NIPES

Journal of Science and Technology Research

Journal homepage: www.nipesjournals.org.ng

Software Process Ontology Evaluation Using Ontoclean

Oveh, R.O[1] and Egbokhare, F.A.[2]

[1]Department of Mathematics and Computer Science, Western Delta University, Oghara, Delta State, Nigeria.
e-mail: omo_rich@yahoo.com

[2]Department of Computer Science, University Of Benin, Benin City, Edo State, Nigeria
e-mail: fegbokhare@uniben.edu

Article Info

***Keywords*:**
Software Process, Software Process Ontology, Ontology, Knowledge, Formal Representation, Ontoclean

Received 10 January 2020
Revised 30 January 2020
Accepted 01 February 2020
Available online 02 March 2020

ISSN-2682-5821/

Abstract

Software process is a knowledge driven process with sub-processes. Harvesting and reuse of this knowledge is key to success in software organisations. An improved use of this knowledge could lead to maximum payoff in software organisations. The purpose of formal representation is to help organisations achieve success by modelling successful organisations. Formal representations must be first evaluated to determine its quality before it can be fit for reuse. In this paper Ontoclean was used to evaluate software process knowledge ontology.

1. Introduction

In Software process is a knowledge driven and knowledge intensive process that involves several other sub-processes. Software process can be defined as the set of related activities that are used in developing software. Knowledge in Software Engineering (SE) is diverse and organizations have problems capturing, retrieving, and reusing it [1]. An improved use of this knowledge is the basic motivation and driver for Knowledge Management (KM) in SE [2]. Harvesting, representing and reusing knowledge within a domain leads to maximum payoff, which is desirable in most organisations [3]. Knowledge Management (KM) is defined as an effort to capture critical knowledge and share it within an organization [4, 5]. It capitalizes on the collective organizational memory to improve decision making, enhance productivity, and promote innovation [6, 7]. It is also the process of transforming information and intellectual assets into persisting value. KM connects people with the knowledge that they need to take action, when they need it [8]. Knowledge management involves the identification and analysis of available and required know [9] and helps an organization to gain insight and understanding from its own experience. Specific knowledge management activities focus on acquiring, storing and utilizing knowledge for problem solving, dynamic leaning, strategic planning and decision making. This prevents intellectual assets from decay, adds to a firm's intelligence and provides increased flexibility [10].

SE comprises several interrelated subdomains such as Requirements, Design, Coding, Testing, Project Management, and Configuration Management. There are several software process models which describe the sequence of activities carried out in developing software. These software process models are a standard way of planning and organizing a software process. The major phases are requirement gathering, design and coding, implementation and maintenance.

It has been identified that there are few works in literature that aim at developing ontologies covering wide portions of the SE domain, such as [11, 12, 13]. A lot of SE domain ontologies model SE subdomains [14, 15, 16, 17, 18]. [19], described these subdomain ontologies as weak or not interrelated, and are often applied in isolation. Thus, he made an attempt to provide an integrated solution for better dealing with KM-related problems in SE by means of a Software Engineering Ontology Network (SEON). It was designed with mechanisms for easing the development and integration of SE domain ontologies, covering the main technical software engineering subdomains (i.e requirements, design, coding and testing). However, he only represented a small portion of software engineering ontology. [5] identified that the combination of ontologies of all SE subdomains would result in an ontology of the complete SE domain. He further stated that the reality is that this goal is extremely laborious, not only due to its size, but also due to the numerous problems related to ontology integration and merging, such as overlapping concepts, diverse foundational theories, and different representation and description levels, among others. He concluded that despite the challenges involved, an ontological representation covering a large extension of the SE domain remains a desired solution. Using Ontoclean, this paper evaluates software process knowledge ontology.

Ontologies have been widely recognized as a key enabling technology for KM. They are used for establishing a common conceptualization of the domain of interest to support knowledge representation, integration, storage, search and communication [17]. A domain ontology identifies the key concepts, objects and entities that exist in some knowledge domain or area of interest and the relationships between them [20, 21]. Ontologies play a significant role for knowledge sharing and as knowledge models in instructional science, technology-enhanced learning, knowledge management and training [20, 21, 22]. Ontologies consist of instances, properties and classes, where instances represent specific project data, properties represent binary relations held among software engineering concepts/instances, and classes represent the software engineering concepts interpreted as sets that contain specific project data [25].

[14] did an extensive review of SE ontologies, where he classified them into generic and specific ontology. Generic SE Ontologies, have the ambitious goal of modelling the complete SE body of knowledge; while Specific SE Ontologies, attempting to conceptualize only part (a subdomain) of this discipline. [23] constructed a software process ontology, which aims to establish a common vocabulary for software organisations to talk about software processes. A mapping between the concepts presented in the ontology and the concepts of some of these standards was also done in order to help software organisations to use those standards in their software process improvement efforts. [24] proposed a knowledge base called DKDOnto, a domain-specific ontology for distributed development. its aim was to help projects with a common vocabulary. Allowing them to assist better the distributed software development process. [25] presented software engineering ontology as software engineering knowledge representation for a multi-site software development. It did not only facilitate the capturing of software engineering knowledge but also enhanced the sharing of software engineering knowledge across geographically multiple software development sites. [26] developed an ontology-based software process assessment tool to support data collection phase of process assessment and to track conformance of software processes to CMMI as the process reference model. [27] produced domain specific knowledge base ontology for core software process subdomain. However, the ontology was not evaluated to check for its efficiency and possible reuse which is key.

The management of knowledge and experience are key means by which systematic software development and process improvement occur. Within the domain of Software Engineering (SE), quality continues to remain an issue of concern. Knowledge Management (KM) gives organizations

the opportunity to appreciate the challenges and complexities inherent in software development [28].

Successful organisations continuously improve their processes. Like organisational standard process definition, systematic process improvement is more effective and efficient if it is done guided by process quality models and standards. The purpose of most standards is to help software organisations achieve excellence by following the processes and activities adopted by the most successful organisations [23].

2. Methodology

[3, 27] used semi structured interviews, socialization and focus group method to explore the views, experiences, beliefs and motivations of Software Process domain experts. Four (4) different software organisations were used for the research. The organisations were selected because of their successes in their past and present software projects. Discussions in the form of key informant interviews were held with four (4) project managers and twelve (12) developers on the experiences and lessons learnt from past projects. Key activities that resulted in project success during the process of software development were elicited. Focus group discussion was used to capture knowledge on the specific activities carried out during software development from the key stakeholders. The interviewees did not grant permission to record the interviews electronically, so the responses were recorded on paper. Each interview session lasted for about one hour and a total of five (5) interviews were conducted over a two-week period. Follow up questions were asked via telephone conversations. The data obtained from the interviews were documented and later transcribed and meaningful knowledge for software development process was extracted using content analysis. The resulting ontology constructed from the data is shown in Figure 1. This research used Ontoclean to evaluate the software process ontology knowledge that was harvested and represented (formally and informally) in [3, 27].

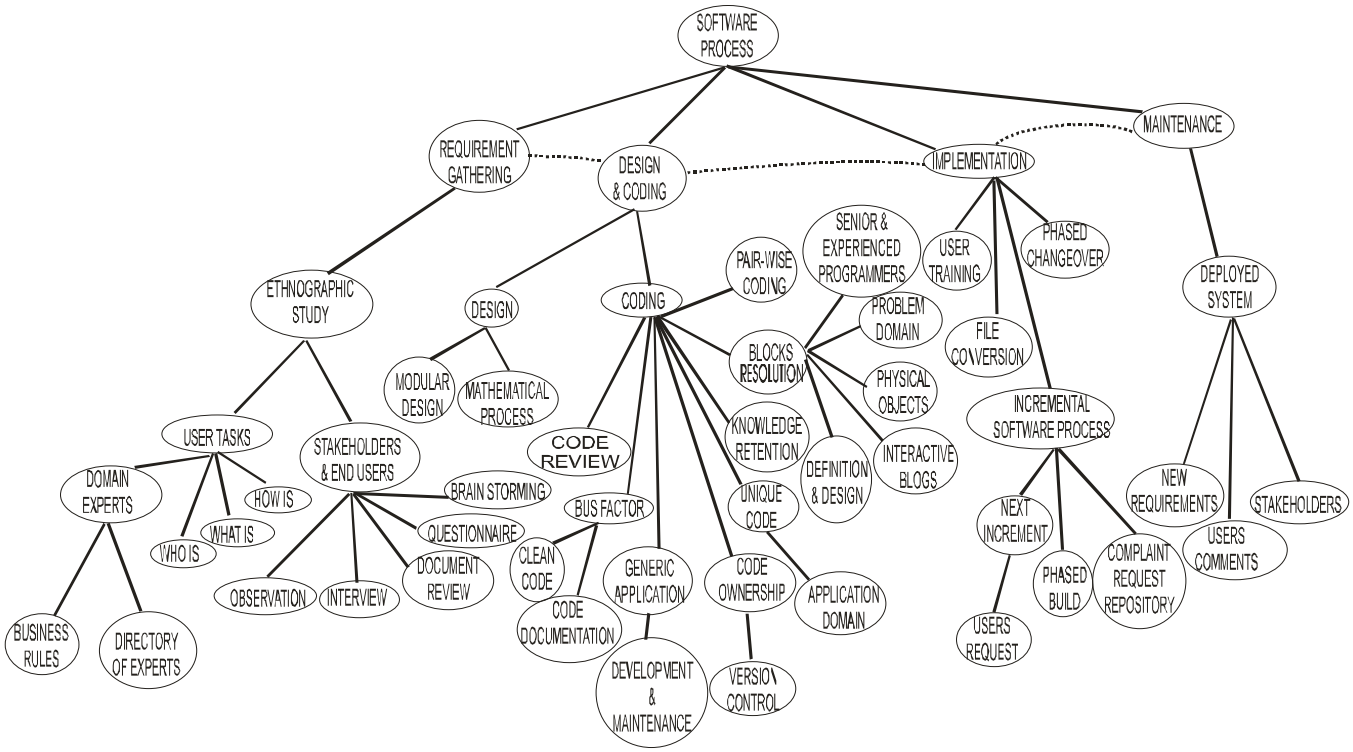

Figure 1: Software Process Ontology [3, 27]

Ontology evaluation is a prerequisite for ontology reuse. Ontoclean was used to validate the software process ontology harvested. The OntoClean methodology is based on formal notions, which are general enough to be used in any ontology effort, independently of a particular domain. We use these notions to define a set of metaproperties which, in turn, are used to characterize relevant aspects of the intended meaning of the properties, classes, and relations that make up an ontology. In addition, the metaproperties impose several constraints on the taxonomic structure of an ontology, which help in evaluating the choices made [29]. Onclean was used to validate the ontology for correctness. Ontoclean provides a logical basis for formally analysing ontologies using formal and domain independent properties called metaproperties of classes. Ontoclean is not an ontology and thus is not concerned with the semantics of the relationships among concepts. It is a methodology used to analyse ontologies using formal and domain independent properties called meta properties [29, 30, 31]. The following steps outlined in [30] was applied for the Ontoclean evaluation:

i. Assign metaproperties to the ontology

ii. Evaluate the metaproperties for violation and correct any errors discovered

Ontoclean is majorly based on four (4) metaproperties of rigidity, identity, unity, and dependence as shown in Table 1.

Table 2: OntoClean Meta Properties [30]

Meta Property	Symbol	Label	Definition
Rigidity	+R	Rigid	All instances will always be instances of this concept in every possible world
	-R	Non-Rigid	There are instances that will stop being instances of the concept
	~R	Anti-Rigid	All instances will no longer be instances of that concept
Identity	+I	Carry Identity	Instances carry a unique identification (IC)criteria from superclass
	-I	Non Carry Identity	There is no identification criteria (IC)
	+O	Supply identity	Instances themselves provide a unique identification criteria (IC)
Unity	+U	Unity	Instances are “whole”, and have a single unit criteria(UC)
	-U	Non-Unity	Instances are “whole”, but they do not have a single unit criteria
	~U	Anti-Unity	Instances are not “whole”
Dependence	+D	External dependence	There is dependence on external concept
	-D	Non External dependence	There is no dependence

Ontoclean have five (5) defined restrictions [29], which are:

1. Anti-rigid class cannot subsume a rigid subclass;
2. A class with identity cannot subsume a non-identity subclass;
3. A class with the unity meta property cannot subsume a subclass without unity criterion;
4. Anti-Unit class cannot subsume unity class;
5. Dependent class cannot subsume non-dependent class

3. Results and discussion

Using the metaproperties in Table 2, we obtained Figure 2 with the assigned metaproperties for the software process ontology.

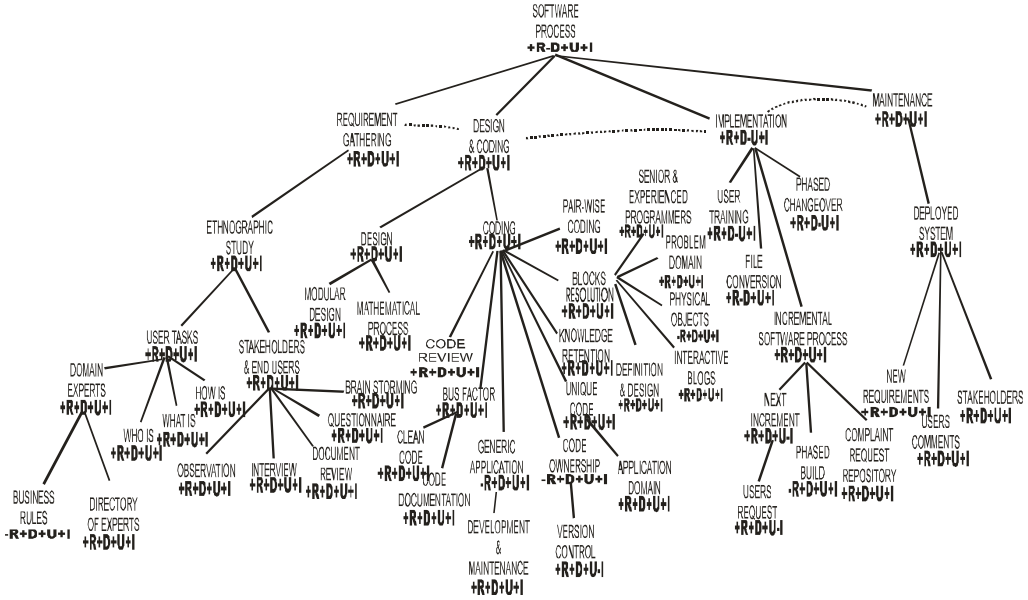

Figure 2: Metaproperties of Software Process Ontology

The metaproperties in Figure 2 was checked against the restrictions of Ontoclean to identify any violation. A violation was identified and cleaned as shown in Figure 3. The cleaned software process ontology in Figure 3 can be said to be free from any violation of the restrictions hence suitable for reuse in software process as posited in [3, 27].

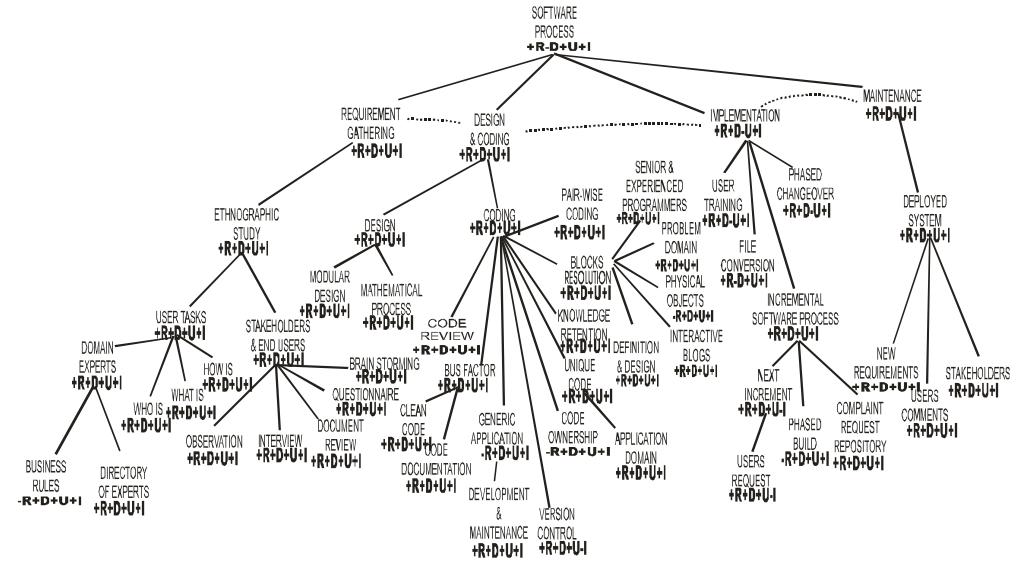

Figure 3: Cleaned Software Process Ontology

The assignment of metaproperties to Figure 2 was done based on the individual properties of each class for example, software process was assigned with the property: rigidity, non-dependence, unity, and identity. Rigidity was assigned because every instance of a software process is a software process. Non-dependence was assigned because a software process does not depend on external concept. Unity was assigned because an instance of a software process cannot be fragmented. Identity was assigned because every instance of a software process has a unique identity criteria. Also, business rules were assigned: non-rigidity, dependence, unity and identity. It was assigned non-rigidity because business rules can change, dependence because every instance of business rule depends on external concept, unity because every instance of business rule cannot be fragmented, identity because every instance of a business rule has unique identity criteria. Metaproperites was assigned to all the concepts in Figure 2 based on their properties in software process domain. Figure 3 was checked for any violation of the five (5) restrictions in Ontoclean. A violation was identified and corrected. The violation was with regard to identity metaproperty in version control (between code ownership and version control) subsuming a non-identity subclass. It was corrected by placing version control as a subclass of coding. The resultant software process ontology produced in Figure 3 is free from violation and said to have a good cohesion in terms of structure as posited in [3, 27].

4. Conclusion

Software process knowledge is a knowledge driven process with sub-processes. This knowledge is latent and could be lost if not formally harvested and documented. An improved use of this knowledge could lead to maximum payoff in software organisations. This is the heart of knowledge management, which focuses on knowledge capturing and sharing. This paper used Ontoclean to evaluate Software Process Knowledge Ontology. The result showed that the ontology was built correctly and it is suitable for reuse in knowledge management of software process.

References

[1] Aurum, A., Jeffery, R., Wohlin, C., Handzic, M. (2003). Managing Software Engineering Knowledge. Springer-Verlag Berlin Heidelberg
[2] Rus, I. and Lindvall M. (2002) Knowledge Management in Software Engineering, IEEE Software, 19(3) 26-38.
[3] Oveh R.O. and Egbokhare F.A. (2019) Harvesting and Informal Representation of Software Process Domain Knowledge. Intelligent Computing Conference, 2, 936–947. Springer Nature Switzerland
[4] Davenport T.H. and Prusak L. (1998) Working Knowledge – How Organizations Manage What They Know. Harvard Business School Press, Boston, Massachusetts.
[5] Alavi M. and Leidner D. E. (2001). Knowledge management and knowledge management systems: Conceptual foundations and research issues. MIS Quarterly, 25(1), 107-136.
[6] Taluja, R.K. , Tewari C.K. and Kaur A. (2010). Concept of Knowledge Management and Its Usage in Higher Learning Institutions. VSRD-TNTJ. I (4), 255-265
[7] Perez E. (1999) Knowledge Management in the Library—Not. Database Magazine 22(2), 75–78
[8] Kidwell, K.M., Vander L and Johnson S.L. (2000). Applying corporate knowledge management practices in higher education, Journal of Educause Quarterly 4, 28-33.
[9] Firestone (2001) Key Issues in Knowledge Management, Knowledge and Innovation. Journal of the KMCI; 1(3), 8-38.
[10] Abdul-Kalam A.P.J. (2004) Digital Library and its multidimensions. President of India's speech at the "Inauguration of International Conference on Digital Libraries (ICDL) retrieved 16/9/18 from: http://www.presidentofindia.nic.in/scripts/sllatest1.jsp?id=282
[11] Mendes, O. and Abran A. (2005) Issues in the Development of an Ontology for an Emerging Engineering Discipline. First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Eng. (ONTOSE). Alcalá Henares, Spain
[12] Sicilia, M.A., Cuadrado, J.J., García, E., Rodríguez, D. and Hilera J.R. (2005) The Evaluation of Ontological Representation of the SWEBOK as a Revision Tool. In: 29th Int. Computer Software and Application Conference (COMPSAC), 26-28. Edinburgh, UK.
[13] Wongthongtham, P.,Chang, E., Dillon, T. and Sommerville I. (2009) Development of a Software Engineering Ontology for Multisite Software Development. IEEE Transactions on Knowledge and Data Engineering, 21 (8) 1205-1217
[14] Calero, C., Ruiz, F. and Piattini M. (2006) Ontologies for Software Engineering and Soft-ware Technology. Springer Science & Business Media.

[15] Souza, E.F., Falbo, R.A. and Vijaykumar, N.L. (2013) Using Ontology Patterns for Building a Reference Software Testing Ontology. In: 17th IEEE Int. Enterprise Distributed Object Computing Conference Workshops (EDOCW), 21-30. Vancouver
[16] González-Pérez, C., and Henderson-Sellers B. (2006) An Ontology for Software Development Methodologies and Endeavours. In: Calero C., Ruiz F., Piattini M. (eds) Ontologies for Software Engineering and Software Technology. Springer, Berlin, Heidelberg
[17] Bringuente, A.C., Falbo, R.A. and Guizzardi G. (2011) Using a Foundational Ontology for Reengineering a Software Process Ontology. Journal of Information and Data Management, 2(3) 511.
[18] Calhau, R.F. and Falbo R.A. (2010) An Ontology-based Approach for Semantic Integration. In: 14th IEEE International Enterprise Distributed Object Computing Conference, Vitória, Brazil. Los Alamitos: IEEE Computer Society, 111-120
[19] Borges-Ruy, F., de Almeida Falbo R. , Perini Barcellos, M.,Dornelas Costa S. and Guizzardi G. (2016) SEON: A Software Engineering Ontology Network. In: Blomqvist E., Ciancarini P., Poggi F., Vitali F. (eds) Knowledge Engineering and Knowledge Management. EKAW 2016. Lecture Notes in Computer Science, vol 10024. Springer, Cham
[20] Gruber T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing?. International Journal of Human-Computer Studies, 43, 907–928. http://dx.doi.org/10.1006/ijhc.1995.1081
[21] Zouaq A. and Nkambou R. (2008). Building domain ontologies from text for educational purposes. IEEE Transactions on Learning Technologies, 1, 49–62. http://dx.doi.org/10.1109/TLT.2008.12
[22] Kickmeier-Rust M. D. and Albert D. (2008). The ELEKTRA ontology model: A learner-centered approach to resource description. Advances in Web Based Learning – ICWL 2007. Lecture Notes in Computer Science. 4823, 78–89. Berlin: Springer.
[23] Falbo R. A. and Bertollo G. (2009) A Software Process Ontology as a Common Vocabulary about Software Processes. International Journal of Business Process Integration and Management. 4(4) 239-250
[24] Rocha, R. Araujo, A., Cordeiro, D. Ximenes, A. Teixeira, J. , Silva, G., Espinhara, D., Fernandes, R., Ambrosio, J. , Duartec, M. and Azevedo R. (2018) DKDOnto: An Ontology to Support Software Development with Distributed Teams. 22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. Elsevier, Procedia Computer Science 126 (2018) 373–382.
[25] Wongthongtham, P., Kasisopha, N. , Chang, E. and Dillon T. (2008) A Software Engineering Ontology as Software Engineering Knowledge Representation. International Conference on Convergence and Hybrid Information Technology. IEEE. 668-675
[26] Gazel S., Sezer, E. A. and Tarhan A. (2011) An Ontology Based Infrastructure To Support CMMI Based Software Process Assessment. Gazi University Journal of Science. 25(1) 155-164
[27] Oveh, R.O., Efevberha-Ogodo O. & Egbokhare, F.A.(2019) Software Process Ontology: A case study of software organisations software process sub domains. Journal of the Nigerian Society of Physical Sciences.1(4), 122-130. Retrieved from https://journal.nsps.org.ng/index.php/jnsps/article/view/28
[28] Ward J. and Aurum A. (2004) Knowledge Management in Software Engineering - Describing the Process. Proceedings of the Australian Software Engineering Conference (ASWEC'04)
[29] Guarino N. and Welty C. (2002) Evaluating Ontological Decisions with Ontoclean. Communications of the ACM, 45(2):61–65.
[30] Rodrigues, C.M., Freitas, F.L., and Azevedo, R.R. (2015). OCIP - An OntoClean Evaluation System Based on a Constraint Prolog Extension Language. *ONTOBRAS*. 1442 retrieved from http://ceur-ws.org/Vol-1442/paper_16.pdf
[31] Welty, C. A. and Guarino, N. (2001). Supporting ontological analysis of taxonomic relationships. Data and Knowledge Engineering, 39(1):51–74