Conceptualizing and Formalizing Requirements for Ontology Engineering

Svitlana Moiseyenko [0000-0001-5451-6350] and Vadim Ermolayev [0000-0002-5159-254X]

Department of Computer Science, Zaporizhzhia National University,
Zhukovskogo st. 66, Zaporizhzhia, Ukraine
svitlana.moiseyenko@gmail.com, vadim@ermolayev.com

Abstract. This paper presents the PhD project, by the first author, that develops, in frame of the OntoElect methodology, the methods, techniques, and software tools for conceptualizing and formalizing the requirements for engineering an ontology in an arbitrary domain. It takes in the terms extracted from a representative collection of high-quality textual documents written by the experts in the target domain and therefore describing this domain. It produces the representative set of the requirements by the knowledge stakeholders as ontological fragments conceptualized as UML class diagrams and formalized in OWL+SWRL. The paper presents the vision of the solution and the plan towards building it based on the background knowledge and related work in the fields of Conceptual Modeling and Ontology Engineering. It also outlines the plan for experimental evaluation and validation of the solution.

Keywords: Ontology, Feature Conceptualization, Semantic Relation Extraction, Ontology Concept Identification, Ontology Engineering, OntoElect.

1 Introduction

This paper presents the Ph.D. project that further develops, in frame of the OntoElect methodology [1,2], the methods, techniques, and software tools for conceptualizing and formalizing the requirements for developing an ontology in an arbitrary domain. It takes in the terms extracted from a representative collection of high-quality textual documents written by the experts in the target domain and therefore describing this domain. The terms are extracted by the prior phase of OntoElect in a way to ensure that these terms indicate the significant features corresponding to the prevailing sentiment of the knowledge stakeholders about this domain. The developed phase of OntoElect outputs the representative set of the requirements by the knowledge stakeholders as ontological fragments conceptualized as UML[1] class diagrams and formalized in OWL 2[2] (and additionally in SWRL[3] if there is a need to use rules in the on-

[1] UML 2.0: http://www.omg.org/spec/UML/2.0/About-UML/
[2] OWL 2: https://www.w3.org/TR/owl2-overview/
[3] SWRL: https://www.w3.org/Submission/SWRL/

tology). OntoElect is the basic methodological and theoretical framework for this project.

Conceptualization and Formalization phase builds and refines the high-level contexts (instances) during ontology development process.

In general, knowledge stakeholders in a domain, like for example, Tourism or Finance, are clearly not knowledge engineers. Therefore, it is naive to request that they provide their requirements using an ontology representation language, like OWL. Moreover, it is naive to expect that they will readily provide their requirements as it is not their business. Therefore, OntoElect solicits their requirements indirectly, relying on the existence of a high-quality and representative collection of documents describing the domain. The documents come as texts in a natural language. Hence, the input and any other supplementary data for conceptualization are natural language text fragments. From the other hand, the output has to come in a formal ontology representation language in order to enforce single interpretation and be machine processable. This is why the task of conceptualizing and formalizing requirements for ontology development is challenging. This challenge includes several complex problems that will be attacked in this PhD project.

The reminder of the paper is structured as follows. Section 2 Outlines these problems and offers a high-level vision of the overall solution. Section 3 looks at the related work in the fields of Conceptual Modeling and Ontology Engineering to seek insights, relevant background knowledge, and the bits of already existing technologies that may help solve the challenge of the project. Section 4 explains how the developed solutions will be evaluated experimentally. Finally, Section 5 concludes the paper.

2 The Vision of the Solution and the Problems to Solve

There is a problem with meaning interpretation for the different terms and contexts which appears between people in different communities. For example, a particular term may have a more specific interpretation by a domain expert which may well differ from the interpretation by an ontology engineer. Regarding engineering, it can be called as a lack of specification when different engineers have different ideas about the particular requirement and its context. Hence, conceptualization and formalization phase is designed to solve this problem. To overcome this mismatch in interpretations, the project bases itself on the approach of stepwise elimination of different aspects that cause mismatches. This approach has been framed out as a part of OntoElect methodology [2] and is outlined in Fig. 1. However, OntoElect relies on manual performance of all these steps. The technical objective of this project is to develop the techniques and software tools to partially automate the process and substantially lower manual effort.

The outlined sequence of steps takes in the ranked list of the terms describing the domain. These terms are guaranteed to be significant and saturated by the Feature Elicitation phase of OntoElect. Therefore, the input is regarded as the list of the required features. The objective is to transform this list of the required features to formalized ontological fragments (requirements), carrying the positive and negative votes of the involved features in their aggregated significance scores. Requirements are further used in Ontology Evaluation phase to compute the fitness of the ontology.

Conceptualization and formalization is done by a knowledge engineer, using the suite of software instruments that will be developed in this project, through: (i) grouping and categorizing extracted required features; (ii) selecting the significant concepts from the list of required features and forming the feature taxonomy; (iii) computing the propagated scores up the concept/property hierarchies; (iv) selecting the most significant concept features; (v) elaborating natural language definitions for the most significant concept features and formalizing these as ontological fragments using UML and OWL; and (vi) documenting requirements.

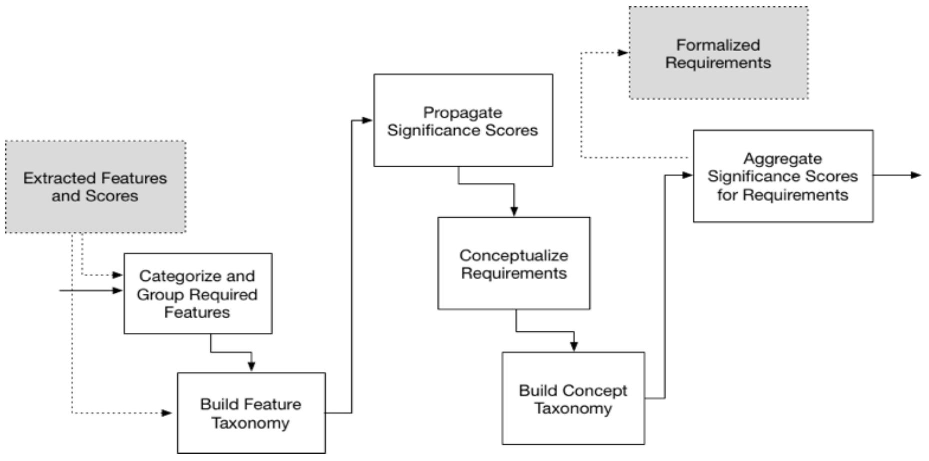

Formalized Requirements
Propagate Significance Scores
Extracted Features and Scores
Aggregate Significance Scores for Requirements
Conceptualize Requirements
Categorize and Group Required Features
Build Feature Taxonomy
Build Concept Taxonomy

Fig. 1. Conceptualization and formalization workflow

Feature grouping is merging several features which are lexically different but carry equivalent semantics. The relevant cases include: plural and singular forms of the same term, for example “temporal constraints” and “temporal constraint” are the same terms and have to be merged; the terms that had or had not lost two-letter combinations due to peculiarities of their representation in PDF documents due to the differences in Adobe versions, for example “de nition” and “definition” are also the same terms. The significance scores of the merged terms are summarized. One possible solution to group features semi-automatically using a proper string similarity measure, as suggested in [3]. The problem could also be attacked by applying an appropriate clustering algorithm (a.k.a. conceptual clustering [4]) based on computing the minimal threshold using string similarity (syntactic) measures to evaluate if a feature belongs to a group.

Analyzing the individuals (see also feature categorization below), for example taken from a relevant linked open data repository or the individual features available from the required feature list, may help receive more confidence in the relevance of a property to a concept.

Looking at feature groups may reveal important information about their subsumption or meronymy (and, possibly, other interesting properties). Indeed, the most abstract feature in a group may be interpreted as the root in the group hierarchy. Consequently, the features which are extended from the root feature by adding words most probably subsume to the root. Meronymy hierarchies involve the features which are either parts of a whole or the wholes for their parts. Putting together all these group hierarchies will result in a **feature taxonomy**, which is the output of the Feature Grouping step.

Building the feature taxonomy for the set of required features is important as both types of hierarchical relationships among features influence the significance of features through inheritance. Indeed, if a feature subsumes to another feature then it inherits some of its properties, so its significance is formed to a particular extent by these inherited properties. Hence, a parent in a hierarchy may expect that it is rewarded by its children through the propagation of their significance scores. OntoElect suggests [1] that **score propagation** adds one fifth of the children' scores to their parent's score. An example of computing propagated scores is pictured in Fig. 2.

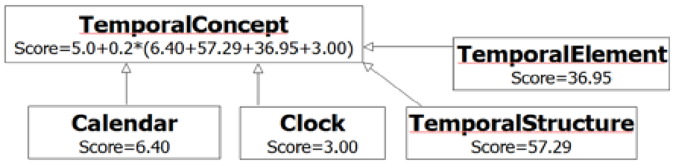


Fig. 2. An example of computing score propagation for required features, adopted from [2]

A step which is somewhat orthogonal to grouping, as it looks into the semantic nature of a feature is categorization. Feature categorization stands for deciding if a feature, due to its semantics, represents a concept, a property, or an individual. **Property features** are further grouped in relevant ontological fragments to represent formalized requirements. **Individual features** – by being the instances of a concept or a property – form the corpus of evidence pointing that the concept possesses the property at the schema level. **Concept features** are further used to form subsumption or meronymy hierarchies in the **concept taxonomy** and form the "anchors" for ontological fragments.

The most significant concept features (due to their scores in the feature taxonomy), having the potential for high impact on the requirements, may be selected. For that, concept features are viewed in a ranked list and the group of features covering the desired proportion of importance is promoted.

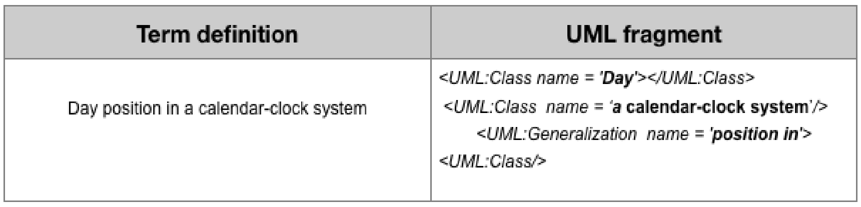
The promoted concept features are used to form the **concept taxonomy** and be the central concepts for the formalized requirements. Each of these promoted concept features is, at the end of the workflow, conceptualized in a **formalized ontological fragment** – as a conceptual model (in UML) and a piece of code in an ontology specification language (in OWL+SWRL). Conceptualization means that all the relevant property features and features representing individuals are consolidated in the ontological fragment in a harmonized way to form a coherent piece of a required descriptive theory for the domain.

Conceptualizing a concept feature starts with elaborating its **natural language definition** based on the high-impact documents describing the domain. These documents may be acquired from the document collection from which the required features have been extracted. To ensure the relevance and high impact of these sources, a snowball sampling in citation networks approach (e.g. [5]) may be tried. The task of a knowledge engineer, supported by a software tool, at this step is to ensure that all the **required property features** are **taken into** this definition and **do not contradict** each other.

Based on the natural language definition, a conceptual model is developed for a concept feature, including also its properties and relationships to the other relevant concept features.

Currently, OntoElect does not recommend any instrumental software tool to help transform the definition of a concept written using a natural (e.g. English) language to a UML model. Current working practice suggests that it is a two-step process. The first step is elaborating the model manually, using the ArgoUML editor[4]. Protégé ontology editor[5] is further used at the second step for manual coding the ontology in OWL 2 with an account for DL restrictions [6]. The transformation patterns from UML to OWL follow the recommendations by Schreiber[6]. There are several possible ways to partially automate this process and hence lower the effort for these operations.

One potentially interesting idea is inspired by the works in automatic program generation. The instructions in a programming language are generated from natural language sentences using different techniques. One of the promising approaches is to employ machine learning approach. To do so, it is required that a set of typical sentences and resulting code instructions is provided to train the model. When done, the trained model is applied to incoming sentences and outputs code instructions. A similar approach may be used for descriptive sentences from one side and UML model outputs from the other side – please see an example in Fig. 3. An appropriate starting point for the training set could be the library of ontology design patterns [7, 8].



Term definition	UML fragment
Day position in a calendar-clock system	*<UML:Class name = '**Day**'></UML:Class>*
	*<UML:Class name = '**a calendar-clock system**'/>*
	*<UML:Generalization name = '**position in**'>*
	<UML:Class/>

Fig. 3. An example of transforming a term to a UML model fragment

An alternative way to transform a text to a UML model is to use NLP technology stack. For example, Stanford Core NLP [7] allows analyzing a text by applying linguis-

[4] ArgoUML is an open source UML modeling tool: http://argouml.tigris.org/

[5] Protégé Ontology Editor: https://protege.stanford.edu/

[6] OWL Restrictions: http://www.cs.vu.nl/~guus/public/owl-restrictions/

[7] Stanford CoreNLP: https://stanfordnlp.github.io/CoreNLP/

tic and syntactic analysis tools. It facilitates extracting dependency structures from phrases or sentences, determining the part of speech of the words, indicating which noun phrases refer/relate to what, etc.

The transformation of a UML model of an ontological fragment to OWL + SWRL could also be automated. The approach could be using a rule-based technique, or a machine-learning approach described above.

OntoElect is more specific in recommending a way for documenting the ontology under development. It suggests that the ontology is documented in a set of Semantic MediaWiki[8] pages. Some of those pages provide the overviews of the ontology modules, but the rest, which are the majority, are dedicated to documenting the concepts – one page per concept. A documentation wiki page of a particular concept contains: the natural language definition of the concept; the UML class diagram of the concept's conceptual model; the description of the properties grouped according to the property types: datatype and object properties. This sort of documentation, for requirements, would be straightforwardly generated based on the results of conceptualization and formalization.

3 Some Insightful Related Work

Ontology engineering is a broad field which a substantially big research community devoting their effort to push forward the State-of-the-Art. Therefore, it is not tractable to overview all the achievements in ontology engineering. In this paper we focus, in our review of the related work, only on those results that gave us insights in developing our vision and circumscribing the problems to be solved, as presented in Section 2. Therefore, our concise related work review is grouped below along the problems that need to be solved.

Feature Grouping and Taxonomy Generation. Feature grouping has been studied in the Knowledge Acquisition and Information retrieval fields. There were plenty approaches developed by applying syntactic, semantic, and linguistics analysis (e.g. Jaccard, Jaro, Euclidean algorithms) be revealing the degree of the measure similarity. Using clustering algorithms like K-means, Agglomerative Hierarchical, or EM and methods such as pattern-based extraction, conceptual clustering, concept learning, ontology learning from instances [9], Aussenac-Gilles method [10]) significantly enhanced the process of assigning terms into groups for discovering concepts or constructing hierarchy.

Feature Categorization. A lot of existing approaches are based on the predefined tokenization or part-of-speech tagging (POS) patterns for identifying relatedness which developed by NLP techniques. The most well-known comprehensive toolkits for natural language processing are General Architecture for Text Engineering (GATE), Natural Language Toolkit (NLTK), and Stanford Core NLP.

Conceptualization. Several methodologies contributed the techniques for conceptualization in ontology engineering. The Klagenfurt Conceptual Pre-Design Model

[8] Semantic MediaWiki: http://semantic-mediawiki.org/

(KCPM) an intermediate phase between requirements analysis and conceptual design. The proposed approach was targeted on harmonization the developer's and user's view. The relevant part of this work for the presented project is in particular in the modeling notions: thing and connection types, a perspective view and constraint [11]. METHONTOLOGY is an ontology engineering methodology that enables constructing ontologies at the knowledge level. This methodology includes the identification of the ontology development process, a life cycle based on evolving prototypes, and techniques to carry out each activity in the management, development-oriented, and support activities [12]. Special attention is paid to the ontology construction. The methodology also provides a detailed description of how to organize and structure the conceptual models in order to build taxonomies. OntoElect [2] offers a methodological framework for all the necessary steps for the presented work. It also outlines the approaches and points to some techniques relevant to develop the software solutions for the required tools, in particular for knowledge extraction, grouping and categorization, building concepts and taxonomies. The work on ontology design patterns, such as Logical and Content Ontology Design Patterns aimed at solving design problems for domain concepts and properties [7], is also relevant for the part of building ontological fragments in our project.

Significance Scores Propagation and Aggregation. The topic about the computing significance scores is very specific in the ontology development process. Hence, to the best of our knowledge, there is no published work that uses feature significance scores for requirements (or ontology) conceptualization.

Assembling the Natural Language Definition for a Concept Feature. Lexico-Syntactic ontology design pattern (OPs) was developed based on the linguistic structures or schemas in order to extract some conclusions about the meaning of the words they express [7].

Text to UML model Transformation. Several publications deal with the transformation of text to UML. Most of them use XML serializations and XSLT [13], [14]. Cranefield and Purvis investigated the use of UML class diagrams in order to represent ontologies and UML object diagrams for representing the knowledge instances [15]. An interesting approach based on using machine learning techniques was presented in [16] to transform natural text sentences to programming language instructions. It would be a good point to use a similar technique in transforming stakeholder requirements to the formalized UML models.

UML model to OWL+SWRL Code Transformation. Transformation, Reengineering, Schema reengineering, and Transformation of Logical Patterns techniques were developed for relevant transformations, for example a non-OWL DL or informal concept models to OWL DL ontology fragments. The conceptualization activity in METHONTOLOGY provide a full-stack guide for converting informally presented concepts and relations into semi-formal specifications. Several tools have also been developed for the purpose of this or similar transformation, such as LEXTER [17], Géditerm [18], TERMINAE [19].

Ontology engineering methods[9] were developed not only for the creating ontology from scratch but also to be able to reuse the already existed ontologies. Here are several purposes that they are used for: (i) building ontology from scratch, (ii) upgrading an existing ontology, (iii) acquiring knowledge for particular tasks, (iv) solving particular problems during ontology development process. The most well-known ontology engineering methods are alignment, merging, evaluating (e.g., Cyc, Uschold and King's, re-engineering (METHONTOLOGY), etc.) which allow ontology engineers to build and edit ontologies using the combination of frames, description logic, first logic order and other different approaches.

Generating Semantic Media Wiki Pages for Documentation. Wiki-based process editor was developed to enable the ontology documenting process. This approach is based on combining graphical process modeling techniques, wiki-based lightweight knowledge capturing approach, and a background semantic knowledge base [20].

4 Planned Evaluation

The envisioned approach, together with the instrumental software tools, for conceptualizing and formalizing requirements in ontology engineering needs to be experimentally evaluated and validated. A straightforward way to evaluate the (correctness) of the solution is to compare its results to a Golden Standard. Consequently, a way to validate the solution is to offer it to knowledge engineers for a trial. Further, their impression of the usability and performance of the solution is compared to their normal mode of work – without the solution.

As a Golden Standard for evaluation, it is planned to use the available working repository of the Syndicated Ontology of Time (SOT). SOT is developed using OntoElect as the ontology engineering methodology. Therefore, this repository contains all the types of intermediate and final results for the key concepts of the currently developed ontology for the Time Representation and Reasoning domain. The repository belongs to our group and therefore is fully available as background knowledge. In evaluation, it is planned to compare the outputs of the developed tools to the same outputs developed by human knowledge engineers.

In validation experiments, the knowledge engineers who developed SOT manually will do the same development for the selected key concept features [2] using the developed instrumental tool suite. Their effort spent in this activity and their subjective assessments of the usability and usefulness of the tools will be collected by offering a questionnaire.

After the evaluation and validation of the solution in the SOT use case, another use case in a different domain will be elaborated. One of the potential candidate domains is Knowledge Management. An industrial use case for this domain is framed out

[9] This description has been deliberately kept concise because of the page limit. Definitely, more relevant ontology engineering methods exist, including ontology learning, and will be reviewed in the planned survey paper.

based on the full text document collection of 15 journals provided by Springer Nature. For a part of this domain the work on extracting features is in progress [21].

5 Some Conclusions

The PhD project presented in this paper is in its initial phase. So, the only preliminary result is the vision of the solution that will help achieve the goal of the project: help domain knowledge stakeholders and knowledge engineers be coherent in interpreting the requirements for representing knowledge to describe the domain formally, in an ontology.

While elaborating our vision of the research problem and possible solution to it, we also presented a workflow that will help solve the challenge in several interrelated steps. These steps are, in their turn, the problems that require solutions. These solutions are planned to be sought in the project.

Our initial plan for evaluating and validating the solution is also presented in the paper. We hope that the outcome of the project will be beneficial for the Ontology Engineering community at broad and also for the experts who carry their knowledge about various domains of interest.

References

1. Tatarintseva, O., Ermolayev, V., Keller, B., Matzke, W.-E.: Quantifying ontology fitness in OntoElect using saturation- and vote-based metrics. In: Ermolayev, V., et al. (eds.) Revised Selected Papers of ICTERI 2013, CCIS, vol. 412, pp. 136–162 (2013)
2. Ermolayev, V.: OntoElecting requirements for domain ontologies: the case of time domain. EMISA Int J of Conceptual Modeling, 13(Sp.I.), 86–109 (2018)
3. Chugunenko, A., Kosa, V., Popov, R., Chaves-Fraga, D., Ermolayev, V.: Refining terminological saturation using string similarity measures. In: Ermolayev, V. et al. (eds.) Proc. ICTERI 2018, CEUR-WS, online (2018) – to appear
4. Michalsky, R.,: Knowledge acquisition through conceptual clustering: a theoretical framework and algorithm for partitioning data into conjunctive concepts. International Journal of Policy Analysis and Information Systems 4(3), 219–243 (1980)
5. Dobrovolskyi, H., Keberle, N., Todoriko, O.: Probabilistic topic modelling for controlled snowball sampling in citation network collection. In: Różewski P., Lange, C. (eds.) KESW 2017. CCIS, vol. 786. Springer, Cham (2017)
6. Motik, B., Patel-Schneider, P. F., Parisa, B. (eds.): OWL 2 Web ontology language. Structural specification and functional-style syntax. 2nd edn. W3C recommendation, W3C (2012) http://www.w3.org/TR/2012/REC-owl2- syntax-20121211/
7. Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. International Handbooks on Information Systems. Springer, Berlin, Heidelberg (2009)
8. Poveda-Villalón, M., Suárez-Figueroa, M.C., Gómez-Pérez, A.: Validating ontologies with OOPS!. In: ten Teije, A. et al. (eds.) Proc. EKAW 2012. LNCS, vol. 7603. Springer, Berlin, Heidelberg (2012)

9. Morik, K., Kietz, J.U.: A bootstrapping approach to conceptual clustering. In: Segre, A.M. (ed.) Proceedings of the Sixth International Workshop on Machine Learning, Ithaca, New York. Morgan Kaufmann Publishers, San Francisco, California, pp. 503–504 (1989)
10. Kietz, J.U., Maedche, A., Volz, R.: A method for semi-automatic ontology acquisition from a corporate intranet. In: Aussenac-Gilles, N., Biébow, B., Szulman, S. (eds.) EKAW'00 Workshop on Ontologies and Texts. Juan-Les-Pins, France. CEUR Workshop Proceedings 51:4.1–4.14. Amsterdam, The Netherlands (2000) http://CEUR-WS.org/Vol-51/
11. Kop, C., Mayr, H.C.: Conceptual predesign bridging the gap between requirements and conceptual design. In: Proceedings of IEEE International Symposium on Requirements Engineering: RE '98, pp. 90–98. IEEE Comput. Soc (1998)
12. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological engineering. 1st ed., Springer-Verlag, London (2003)
13. Cranefield, S.: Networked knowledge representation and exchange using UML and RDF. Journal of Digital information 1(8), (2001)
14. Djuri ́c, D.: MDA-based ontology infrastructure. Computer Science and Information Systems 1(1), 91–116 (2004).
15. Cranefield, S., Purvis, M.: UML as an ontology modeling language. In: Proc. of the Workshop on Intelligent Information Integration, 16th Int. Joint Conference on AI (IJCAI-99), Stockholm (1999)
16. Stehnii, A., Hryniv, R.: Generation of code from text description with syntactic parsing and Tree2Tree model. Master thesis, Ukrainian Catholic University (2017)
17. Bourigault, D., González, I., Gros, C.: LEXTER, a natural language tool for terminology extraction. In: Gellerstam, M., Järborg, J., Malmgren, S.G., Norén, K., Rogström, L., Papmehl, C.R. (eds.) 7th EURALEX International Congress, Part II, pp. 771–779. Goteborg, Sweden (1996)
18. Aussenac-Gilles, N.: Gediterm, un logiciel de gestion de bases de connaissances terminologiques. Terminologies Nouvelles 19:111–123 (1999)
19. Aussenac-Gilles, N., Biébow, B., Szulman, S.: Modelling the travelling domain from a NLP description with TERMINAE. In: Angele, J., Sure, Y. (eds.) EKAW'02 Workshop on Evaluation of Ontology-based Tools (EON2002), Sigüenza, Spain. CEUR Workshop Proceedings 62:112–128. Amsterdam, The Netherlands (2002) http://CEUR-WS.org/Vol-62/
20. Dengler, F., Vrandečič, D., Simperl, E.: Comparison of wiki-based process modeling systems. In: Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies. i-KNOW'11, pp. 30:1–30:4. ACM, New York, NY, USA (2011)
21. Kosa, V., Chaves-fraga, D., Naumenko, D., Yuschenko, E., Badenes-Olmedo, C., Ermolayev, V, Birukou, A.: Cross-evaluation of auomated term extraction tools by measuring terminological saturation. In: Bassiliades, N., et al. (eds.) ICTERI 2017, Revised Selected Papers,.CCIS, vol. 826, pp. 135–163, Springer, Cham (2018)