A Method and Tool for Fact Type Reuse in the DOGMA Ontology Framework

Christophe Debruyne, Pieter De Leenheer, and Robert Meersman

Semantic Technology and Application Research Laboratory (STARLab),
Department of Computer Science,
Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium
{chrdebru,pdeleenh,meersman}@vub.ac.be

Abstract. In the DOGMA approach to ontology engineering, the construction of an ontology starts from a "Lexon Base", a possibly very large and un-interpreted base of plausible elementary fact types called lexons. Lexons - mined from various linguistic sources such as schemas, texts or domain experts - are used to create ontological commitments by selecting or reusing a meaningful set of lexons and together with constraints so that the intended conceptualization is well approximated. All too often, the same or similar lexons are entered in the Lexon Base, which causes heterogeneity among different ontological commitments. Due to this heterogeneity, meaning negotiation to agree upon a common commitment becomes more difficult. Encouraging lexon reuse by providing knowledge engineers and domain experts an automated process for finding relevant lexons in the Lexon Base or existing ontological commitments can tackle this problem. In this paper, we introduce a novel approach to fact type reuse that we will apply to DOGMA MESS, a state-of-the-art collaborative ontology engineering methodology. The method we propose uses several heuristics that reside in one of the six semiotic levels described by Ronald Stamper's semiotic ladder and adds a pragmatic and social layer onto the current methodology and tools. We provide a proof of concept by implementing our method in a tool for finding relevant lexons while building an ontological commitment in a production environment called DOGMA Studio Workbench.

Keywords: Ontology reuse, knowledge reuse, ontology engineering.

1 Introduction

The amount of data within information systems all over the world keeps growing each year. Many organizations have their own closed information system that captures, at a certain level, the knowledge within their business. However, for some organizations it will be necessary to communicate with each other, especially when their domains overlap. Interoperability between information systems and sharing knowledge across organizations are just few of the motives for communication. This need for communication resulted in the introduction of *open*

R. Meersman, T. Dillon, P. Herrero (Eds.): OTM 2009, Part II, LNCS 5871, pp. 1147–1164, 2009.

information systems, which handle open networks, heterogeneous sources of information and support ontology evolution for the fast changing interoperability requirements [8].

A requirement for different organizations in a certain domain to communicate is to have a common understanding about that domain on which all parties can rely on. The process of reaching that common understanding often comes with disputes and *conflicts*, since every stakeholder has a different opinion on the situation. This observation also holds for building open information systems, where this situation is a conceptualization of (a part of) the real world. This conceptualization thus represents real-life objects, on which every stakeholder might have a different view. But in information systems, conflicts also rise from stakeholders entering the same *observation multiple* times or not *reusing* earlier conceptualizations where an agreement has already been achieved.

This conceptualization is defined into an *ontology*, which is a formal, shared understanding between different parties in the same domain [15,16]. Because of the fast changing requirements, the ontology engineered by the different stakeholders constantly evolves over time. This engineering process exists of two phases: *ontology elicitation*, for example by performing brainstorm sessions and *ontology application*.

2 Research Problem

The development process of an ontology can be intensive and time-consuming; many stakeholders need to reach a common agreement, and this proves to be sometimes difficult [10]. Because of the communal aspect in this agreement, it should be based on the *perspectives* of all involved stakeholders [8]. A perspective captures the meaning on what the stakeholder thinks is currently relevant to the community he makes part of. Perspectives are expressed in fact types, where a fact type is a type or kind of fact [17]. Evolving towards an agreement is thus a social process. All stakeholders have enclosed their own "opinion" in their information system. As the different organizations all have their own closed information system, they prefer their own model and promote it in order to limit their expenses.

Different opinions create conflicts between the perspectives of the different stakeholders and the current commonly agreed insights whenever new requirements must be implemented into the ontology. It is clear that an efficient way of tackling these conflicts can dramatically reduce the subsequent meaning interpretation and negotiation process. Earlier work on *Perspective management* [9] can provides means for conflict detection, permitting the different stakeholders to identify and explore them. The study, however, does not prevent one of the causes heterogeneous perspectives: the introduction of the same intended conceptualization by using different fact types by different stakeholders or reinventing fact types which are already available for reuse.

This paper introduces a framework and tool to support the user in reusing existing fact types and is organized as follows: we first elaborate on DOGMA

and ontology reuse. We then propose a methodology for ontology reuse built upon the collaborative ontology engineering methodology, DOGMA MESS, and present our tool that supports this methodology.

3 The DOGMA Approach

DOGMA is an ontology approach and framework that is not restricted to a particular representation language. One of the most characteristic features of DOGMA is the use of a layered architecture, in particular the possibility of having multiple views on and uses of the same stored conceptualization. The *meaning space* can be subdivided in different ways according to the needs of a specific application that will add its particular semantic restrictions according to its intended usage [20]. A DOGMA ontology consists of a *Lexon Base* layer and a *Commitment Layer*. This layering approach enhances the potential for reuse and allows for scalability in representing and reasoning about formal semantics [30]. In analogy with a principle stemming from the linguistics fields, this has been dubbed the *double articulation principle* [23]. This principle is an orthodox *model-theoretic* approach to ontology representation and development [23].

The *Lexon Base* holds (multiple) intuitive conceptualizations of a particular domain. Each conceptualization is simplified to a "representation-less" set of context-specific binary fact types called *lexons*. A lexon represents a plausible binary fact type and is formally described as a 5-tuple $\langle \gamma$*, headterm, role, co-role, tailterm* $\rangle$, where γ is an abstract *context identifier* used to group lexons that are logically related to each other. For example the lexon: $\langle$ *Comics, The Hero, beating, beaten by, The Villain* $\rangle$, can be read as: in the context *Comics*, *Hero* plays the role of *beating Villain* and *Villain* plays the role of being *beaten by Hero*. The goal of the Lexon Base is to reach a common and agreed understanding about the ontology terminology and is thus aimed at human understanding.

The *Commitment Layer*, together with its formal constraints [18,26], is meant for interoperability issues between information systems, software agents and web services. It consists of a finite set of axioms that specify which lexons of the Lexon Base are interpreted and how they are visible in the committing application, and (domain) rules that semantically constrain this interpretation. Experience shows that it is much harder to reach an agreement on domain rules than on a conceptualization [19], also motivating this separation.

DOGMA MESS [10,6,8] extends DOGMA by adding a community layer that enables scalable, community-grounded ontology engineering. The main focus lies on how to capture similarities and differences in meaning from domain experts, who can give different views on the domain ontology. Assigning them scalable knowledge elicitation tasks does this. DOGMA MESS is not only a collaborative process, but also a context-driven ontology engineering approach. The collaborative process implies that requirements of stakeholders are subject to constant evolution, resulting in many changes. These changes reflect themselves in multiple "perspective policies" [8] (with each perspective residing in a different context). Since requirements might change quickly, developing an inter-organizational ontology is an iterative (and therefore complex) process. DOGMA

MESS implements a versioning system for ontologies that not only allows domain experts in gradually building increasing complex versions of their conceptualizations, but also tackles the complexity partially by providing version management [4,9].

4 Ontology Reuse

There is a strong argument for *ontology reuse*. Reuse already proved its benefits in software engineering where the code-reuse reduces costs. In software engineering, reusing an existing component implies costs for its discovery, comprehension, evaluation, adaptation and actualization [2]. The cost of reusing a component often does not weigh against planning, designing and implementing the component from scratch. These costs, such as efforts in person months or duration, hold for ontology engineering as well, where the same concepts are all too often modeled over and over again. When building an ontology from scratch, the typical stages as defined by Fernandez [14] will be: (i) domain analysis, resulting in a requirements analysis; (ii) conceptualization, resulting in a conceptual model; (iii) implementation, where a specification of the conceptual model in the selected representation language is the result; and (iv) ontology population, generating instances and aligning them to the model, results in an instantiated ontology.

Ontology reuse, on the other hand, involves the discovery and reuse of existing (source) ontologies in order to generate a new (target) ontology. This means that ones needs to understand, evaluate and adapt the source ontology in order to fit in the requirements of the target ontology [1]. Ontology engineering is already considered a mature discipline in the context of the Semantic Web [2]. However, most of the currently available ontologies are not aligned to a specific ontology. They are often the product of ad hoc application-dependent engineering processes.

Ontology reuse starts with the identification of potentially relevant knowledge sources [2]. Like ontology building, this identification is mostly done manually. Bontas and Mochol's approach [2] to ontology reuse copes with limitations such as heterogeneity of different sources by proposing an incremental process that concentrates on the concepts represented in the input sources and subsequently takes into account additional information like semantic relationships and axioms depending on the application needs to integrate those concepts. Bontas *et al.* [3] also proposed a cost model for ontology engineering, called ONTOCOM, to determine the cost of building an ontology from scratch, reusing existing ontologies or a combination of both.

One of the phases in their approach is ontology integration, from which the steps can be found again in the work of Pinto and Martins [22], where they propose a methodology for ontology integration. They also start from choosing suitable sources and adapt them to the desired ontology. Suitable ontologies are therefore ontologies that are more easily adapted, e.g., using less operations. Pinto *et al.* identified several sub-processes in ontology integration [21]; reusing

an ontology involves translation, rearrangement, correction and extension of an ontology.

We observe that approaches to ontology reuse mentioned above often focus on integrating different ontologies to create a new ontology rather then presenting the user fact types he can reuse in the ontology he's building. This observation holds for others in that same community. Uschold *et al.* stated that when one decides to reuse ontologies, different sources will often be compared, from which a few will form the basis of the new ontology [28,29]. Ontology integration, for which an survey is given by Euzenat and Shvaiko [11] suffers from the same problem.

5 Methodology

In this section we present our approach to ontology reuse, where we consider different aspects of knowledge, moving into the field of semiotics. Traditional semiotics distinguishes syntactics, semantics and pragmatics; dealing with the structures, meanings and usage of representations, respectively. In an attempt to provide a definition for information systems, Falkenberg *et al.* find that these aspects alone do not suffice [12]. They found that defining "information" proved to be difficult [24,12], for which Stamper found a solution [24] by seeing information as signs and to define the different aspects of these signs based on the different operations you can do upon them. His research into the operational definition of signs has led to the addition of three new views on signs (physics, empirics, and the social world), resulting in Stamper's *semiotic ladder* (see Fig. 1).

More concretely: the physical properties of representations are for instance the bits in hardware. Closely related, but more abstract in nature, are the empirical characteristics of representations, such as the pattern of symbols, the "entropy", etc. These two aspects are considered on a lower level than the syntactic, semantic and pragmatic levels. All these aspects inhabit a world in which persons interact, agreeing or disagreeing about meanings, expressing beliefs, entering into commitments, etc. In other words, there is also a social angle to the use of representations [12]. In order to understand which aspect or aspects one discusses it is important to define these aspects (or views, or representations) as semiotic levels. De Leenheer [8] applied the different semiotic levels defined by Falkenberg *et al.* [12] on ontologies; moving from information systems to the so called "open information systems". This application is as follows [8]:

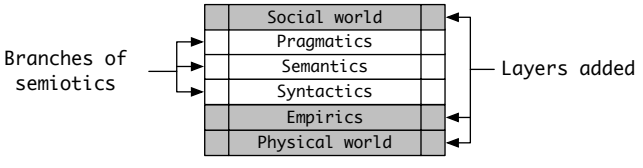

Fig. 1. Stamper's "semiotic ladder"

- *Social world*: A stable version of an ontology is an agreement on a symbolic system, which is a single version of the truth for the time being within and between organisations and their participants [8]. A stable version of an ontology can be therefore considered as a semiotic contract. It is socially accepted because it enables communication that ultimately brings added value.
- *Pragmatics* considers the use of meaningful combinations of representations for performing meaningful communication within the context of a common goal. The pragmatic level of a representation would therefore be the extent to which it supports meaningful actions that makes the community thrive in achieving their goals.
- *Semantics* considers the things in the domain the concept representation approximately refers to by using attributes and rules.
- *Syntactics* constitutes the structure using symbols. These symbols should enable interpretation, but also needs to be reusable while allowing disambiguation as well.
- *empirics*: Because subjectivity and variability in a natural language create conflicts that need to be negotiated about, they are essential prerequisites for patterns to emerge. Through repeated meaning evolution rounds, these patterns could become stable and reusable, providing empirically proven building blocks.
- *Physical world*: Signals, traces, physical distinctions, hardware, physical tokens, component density, speeds, economics, laws of nature, etc. We can basically categorize the means to make ontologies and information operational in this category.

Now that the different levels or aspects of an open information system are defined [8], we present our framework for retrieving possibly interesting lexons that users can reuse, by defining heuristics that reside in one of these levels.

Creating or editing a commitment would require some sort of interactive mode for users to introduce their input, knowledge or decisions into the systems. This manipulation is usually done on the client-side, using the *DOGMA Studio Workbench*[1], resulting in a series of operations that are sent to the server. We propose that users can build a "Lexon Suggester" by picking one or more heuristics and assigning them weights. Weights represent the importance given by the user to that particular heuristic. These heuristics are classified according to the semiotic levels of an open information system, since that ladder captures all of such a system's aspects. Since user can define an almost unlimited set of combinations of heuristics and weights[2], he can nuanciate the outcome of one heuristic with another. Not only does the content of the user evolve, but also the vast amount of data on a remote server, namely the *DOGMA Server*. Some heuristics give results depending on what can be found on the server, so they should update their results in regular intervals since the remote data evolves as well. Fig. 2

[1] http://starlab.vub.ac.be/website/dogmastudio

[2] By default, the weight of each heuristic is equal. Exploring what combinations of heuristics and their weights will be explored in the future, see Section 8.

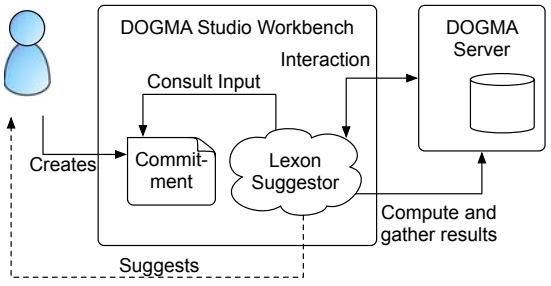

Fig. 2. The Lexon Suggester's position between user, client and server

illustrates the role of the Lexon Suggester between the user, the Workbench and the DOGMA Server.

The skeleton of a heuristic is given in Algorithm 1 the methods called in this procedure are: (i) *setup*, for setting up heuristic specific parameters (ii) *processLexons* to retrieve the lexons from a specific source (contexts, commitments, votes) and assign them a score, (iii) *normalizeScores* to normalize the scores, and (iv) *filter* to remove the lexons which do not fulfill the necessary requirements. Filtering lexons can only happen after all scores have been obtained and normalized.

Algorithm 1. Aggregating candidate lexons, given L, the lexons in the current editor and h a heuristic

```
1: C = ∅ {New list of candidates}
2: setup()
3: lexonscores ← processLexons()
4: normalizeScores(lexonscores)
5: filter(lexonscores)
6: for all lexon in lexonscores do
7:     c ← newCandidate()
8:     c.score ← getLexonScore(lexon)
9:     c.lexon ← lexon
10:    C ← c {Add new candidate to list}
11: end for
12: return C
```

We now describe the heuristics we have defined for five of the six semiotic levels, since we have not defined one for the physical level.

5.1 Empirical Level

For patterns to emerge, subjectivity and variability are essential prerequisites in a language. Because lexons are expressed in a natural language, the subjectivity

and variability of the natural language can trigger interest in a user. Users might click on terms and lexons, while browsing the Lexon Base. Lexons are graphically represented using NORM Trees [27], which are undirected rooted trees constructed by concatenating lexons. Parts of a NORM Tree can be expended or collapsed by respectively showing or hiding all the lexons around a certain term. Expanding a term in a NORM Tree means that the user wants to explore these lexons either to learn more about that term or to traverse the tree. Either way, that particular term triggered an interest. This information is useful and we therefore store it to define a heuristic giving scores to lexons based on that information. This heuristic, which does not depend on the input of the editor, will be useful as a deciding factor when choosing between two lexons.

5.2 Syntactical Level

By only looking at the lexical representation of terms, we do face the problem of homonyms and synonyms. This problem will be tackled when one takes the Concept Definition Server into account or uses an external dictionary such as WordNet [13], which levers the terms to the semantic level. A motivation to use this method, however, is that one can never be sure that all terms are articulated. Articulation means linking a term, together with its context, to a concept in the Concept Definition Server [7]. Articulation is the process used in DOGMA to link semantics to a term.

On the syntactic level we defined two approaches. The first approach is to search for all lexons where either the head-term or tail-term has an exact match with one of the terms found in the commitment the user is editing. The second approach is to use string metrics, a method for finding similar strings. String metrics are a class of textual based metrics resulting in a similarity or dissimilarity (distance) score between two pairs of text strings for approximate matching or comparison and in fuzzy string searching. For example the strings "Neighbour" and "Neighbor" can be considered to a degree similar. A string metric provides a floating point number indicating an heuristic-specific indication of similarity [5].

5.3 Semantical Level

By using WordNet [13] to lift terms in lexons to the semantic level, we are able to find relevant lexons using WordNet's linguistic relationships. WordNet is a semantic lexicon for the English language. It groups English words into sets of synonyms called synsets, provides general definitions, and records the various semantic relations between these synonym sets. WordNet also defines relations between words such as *hyponyms* and *hypernyms*. In linguistics, a hyponym is a word or phrase whose semantic range is included within that of another word. For example, steel, iron, silver, and charcoal are all hyponyms of gray; and gray is a hyponym of color. Computer science often terms this relationship an *is-a relationship*. For example, *Gray is a color* can be used to describe the hyponymic relationship between gray and color. Hypernyms denote a word, usually somewhat vague and broad in meaning, that other more specific words fall under or

are fairly encompassed by. Here, gray is a hypernym of steel. These relationships can be interpreted since their semantics are known. WordNet can thus be used to find relevant lexons on a semantic level. The user is able to look for synonyms, hypernyms, hyponyms or a combination of one of the tree and also the score assigned to each of those relations.

5.4 Pragmatical Level

This level considers the use of meaningful combinations of commitments for performing meaningful communication within the context of a common goal. The pragmatic level of a commitment would therefore be the extent to which it supports meaningful actions that makes the community thrive in achieving their goals. At this level we define two approaches; the first approach is to count the number of commitments to a lexon. A user is then able to use this heuristic and give it a certain threshold, and collect all lexons satisfying that threshold. This method does not take as input the lexons found in the editor, but combined with another heuristics provides a powerful means of ordering the relevance of a lexon. A syntactic heuristic might propose two different lexons that denote the same, but happen to both exist at the same time (syntactic heterogeneity) in two different contexts, but one of these lexons has been more committed to. This approach could then rank this lexon higher and therefore suggest the user to prefer this lexon over the other.

The second approach is to discover patterns that emerge from the different commitments or contexts. Discovering patterns that describe associated features in data is the goal of a data-mining task called *Association Analysis* [25]. One of the processes in association analysis gathers frequent itemsets, bags of items or facts that are often seen together. We use these frequent sets to find relevant lexons for a user. Whenever a user has lexons in his editor appearing in of the frequent itemsets, the lexons within the itemset are proposed to the user.

5.5 Social Level

Ontologies are not only a "contract" or an "agreement" between organizations, they also express a certain belief in a conceptualization. In Section 5.4 we already defined a heuristic looking at the number of commitments a lexon has. We now want to nuancate "belief in a lexon" and made a distinction between rating a lexon and committing to a lexon. This allows us to define a heuristic that takes into account the number of votes a lexon has. This will be a powerful means of ordering the relevance of a lexon when combined with another heuristic.

The motivation for this distinction is as follows: commitments are made for a certain application and committing to a lexon implies the acknowledgement of a lexon. The inverse, however, is not true; a user can find a set of lexons he does not need it for his commitment, but wants to acknowledge. We therefore add a voting system within DOGMA in which people can express their belief in a set of lexons by "promoting" them. Commitments, and their heuristics, belong

to the pragmatic level since they are used to enable meaningful communication. Rating lexons resides at the social level.

Another approach we took is to look at organizations and their participants; they either have a common *goal* or reside in the same domain. This can be exploited to reuse lexons resulted from reaching that common goal or projects within a common domain. We therefore defined a heuristic that gathers lexons from commitments concerning the same goal and/or the same domain. De Leenheer [6,8] defines these goals as *semantic interoperability requirements*, being the conceptions that need to be represented in the shared ontology in order to enable or restate semantic interoperability.

6 Tool

We applied this methodology to the DOGMA approach and implemented a tool supporting this methodology that we added to the already existing tool suite supporting DOGMA MESS, called DOGMA Studio Workbench, developed at STARLab and over the years extended with a graphical editor [27] and operation based ontology engineering and versioning [9]. It also provides the Knowledge Engineer with advanced perspective management, as well as support for detailed semantic constraints.

The current version of the Workbench supports: (i) connecting to the Lexon Base, (ii) textual and graphical browsing of the Lexon Base, (iii) input of lexons using a simple textual editor, (iv) support for browsing and editing the Concept Definition Server, (v) a set of tools for context-based ontology engineering and evolution and (vi) detecting and providing solutions for conflicts between different perspectives. These functions, available in the different existing plug-ins, allow knowledge engineers to create ontologies and commitments, but still a danger exists for them to reinvent the wheel by introducing the same fact types over and over again. What was missing was a mechanism that presents them with relevant fact types they can reuse.

The Lexon Suggester Suite contains two plug-ins: *The Suggester* and *The Controller*. The first is responsible for configuring the different heuristics and combining the results. The latter functions as a bridge between the different UI components and the server. Fig. 3 shows the Suggester within the Workbench. To situate the Suggester, we briefly explain each of the plug-ins depicted in Fig. 3 and how they interact.

The *Context Treeview* (1) lists all the contexts available on DOGMA Server. Each directory lists the terms found within that concept. Such a term, when clicked, will appears in the *Lexon Base Browser* as the root of a NORM-tree [27]. Clicking on a context displays its lexons in the *Lexon Viewer*. The *Suggester* (2) is an editor for a *Lexon Suggester configuration file* that can be configured by a user. It will look at the current content of, for instance, a commitment in the *Commitment Editor* [27] to ask relevant lexons from the Lexon Reuse framework on the DOGMA Server. These lexons can be dragged from the list of the Lexon Suggester and dropped onto the *Commitment Editor*. The *Commitment Editor*

(3) serves as a tool to create a commitment, graphically represented by a NORM-tree. The framework presented in this paper will look for relevant lexons that can be reused for such a commitment.

The *Navigator* (5) is a standard Eclipse IDE plug-in that handles the visualization of projects, their files and their structure. It is used to create and manage the commitment files and Lexon Suggester configuration files. The *Lexon Base Browser* (4) serves as a graphical tool to explore a context using NORM-trees and the *Lexon Viewer* (6) is a simple table view that displays all the lexons of a given context.

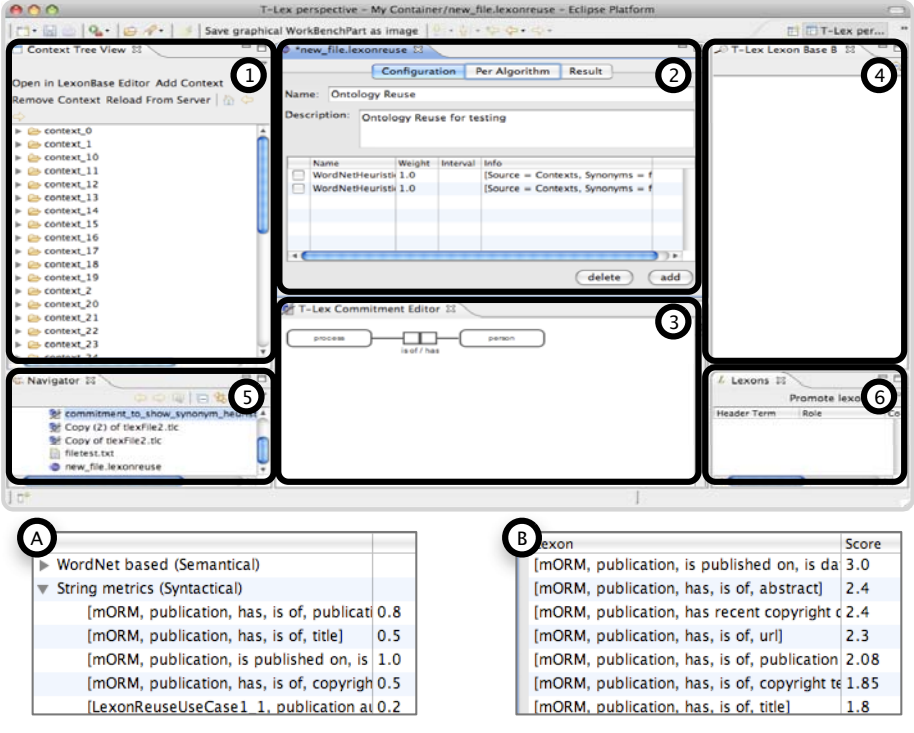

Fig. 3. DOGMA Studio Workbench with the Lexon Suggester

Users are able to configure a heuristic before adding them to the list of heuristics. Each heuristic can be given a weight and an interval that for heuristic to re-consult the DOGMA Server. When no interval is defined, the heuristic will be applied only once. This is interesting for heuristics that do not depend on input. The client is updated after fetching the results from the server of a particular heuristic. There are two views for presenting the results. One view shows the lexons grouped by heuristic, see Fig. 3(A) and another groups the lexons all together, where they can be sorted lexicographically or by their weighted means, Fig. 3(B). From each view, a lexon can be dragged onto the commitment editor.

7 An Illustrative Example

To illustrate the methodology in Section 5, we simulate the perspective rendering of the concept *publication* using the tools described in Section 6, where five participants have already given their perspective as a running example to show what the result of a heuristic (or a combination of heuristics) might be. The perspective of each participant can be found in Appendix.

Assuming we have created a blank commitment file; we first start by choosing an appropriate term from an existing context (γ_1) in the Context Tree View to display the NORM tree beginning with that term. We select a lexon from that context as well as add a new lexon, as seen in the commitment below. The new lexon will be added to a new context[3].

$\langle \gamma_1$, publication, has, is of, abstract$\rangle$
$\langle \gamma_5$, publication, is published on, is date of publication of, datum$\rangle$

We create a new Lexon Suggester (see Fig. 4(A)), to automate the process of finding reusable lexons. We can also open an existing Lexon Suggester, since we store its configuration in a file. We can now add and remove heuristics from the Lexon Suggester. To add a heuristic, we first choose a type of heuristic (based on the semiotic levels) before choosing the heuristic itself and setting their parameters (intervals, thresholds, ...) as seen in Fig. 4(B).

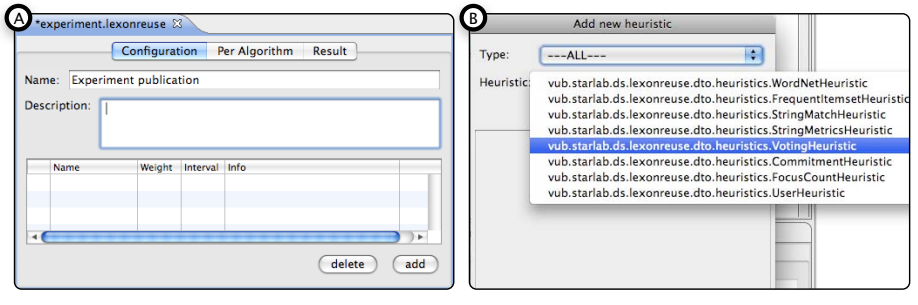

Fig. 4. (A) The new Lexon Suggester and (B) adding a new heuristic

7.1 Exact String Matching and String Metrics

We can choose between looking at lexons in contexts or commitments. Since a commitment is a subset of lexons from the lexon base with additional constraints, it is possible that the same lexon resides in more than one commitment. In our example, using exact string matching on commitments returned 11 lexons, whereas

[3] A context-label should denote the source form where that lexon is extracted form (a document, a domain expert, ...). To enhance readability, however, we chose to denote them as γ_1, γ_2, γ_3, ...

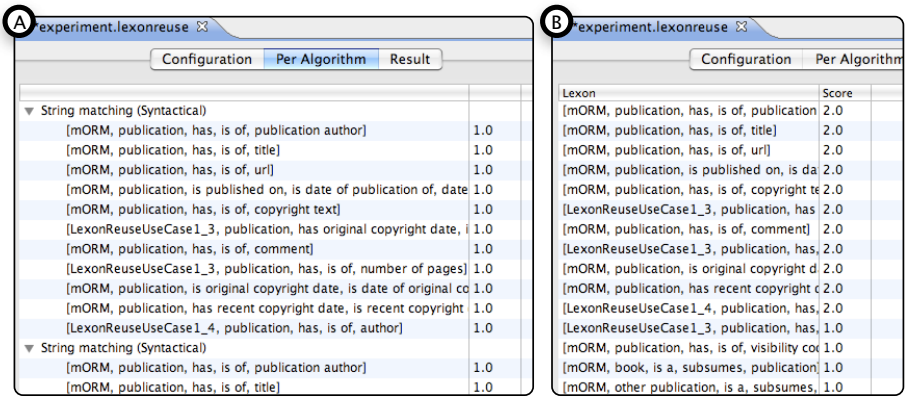

Fig. 5. Using string matching on contexts and commitments: (A) Results per algorithm and (B) Combined result

on contexts 21 (see Fig. 5(A)). The first method provides information about the commitment to a lexon whereas the latter provides means for discovering unused lexons. The combination of the two is powerful since the result from the contexts "confirm" the lexons found in the commitments by augmenting their score, ranking them higher than the lexons not used in any commitment yet (see Fig. 5(B)).

String metrics return similar results; the extra parameters available in this heuristic are a string metric and a threshold. This heuristic will return lexons with terms that look similar and might therefore be related. In our example "date" resembles "datum" well enough to be taken into account. The heuristic found 14 lexons in commitments and 29 lexons in contexts. These results are obtained using the Level2JaroWinkler [5] metric with a threshold of 90% similarity.

7.2 Number of Commitments, Voting and Focus Counts

In Section 5 we discussed that using heuristics such as the commitment heuristic (Section 5.4), voting heuristic (Section 5.5) and the heuristic counting the focus around a lexon (Section 5.1) can be used as a deciding factor for ranking lexons. Since these three heuristics can be applied in a similar way, we only demonstrate the voting-heuristic.

In our example, two parties happen to have introduced the same lexon in their newly created context: $\langle \gamma_3$, publication, has, is of, pages$\rangle$ and $\langle \gamma_4$, publication, has, is of, pages$\rangle$. These two lexons have not been used in any commitment. Heuristics such as the string matching heuristic or string metrics heuristic will give these two lexons an equal score.

Assume that two users have voted for two lexons in the workbench. One user voted for $\langle \gamma_4$, publication, has, is of, pages$\rangle$ and the other for $\langle \gamma_4$, publication, has, is

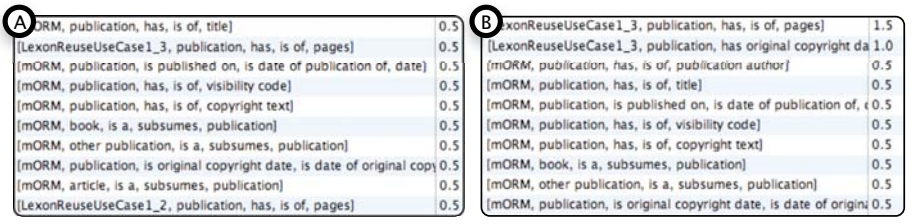

Fig. 6. (A) Merely using string matching might make decisions difficult (B) Combining string matching with voting

of, pages⟩ and ⟨γ_4, publication, has original copyright date, is date of original copyright of, date⟩. Voting for a set of lexons is either done on a selection in the Lexon Viewer, or on a selection in the NORM tree of the Lexon Base Browser. Such a set of lexons that received a vote has a creator and a list of supporters, which are people who have voted on the same set.

We select a string match heuristic on contexts and sees that both lexons are returned with an equal score, as seen in Fig. 6(A). Fig. 6(B) then illustrates how the voting-heuristic acts as a deciding factor.

7.3 Frequent Item Sets

We are also interested in retrieving lexons frequently appearing together. In Section 5 we proposed a heuristic looking at maximal frequent itemsets, where the items are lexons. In the example, the user wants to look for sets of lexons that appear in more than one commitment. Fig. 7(A) show the maximal frequent "lexonsets" that appear in 60% of the commitments. In this example we add the heuristic with a 60% support count and 8 lexons are returned. The heuristic unified the two maximal frequent lexonsets, as seen in Fig. 7(B) and gives a higher score to the lexons that appear in more than one set.

7.4 WordNet

This heuristic helps the user to find lexons where there might a semantical relation between terms. This is useful if terms in lexons are not articulated, which means they do not point a specific concept. However, this heuristic's disadvantage is ambiguity. Since WordNet might have more than one meaning for a word, finding appropriate hypernyms and hyponyms might be difficult.

Take for instance the word "date", is has at least three meanings[4], each with a hyponym: (1) meeting arranged in advance → rendez-vous (2) particular but unspecified point in time → point in time and (3) a participant in a date → escort.

[4] These can be found by looking for "date" in WordNet Online `http://wordnetweb.princeton.edu/perl/webwn`

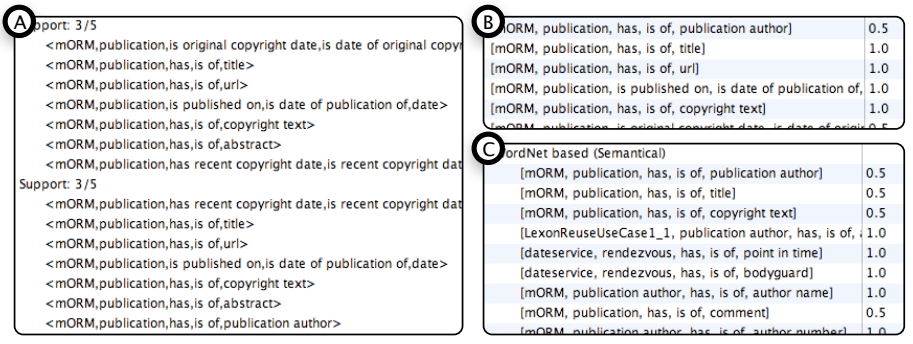

Fig. 7. (A) Maximal frequent lexon-sets in the 5 commitments with a 60% support count (B) Result of frequent itemset heuristic (C) Result of the WordNet-Heuristic

The heuristic will find these relations and present the user with lexons that contains the hyponym. We have created a commitment for a "dating service" containing the following lexons: ⟨dateservice, rendezvous, has, is of, point in time⟩ and ⟨dateservice, rendezvous, has, is of, escort⟩ and in the commitment we are creating we added a concept "datum". Since both terms are, in some sense, hyponyms of "datum", the lexon gets assigned a rather high score (see Fig. 7(C)).

8 Conclusions and Future Work

In this paper, we have presented our approach to ontology reuse for a user to render their perspective or commitment. We chose the DOGMA MESS approach for ontology engineering and to further improve this approach we developed an extra tool that helps users to reuse existing lexons, based on heuristics that reside on different semiotic levels. We claim this will reduce heterogeneity between different perspectives and commitments, meaning negotiation will be facilitated, speeding up and smoothing the iterative process of DOGMA MESS. To prove this claim we need to conduct a proper validation and evaluation in the future.

In this paper we only treated lexons and not the axiomatization in the different commitments, the so called "domain rules". Support for reuse on domain rules would increase the approach's usefulness and will be worthwhile investigating in the future.

We also defined a few of the possible many heuristics and in an attempt to add a pragmatic and a social layer, we could have proposed additional heuristics on these levels to exploit the user's behavior and creativity in DOGMA MESS. In this paper, we added a voting mechanism within DOGMA MESS to rate lexons, other heuristics on the social level might involve, for example, the discussion around a lexon on a wiki. Exploring the different heuristics, especially at these two levels, are part of our future work. Another subject to explore in the future is to examine what combinations of heuristics and weights work best, as we have showed that some heuristics provide a deciding factor when another heuristic gave equal scores to some lexons.

Acknowledgments

We would like to thank Johannes Peeters and Ben Maene, both students at the VUB, with whom we have created the lexons used for the illustrative example of our tool and methodology.

References

1. Bontas, E.P., Mochol, M.: Towards a cost estimation model for ontology engineering. In: Eckstein, R., Tolksdorf, R. (eds.) Berliner XML Tage, pp. 153–160 (2005)
2. Bontas, E.P., Mochol, M.: Towards a reuse-oriented methodology for ontology engineering. In: Proc. of 7th International Conference on Terminology and Knowledge Engineering, TKE 2005 (2005)
3. Bontas, E.P., Tempich, C., Sure, Y.: Ontocom: A cost estimation model for ontology engineering. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 625–639. Springer, Heidelberg (2006)
4. Christiaens, S., De Leenheer, P., de Moor, A., Meersman, R.: Business use case: Ontologising competencies in an interorganisational setting. In: Hepp, M., De Leenheer, P., de Moor, A., Sure, Y. (eds.) Ontology Management for the Semantic Web, Semantic Web Services, and Business Applications, from Semantic Web and Beyond: Computing for Human Experience. Springer, Heidelberg (2008)
5. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string metrics for matching names and records. In: Data Cleaning Workshop in Conjunction with KDD (2003)
6. De Leenheer, P., Christiaens, S., Meersman, R.: Business semantics management: a case study for competency-centric HRM. Journal of Computers For Industry (2009)
7. De Leenheer, P., de Moor, A., Meersman, R.: Context dependency management in ontology engineering: a formal approach. In: Spaccapietra, S., Atzeni, P., Fages, F., Hacid, M.-S., Kifer, M., Mylopoulos, J., Pernici, B., Shvaiko, P., Trujillo, J., Zaihrayeu, I. (eds.) Journal on Data Semantics VIII. LNCS, vol. 4380, pp. 26–56. Springer, Heidelberg (2007)
8. De Leenheer, P.: On Community-based Ontology Evolution. In: Manuscript, Vrije Universiteit Brussel (2008)
9. De Leenheer, P., Debruyne, C.: DOGMA-MESS: A tool for factoriented collaborative ontology evolution. In: On the Move to Meaningful Internet Systems 2008: ORM (ORM 2008), Monterrey, Mexico. LNCS. Springer, Heidelberg (2008)
10. de Moor, A., De Leenheer, P., Meersman, R.: DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) ICCS 2006. LNCS (LNAI), vol. 4068, pp. 189–203. Springer, Heidelberg (2006)
11. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)
12. Falkenberg, E.D., Hesse, W., Lindgreen, P., Nilsson, B.E., Oei, J.L.H., Rolland, C., Stamper, R.K., Assche, F.J.M.V., Verrijn-Stuart, A.A., Voss, K.: Frisco: A framework of information system concepts. Technical report, The IFIP WG 8. 1 Task Group FRISCO (1996)
13. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)

14. Fernández-López, M., Gómez-Pérez, A.: Overview and analysis of methodologies for building ontologies. Knowl. Eng. Rev. 17(2), 129–156 (2002)
15. Gruber, T.R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In: Guarino, N., Poli, R. (eds.) Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers, Deventer (1993)
16. Guarino, N., Giaretta, P.: Ontologies and knowledge bases: Towards a terminological clarification. In: Mars, N.J.I. (ed.) Towards Very Large Knowledge Bases. IOS Press, Amsterdam (1995)
17. Halpin, T.A.: Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design. Morgan Kaufmann, San Francisco (2001)
18. Jarrar, M., Meersman, R.: Formal ontology engineering in the DOGMA approach. In: Meersman, R., Tari, Z., et al. (eds.) CoopIS 2002, DOA 2002, and ODBASE 2002. LNCS, vol. 2519, pp. 1238–1254. Springer, Heidelberg (2002)
19. Meersman, R.: Semantic web and ontologies: Playtime or business at the last frontier in computing? In: NSF-EU Workshop on Database and Information Systems Research for Semantic Web and Enterprises, NSF-EU, pp. 61–67 (2002)
20. Meersman, R.: Semantics ontology tools in information system design. In: Raś, Z., Zemankova, M. (eds.) ISMIS 1999. LNCS, vol. 1609. Springer, Heidelberg (1999)
21. Pinto, H.S., Peralta, D.N., Mamede, N.J.: Using protege-2000 in reuse processes. In: Proceedings of the OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management EKAW (2002)
22. Pinto, H.S., Jo a, P.M.: A methodology for ontology integration. In: K-CAP 2001: Proceedings of the 1st international conference on Knowledge capture, pp. 131–138. ACM, New York (2001)
23. Spyns, P., Meersman, R., Jarrar, M.: Data modelling versus ontology engineering. SIGMOD Record Special Issue 31 (4), 12–17 (2002)
24. Stamper, R.: Signs, information, norms and systems. In: Holmqvist, B., Andersen, P., Klein, H., Posner, R. (eds.) Signs at Work: Semiosis and Information Processing in Organisations, De Gruyter, Berlin (1996)
25. Tan, P., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Reading (2005)
26. Trog, D., Tang, Y., Meersman, R.: Towards ontological commitments with Ω-ridl markup language. In: Paschke, A., Biletskiy, Y. (eds.) RuleML 2007. LNCS, vol. 4824, pp. 92–106. Springer, Heidelberg (2007)
27. Trog, D., Vereecken, J., Christiaens, S., Leenheer, P.D., Meersman, R.: T-lex: A role-based ontology engineering tool. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1191–1200. Springer, Heidelberg (2006)
28. Uschold, M., King, M.: Towards a methodology for building ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI 1995 (1995)
29. Uschold, M., Healy, M., Williamson, K., Clark, P., Woods, S.: Ontology reuse and application. In: Proceedings of the 1st International Conference on Formal Ontology in Information Systems(FOIS 1998), pp. 179–192. IOS Press, Amsterdam (1998)
30. Zhao, G., Meersman, R.: Architecting ontology for scalability and versatility. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3761, pp. 1605–1614. Springer, Heidelberg (2005)

Appendix: Perspectives for the Running Example

– Commitment$_1$
{⟨γ_1, publication, has, is of, title⟩ ⟨γ_1, publication, has, is of, publication author⟩
⟨γ_1, publication, is published on, is date of publication of, date⟩
⟨γ_1, publication, is original copyright date, is date of original copyright of, date⟩
⟨γ_1, publication, has recent copyright date, is recent copyright date of, date⟩
⟨γ_1, publication, has, is of, url⟩ ⟨γ_1, publication author, has, is of, author name⟩
⟨γ_1, publication, has, is of, abstract⟩ ⟨γ_1, publication, has, is of, copyright text⟩
⟨γ_1, publication author, has, is of, author number⟩}
– Commitment$_2$
{⟨γ_1, publication, has, is of, title⟩ ⟨γ_1, publication, has, is of, publication author⟩
⟨γ_1, publication, is published on, is date of publication of, date⟩
⟨γ_1, publication, is original copyright date, is date of original copyright of, date⟩
⟨γ_1, publication, has recent copyright date, is recent copyright date of, date⟩
⟨γ_1, publication, has, is of, url⟩ ⟨γ_2, publication author, has, is of, author name⟩
⟨γ_1, publication, has, is of, abstract⟩ ⟨γ_1, publication, has, is of, copyright text⟩}
– Commitment$_3$
{⟨γ_1, publication, has, is of, title⟩ ⟨γ_2, publication author, has, is of, author⟩
⟨γ_1, publication, is published on, is date of publication of, date⟩
⟨γ_1, publication, has, is of, url⟩ ⟨γ_1, publication, has, is of, publication author⟩
⟨γ_1, publication, has, is of, abstract⟩ ⟨γ_1, publication, has, is of, copyright text⟩}
– Commitment$_4$
{⟨γ_1, publication, has, is of, title⟩ ⟨γ_3, publication, has, is of, number of pages⟩
⟨γ_1, publication, is published on, is date of publication of, date⟩
⟨γ_3, publication,has original copyright date,is date of original copyright of, date⟩
⟨γ_1, publication, has recent copyright date, is recent copyright date of, date⟩
⟨γ_1, publication, has, is of, url⟩ ⟨γ_1, publication author, has, is of, author number⟩
⟨γ_1, publication, has, is of, abstract⟩ ⟨γ_1, publication, has, is of, publication author⟩
⟨γ_1, publication, has, is of, copyright text⟩
⟨γ_1, publication author, has, is of, author name⟩}
– Commitment$_5$
{⟨γ_1, publication, has, is of, title⟩ ⟨γ_1, publication, has, is of, abstract⟩
⟨γ_1, publication, is published on, is date of publication of, date⟩
⟨γ_1, publication, is original copyright date, is date of original copyright of, date⟩
⟨γ_1, publication, has recent copyright date, is recent copyright date of, date⟩
⟨γ_1, publication, has, is of, url⟩ ⟨γ_1, publication, has, is of, copyright text⟩
⟨γ_1, publication, has, is of, comment⟩ ⟨γ_4, publication, has, is of, author⟩}