Web Knowledge Representation with Autonomic Ontologies

Stainam Brandao[1], Jonice Oliveira[2], Jano Moreira de Souza[1]

[1]COPPE/UFRJ - Coordenação dos Programas de Pós-graduação de Engenharia – Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro, RJ - Brasil

[2]DCC/IM/UFRJ – Instituto de Matemática – Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro, RJ - Brasil.

{stainam,jano}@cos.ufrj.br; jonice@dcc.ufrj.br

Abstract. *The success of the semantic web is intrinsically related to the use of ontologies. This important role given to ontologies in the semantic web implies increasing need for development and maintenance of domain ontologies, even with the scarcity of tools supporting the life cycle from creation to management and adaptation. In this article, we presented an ongoing doctoral work that focus on the Web knowledge representation available with domain ontologies and processing of this knowledge autonomously through a systematic evaluation and redesign of the ontology. The autonomic computing approach was used to provide the ability to adapt ontologies on the independent environment like the Web, where there is no restriction on the information being published, which can damage their quality.*

1. Introduction

Ontologies are defined in this work as a representation of the conceptualization [Gruber 1993] and thus, a symbolic representation of world concepts taking an epistemological view of reality and therefore a possibility of consistent world [Guizzardi 2005]. Also, the language and common sense influence on the ontology building, because our mind filters out this reality, where a concept is used in different contexts and its meaning depends on context.

Ontologies have proven beneficial to the representation of domain knowledge [Alani and Brewster 2005], and its importance in the semantic web [Berners-Lee , Hendler and Lassila 2001]. In this case, ontologies often support the process of indexing resources content, which is called semantic annotation and can result in the representation of explicit knowledge that cannot be assessed and managed because of their mess.

However, ontologies as a conceptual model for a business domain should react to any changes in business environment, without affecting the intended model and also incorporating additional functionality in accordance with changes in the user's needs, organizing information in a better way, etc. If the ontology update or semantic

annotation is performed in an inconsistent, redundant or incomplete way, then reliability, accuracy and efficiency of the system decrease significantly [Maedche, Motik, Stojanovic, Studer and Volz 2003]. According to these authors, to avoid these real problems, ontologies-based applications may support mechanisms for changes detection, analyze and resolve it in a consistent way.

In this paper, we present an autonomic mechanism for ontology concept fragmentation and assessment criteria to achieve a quality level of configuration management that uses instances, user queries and the web to identify undesirable situations and therefore guarantee the ontology evolution on the conceptualization, without affecting the intended model defined by the ontology engineer. Few studies explicitly examine approaches using ontologies in real environments. However, working with statistics of its use, we could analyze features that would affect their use or even would evolve it over time when it is taking dimensions that do not allow the engineers to pay attention to every detail.

Within this context, we define in our project autonomic ontology as an ontology that encompasses the knowledge represented and also guidelines for capturing and analyzing data related to their use and, planning and execution of behaviors to ontology evolution that leads to an optimal configuration based on the represented knowledge, on existing concepts instances and user queries performance.

Likewise, we present a repository that will support the lifecycle of the ontology from creation to evolution, focusing developing semi-automated ontology evolution to determine the side effects of any development or use of ontology. Always aiming to preserve the intended model already specified by the ontology engineer, but systematize the process of monitoring, analysis, planning and execution of adaptive actions. For this repository will give the name Onto-CHOP.

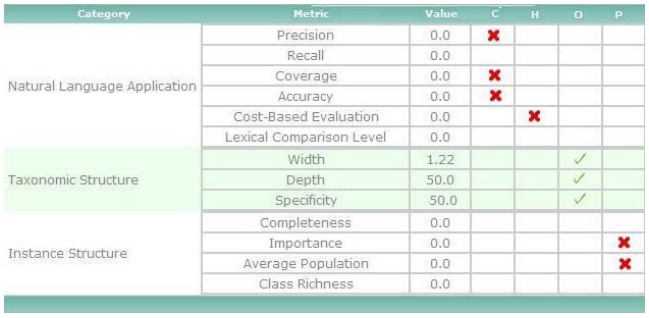
2. Autonomic Framework for Web Knowledge Representation

Since the manual ontologies construction requires a huge human effort, the acquisition of which is considered a bottleneck for the Semantic Web [Omelayenko 2001]. Furthermore, it is difficult to maintain, where small ontologies cannot be representative and the bigger can be complex enough to difficult its re-use. In this paper, maintenance involves ontology enrichment or tuning through an autonomic action on the vocabulary (taxonomy), which is representing web documents.

Below, we describe a repository with autonomic feature to evolve domain ontologies through indexed web documents and user queries performed on them. It will allow self-management of domain ontologies to treat the dynamism of the web environment by identifying events that demands ontologies fragmentation action. The monitoring of events is associated with ontology evolution to reduce the need for human intervention and are identified through ontology-based metrics already defined in the literature. On the other hand, the action patterns have been specified in this research aiming to use metrics to define guard expressions to guarantee autonomic actions performed on the conceptualization without affecting the intended model of the ontology. Notice that ontology engineers remain responsible for auditing the autonomic actions, confirming or revoking the evolution proposed for the domain ontology.

To achieve this, the first step was to define the categories of metrics that will be monitored by the repository. Any changes in metrics values of ontologies are detected and reported to the module responsible for analyzing this event. Event in this research is defined as any change in metrics values specified in the ontology repository. The metrics currently used are into three (3) categories: Natural Language Application, Taxonomic Structure and Instance Structure.

For the metrics related to the need for ontology fragmentation are created guard expressions, that are defined to values that trigger autonomic action of Fragmentation. The figure below shows an example where some metrics have guard expressions triggred represent by a (x) on the columns: Configuration (C), Healing (H), Optimization (O) and/or Protection (P). More than one expression can be defined for each metric.



Category	Metric	Value	C	H	O	P
Natural Language Application	Precision	0.0	✖			
	Recall	0.0				
	Coverage	0.0	✖			
	Accuracy	0.0	✖			
	Cost-Based Evaluation	0.0		✖		
	Lexical Comparison Level	0.0				
Taxonomic Structure	Width	1.22			✓	
	Depth	50.0			✓	
	Specificity	50.0			✓	
Instance Structure	Completeness	0.0				
	Importance	0.0				✖
	Average Population	0.0				✖
	Class Richness	0.0				

Figure 1. Guard expressions for metrics: 'x' means expressions triggered and 'v' metrics values in accordance

Therefore, when a set actions are triggered after an event, a planning may be build to define the execution script of this actions avoiding endless shots and unexpected results for the evolution of ontology.

The challenge related to the capture, modeling, storage and monitoring events justifies the need for an appropriate architecture for this purpose. The system architecture was developed using the approach of autonomic computing to provide capacity for self-management, hiding from the user complexity of programming and providing a system that works 24 / 7 [Huebscher and McCann 2008], [IBM 2005], [Kephart and Chess 2003], [Miller 2005].

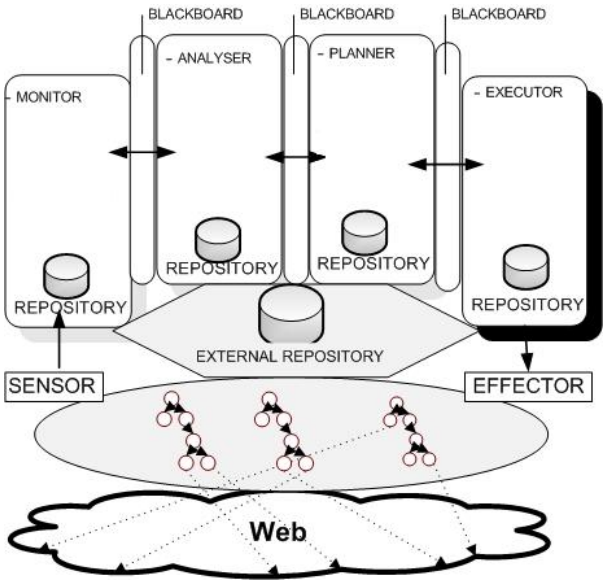


Figure 2. Framework Architecture

The figure 2 shows the architecture with the following four modules and a brief description, which will be detailed in the next section:

- Monitor has sensors that detect events in domain ontologies. We defined as an event any changes related to user queries, new web documents associated on the ontology and manual evolution performed by the ontology engineer. When an event is identified, the framework recalculates the metrics and publishes the changes to the Analyzer module.
- Analyzer has active rules to verify guard expression that are activated after the occurrence of correspondent event. Every guard expression is associated with autonomic action.
- Planner is responsible to orchestrate multiple expressions triggered simultaneously to avoid endless shots and unexpected results.
- Executor manages the action patterns specified by the framework and executes one when requested by the Planner, generating the expected result for ontology evolution (with intervention of the ontology engineer).

For communication between the modules the blackboard approach is used for modules register all the ontologies events. Blackboard is defined as an area of knowledge where loosely coupled entities share information [Shaw and Garlan 1996]. In the next section we present the autonomic actions in the form of standard to facilitate understanding.

3.1. Monitoring: methodology to capture metrics

[Gomez-Perez 1998] distinguished the following stages on the construction of ontologies: knowledge acquisition, requirements specification, conceptualization, implementation, evaluation and documentation. The monitoring is related to the

ongoing evaluation of the ontology using metrics that can be monitored and precursor of actions. The edition, manipulation (for semantic annotation or instantiation) and user queries are monitored by the framework from the moment that ontologies are published and made available on the Web.

Nevertheless, there is no framework available in the literature to determine the quality of domain ontologies. However, many criteria have been proposed and the most accepted is the set of criteria presented in [Gruber 1993] and cited also in [Geert 2000].

The monitor module is one that is aware of events on the ontologies, recalculating the metrics and reporting when any values changes. The quality metrics consolidated in the literature and presented in the framework are divided into three (3) categories: Intentional Size, Taxonomic Structure and Natural Language Application.

The Natural Language Application category provides metrics to determine whether or not the results of ontology content mining, capture most of the notions of the input text by using a number of metrics. Seven measures have been defined that should express how well the resulting triples lexically represent the important notions of the application domain:

- Precision: The intersection of the words of the triples and the related words divided by the number of words of the triples.
- Recall: Measures the ratio between the number of items correctly identified and the total number of correct items.
- Coverage: The average of the overlap between the vocabulary of the triples and the input text for each frequency class.
- Accuracy: The intersection of the words of the triples and the related words divided by the number of relevant words.
- Cost-Based Evaluation (CBE): An alternative method to obtain a low level to measure performance.
- Lexical Comparison Level: Method to compare two words by the smallest number of insertions, deletions and substitutions.

The Taxonomic Structure category evaluates the structure of the ontology:

- Width: Average number of subclasses per class.
- Depth: Measure the ration between the total of concept and the width.
- Specificity: The ration between the Width and Depth

Finally, the Intentional Size category evaluates the instances of ontologies through its distribution on them:

- Class Richness: This metric is related to the distribution of instances between classes. Where the number of classes that have instances is compared with the total of classes defined in the schema. The result is the percentage indicating how classes wealthy in the ontology are.

- Average Population: This metric compares the number of instances by the number of classes.

- Importance: Quantifies instances that belong to a class in a sub-tree, aiming to identify, which areas of the scheme are in focus when the instances are extracted, assessing the distribution of instances for classes. The importance of a class Ci is defined by the number of instances that belong to sub-trees that have Ci as the root, compared with the total number of instances.

- Completeness: The completeness is defined by the current number of instances that belong to subtrees that have root Ci as compared with the expected number of instances for the sub-trees that have Ci as root. The result is the percentage of instances compared with expected coverage in the ontology instances.

Whereas this metrics was defined, the framework may monitor them and report any change that occurs in their values to other modules through the blackboard to determine whether the new value trigger an action.

3.2. Analysis: methodology for quality assessment

The metrics are associated with values through the logical operator to build a condition that works as a guard expression for the metric and triggers an autonomic action. An action is defined as an intervention in the ontology to modify the value of a particular metric through ontology evolution (with intervention of the ontology engineer). Notice that we do not use explicitly the Event-Condition-Action (ECA) nomenclature, because the characteristics of our framework are closer to expert systems paradigm. Likewise, its behavior presented is the same as in active rules and implemented with Jess, which is based on C Language Integrated Production System (CLIPS) [CLIPS 2009].

The Analyzer module is responsible to evaluate any change in the values of ontological metrics identified by the Monitor module and available through the data sharing area called blackboard. Also, the Analyzer module maintains a database with the rules and its correspondent actions that must be triggered when the condition of the rule is satisfied by ontologies metric. Notice that all triggered actions will be forwarded to the Planner module before been executed in order to be orchestrated, avoiding endless shots and unexpected results.

The ontology evolution occurred through the Fragmentation Action that is triggered to treat three events: (1) overloading of concepts with lots of instances or indexed Web documents, (2) low query performance and (3) inefficient representation of Web knowledge by the domain ontologies. For the first case, guard expressions are created with metrics Average Population - eAvCnt (C) and Class Richness - eMaxCnt (C) from Intentional Size Category that trigger an action when at least one ontology concept has two (2) times instances more than the overall ontology concepts. These guard expressions identify the concept concerned and the ontology scenario to know where to act with fragmentation.

```
(defrule INSTANCES_OVERLOAD
    ?o <-(OntologyVO (e_av_cnt_c  ?e_av_cnt_c) (e_max_cnt_c ?e_max_cnt_c)
        (e_max_cnt_c > (* e_av_cnt_c 3)))

    =>

    (save-problem "PROTECTION" "INSTANCES_OVERLOAD")
    (printout t "Self-protection -> e_av_cnt_c: " ?e_av_cnt_c
        " e_max_cnt_c: " ?e_max_cnt_c crlf)
    )
```

Figure 3. Active Rule in Jess

In the figure 3, we can see that when the rule is triggered, the action is saved in the blackboard (via the 'save-problem' function) and consequently the Planner module will be aware of this fact.

On the other hand, the low performance of queries is addressed with a guard expression of the 'Cost-Based Evaluation' from the Natural Language Application category. The figure 4 shows an expression guard that is triggered when a query reaches a maximum time for processing defined by the framework to treat a Healing action through Fragmentation pattern.

```
(defrule OVERLOAD_LOW_PERFORMANCE
    ?o <-(OntologyVO (last_query_process_time ?last_query_process_time)
        (e_av_cnt_c  ?e_av_cnt_c) (e_max_cnt_c ?e_max_cnt_c)
            (last_query_process_time > 1));

  =>

    (save-problem "HEALING" "OVERLOAD_LOW_PERFORMANCE")
        (printout t "Self-healing -> last_query_process_time: " ?last_query_process_time
            "e_av_cnt_c: " ?e_av_cnt_c " e_max_cnt_c: " ?e_max_cnt_c crlf)
)
```

Figure 4. Active Rule to low performance of user queries

Nonetheless, only a few guard expressions are pre-configured and enabled in the framework for autonomic action. The ontology engineer can specify and enable new guard expressions.

For the third case, the autonomic action to represent and process Web document is executed when the documents indexed by the domain ontology do not have their knowledge represented in a satisfactory manner in accordance with the set of metrics from Natural Language Application category.

```
(defrule PRECISION
    ?o <-(OntologyVO  (precision  ?precision))
    ?mCon <-(MetricaVO ( nomeMetrica "precision") (vlMinSelfCon > ?precision ) )

  =>

    (save-problem "CONFIGURATION" "PRECISION")
    (printout t "Self-configuration -> ontology availability: " ?availability
        " vlMinSelfCon: " ?mCon.vlMinSelfCon crlf)
)
```

Figure 5. Active Rule to Precision Metric

In the figure 5, we can see that when the rule is triggered, the action is saved in the blackboard (via the save-problem function), and consequently the Planner module planner will be aware of this fact.

3.3. Planner: orchestrating the actions

The evolution of domain ontology occurred through the Autonomic Fragmentation Action that is performed when quality metrics evaluated and monitored reaches undesirable levels that was pre-defined. Below, we formalize the action to facilitate the design, analysis and correction of some module that do not work perfectly.

Motivation

There is a scenario where the ontology has few concepts with many of the instances or generalizes a knowledge in which they are representing. This scenario may point to a necessity for defining new concepts from these ontology's centralizing concepts. Likewise, ontologies with this scenario may have low performance of their queries when these concepts or their instances are accessed.

Applicability

In a dynamic environment like the Web, where we want to semantically annotate Web documents, the ontology must evolve with the creation of new concepts to avoid overload in those who are concentrating great part of the instances. This scenario points to a protection action with the specialization of concepts and distribution of their instances. Guard expressions are built to the following metrics: Importance (instances distribution), Class Richness (concepts and their instances) and Cost-Based Evaluation (CBE -to measure performance). These metrics will be presented in the next section.

Model

The fragmentation action is triggered when a guard expression associated with it and controlled by the Analyzer module is enabled. When this occurs, the action uses the External Repository that contain the metrics and the current location of the ontology, the internal Repository that maintain the metrics that must be addressed by the concepts involved on the Fragmentation, the Function name to be used in action and where the evolved ontology will stored before engineer validation.

Upon receiving an event initiator of the fragmentation action, the first step is to identify the ontology concept to be fragmented (IDENTIFY CONCEPT ONTOLOGY). After, the extraction of domain vocabulary is performed on the summary of Web documents collection associated with the concept to be fragmented (GET IDF). The IDF (inverse document frequency) is retrieved from the Semantic Annotation repository that is populated before any autonomic action (see section 4.4). In the next step (GET MORE FREQUENT TERMS) we retrieve the most frequent terms (the minimum value pre-

defined by the ontology engineer), which will be the candidates terms for the ontology. In possession of these terms, the next step (IDENTIFY RELATIONS) is the identification of relations between the candidate terms in the documents collection. This step uses taxonomy of natural language (WordNet) [Miller 1995] and the domain ontology. On the next step the taxonomy built with candidate term and the relations identified is showed for the ontology engineer responsible for integrating this with the domain ontology (WAIT APROVALE). After the engineer finalize the manual integration, the last step (PERFORM SEMANTIC ANNOTATION) is executed to perform semantic annotation up on the documents previously associated with the concept(s) fragmented (s), since that can be replaced by the candidate terms.

Indeed, if the engineer has done some integration between the domain ontology and taxonomy extracted autonomically, we can deem the ontology suffered concepts fragmentation action.

3.4. Executor: autonomic actions modeling

As ontology is the specification of conceptualization [Gruber 1993] and we want evolves it, thus we have to guarantee the accuracy in evolve the conceptualization [Guarino 1998] through ensuring that development occurs on the conceptualization, but that the intended model remains as the original. In other words, we must keep the focus on relevant conceptual relationships to a specific context, in this case, the Web documents that are being represented.

For this, the Executor module is responsible for ontology changes presented by Planner module. The Executor receives as input the default action, performs the action on the ontology and finally generates a label with the new version of it. This, however, has the annotation semantic action as a preliminary step, in which associate web documents to the domain ontologies.

The Web documents have knowledge about a domain that is represented in the ontology and has the document content indexed for future user queries. From this point, we identify the need for ontology evolves either by creating new concepts and new meanings to existing concepts. The semantic annotation proposed here analyzes the content and index on the proposed framework as a way to provide semantics to the documents without these be aware of this and even been modify. Which we consider an annotation "top down", which is different of annotation manual, known in the literature, where the user inserts annotation with a markup (tag) language in the pages using an ontology (or vocabulary) to give semantic to the Web document.

The semantic notation is presented in the figure 6, where Web documents defined by the user are crawled and persisted on the ontologies repository to provide semantic to them and allow domain analysis and domain ontology evolution.

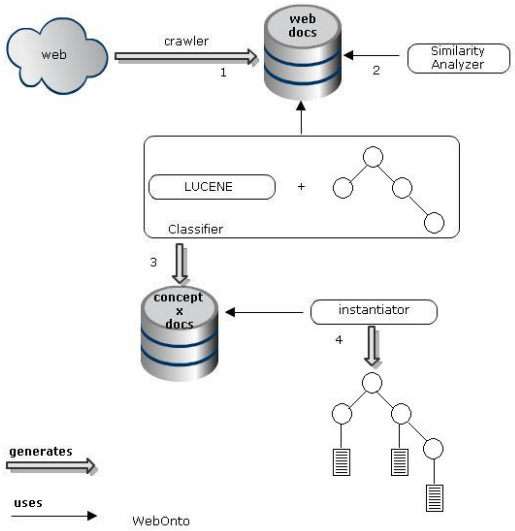


Figure 6. Semantic Annotation Methodology

On the first step, the COPCrawler (Pacheco, 2010) crawler the Web documents from URLs specified by the user and persists them in the database. After, the document content is processed, eliminating advertising and other links, and finally summarize the content to capture only the terms already in the domain ontology and future candidates terms for the ontology concept. The search for candidate terms in the document content implies a syntactic analysis of sentences. For this operation, the first step is to define a context for each document that is formed by the concepts that exist in the ontology and are present in the document, regardless of syntactic functions they perform. The second step is to define the candidates terms as those 10 (ten) words that occur before and after the words belonging to the context.

Follow the figure 6, the step 2 uses the vector model (TF-IDF) from Lucene [Lucene 2009] to calculate the similarity between Web documents persisted in the database by the crawler. With this step we want to ensure minimal accuracy between the documents and the domain. The documents are represented using values of terms absolute frequency based on expression reduced to the terms considered "important" by measures of information retrieval, eliminating the most and least frequent terms. Thus, based on this input, each document is represented by a vector of concepts and the frequency with which concept occurs in the document. After, the similarity value is defined between each document with the ontology vocabulary represented by the metadata name and description of the ontology concepts. The following formula (figure 7) corresponds to the cosine distance between the metadata words set q and each document dj. The value Sim(d_j, q) allows to cluster the documents by the similarity feature.

$$sim(d_j \cdot q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|},$$

Figure 7. Similarity measurement between two vectors

In step 3, the Classifier module applies the Jaccard Similarity Coefficient [Jaccard 2009] to determine the similarity between the Web documents persisted and the ontology concepts to define the domain vocabulary used in the text content and therefore, provide a semantic annotation to the document. The similarity is checked by comparing the entire meta-data ontology concept with the text content of the web document, generating a value between 0-1. Thus, we generate a database associating each Web document to concepts with a minimal value of similarity defined by the framework.

In the last step 4, from the base created in step 3, for each web document is created an instance on the associated ontology concept. So after the semantic annotation, the autonomic actions are able to run.

5. Preliminary Results

This section presents a brief case study to demonstrate how the architecture works. The example used in this paper is adapted from Brazilian Social Security ontology that represents the vocabulary and some business rules related to benefits granting for citizen retirement.

As explained in section 2, the domain ontology must be made available in the framework and the desirable values for the metrics configured by the ontology engineer as shown in figure 8. In this example, we can observe that the Average Population metric related to Fragmentation action was enable to perform as a Healing when identify a concept with more than 90% of instances. Or as a Protection when identify a concept with between 50% and 60% of ontology instances.

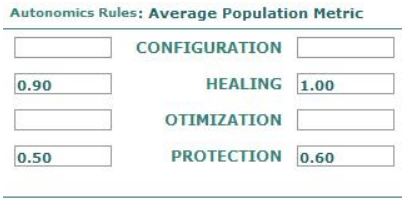


Figure 8. Average Population metric configuration

After, the ontology engineer may inform the URL of Web documents to be semantically annotated. In our case, the URL came from Intranet and Internet sites available by Brazilian Social Security. The ontology is defined in the Portuguese language, has 352 concepts defined and 18 object property for the business rules. The idea is to have the knowledge represented on the ontology available to technicians and the Brazilian citizens. Nonetheless, the Web documents are always growing and being modified due to the constant changes in benefits granting for citizen retirement, which is very common in Brazil. One example is the current debate in Congress by the end of Social Security Factor concept that was created with the aim of reducing the value of pension benefits at the time of grant, so inversely proportional to the retirement age of the insured.

In this case study, we used an early version of the ontology with the basic concepts considered to benefit: Allowance, Pension, Retirement, Help and Annuities. The objective is the identification by the autonomic ontology evolution of these concepts that are overloaded.

Instance Structure	Completeness	0.0				
	Importance	0.65				
	Average Population	0.59		✓		✖
	Class Richness	0.3				✓

Figure 9. Exemple of a concept with 59% of the instance which trigger the protection rule

After that, as expected, the figure 9 depicts the scenario where there is a concept with 59% of the instance which trigger the protection rule that monitors values between .50 and .60 as shown in figure 9. Finally, it is up to the ontology engineer access this flag, see the fragmentation of autonomic action framework to consolidate the ontology evolution, since this is not done automatically.

Our document corpus has been crawled from a WWW provider for Brazilian Social Security (URL: http://www.previdencia.gov.br) and (http://www-dtpnet) consisting about of 1800 HTML documents with a total sum of over 1.2 million words.

The Fragmentation action identified 78 new concepts for the five concepts mentioned above. One example is the figure 10, where seven new types were identified for the Saving (Pecúlio, in Portuguese) Benefits.

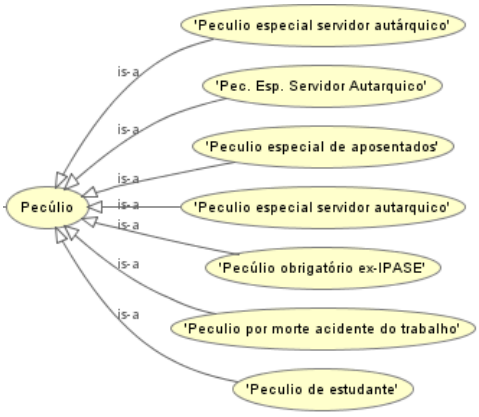


Figure 10. Seven new types were identified for the Saving (Pecúlio, in Portuguese) Benefits.

6. Conclusions and Future Works

This paper proposes a repository for domain ontologies that encompasses guidelines for capturing and analyzing data related to their use and also, planning and execution of behaviors to ontology evolution that leads to an optimal configuration based on the represented knowledge, on existing concepts instances and user queries performance. The proposed autonomic ontology repository has shown to be simple, yet powerful. However, we do know that challenges still exist regarding:

- Action quality, since, as an autonomic system, it should be able to distinguish trustworthy from untrustworthy action;
- Analysis of web content indexed in the ontology;
- Summarization through autonomic aggregation of content in heterogeneous format
- Specification of new action patterns related to other metrics defined in the literature.

One future work is the development of a Context-Sensitive Search based on the content web documents analyzed. Another future work is allow navigation through Web documents by the domain ontology, deeming the correlation of documents already defined with the Vector Model (TF-IDF) approach from Lucene [Lucene 2009] to calculate the similarity between Web documents.

A future target is the integration with Formal Ontology to establish formal meanings for domain vocabulary allowing axiomatization and integration of domain ontologies from different sources. We notice that domain ontologies consist of specialized terminology and a particular vision of reality. But the meaning remains dependent on the context. The use of Formal Ontology to integrate the domain ontology can promote the reuse, integration and management through the construction of ontologies from web documents.

References

Alani, H. and Brewster, C. (2005) “Ontology ranking based on the analysis of concept structures,” Proceedings of the 3rd international conference on Knowledge capture, Banff, Alberta, Canada: ACM, pp. 51-58.

Berners-Lee, T., Hendler, J. and Lassila, O. (2001) “The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities”.

CLIPS (2009) “http://clipsrules.sourceforge.net/” Acessed in august 22th 2009.

Geert, M.W.G. (2000) “The Ontological Foundation of REA Enterprise Information Systems,” Ago.

Gomez-Perez, A. (1998) Knowledge sharing and reuse. In J. Liebowitz, ed., The Handbook of Applied Expert Systems, CRC Press.

Gruber, T.R. (1993) “A translation approach to portable ontology specifications,” Knowl. Acquis., vol. 5, pp. 199

Guarino, N. (1998)“Formal Ontology and Information Systems”. . In: Proceedings of the international conference on Formal Ontology in Information Systems, pp. 3-15.

Guizzardi, G. (2005) “Ontological foundations for structural conceptual models”. Doctoral Thesis.

Huebscher, M. C. and McCann, J.A. (2008) “A survey of autonomic computing—degrees, models, and applications”, ACM Computing Surveys.

IBM (2005) “An architectural blueprint for autonomic computing”, Autonomic Computing White Paper, Third Edition, June.

Jaccard (2009) “http://sourceforge.net/projects/simmetrics/” Acessed in august 22th.

Jess (2009) “http://www.jessrules.com/” Acessed in august 22th.

Kephart, J.O. and Chess, D.M. (2003) “The Vision of Autonomic Computing”, IBM Thomas J. Watson Research Center.

Lucene, (2009) “http://lucene.apache.org/” Acessed in november 12th.

Maedche, A., Motik, B., Stojanovic, L., Studer, R. and Volz, R. (2003) “An infrastructure for searching, reusing and evolving distributed ontologies,” WWW, pp. 439–448.

Miller, G.A. (1995) “WordNet: A Lexical Database for English”. In Communication of the ACM, vol. 38, pp. 39--41.

Miller, B. (2005) The autonomic computing edge: Can you CHOP up autonomic computing? IBM Corporation.

Omelayenko, B. (2001)“Learning of Ontologies for the Web: the Analysis of Existent Approaches,” Proceedings of the International Workshop on Web Dynamics, held in conj. with the 8th International Conference on Database Theory (ICDT’01), London, UK.

Shaw, M. and Garlan, D. (1996) Software Architecture: Perspectives on an Emerging Discipline, Prentice Hall.