creates a structure to describe the semantic object – a subject of action or an action as a type of link between subject and object
17	OntologyMapClass.java	Adds classes to the indicated level of an ontology **(addClassesToLevel)**
18	OWLEvaluation.java	The most applied procedure of this class (its objects) • Calcuting the weight of concepts and link of an ontology • Output an antology into <file.owl> using DumpOWLModel(String file_name) • ShowAll – making the procedure of visualization into the standart output channel.
19	Parser.java	Initialization of Link Grammar Parser, its customization, inputting the text file with sentences, saving the result of Parser execution into it.
20	Part_of.java	Recognizes semantic links „Part-of" type into English sentences (after their processing into Link Grammar Parser).
21	Pattern.java	A class Pattern constrictor, in which creates a semantic link pattern **for its recognition into sentences using static method.** 3 main elements are collected „subject -> meta link -> object", in particular „subject -> metalink" and „metalink -> object". They became during education of this semantic link so they play the main role (because they are the most common). Statistics saves as exemplars of the semantic link class in an ontology: default_<semantic_link_name>_ <metalink:{D, S, O,...}>_<Subject|Object>_ <subject/object_name>
22	SemLinkDescriptArr.java	Chooses a sequence of SemLinkDescriptor type descriptors from the investigated sentence and creates a dynamic array of them.
23	SemLinkDescriptor.java	Triplet semantic link descriptor: metaSubject (string) -> link-type (string) -> metaObject (string) A semantic link signs vector has to consist of such triplets. Each triplet of each semantic link type has its value coefficient. A size of such a vector is 10.

The basic system control element in CROCUS is ControlGUI module (user graphical control interface). This module has a graphical interface by which user can

execute procedures, which are provided by the system functionality. Module has the main menu and its main functions are in the toolbar. Control output is carried out at the appropriate text panel. There is an output field and an input field at the underside of the main window to specify the semantic link type at the process of ontology learning using learning sentences (fig. 3).

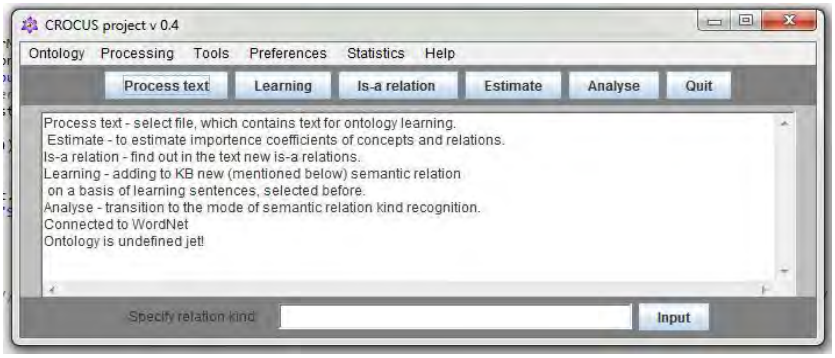

Fig. 3. The main window of CROCUS user interface

Despite the importance of the effective dialog between system and user, a great attention during developing was to the graphical interface design, its intuitiveness and pithiness with a functional completeness of project tasks realization. In addition, there is a possibility to zoom interface to expand its functionality. Despite the intense competition between the similar projects, a great importance was to create a recognizable logotype, which will be replied to the content of the word CROCUS. An experience of such foreign projects confirms an efficiently of such an approach (Protégé to work out with OWL-Ontology, project GATE, etc.). That is why all the windows in dialogue with user are decorated with CROCUS logotype – an illustration of 6-petal saffron (crocus) flower. Professional designers developed an image and the design of main windows interface.

System has an internationalization of whole text dialogues. User can choose a comfortable interface language from four available. There is no problem in language addition. A dialog language file MessagesBundle_xx_XX.properties has to be translated, where XX – the code of a language (RU – Russian, UA – Ukrainian etc.). To choose the dialogue language you have to choose a subparagraph ‗language' into the paragraph of the main menu ‗Preferences'.

4 Justification of the SDK choose

A using of SDK common libraries gives a chance to avoid unjustified overrun of time, finances and human resources for their redevelopment. Therefore, there is a wide list of investigated currently working analogue projects in this work. Most of them use the concept of open source code and free licensing. The leading developers groups provide their projects with API (Application Programming Interface) means, so the functionality of these objects may be efficiently used by cataloged and well-

documented procedures and functions with appropriate settings.

Co-authorship of SDK developments worked out principles of software application usage with different license agreements. In addition, they can take part in support and development of existence projects, so each developer has a possibility to get and install these project or libraries and use them as he or she wants. Such internet portals as SourceForge.net contain all the necessary instrumentals for documentation and support projects of every level of difficulty, readiness, access level and popularity between users. Developers actively use special foundation servers, which provide collective (let it be 1000 developers) software developing. The most popular foundation server is Git. It can be installed separately as individual or corporative server and it is possible to use a global GitHub server.

Researches show, that most it develops in text documents natural processing, almost all develops in ontology learning are performed on Java. Moreover, Java is dominating between projects languages at SourceForge resource.

A decisive argument to Java usage is an accessibility of Protégé-OWL Java API of Stanford University (USA), because Stanford Center for Biomedical Informatics Research became a flagman of practice developments in OWL SDK-s.

Projects made by Java:

- Gate [http://gate.ac.uk/] – a couple of text documents processing means to find a new knowledge;
- owlapi.sourceforge.net – another one Java project, which is an OWL documents processing Java classes library with broad functionality;
- Pellet [http://clarkparsia.com/pellet/] – a logical output machine to realize thinking (new knowledge output) from OWL 2.0 knowledge base.

5 Conclusions

Therefore, this work shows an approach to develop an automated basic ontology building. An architecture of ontology synthesis system as CROCUS (Cognition Relations or Concepts Using Semantics) software model is created. The main system modules and their appointment are described. A decision of SDK for system realization is substantiated. A usage of such a system can fill an ontology of subject area automatically.

References

1. Ourania Hatzi, Dimitris Vrakas, Nick Bassiliades, (2010). Dimosthenis Anagnostopoulos, and Ioannis Vlahavas. The PORSCE II Framework: Using AI Planning for Automated Semantic Web Service Composition the Knowledge Engineering Review, Cambridge University Press, Vol. 02:3, 1–24 p. (In English)
2. Link Grammar – Carnegie Mellon University, available at: http://bobo.link.cs.cmu.edu/link.
3. Qiu Ji, Peter Haase, and Guilin Qi (2008). Combination of Similarity Measures in Ontology Matching using the OWA Operator, In Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Base Systems.

4. Lytvyn V. (2013). Design of intelligent decision support systems using ontological approach, *An international quarterly journal on economics in technology, new technologies and modelling processes*, Krakiv-Lviv, Vol. II, No 1, 31 – 38 (In English).
5. Gruber T. A. (1993). Translation approach to portable ontologies. Knowledge Acquisition, № 5 (2):199–220.
6. Guarino N. (1995). Formal Ontology, Conceptual Analysis and Knowledge Representation. International Journal of Human-Computer Studies, 43(5-6):625–640.
7. Sowa J. (1992). Conceptual Graphs as a universal knowledge representation. In: Semantic Networks in Artificial Intelligence, Spec. Issue of An International Journal Computers & Mathematics with Applications. (Ed. F. Lehmann), № 2–5:75–95.
8. Montes-y-Gómez M. (2000). Comparison of Conceptual Graphs [Електронний ресурс]. Lecture Notes in Artificial Intelligence, Vol. 1793. – Springer-Verlag. Режим доступу до журналу: http://ccc.inaoep.mx/~mmontesg/publicaciones/ 2000/ComparisonCG.
9. Muller H.M., Kenny E.E., Sternberg P.W. (2004). ―An Ontology-Based Information Retrieval and Extraction System for Biological Literature". PLoS Biol. 2(11):e309. doi:10.1371/journal.pbio.0020309.
10. Knappe R., Bulskov H., Andreasen T. (2004) Perspectives on Ontology-based Querying // International Journal of Intelligent Systems. – http://akira.ruc.dk/~knappe/publications/ ijis2004.pdf.
11. Jacso, Peter. (2010). ―The impact of Eugene Garfield through the prizm of Web of Science,". Annals of Library and Information Studies, Vol. 57, p. 222.
12. Christoph Meinel Serge Linckels (2007). Semantic interpretation of natural language user input to improve search in multimedia knowledge base, Information Technologies, 49(1):40–48.
13. Giorgos Stoilos, Giorgos Stamou, and Stefanos Kollias (2005) A String Metric For Ontology Alignment, Proc. of the 4rd Int. Semantic Web Conf. (ISWC), vol 3729 of LNCS, p. 624–637, Berlin. Springer.
14. Lytvyn V., DosynD., Smolarz A. (2013). An ontology based intelligent diagnostic systems of steel corrosion protection, *Elektronika*, Lodzj. – No. 8. – 2-13. – Pp. 22-24 (In English).
15. Lytvyn V. (2011), The similarity metric of scientific papers summaries on the basis of adaptive ontologies , *Proceedings of VIIth International Conference on Perspective Technologies and Methods in MEMS Design*, Polyana, Ukraine, pp. 162. (In English)
16. Lytvyn V., Pukach P., Bobyk I., Vysotska V. (2016). The method of formation of the status of personality understanding based on the content analysis, *Eastern-European Journal of Enterprise Technologies*, no5/2(83), 4–12.
17. Lytvyn V., Vysotska V., Veres O., Rishnyak I., and Rishnyak H. (2017). Classification Methods of Text Documents Using Ontology Based Approach, *Advances in Intelligent Systems and Computing* 512, Springer International Publishing AG: 229-240.