Facilitating Ontology Development with Continuous Evaluation

Dejan Lavbič and Marjan Krisper

Dejan Lavbič and Marjan Krisper. 2010. **Facilitating Ontology Development with Continuous Evaluation**, Informatica **(INFOR)**, 21(4), pp. 533 - 552.

Abstract

In this paper we propose facilitating ontology development by constant evaluation of steps in the process of ontology development. Existing methodologies for ontology development are complex and they require technical knowledge that business users and developers don't poses. By introducing ontology completeness indicator developer is guided throughout the development process and constantly aided by recommendations to progress to next step and improve the quality of ontology. In evaluating the ontology, several aspects are considered; from description, partition, consistency, redundancy and to anomaly. The applicability of the approach was demonstrated on Financial Instruments and Trading Strategies (FITS) ontology with comparison to other approaches.

Keywords

Ontology development methodology, ontology evaluation, ontology completeness, rapid ontology development, semantic web

1 Introduction

The adoption of Semantic Web technologies is less than expected and is mainly limited to academic environment. We are still waiting for wide adoption in industry. We could seek reasons for this in technologies itself and also in the process of development, because existence of verified approaches is a good indicator of maturity. As technologies are concerned there are numerous available for all aspects of Semantic Web applications; from languages for capturing the knowledge, persisting data, inferring new knowledge to querying for knowledge etc. In the methodological sense there is also a great variety of methodologies for ontology development available, as it will be further discussed in section 2, but the simplicity of using approaches for ontology construction is another issue. Current approaches in ontology development are technically very demanding and require long learning curve and are therefore inappropriate for developers with little technical skills and knowledge. In majority of existing approaches an additional role of knowledge engineer is required for mediation between actual knowledge that developers possess and ontology engineers who encode knowledge in one of the selected formalisms. The use of business rules management approach (Smaizys and Vasilecas, 2009) seems like an appropriate way to simplification of development and use of ontologies in business applications. Besides simplifying the process of ontology creation we also have to focus on very important aspect of ontology completeness. The problem of error-free ontologies has been discussed in (Fahad and Quadir, 2008; Porzel and Malaka, 2004) and several types of errors were identified - inconsistency, incompleteness, redundancy, design anomalies etc. All of these problems have to already be addressed in the development process and not only after development has reached its final steps.

In this paper we propose a Rapid Ontology Development (ROD) approach where ontology evaluation is performed during the whole lifecycle of the development. The idea is to enable developers to rather focus on the content than the formalisms for encoding knowledge. Developer can therefore, based on recommendations,

improve the ontology and eliminate the error or bad design. It is also a very important aspect that, before the application, the ontology is error free. Thus we define ROD model that introduces detail steps in ontology manipulation. The starting point was to improve existing approaches in a way of simplifying the process and give developer support throughout the lifecycle with continuous evaluation and not to conclude with developed ontology but enable the use of ontology in various scenarios. By doing that we try to achieve two things:

- guide developer through the process of ontology construction and
- improve the quality of developed ontology.

The remainder of the paper is structured as follows. In the following section 2 state of the art is presented with the review of existing methodologies for ontology development and approaches for ontology evaluation. After highlighting some drawbacks of current approaches section 3 presents the ROD approach. Short overview of the process and stages is given with the emphasis on ontology completeness indicator. The details of ontology evaluation and ontology completeness indicator are given in section 3.3, where all components (description, partition, redundancy and anomaly) that are evaluated are presented. In section 4 evaluation and discussion about the proposed approach according to the results obtained in the experiment of **Financial Instruments and Trading Strategies (FITS)** is presented. Finally in section 5 conclusions with future work are given.

2 Related work

2.1 Review of related approaches

Ontology is a vocabulary that is used for describing and presentation of a domain and also the meaning of that vocabulary. The definition of ontology can be highlighted from several aspects. From taxonomy (Corcho et al., 2003; SanJuan and Ibekwe-SanJuan, 2006; Veale, 2006) as knowledge with minimal hierarchical structure, vocabulary (Bechhofer and Goble, 2001; Miller, 1995) with words and synonyms, topic maps (Dong and Li, 2004; Park and Hunting, 2002) with the support of traversing through large amount of data, conceptual model (Jovanović and Gašević, 2005; Mylopoulos, 1998) that emphasizes more complex knowledge and logic theory (Corcho et al., 2003; Dzemyda and Sakalauskas, 2009; Waterson and Preece, 1999) with very complex and consistent knowledge.

Ontologies are used for various purposes such as natural language processing (Staab et al., 1999), knowledge management (Davies et al., 2006), information extraction (Wiederhold, 1992), intelligent search engines (Heflin and Hendler, 2000), digital libraries (Kesseler, 1996), business process modeling (Brambilla et al., 2006; Ciuksys and Caplinskas, 2007; Magdalenic et al., 2009) etc. While the use of ontologies was primarily in the domain of academia, situation now improves with the advent of several methodologies for ontology manipulation. Existing methodologies for ontology development in general try to define the activities for ontology management, activities for ontology development and support activities. Several methodologies exist for ontology manipulation and will be briefly presented in the following section. CommonKADS (Schreiber et al., 1999) is in fact not a methodology for ontology development, but is focused towards knowledge management in information systems with analysis, design and implementation of knowledge. CommonKADS puts an emphasis to early stages of software development for knowledge management. Enterprise Ontology (Uschold and King, 1995) recommends three simple steps: definition of intention; capturing concepts, mutual relation and expressions based on concepts and relations; persisting ontology in one of the languages. This methodology is the groundwork for many other approaches and is also used in several ontology editors. METHONTOLOGY (Fernandez-Lopez et al., 1999) is a methodology for ontology creation from scratch or by reusing existing ontologies. The framework enables building ontology at conceptual level and this approach is very close to prototyping. Another approach is TOVE (Uschold and Grueninger, 1996) where authors suggest using questionnaires that describe questions to which ontology should give answers. That can be very useful in environments where domain experts have very little expertise of knowledge modeling. Moreover authors of HCONE (Kotis and Vouros, 2003) present decentralized approach to ontology development by introducing regions where ontology is saved during its lifecycle. OTK Methodology (Sure, 2003) defines steps in ontology development into detail and introduces two processes – Knowledge Meta Process and Knowledge Process.

The steps are also supported by a tool. UPON (Nicola et al., 2005) is an interesting methodology that is based on Unified Software Development Process and is supported by UML language, but it has not been yet fully tested. The latest proposal is DILIGENT (Davies et al., 2006) and is focused on different approaches to distributed ontology development.

From information systems development point of view there are several methodologies that share similar ideas found in ontology development. Rapid Ontology Development model, presented in this paper follows examples mainly from blended, object-oriented, rapid development and people-oriented methodologies (Avison and Fitzgerald, 2006). In blended methodologies, that are formed from (the best) parts of other methodologies, the most influential for our approach was Information Engineering (Martin and Finkelstein, 1981) that is viewed as a framework within which a variety of techniques are used to develop good quality information systems in an efficient way. In object-oriented approaches there are two representatives – Object-Oriented Analysis (OOA; Booch (1993)) and Rational Unified Process (RUP; Jacobson et al. (1999)). Especially OOA with its five major activities: finding class and objects, identifying structures, indentifying subjects, defining attributes and defining services had profound effect on our research, while it was extended with the support of design and implementation phases that are not included in OOA. The idea of rapid development methodologies is closely related to ROD approach and current approach addresses the issue of rapid ontology development which is based on rapid development methodologies of information systems. James Martin's RAD (Martin, 1991) is based on well known techniques and tools but adopts prototyping approach and focuses on obtaining commitment from the business users. Another rapid approach is Dynamic Systems Development Method (DSDM; Consortium (2005)) which has some similarities with Extreme Programming (XP; Beck and Andres (2004)). XP attempts to support quicker development of software, particularly for small and medium-sized applications.

Comparing to techniques involved in information systems development, the ontology development in ROD approach is mainly based on *holistic techniques* (rich pictures, conceptual models, cognitive mapping), *data techniques* (entity modeling, normalization), *process techniques* (decision trees, decision tables, structured English) and *project management techniques* (estimation techniques).

The ROD approach extends reviewed methodologies by simplifying development steps and introducing continuous evaluation of developed ontology. This is achieved by ontology completeness indicator that is based on approaches for ontology evaluation. Based on existing reviews in (Brank et al., 2005; Gangemi et al., 2006; Gómez-Pérez, 1999; Hartmann et al., 2004) we classify evaluation approaches into following categories:

- compare ontology to *"golden standard"* (Maedche and Staab, 2002),
- using ontology in an *application* and evaluating results (Porzel and Malaka, 2004),
- compare with source of data about the *domain to be covered* by ontology (Brewster et al., 2004) and
- *evaluation* done *by humans* (Lozano-Tello and Gómez-Pérez, 2004; Noy et al., 2005).

Usually evaluation of different levels of ontology separately is more practical than trying to directly evaluate the ontology as whole. Therefore, classification of evaluation approaches based on the level of evaluation is also feasible and is as follows: lexical, vocabulary or data layer, hierarchy or taxonomy, other semantic relations, context or application level, syntactic level, structure, architecture and design. Prior the application of ontologies we have to assure that they are free of errors. The research performed by Fahad and Quadir (2008) resulted in classification and consequences of ontology errors. These errors can be divided into inconsistency errors, incompleteness errors, redundancy errors and design anomalies.

2.2 Problem and proposal for solution

The review of existing approaches for ontology development in this section pointed out that several drawbacks exist. Vast majority of ontology development methodologies define a complex process that demands a long learning curve. The required technical knowledge is very high therefore making ontology development very difficult for nontechnically oriented developers. Among methodologies for ontology development there is a lack of rapid approaches which can be found in traditional software development approaches. On the other hand methodologies for traditional software development also fail to provide sufficient support in ontology development. This fact can be confirmed with the advent of several ontology development methodologies

presented at the beginning of this section. Majority of reviewed methodologies also include very limited evaluation support of developed ontologies. If this support exists it is limited to latter stages of development and not included throughout the process.

This paper introduces a novel approach in ontology modeling based on good practices and existing approaches (Allemang and Hendler, 2008; Cardoso et al., 2007; Fahad and Quadir, 2008; Fernandez-Lopez et al., 1999; Sure, 2003; Uschold and King, 1995) while trying to minimize the need of knowing formal syntax required for codifying the ontology and therefore bringing ontology modeling closer to business users who are actual knowledge holders. Based on the findings from the comparison of existing methodologies for ontology development and several evaluation approaches it has been noted that no approach exist that would constantly evaluate ontology during its lifecycle. The idea of proposed ROD approach with ontology completeness evaluation presented in section 3 is to create a feedback loop between developed ontology and its completeness by introducing indicator for completeness. With ROD approach detailed knowledge of development methodology is also not required as the process guides developers through the steps defined in methodology. By extending existing approaches with constant evaluation the quality of final artifact is improved and the time for development is minimized as discussed in section 3.3.

3 Rapid Ontology Development

3.1 Introduction to ROD process

The process for ontology development ROD (Rapid Ontology Development) that we propose is based on existing approaches and methodologies (see section 2) but is enhanced with continuous ontology evaluation throughout the complete process. It is targeted at domain users that are not familiar with technical background of constructing ontologies.

Developers start with capturing concepts, mutual relations and expressions based on concepts and relations. This task can include reusing elements from various resources or defining them from scratch. When the model is defined, schematic part of ontology has to be binded to existing instances of that vocabulary. This includes data from relational databases, text file, other ontologies etc. The last step in bringing ontology into use is creating functional components for employment in other systems.

3.2 ROD stages

The ROD development process can be divided into the following stages: *pre-development*, *development* and *post-development* as depicted in Figure 1. Every stage delivers a specific output with the common goal of creating functional component based on ontology that can be used in several systems and scenarios. In pre-development stage the output is feasibility study that is used in subsequent stage development to construct essential model definition. The latter artifact represents the schema of problem domain that has to be coupled with instances from the real world. This is conducted in the last stage post-development which produces functional component for usage in various systems.

The role of constant evaluation as depicted in Figure 1 is to guide developer in progressing through steps of ROD process or it can be used independently of ROD process. In latter case, based on semantic review of ontology, enhancements for ontology improvement are available to the developer in a form of multiple actions of improvement, sorted by their impact. Besides actions and their impacts, detail explanation of action is also available (see Figure 2).

In case of following ROD approach, while developer is in a certain step of the process, the OC measurement is adapted to that step by redefinition of weights (see Figure 5 for distribution of weights by ROD steps) for calculation (e.g., in Step 2.1 of ROD process where business vocabulary acquisition is performed, there is no need for semantic checks like instance redundancy, lazy concept existence or inverse property existence, but the emphasis is rather on description of TBox and RBox component and path existence between concepts).

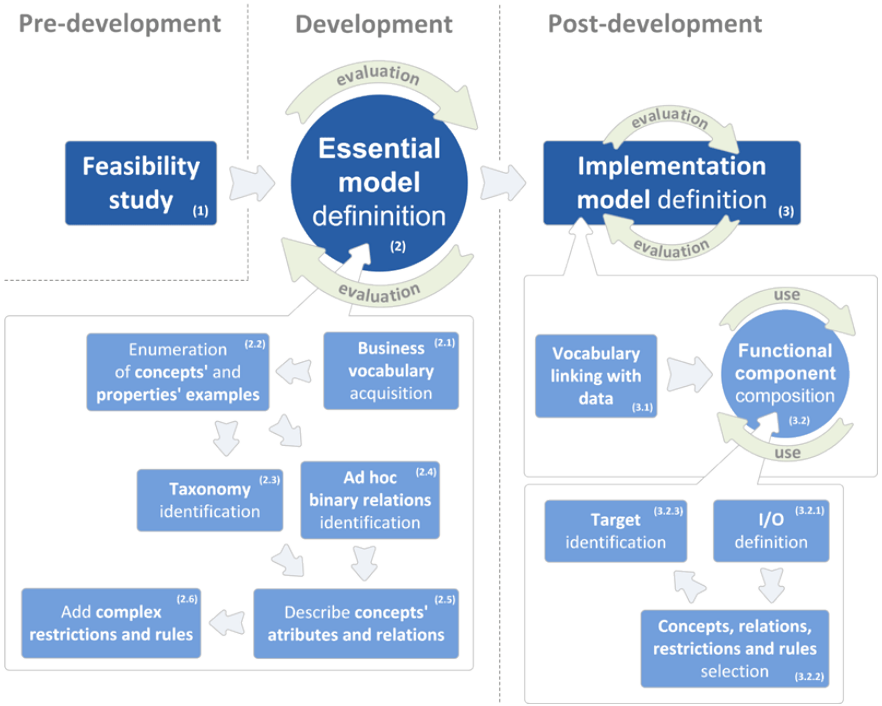

Figure 1: Process of rapid ontology development (ROD)

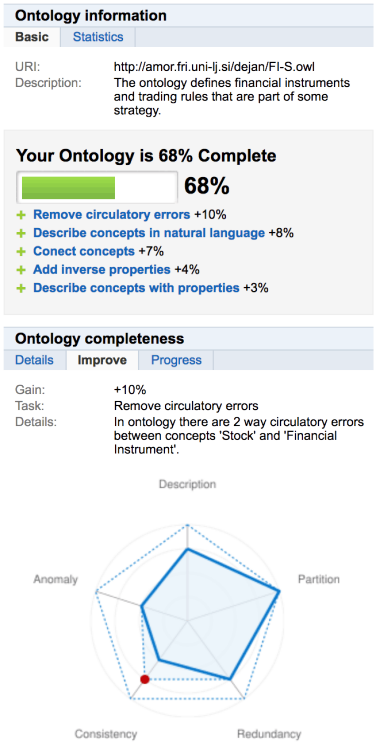

Figure 2: Display of ontology completeness (OC) results and improvement recommendations

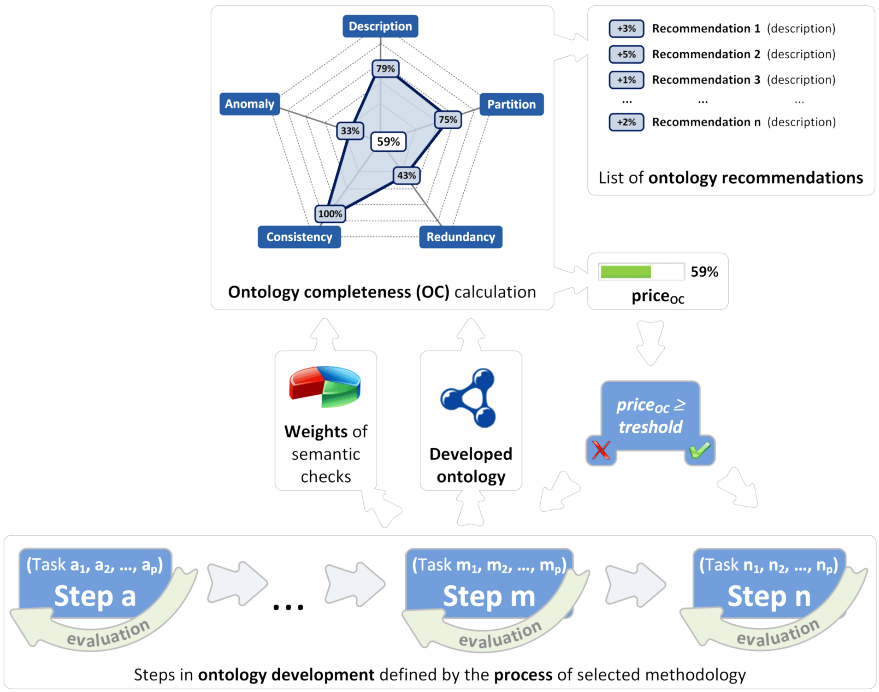

Figure 3: OC calculation

When OC measurement reaches a threshold (e.g., 80%) developer can progress to the following step (see Figure 3). The adapted OC value for every phase is calculated on-the-fly and whenever a threshold value is crossed, a recommendation for progressing to next step is generated. This way developer is aided in progressing through steps of ROD process from business vocabulary acquisition to functional component composition.

In case that ontology already exists, with OC measure we can place the completeness of ontology in ROD process and start improving ontology in suggested phase of development (e.g., ontology has taxonomy already defined, so we can continue with step 2.4 where ad hoc binary relations identification takes place).

3.3 Ontology evaluation and ontology completeness indicator

Ontology completeness (OC) indicator used for guiding developer in progressing through steps of ROD process and ensuring the required quality level of developed ontology is defined as

$$OC = f\left(C, P, R, I\right) \in [0, 1] \tag{1}$$

where C is set of concepts, P set of properties, R set of rules and I set of instances. Based on these input the output value in an interval $[0, 1]$ is calculated. The higher the value, more complete the ontology is. OC is weighted sum of semantic checks, while weights are being dynamically altered when traversing from one phase in ROD process to another. OC can be further defined as

$$OC = \sum_{i=1}^{n} w_i^{'} \cdot leafCondition_i \tag{2}$$

where n is the number of leaf conditions and $leafCondition$ is leaf condition, where semantic check is executed. For relative weights and leaf condition calculation the following restrictions apply $\sum_i w_i^{'} = 1$, $\forall w_i^{'} \in [0, 1]$ and

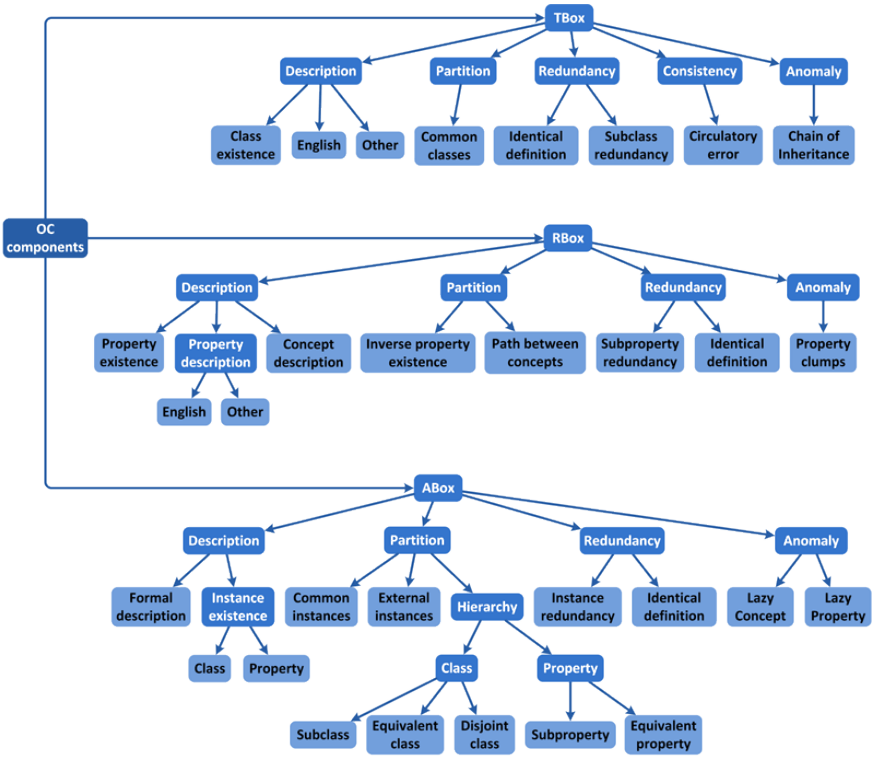

Figure 4: Ontology completeness (OC) tree of conditions, semantic checks and corresponding weights

$\forall leafCondition_i \in [0,1]$. Relative weight $w_i^{'}$ denotes global importance of $leafCondition_i$ and is dependent on all weights from leaf to root concept.

The tree of conditions in OC calculation is depicted in Figure 4 and contains semantic checks that are executed against the ontology. The top level is divided into *TBox*, *RBox* and *ABox* components. Subsequent levels are then furthermore divided based on ontology error classification (Fahad and Quadir, 2008). Aforementioned sublevels are *description*, *partition*, *redundancy*, *consistency* and *anomaly*.

This proposed structure can be easily adapted and altered for custom use. Leafs in the tree of OC calculation are implemented as semantic checks while all preceding elements are aggregation with appropriate weights. Algorithm for ontology completeness (OC) price is depicted in Definition 3.1, where X is condition and $w = w(X, Y)$ is the weight between condition X and condition Y.

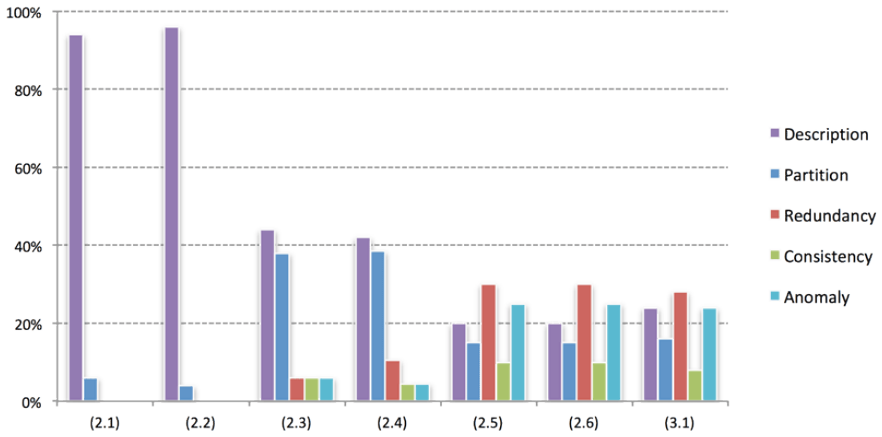

Figure 5: Impact of weights on OC sublevels in ROD process

Definition 3.1 (Ontology completeness evaluation algorithm)**.**

' Evaluation is executed on top condition "OC components" with weight 1
Evaluate $(\boldsymbol{X}, \boldsymbol{w})$
  $price_{OC} = 0$
  mark condition X as visited
  if not exists sub-condition of X
    ' Execute semantic check on leaf element
    return $w \cdot exec(X)$
  else for all conditions Y that are sub-conditions of X such that Y is not visited
    ' Aggregate ontology evaluation prices
    if $w(X, Y) \neq 0$
      $price_{OC} = price_{OC} + Evaluate(Y, w(X, Y))$
  return $w \cdot price_{OC}$
End

Each leaf condition implements a semantic check against ontology and returns value $leafCondition \in [0, 1]$.

Figure 5 depicts the distribution of OC components (description, partition, redundancy, consistency and anomaly) regarding individual phase in ROD process (see section 3.2). In first two phases 2.1 and 2.2 developer deals with business vocabulary identification and enumeration of concepts' and properties' examples. Evidently with aforementioned steps emphasis is on description of ontology, while partition is also taken into consideration. The importance of components description and partition is then in latter steps decreased but it still remains above average. In step 2.3 all other components are introduced (redundancy, consistency and anomaly), because developer is requested to define taxonomy of schematic part of ontology. While progressing to the latter steps of ROD process emphasis is on detail description of classes, properties and complex restriction and rules are also added. At this stage redundancy becomes more important. This trend of distributions of weights remains similarly balanced throughout the last steps 2.5 and 2.6 of development phase. In post-development phase when functional component composition is performed, ontology completeness calculation is mainly involved in redundancy, description and anomaly checking. The details about individual OC components are emphasized and presented in details in the following subsections.

3.3.1 Description

Description of ontology's components is very important aspect mainly in early stages of ontology development. As OC calculation is concerned there are several components considered:

- *existence of entities* (classes and properties) and *instances*,
- (multiple) *natural language descriptions* of TBox and RBox components and
- *formal description* of concepts and instances.

The notion of existence of entities is very straightforward; if ontology doesn't contain any entities than we have no artifacts to work with. Therefore the developer is by this metric encouraged to first define schematic part of ontology with classes and properties and then also to add elements of ABox component in a form of individuals.

Next aspect is natural language descriptions of entities. This element is despite of its simplicity one of the most important, due to ability to include these descriptions in further definition of complex axioms and rules (Vasilecas et al., 2009). Following business rules approach (Vasilecas and Sosunovas, 2008) it's feasible to create templates for entering this data on-the-fly by employing this natural description of entities. Developer is encouraged to describe all entities (classes and properties) with natural language using readable labels (e.g., `rdfs:label` and `rdfs:comment`) that don't add to the meaning of captured problem domain but greatly improves human readability of defined ontology. When constructing ontology it is always required to provide labels and description in English, but the use of other languages is also recommended to improve employment of ontology.

The last aspect of ontology description is formal description of TBox and ABox components that concerns concepts and instances. When describing classes with properties ontologists tend to forget defining domain and range values. This is evaluated for schematic part of ontology while for instances all required axioms are considered that are defined in TBox or ABox. Ontologists tend to leave out details of instances that are required (e.g., cardinality etc.).

3.3.2 Partition

Partition errors deal with omitting important axioms or information about the classification of concept and therefore reducing reasoning power and inferring mechanisms. In OC calculation several components are considered:

- *common classes* and *instances*,
- *external instances* of ABox component,
- *connectivity of concepts* of TBox component and
- *hierarchy of entities*.

The notion of common classes deals with the problem of defining a class that is a sub-class of classes that are disjoint. The solution is to check every class C_i if exist super-classes C_j and C_k that are disjoint. Similar is with common instances where situation can occur where instance is member of disjointing classes.

When decomposing classes in sub-class hierarchy it is often the case that super-class instance is not a member of any sub-class. In that case we deal with a problem of external instances. The solution is to check every class C_i if exist any instance that is a member of C_i, but not a member of any class in set of sub-classes.

The aspect of connectivity of concepts deals with ontology as whole and therefore not allowing isolated parts that are mutually disconnected. The first semantic check deals with existence of inverse properties. If we want to contribute to full traversal among classes in ontology the fact that every object property has inverse property defined is very important.

The second semantic check deals with existence of path between concepts. Ontology is presented as undirected graph $G = (V, E)$ and we try to identify maximum disconnected graphs.

The last aspect of ontology completeness as partition is concerned with hierarchy of entities. We introduce data oriented approach for definition of hierarchy of entities where technical knowledge from domain user is not required. This is based on requirement that for every class and property defined ontologist is requested to insert also few instances (see preliminary steps in ROD process introduced in section 3.2). After this requirement is met, set of competency questions are introduced to the domain user and the result are automatically defined hierarchy axioms (e.g., `rdfs:subClassOf`, `owl:equivalentClass`, `owl:disjointWith`, `rdfs:subPropertyOf` and `rdfs:equivalentProperty`).

The approach for disjoint class recommendation is depicted in Definition 3.2, while approach for other hierarchy axioms is analogous.

Definition 3.2 (Recommend disjoint axiom between classes)**.**

recommendDisjointWithClasses

$\tau_{\subseteq}^{sibling} = \{\} \leftarrow$ Set of all sub-class pairs (C, D)

$Q_n \leftarrow$ Competency questions

$disjointClassRecommend = \{\}$

for each $C_i \in TBox$

add all sub-class pairs of class C_i to $\tau_{\subseteq}^{sibling}$

for each sub-class pair $(C_j, C_k) \in TBox$ where $C_j \subseteq C_i \wedge C_k \subseteq C_i \wedge C_j \neq C_k$

if $\exists i(C_j), i(C_k) \in ABox : (\neg Q_1(C_j, C_k) \wedge \neg Q_3(C_j, C_k))$ then

if $C_j \cap C_k \neq \{\}$ then

$disjointClassRecommend = disjointClassRecommend \cup (C_j, C_k)$

end if

end if

end for

end for

$$price = 1 - \frac{|disjointClassRecommend|}{\left|\tau_{\subseteq}^{sibling}\right|}$$

return ***disjointClassRecommend*** and ***price***

end

Using this approach of recommendation, domain users can define axioms in ontology without technical knowledge of ontology language, because with data driven approach (using instances) and competency questions the OC calculation indicator does that automatically.

Redundancy occurs when particular information is inferred more than once from entities and instances. When calculating OC we take into consideration following components:

- *identical formal definition* and
- *redundancy in hierarchy of entities.*

When considering identical formal definition, all components (TBox, RBox and ABox) have to be checked. For every entity or instance Ai all belonging axioms are considered. If set of axioms of entity or instance A_i is identical to set of axioms of entity or instance A_j and $A_i \neq A_j$, then entities or instances A_i and A_j have identical formal definition. This signifies that A_i and A_j describe same concept under different names (synonyms).

Another common redundancy issue in ontologies is redundancy in hierarchy. This includes sub-class, sub-property and instance redundancy. Redundancy in hierarchy occurs when ontologist specifies classes, properties or instances that have hierarchy relations (`rdfs:subClassOf`, `rdfs:subPropertyOf` and `owl:instanceOf`) directly or indirectly.

3.3.3 Consistency

In consistency checking of developed ontology the emphasis is on finding circulatory errors in TBox component of ontology. Circulatory error occurs when a class is defined as a sub-class or super-class of itself at any level of hierarchy in the ontology. They can occur with distance 0, 1 or n, depending upon the number of relations involved when traversing the concept down the hierarchy of concepts until we get the same from where we started traversal. The same also applies for properties. To evaluate the quality of ontology regarding circulatory errors the ontology is viewed as graph $G = (V, E)$, where V is set of classes and E set of `rdfs:subClassOf` relations.

3.3.4 Anomaly

Design anomalies prohibit simplicity and maintainability of taxonomic structures within ontology. They don't cause inaccurate reasoning about concepts, but point to problematic and badly designed areas in ontology. Identification and removal of these anomalies should be necessary for improving the usability and providing better maintainability of ontology. As OC calculation is concerned there are several components considered:

- *chain of inheritance* in TBox component,
- *property clumps* and
- *lazy entities* (classes and properties).

The notion of chain of inheritance is considered in class hierarchy, where developer can classify classes as `rdfs:subClassOf` other classes up to any level. When such hierarchy of inheritance is long enough and all classes have no appropriate descriptions in the hierarchy except inherited child, then ontology suffers from chain of inheritance. The algorithm for finding and eliminating chains of inheritance is depicted in Definition 3.3.

Definition 3.3 (Find chain of inheritance)**.**

findChainOfInheritance

$price = 1$

$axiom(C) = [type, entity, value] \leftarrow$ Axiom of class C

$A(C) = \forall axiom(C) : entity = C \leftarrow$ Set of asserted axioms of class C

$A^{-}_{\subseteq} \leftarrow$ Set of asserted axioms of class C without rdfs:subClassOf axiom

$chainOfInheritance = \{\}$

while $\exists C_i, C_j \in TBox \wedge \exists C_1, C_2, \ldots, C_n \in TBox : (C_j \subseteq C_n \subseteq C_{n-1} \subseteq \ldots \subseteq C_2 \subseteq C_1 \subseteq C_i) \wedge$

$(\forall C_1, C_2, \ldots, C_n : |superClass(C_n)| = 1 \wedge A^{-}_{\subseteq}(C_n) = \{\}) \wedge \left|A^{-}_{\subseteq}(C_i)\right| > 0 \wedge \left|A^{-}_{\subseteq}(C_j)\right| > 0$ then

$price = price - \frac{n}{n^{direct}_{\subseteq}}$

$chainOfInheritance = chainOfInheritance \cup \{C_i, C_j, \{C_1, C_2, \ldots, C_n\}\}$

end while

chainsOfInheritance and ***price***

end

The next aspect in design anomalies is property clumps. This problem occurs when ontologists badly design ontology by using repeated groups of properties in different class definitions. These groups should be replaced by an abstract concept composing those properties in all class definitions where this clump is used. To identify property clumps the following approach depicted in Definition 3.4 is used.

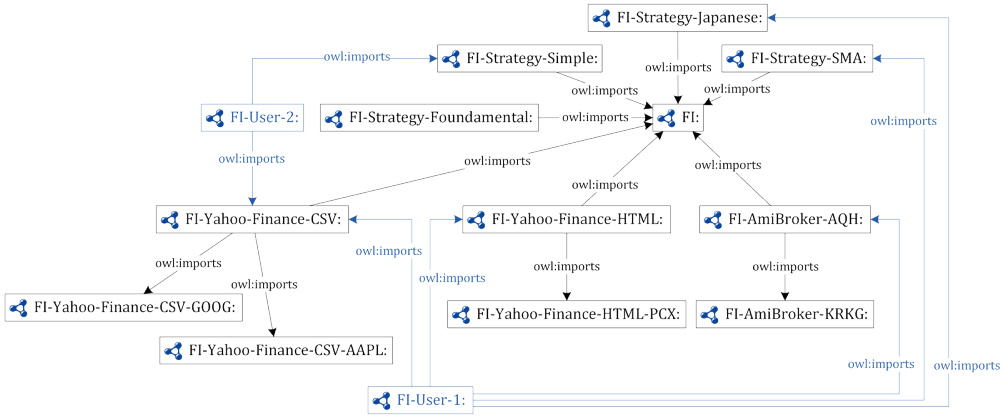

Figure 6: Financial instruments and trading strategies (FITS)

Definition 3.4 (Find property clumps)**.**

findPropertyClumps

$price \leftarrow 1$

$n_R \leftarrow$ Number of properties (datatype and object)

$V \leftarrow$ Classes and properties

$E \leftarrow$ Links between classes and properties

$propertyClumps = \{\}$

while exist complete bipartite sub-graph $K'_{m,n}$ of graph $G(V, E)$

select $K''_{m,n}$ from $K'_{m,n}$, where $\max(\frac{m'' \cdot n''}{m'' + n''})$

$propertyClumps = propertyClumps \cup K''_{m,n}$

remove all edges from $G(V, E)$ that appear in $K''_{m,n}$

$price = price - \frac{m'' \cdot n'' - (m'' + n'')}{n_R}$

end while

return ***propertyClumps*** and ***price***

end

The last aspect of design anomalies is lazy entities, which is a leaf class or property in the taxonomy that never appears in the application and does not have any instances. Eliminating this problem is quite straightforward; it just requires checking all leaf entities and verifying whether it contains any instances. In case of existence those entities should be removed or generalized or instances should be inserted.

4 Evaluation

4.1 Method

The ROD process was evaluated on Financial Instruments and Trading Strategies (FITS) ontology that is depicted in Figure 6.

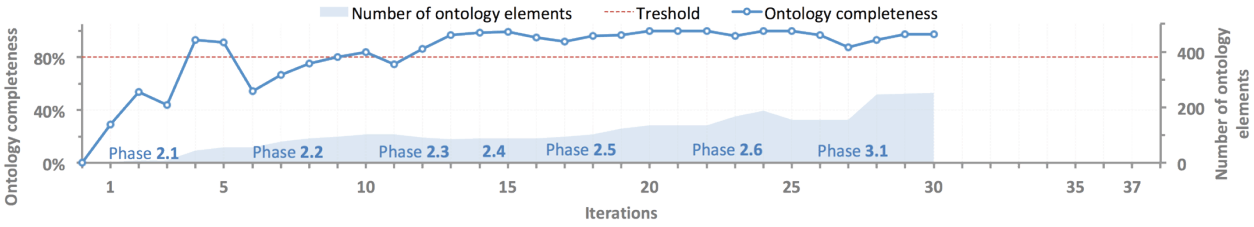

Figure 7: OC assessment and number of ontology elements through iterations and phases of ROD process

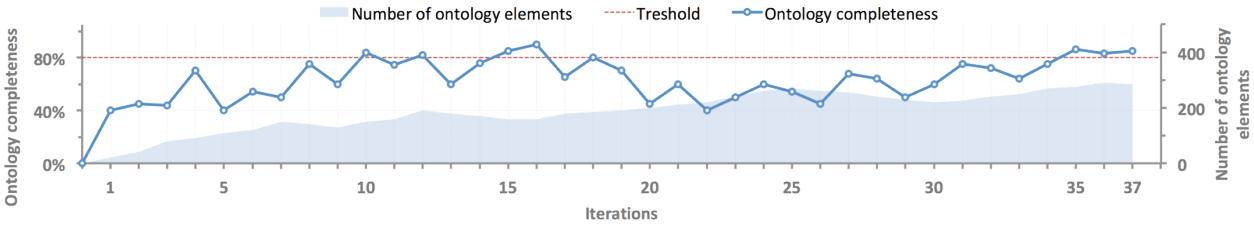

Figure 8: OC assessment and number of ontology elements through iterations of ad-hoc development process

When building aforementioned ontology one of the requirements was to follow Semantic Web mantra of achieving as high level of reuse as possible. Therefore the main building blocks of FITS ontology are all common concepts about financial instruments. Furthermore every source of data (e.g., quotes from Yahoo! Finance in a form of CSV files and direct Web access, AmiBroker trading program format etc.) is encapsulated in a form of ontology and integrated into FITS ontology. Within every source of data developer can select which financial instrument is he interested in (e.g., `GOOG`, `AAPL`, `PCX`, `KRKG` etc.). The last and the most important component are financial trading strategies that developers can define. Every strategy was defined in its own ontology (e.g., `FI-Strategy-Simple`, `FI-Strategy-SMA`, `FI-Strategy-Japanese` etc.). The requirement was also to enable open integration of strategies, so developer can select best practices from several developers and add its own modification.

Two different approaches in constructing ontology and using it in aforementioned use case were used. The approach of rapid ontology development (ROD) was compared to ad-hoc approach to ontology development, which was based on existing methodologies CommonKADS, OTK and METHONTOLOGY. With ROD approach the proposed method was used with tools IntelliOnto and Protégé. The entire development process was monitored by iteration, where ontology completeness price and number of ontology elements (classes, properties and axioms with rules) were followed.

At the end the results included developed ontology, a functional component and information about the development process by iteration. The final version of ontology was reviewed by a domain expert, who confirmed adequateness. At implementation level ontology was expected to contain about 250 to 350 axioms of schematic part and numerous instances from various sources.

4.2 Results and Discussion

The process of ontology creation and exporting it as functional component was evaluated on FITS ontology and the results are depicted in Figures 7 and 8. Charts represent ontology completeness price and number of ontology elements regarding to iterations in the process.

Comparing ROD to ad-hoc approach the following conclusions can be drawn:

- the number of iterations to develop required functional component using ROD approach (30) is less than using ad-hoc approach (37) which results in 23% less iterations;
- ontology developed with ROD approach is throughout the development process more complete and more appropriate for use than in ad-hoc, due to continuous evaluation and simultaneous alerts for

developers.

During the process of ontology construction based on ROD approach the developer was continuously supported by ontology evaluation and recommendations for progressing to next steps. When developer entered a phase and started performing tasks associated with the phase, ontology completeness was evaluated as depicted in Figure 2. While OC was less than a threshold value, developer followed instructions for improving ontology as depicted in Figure 3. Results of OC evaluation are available in a simple view, where basic statistics about ontology is displayed (number of concepts, properties, rules, individuals etc.), progress bar depicting completeness, and details about evaluation, improvement recommendations and history of changes. The core element is progress bar that denotes how complete ontology is and is accompanied with a percentage value. Following are recommendations for ontology improvement and their gains (e.g., remove circulatory errors (+10%), describe concepts in natural language (+8%), connect concepts (+7%) etc.). When improvement is selected (e.g., remove circulatory errors) the details are displayed (gain, task and details). The improvement and planned actions are also clearly graphically depicted on radar chart (see Figure 2). The shaded area with strong border lines presents current situation, while red dot shows TO-BE situation if we follow selected improvement.

When OC price crosses a threshold value (in this experiment 80%) a recommendation to progress to a new phase is generated. We can see from our example that for instance recommendation to progress from phase 2.5 to phase 2.6 was generated in 20th iteration with OC value of $91,3\%$, while in 19th iteration OC value was $76,5\%$.

As Figure 7 depicts ontology completeness price and number of ontology elements are displayed. While progressing through steps and phases it's seen that number of ontology elements constantly grow. On the other hand OC price fluctuate – it's increasing till we reach the threshold to progress to next phase and decreases when entering new phase. Based on recommendations from the system, developer improves the ontology and OC price increases again. With introduction of OC steps in ontology development are constantly measured while enabling developers to focus on content and not technical details (e.g. language syntax, best modeling approach etc.).

5 Conclusions and Future work

Current methodologies and approaches for ontology development require very experienced users and developers, while we propose ROD approach that is more suitable for less technically oriented users. With constant evaluation of developed ontology that is introduced in this approach, developers get a tool for construction of ontologies with several advantages:

- the required technical knowledge for ontology modeling is decreased,
- the process of ontology modeling doesn't end with the last successful iteration, but continues with post-development activities of using ontology as a functional component in several scenarios and
- continuous evaluation of developing ontology and recommendations for improvement.

In ontology evaluation several components are considered: description, partition, redundancy, consistency and anomaly. Description of ontology's components is very important aspect mainly in early stages of ontology development and includes existence of entities, natural language descriptions and formal descriptions. This data is furthermore used for advanced axiom construction in latter stages. Partition errors deal with omitting important axioms and can be in a form of common classes, external instances, hierarchy of entities etc. Redundancy deals with multiple information being inferred more than once and includes identical formal definition and redundancy in hierarchy. With consistency the emphasis is on finding circulatory errors, while anomalies do not cause inaccurate reasoning about concepts, but point to badly designed areas in ontology. This includes checking for chain of inheritance, property clumps, lazy entities etc. It has been demonstrated on a case study from financial trading domain that a developer can build Semantic Web application for financial trading based on ontologies that consumes data from various sources and enable interoperability. The solution can easily be packed into a functional component and used in various systems.

The future work includes improvement of ontology completeness indicator by including more semantic checks and providing wider support for functional components and creating a plug-in for most widely used ontology editors for constant ontology evaluation. One of the planned improvements is also integration with popular social networks to enable developers rapid ontology development, based on reuse.

References

Allemang, D. and Hendler, J. (2008). *Semantic Web for Working Ontologist: Effective Modeling in RDFS and OWL*. Elsevier.

Avison, D. and Fitzgerald, G. (2006). *Information Systems Development: Methodologies, Techniques and Tools, 4th edition*. McGraw-Hill, Maidenhead, UK.

Bechhofer, S. and Goble, C. (2001). Thesaurus construction through knowledge representation. *Data & Knowledge Engineering*, 37(1):25–45.

Beck, K. and Andres, C. (2004). *Extreme Programming explained: Embrace change, 2nd edition*. Addison-Wesley, USA, Boston.

Booch, G. (1993). *Object Oriented Analysis and Design with Applications, 2nd edition*. Addison-Wesley, Santa Clara, USA.

Brambilla, M., Celino, I., Ceri, S., and Cerizza, D. (2006). A Software Engineering Approach to Design and Development of Semantic Web Service Applications. In *5th International Semantic Web Conference*.

Brank, J., Grobelnik, M., and Mladenić, D. (2005). *A Survey of Ontology evaluation techniques*.

Brewster, C., Alani, H., Dasmahapatra, S., and Wilks, Y. (2004). *Data driven Ontology evaluation*.

Cardoso, J., Hepp, M., and Lytras, M. (2007). *The Semantic Web: Real World Applications from Industry*. Springer.

Ciuksys, D. and Caplinskas, A. (2007). Reusing ontological knowledge about business process in IS engineering: process configuration problem. *Informatica*, 18(4):585–602.

Consortium, D. (2005). *DSDM Manual version 4.2*. Tesseract Publishing, UK, Surrey.

Corcho, O., Fernandez-Lopez, M., and Gomez-Perez, A. (2003). Methodologies, tools and languages for building ontologies. Where is their meeting point? *Data & Knowledge Engineering*, 46(1):41–64.

Davies, J., Studer, R., and Warren, P. (2006). *Semantic Web technologies - trends and research in ontology-based systems*. John Wiley & Sons, Chichester, England.

Dong, Y. and Li, M. S. (2004). HyO-XTM: a set of hyper-graph operations on XML Topic Map toward knowledge management. *Future Generation Computer Systems*, 20(1):81–100.

Dzemyda, G. and Sakalauskas, L. (2009). Optimization and Knowledge-Based Technologies. *Informatica*, 20(2):165–172.

Fahad, M. and Quadir, M. A. (2008). Ontological errors - Inconsistency, Incompleteness and Redundancy. In *International Conference on Enterprise Information Systems (ICEIS) 2008*.

Fernandez-Lopez, M., Gomez-Perez, A., Sierra, J. P., and Sierra, A. P. (1999). Building a chemical ontology using methontology and the ontology design environment. *Intelligent Systems*, 14(1).

Gangemi, A., Catenacci, C., Ciaramita, M., and Lehmann, J. (2006). Modelling ontology evaluation and validation. In Sure, Y. D. J., editor, *3nd European Semantic Web conference (ESWC 2006)*, pages 140–154. Springer-Verlag Berlin.

Gómez-Pérez, A. (1999). *Evaluation of Taxonomic Knowledge in Ontologies and Knowledge Bases.*

Hartmann, J., Sure, Y., Giboin, A., Maynard, D., Suárez-Figueroa, M. d. C., and Cuel, R. (2004). D1.2.3 Methods for ontology evaluation. Technical report.

Heflin, J. and Hendler, J. (2000). Searching the web with SHOE. In *Artificial Intelligence for Web Search*, pages 36–40. AAAI Press, Menlo Park, USA.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process.* Addison-Wesley, Boston, USA.

Jovanović, J. and Gašević, D. (2005). Achieving knowledge interoperability: An XML/XSLT approach. *Expert Systems with Applications*, 29(3):535–553.

Kesseler, M. (1996). A schema based approach to HTML authoring. *World Wide Web Journal*, 96(1).

Kotis, K. and Vouros, G. (2003). Human centered ontology management with HCONE. In *IJCAI '03 Workshop on Ontologies and Distributed Systems.*

Lozano-Tello, A. and Gómez-Pérez, A. (2004). ONTOMETRIC: A method to choose the appropriate ontology. *Journal of Database Management*, 15(2):1–18.

Maedche, A. and Staab, S. (2002). *Measuring Similarity between Ontologies.*

Magdalenic, I., Radosevic, D., and Skocir, Z. (2009). Dynamic Generation of Web Services for Data Retrieval Using Ontology. *Informatica*, 20(3):397–416.

Martin, J. (1991). *Rapid Application Development.* MacMillan Publishing, Indianapolis, USA.

Martin, J. and Finkelstein, C. (1981). *Information Engineering*, volume Volume 1 and 2. Prentice Hall, New Jersey, USA.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Mylopoulos, J. (1998). Information modeling in the time of the revolution. *Information Systems*, 23(3-4):127–155.

Nicola, A. D., Navigli, R., and Missikoff, M. (2005). Building an eProcurement ontology with UPON methodology. In *15th e-Challenges Conference*, Ljubljana, Slovenia.

Noy, N. F., Guha, R., and Musen, M. A. (2005). *User ratings of ontologies: Who will rate the raters?*

Park, J. and Hunting, S. (2002). *XML Topic Maps: Creating and Using Topic Maps for the Web.* Addison-Wesley, Boston, USA.

Porzel, R. and Malaka, R. (2004). *A Task-based Approach for Ontology Evaluation.*

SanJuan, E. and Ibekwe-SanJuan, F. (2006). Text mining without document context. *Information Processing & Management*, 42(6):1532–1552.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., van de Velde, W., and Wielinga, B. (1999). *Knowledge engineering and management - The CommonKADS methodology.* The MIT Press: Cambridge, Massachusetts, London, England.

Smaizys, A. and Vasilecas, O. (2009). Business Rules based agile ERP systems development. *Informatica*, 20(3):439–460.

Staab, S., Braun, C., Bruder, I., Duesterhoeft, A., Heuer, A., Klettke, M., Neumann, G., Prager, B., Pretzel, J., Schnurr, H. P., Studer, R., Uszkoreit, H., and Wrenger, B. (1999). A system for facilitating and enhancing web search. In *International working conference on artificial and natural neural networks: Engineering applications of bio-inspired artificial neural networks (IWANN '99).*

Sure, Y. (2003). *Methodology, Tools & Case Studies for Ontology based Knowledge Management*. PhD Thesis.

Uschold, M. and Grueninger, M. (1996). Ontologies: principles, methods and applications. *Knowledge Sharing and Review*, 11(2).

Uschold, M. and King, M. (1995). Towards a methodology for building ontologies. In *Workshop on basic ontological issues in knowledge sharing (IJCAI '95)*.

Vasilecas, O., Kalibatiene, D., and Guizzardi, G. (2009). Towards a formal method for transforming ontology axioms to application domain rules. *Information Technology and Control*, 38(4):271–282.

Vasilecas, O. and Sosunovas, S. (2008). Practical application of BRTL approach for financial reporting domain. *Information Technology and Control*, 37(2):106–113.

Veale, T. (2006). An analogy-oriented type hierarchy for linguistic creativity. *Knowledge-Based Systems*, 19(7):471–479.

Waterson, A. and Preece, A. (1999). Verifying ontological commitment in knowledge-based systems. *Knowledge-Based Systems*, 12(1-2):45–54.

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49.