

Semantic Interoperation of Information Systems by Evolving Ontologies through Formalized Social Processes

Christophe Debruyne and Robert Meersman

¹Semantics Technology & Applications Research Laboratory,
Vrije Universiteit Brussel, Brussels, Belgium
{chrdebru,meersman}@vub.ac.be

Abstract. For autonomously developed information systems to interoperate in a meaningful manner, ontologies capturing the intended semantics of that interoperation have to be developed by a community of stakeholders in those information systems. As the requirements of the ontology and the ontology itself evolve, so in general will the community, and vice versa. Ontology construction should thus be viewed as a complex activity leading to formalized semantic agreement involving various social processes within the community, and that may translate into a number of ontology evolution operators to be implemented. The *hybrid* ontologies that emerge in this way indeed need to support both the social agreement processes in the stakeholder communities and the eventual reasoning implemented in the information systems that are governed by these ontologies. In this paper, we discuss formal aspects of the social processes involved, a so-called fact-oriented methodology and formalism to structure and describe these, as well as certain relevant aspects of the communities in which they occur. We also report on a prototypical tool set that supports such a methodology, and on examples of some early experiments.

Keywords: ontology development, methodology, social process, business semantics management, fact-orientation, natural language.

1. Introduction

Ontologies are keystone technologies for the meaningful and efficient interoperation of information systems. Information systems on the Web are in general developed and maintained *autonomously*, which necessitates agreement to be negotiated between Web services. This, in turn, requires agreement between the stakeholders and designers on the semantics of the shared concepts involved. As a consequence, ontologies in general will *evolve* while such agreements are developed and finally put in place. These ontologies are approximations of a real world; in fact to the Web services involved, ontologies *are* the world. Ontologies represent an *externalization* of the semantics *outside* of the information system. The basic techniques and architecture for semantic interoperation is based on *annotation* (of an application system) and *reasoning* (about the concepts involved, in terms of the ontology).

From above it follows that the modeling of ontologies within a community of stakeholders and designers is a critical activity for the eventual success of interoperability. In this paper, we discuss the social processes involved, a methodology and

formalism to structure these and the communities in which they occur, and a prototypical tool set that supports such a methodology.

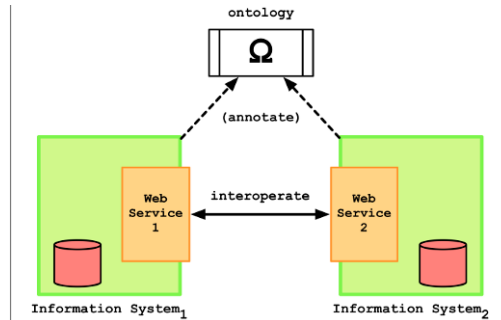


Fig. 1: Two autonomously developed information systems interoperating at runtime, exchanging messages via the application symbols annotated with the shared ontology.

Fundamental to our approach is the involvement of structured natural language as a vehicle to elicit useful and relevant concepts from community communication, and the mapping of these social processes to evolutionary processes in the emerging ontology. The formalism and language presented here are therefore “upstream” from the usual ontology languages such as RDF(S) and OWL and should not be confused with those; in fact it is relatively straightforward to compile the resulting/emerging ontologies into, for example, RDF(S) and OWL at any time.

One fundamental principle of all large system design is the so-called *separation of concerns* resulting in architectures that delegate respective functionalities to the stakeholders responsible for them. Examples are modules, etc. provided by the (generic) architecture of information systems driven by a database, largely separating the concern of basic data management from that of application development, the famous paradigm of *data independence*.

We reapply this principle in our approach by the rigorous separation in conceptualizations of “fact modeling” from all application-specific interpretations. It is this interpretation process (formally, of statements shared in the application system in terms of ontology concepts) that usually is called “reasoning” in the Semantic Web literature. However, there is little or no attention to such separation of concerns in the usual reasoning formalisms of Semantic Web in terms of Description Logic and its syntactical manifestations such as OWL and its dialects. In our approach, this interpretation is *exclusively* delegated to the mapping between application system and the “lexon base”¹ of the ontology. We shall call these mappings *ontological commitments* after Guarino [10], but we shall reify them in a well-defined manner suited to our formalism. Intuitively, our commitments select the facts needed, map application symbols to ontology concepts, and contain the rules and constraints, expressed in ontology terms, under which application symbols, relationships and business rules must be interpreted when they are to be shared with other autonomous systems. Those systems will share the concepts, but of course will have their own symbols, business rules, etc.

¹ We shall call the facts in the ontology lexons to distinguish the terminology from application context. See Section 3 for details.

This separation of concerns now allows a natural introduction of formalized social processes in goal-oriented communities such as exist in enterprises, professional networks, standardization groups, etc. and in fact in any “human agent” context for which agreement about facts is more efficient than reasoning from axioms. Note that nearly all data models for databases and business information systems were arrived at in this manner for the last 50 or so years.

And finally, as we shall argue in the next sections, this provides for a suitable and elegant context in which such business information systems can be made truly *open* by “lifting” their data models to an ontological level by widening the scope of the social processes. It follows that data schemas (e.g. defining a relational database) should not be seen as equal to ontologies. At best they will serve, after lifting by a community into a more “agreed” and “shared” form, as first approximations of one. In fact the same is true for any “conceptual schema” [1] that was used for designing an information system within one given enterprise. For more details on the distinction between a data model and an ontology, we refer to [18].

2. Related Work

Social interactions have been studied by observation in *mediawiki* talk pages by [21], which resulted in a (fairly limited) taxonomy of possible discussion items on a Wikipedia article. [17] also noted a correlation between the number of edits and the amount of discussion in an article. [17] extended this classification by using a larger dataset and added about 5 extra types. The goal of [17] was to create subclasses of `sioc:Post` so different types of discussion items can be easily accessed and mined upon. The SIOC Ontology² focuses on the integration of online community socialization and is used in conjunction with the FOAF³ vocabulary for expressing personal profile and social networking facts. In the context of a discussion, forum topics can range from conceptions that must be added to the ontology to meta-concept types that constitute the community meta-model itself.

Fact-oriented modeling, such as ORM [11] and NIAM [22], is a method for analyzing and creating conceptual schemas for information systems starting from (usually binary) relationships expressed as part of human-to-system communication. Using concepts and a language people are intended to readily understand, fact-oriented modeling helps ensuring the quality of a database application without caring about any implementation details of the database, including e.g. the grouping itself of linguistic concepts into records, relations, ... In fact-oriented approaches, every concept plays roles with other concepts, and those roles may be constrained. It is those constraints that allow the implementer of a database (or in fact an algorithm) to determine whether some linguistic concept becomes an entity or an attribute, or whether a role turns out to be an attribute relationship or not. This is different from other approaches such as (E)ER and UML, where these decisions are made at design time.

² <http://www.sioc-project.org/>

³ <http://xmlns.com/foaf/spec/>

3. DOGMA

In [14, 15] a formalism and methodology for ontology development called DOGMA⁴ was defined that illustrated and implemented these principles, now lifted to domain level from the mere enterprise system level. We first define a DOGMA *ontology description*. As indicated in the introduction, such descriptions must be seen as different from their eventual implementations, e.g. using RDF(S) and/or OWL. In the methodology and lifecycle of semantic systems the creation of DOGMA ontology descriptions belongs upstream from such implementation – although of course in many cases one will have to “mine” or elicit the required knowledge from existing information systems and their enterprise environments.

3.1. Towards Hybrid Ontology Descriptions

Definition 1. A DOGMA Ontology Description Ω is an ordered triple $\langle \Lambda, ci, K \rangle$ where Λ is a *lexon base*, i.e. a finite set of *lexons*. A lexon is an ordered 5-tuple $\langle \gamma, t_1, r_1, r_2, t_2 \rangle$ where $\gamma \in \Gamma$ is a context identifier, $t_1, t_2 \in T$ are *terms*, and $r_1, r_2 \in R$ are *role labels*. A lexon is a binary fact type that can be read in two directions: t_1 playing the role of r_1 on t_2 and t_2 playing the role of r_2 on t_1 . We omit here for simplicity the usual alphabets for constructing the elements of $T \cup R$. $ci: \Gamma \times T \rightarrow C$ is a partial function mapping pairs of context identifiers and terms to (unique) elements of C , a finite given set of *concepts*. The nature of these concepts is intentionally left unspecified but intuitively it is assumed that all users of an ontology described by Ω , i.e. sharing Λ and K , agree on the nature of all concepts in C . In a concrete way, within a context γ , $ci(\gamma, t)$ is the definition itself of the concept agreed by all such users. To emphasize this explicit agreement, we shall avoid to label concepts as such in our formalism, and assume they are “computed” by the community from the term labels. K is a finite set of ontological *commitments*. Each commitment is an ordered triple $\langle \sigma, \alpha, c \rangle$ where $\sigma \subseteq \Gamma$ is a selection of lexons, $\alpha: \Sigma \rightarrow T$ is a mapping called an *annotation* from the set Σ of application (information, system, database) symbols to terms, and c is a predicate over $T \cup R$ expressed in a suitable first-order language, not defined further in this paper but an example syntax named Ω -RIDL may be found in [19,20].

Context identifiers are pointers to a community, they can be a name, a URI to a website or even a URI to a document describing the community. To improve readability, we use names as a context identifier in the following examples. For example, the DOGMA ontology description might contain the following plausible lexons in its lexon base:

- $\langle VCard\ Community, VCard, with, of, Email\ Address \rangle$
- $\langle Vendor\ Community, Offer, with, of, Title \rangle$
- $\langle Vendor\ Community, Offer, contains, contained\ in, Product \rangle$
- $\langle Vendor\ Community, Offer, made\ by, makes\ Vendor \rangle$
- $\langle RFP\ Community, Request\ For\ Proposals, corresponds\ to, matches, Offer \rangle$

⁴ Developing Ontologies-Grounded Methodology and Applications

The function ci maps terms in those lexons to concepts, e.g. $ci(\text{Vendor Community}, \text{Offer})$ points to a URL of a concept definition in which all synonyms are centralized, e.g. $ci(\text{RFP Community}, \text{Offer})$ of a fact entered in the lexon base by a different community with overlapping concepts in their domains. These lexons are then used to construct commitments. **Fig. 2** depicts an example commitment. The characters in boldface are reserved words for creating the constraints and mappings. The underlined characters represent variables. For more details on the syntax of commitments, we refer to [19,20].

```

BEGIN SELECTION
  <Vendor Community, Offer, with, of, Title>
  <Vendor Community, Offer, contains, contained in, Product>
  <Vendor Community, Offer, made by, makes, Vendor>
  <Vendor Community, Vendor, located on, location of, Address> ...
END SELECTION
BEGIN CONSTRAINTS
  Offer contains at least 1 Product.
  Vendor located on exactly 1 unique Address. ...
END CONSTRAINTS
BEGIN MAPPING
  map "APP_OFF.TITLE" on Title of Offer.
  map "APP_VEN.ADDR" on Address location of Vendor. ...
END MAPPINGS

```

Fig. 2: Example of a commitment for a particular application showing pieces of the three parts: selection σ , constraints c and annotations (or application symbol mappings) α .

Note that the separation of concerns mentioned in the previous section is reflected here through the set of plausible facts in the lexon base on one side, and the constraints, rules, ... on a relevant selection of those lexons on the other. In fact there are *no* constraints or any other reasoning supports included in the lexon base, making for a so-called *light* ontology.

Also note this definition imparts a well-defined *hybrid aspect* on ontologies as they are to be resources shared among humans working in a community as well as among networked systems such as exist in the World Wide Web. As the “unique concept” property mentioned above informally and intuitively results from a community agreement, for the purpose of this paper we find it useful therefore to formalize a community precisely as such a context, and to name the resulting notion a *hybrid ontology* (see also [16]). We also introduce a special linguistic resource, called a *glossary*, recording and supporting all the social processes.

Definition 2. A Hybrid Ontology Description (HOD) is an ordered pair $H\Omega = \langle \Omega, G \rangle$ where Ω is a DOGMA ontology description where the contexts in Γ are labeled *communities* and G is a *glossary*, a finite set of functions either of the form $g_1 : \Gamma \times T \rightarrow \text{Gloss}$, the *Term Glossary* or of the form $g_2 : \Lambda \rightarrow \text{Gloss}$, the *Lexon Glossary*. Gloss is a set of linguistically interpretable objects. We shall write $G = G_1 \cup G_2$ if the distinction needs to be made explicit.

For example, given the DOGMA ontology description Ω from the previous example, a HOD can be constructed where G contains (among others):

- (*VCard, Email Address*), “The address of an email (system of world-wide electronic communication in which a user can compose a message at one terminal that can be regenerated at the recipient's terminal when the recipient logs in)”
- (*Vendor Community, Offer*), “Represents the public announcement by a vendor to provide a certain business function for a certain product or service to a specified target audience.”
- (*Vendor Community, Offer, contains, contained in, Products*), “Represents the relation of a product for sale being included in an offer.”

Note that in this paper, we shall not concern ourselves with the precise nature of the elements of Gloss. For the sake of simplicity and understanding it will be sufficient to think of Gloss as a set of natural language documents each providing an “explanation” for a term in T or a lexon in Λ adequate within a given community.

3.2. Glossaries

Glossaries turn out to require a fairly rich structure when to be deployed for the purpose of (hybrid) ontology engineering, as they are used to build agreements in communities about concepts. It is natural to associate them with concepts (in a DOGMA ontology description through the terms of lexons).

Definition 3. Given a HOD $H\Omega = \langle \Omega, G \rangle$, we call a glossary *coherent* if $\forall \gamma \in \Gamma, \forall \lambda = \langle \gamma, t_1, r_1, r_2, t_2 \rangle \in \Lambda : g_1(\gamma, t_1) \rightarrow g_2(\gamma, \lambda) \wedge g_1(\gamma, t_2) \rightarrow g_2(\gamma, \lambda)$. Where \rightarrow stands for “is subsumed by”, which is not a logic property, but a binary (linguistic) predicate on the set Gloss, intended to express that any community agreement on its first argument is implied by a community agreement about its second. (One way to implement such a predicate may be by simply listing its extension.)

Indeed, it would be undesirable to describe a relation between two terms if one or both terms playing the roles in that relation are not described themselves, meaning that their intended meaning has not yet been made explicit.

Definition 4. The glossary consistence property. We say that a hybrid ontology satisfies this property if for every two pairs $\langle \gamma_1, t_1 \rangle, \langle \gamma_2, t_2 \rangle \in \Gamma \times T$, if $g_1(\gamma_1, t_1) = g_1(\gamma_2, t_2)$ then $ci(\gamma_1, t_1) = ci(\gamma_2, t_2)$. The converse does not necessarily hold. In other words, if two terms in two communities point to exactly the same gloss, then they must refer to the same concept as well. For most purposes this condition is too limiting since often glosses will express “the same thing” without being textually identical. It suffices that the communities agree on their equivalence; this lead to the following definition.

Definition 5. Two term glosses are *community-equivalent* EQ_γ if that community agrees that the described terms refer to the same (abstract) concept. A similar definition may be given for lexon glosses; it is omitted here. Two term glosses are *term-equivalent* EQ_T if any two communities agree that a given term refers to the same concept for both. It is easy to see that EQ_γ and EQ_T define equivalence relations (reflexive, symmetric and transitive) on G . Again, implementation could be by listing (or e.g. logging) its extension.

Definition 6. Given a hybrid ontology description $H\Omega = \langle \Omega, G \rangle$ and communities $\gamma_1, \gamma_2 \in \Gamma$ and $t_1 \in T$, we say that community γ_2 *adopts* $\langle \gamma_1, t_1 \rangle$ when $gloss_1 = g_1(\gamma_1, t_1)$ and $gloss_2 = g_1(\gamma_2, t_1)$ are defined, and we have (i) $EQ_T(gloss_1, gloss_2)$, i.e. first “match” the two glosses; and (ii) $ci(\gamma_1, t_1) = ci(\gamma_2, t_1)$, i.e. agree that both concepts are equal.

In other words, γ_1 and γ_2 agree behind the concepts on their respective glosses (a symmetric condition) and γ_2 agrees to use t_1 as a term to refer to γ_1 's concept behind it (an asymmetric condition). **Fig. 3** shows an example of agreeing on the equivalence of two glosses that are synonyms inside the Business Semantics Glossary (see Section 5). If their synonym relation was not already established, it will be after this action.

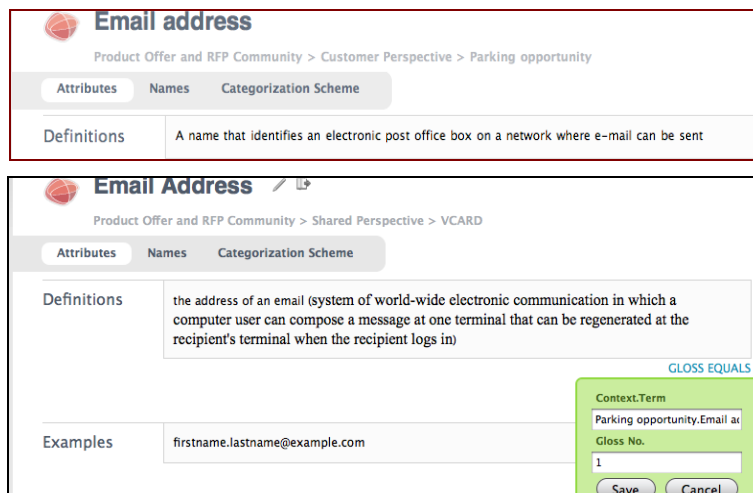


Fig. 3: Example of two glosses the two different communities deem to be the same. The first is a definition for the term “email address” in community interested in exchanging information about offers of parking spots. Their definition came from their domain expert. The second was the definition of a different community whose effort went to lifting vCard into a hybrid ontology description. The result of agreeing on the equality was a link stating that these two terms in those two communities were synonyms.

What the definition does *not* tell is *how* to achieve the shared understanding from those glosses that the concepts are the same. For this, one option is to let the communities involve their commitments to t_1 from their respective intended applications; in particular we need to study the reference structures for t_1 in those commitments.

Several guidelines on the construction of such glosses were given in [13]. While formally adequate to be useful in practice, we shall require more details of the *structure* (i.e. organizations) and the *processes* by which such a community achieves agreement about lexons and about the commitment of a specific information system. An immediate consequence is the requirement that a community viewed as a context must agree on unique concepts based on terms used in lexons. In this paper, we pro-

pose a formalism and methodological approach for such interactions of communities with the repositories of the knowledge they own.

The hybrid aspect is reflected in a dual perspective on the ontology Ω , and in particular on the glossary underpinning its lexons within a community: the community members agree on unique concepts based on glosses while systems interoperate (reason) based on the relationships (facts) that are deemed to exist between terms that refer to those same concepts.

Fig. 4 depicts a simplification of the iterative process involved. The Hybrid Ontology Description is used downstream (ref. **Fig. 5**) to generate a knowledge base, e.g. as RDF(S)-defined “storage structures” and constraints/rules implementing relevant commitments for the enterprise information systems to be served. The *co-evolution* of a community and its Hybrid Ontology Description is a natural consequence of this process. Externalization - identifying the key conceptual patterns that are relevant from the discussions [7]- results in a series of ontology evolution operators for the next version and (re-)internalization – by committing instance bases to the new version of the ontology [7] - changes the community’s composition: members depart when their goals differ too much from the common goal, or others join.

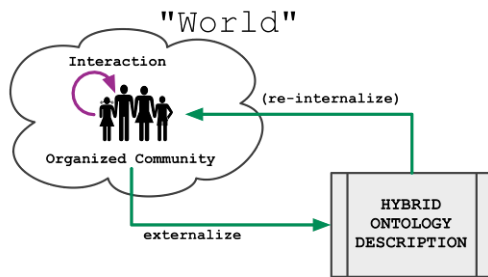


Fig. 4: Feedback loop between an organized community and an ontology. The interactions between the community result in ontology evolution operators applied to the ontology. These operators thus enact the externalization of the reflections of that community. The new ontology description, after a while, will be re-internalized as the community achieves (and discusses about) new insights.

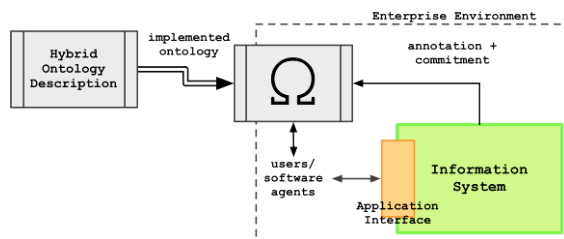


Fig. 5: “Downstream” usage of the Hybrid Ontology Description to implement the ontology, used for annotating the application symbols of an information system. Users and software agents “recognize” the kind of annotated data provided by the information system and the ontology.

Before describing the procedure of the methodology and the description of social processes in the next sections, we would like to note that communities in a collaborative ontology engineering methodology are *relevant only* if there are *two or more autonomously developed information systems* that need to interoperate. When there is only one information system, the semantics resulting from that community (even if the number of people is greater than one) are of that application. This would bring us no step further from going from a closed information system to open information systems.

3.3. Procedure of the Methodology

To achieve this we first define a set of operations on a community that are intended to reflect its member interaction with the “real world” and with each other. Then we “map” those operations onto a sequence of ontology evolution operators, as defined by [4]. It is essential to observe that the ontology description evolves *only* as the result of agreements, viz. actions performed in principle by multiple community members.

Every ontology evolution operation is subject to discussion before approval during the re-internalization phase of mentioned earlier. Members request certain changes under the form of those operators with a motivation. The status of each such request is initially “candidate”. Depending on the outcome of the discussion, the request is “approved”, “denied” or “postponed for future iteration”. The latter is useful when the community agrees that the request falls out the scope of the current iteration. Once proposed changes have been accepted and the community decides to go to a next version of the hybrid ontology description, all changes are translated into ontology evolution operators.

The next section starts from an existing collaborative ontology engineering approach built on top of DOGMA in which – for every iteration – a set of ontology engineering phases are identified. For every phase, the different social processes (e.g. the request to add a lexon and its discussion) and corresponding ontology evolution operators are identified (e.g. adding a lexon).

4. Social Processes in Ontology Engineering

These operations can be classified according to the different phases of a collaborative ontology engineering process. Business Semantics Management (BSM) [5], developed by Collibra⁵, is such a collaborative ontology engineering methodology drawing from best practices in ontology management [12] and ontology evolution [6]. The representation of business semantics is based on the DOGMA approach⁶. BSM consists of two complementary cycles: semantic reconciliation and semantic application (see **Fig. 6**) that each groups a number of activities.

Semantic Reconciliation is the first cycle of the methodology. In this phase, business semantics are modeled by extracting, refining, articulating and consolidating lexons from existing sources such as natural language descriptions, existing metadata, etc. Ultimately, this results in a number of consolidated language-neutral semantic patterns that are articulated with glosses (e.g. WordNet [9] word senses). These patterns are reusable for constructing various semantic applications. This process is supported by the Business Semantics Glossary, which will serve as basis for our prototype (see Section 5).

⁵ Collibra nv/sa, launched in 2008, is a spin-off company of the Vrije Universiteit Brussel that validates and further develops the technology of the DOGMA research project. <http://www.collibra.com/>

⁶ Recently, BSM adopted Semantics of Business Vocabulary and Business Rules (SBVR) [3], a recent OMG standard pushed by the business rule community and the fact-oriented modeling community (and fully compatible with DOGMA).

Semantic Application is the second cycle. During this cycle, existing information sources and services are committed to a selection of lexons, as explained earlier. In other words, a commitment creates a bidirectional link between the existing data sources and services and the business semantics that describe the information assets of an organization. The existing data itself is not moved nor touched.

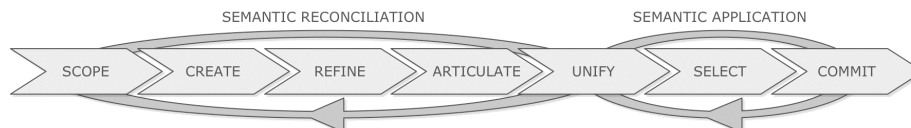





Fig. 6: Business Semantics Management consists of two complementary cycles: semantic reconciliation and semantic application. Both cycles communicate via the unify-activity.

4.1. Semantic Reconciliation and its Social Processes

Scope defines the borders of the current ontology engineering iteration and helps grounding discussions, preventing members of a community to go “off topic” on the current problem. The first iteration consists of the initial community of members representing autonomously developed information systems that need to interoperate. The discussions are grounded on the basis of a motivation and a problem scope. The motivation expresses why a HOD⁷ or an incremental extension of a HOD is needed. The scope of the next iteration limits the problem that needs to be tackled, e.g. the definition and relations around one particular term, to avoid divergent discussions. During this phase, members can also propose the use of relevant sources from which inspiration can be drawn upon. Sources of inspiration can be legacy database schemas, standards, documentation, etc. Before going on to the next phase, the relevance of these sources is agreed upon by the community. The social processes here are:

- *Creating the motivation* by a member with sufficient rights to start the process. In the case of bootstrapping the ontology, the creator is the community-leader (founder) or one of the founders of the community. The motivation is discussed by all members and refined in terms of the discussion until a consensus is achieved.
- *Scoping the problem*. Similarly to the motivation, a founder or a member with sufficient rights creates the scope. It is also the subject of discussion and refinements until a consensus is achieved.
- *Add (invite) member and remove member*. Members can join (or invited) to take part in refining the motivation and scope and the subsequent ontology engineering processes. When the goals of a community differ to greatly from the interest of one of the stakeholder, a member can decide to leave the community.
- *Proposing resources* that can be used to draw inspiration from. At any time, users can propose a list of resources that can be accepted or not. Examples of such resources can be the use of existing standards.

[Sojo-Duarte](#), 2011/04/10 18:29   

Can't we use an ISO standard for this? The ISO8601 Calenderdate standard for example..

⁷ In the case of a first iteration, the hybrid ontology description is initially empty.

Fig. 7: A proposition to use an existing standard to create lexons concerning dates.

From the creation phase onwards, every ontology operation is subject to discussion before approval as mentioned in the previous section. During the *create* process, lexons are generated from the collected sources in the scoping activity (e.g. documents or legacy database schemas). The operations in this phase are:

- *Request to add a lexon* $\langle \gamma, t_1, r_1, r_2, t_2 \rangle$ on which the community members as a whole accept or *refuse* the new lexon (see **Fig. 8**).
- *Request to add a constraint* such as internal and external mandatory constraints. The role or roles on which the constraint is put on have to exist in the ontology. Constraints on lexons are modeled using a predicate language such as Ω -RIDL [19].

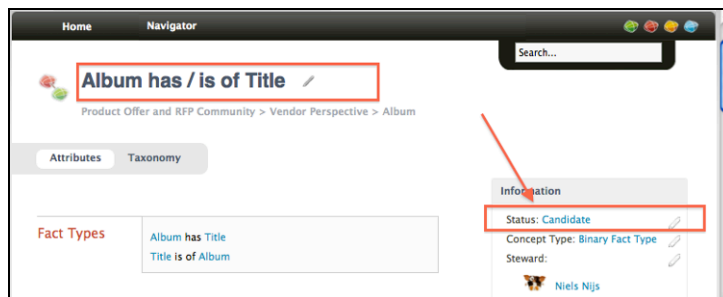


Fig. 8: The addition of a lexon is initially pending or “candidate”. In this example, a user added a lexon around a specific type of product (see Section 5). Users can discuss this lexon before accepting, denying or postponing the decision.

The *refine* process is used to refine existing lexons and constraints in the HOD. Actions that correspond in this phase are:

- *Request to remove lexon*. All constraints involving roles in this lexon will be affected as well as the glosses around this lexon.
- *Request to change the supertype of a term*. The class hierarchy is constructed with a lexon whose roles bear a special meaning (the taxonomic relation, e.g. with *role* and co-role “is a” and “subsumes”). When no supertype was defined, a taxonomic relation is added between the two terms. When such a relation already exists between the terms and another super terms, the existing taxonomic relation is removed before the creation of the new one.
- *Request to change “super lexon” of a lexon*. Which indicates that the population of a lexon is a subset of the more general lexon. A special operation, as it corresponds with a *subset* constraint on both roles of the two lexons [11,19].
- *Request to remove a constraint*.

Articulate is used to create informal meaning description, i.e. glosses, as extra documentation that can serve as anchoring points when stakeholders have used different terms for the same terms (synonyms). When descriptions are already available, e.g. in source documents, they can already be imported to speed up this process. The operations in this phase are:

- *Request to add gloss*, for a particular term $t \in T$ or lexon $\lambda \in \Lambda$ in a community $\gamma \in \Gamma$, or request to $g_1(\gamma, t) \leftarrow gloss$ or $g_2(\lambda) \leftarrow gloss$. Lexon glosses have to follow the glossary coherence property, the glosses of those terms have to be delivered or the request is ignored.
- Request to remove a gloss. When all glosses of a term are removed, the lexons around this term that are articulated lose their glosses as well. This impact is shown to the user. All EQ_γ around this gloss will be removed.
- Request to *change a gloss*. When accepted, all EQ_γ assertions around this gloss are removed and the other communities are notified to reconsider whether the changed gloss still EQ_γ theirs.
- *Request to add synonym*. A request to link terms across different communities, such that $ci(\gamma_1, t_1) = ci(\gamma_2, t_2)$ where $t_1, t_2 \in T$ and $\gamma_1, \gamma_2 \in \Gamma$. This action will make those two terms term-equivalent EQ_T .
- *Request to remove synonym*. This action happens when the definitions of both concepts diverge in such a way that they do not handle about the same anymore. Naturally, different lexons will emerge around those terms using other operations. The EQ_T assertion is thus removed.

At any given point, in order to achieve unification, discussions between users can take place. One can compare such discussions with posts and Web forums. By linking posts with their replies, one can create threads. An item in such discussion can be a trigger for an ontology operation or a task assigned to a person. A link is therefore kept between a task and an ontology operation if the post in question was the source of this action. For example, users who do not feel comfortable with formal ontology operators or do not know how to solve a problem might request an edit.

- *Request for edit*. A general request for edit (or solving a problem). For instance used when a member feels he has not enough responsibility over the concept to propose the actual changes.
- *Request for information*. Not to be confused with a request for edit for glosses, but rather a request for clarification. Such a request might result in a request for edit or as a request for an ontology operation.
- *Request for peer review*. A invitation to review some aspects of the ontology, e.g. inviting members of the community to give comments to certain proposed changes, even though they are not immediately affected by the concepts in question.
- *Request for help*, in contributing to certain aspects of the HOD.
- *Comment*. A comment to a post or a concept, a general class of posts that are not related to the other types of posts.
- *Reply*. All posts not belonging to any category in a thread.

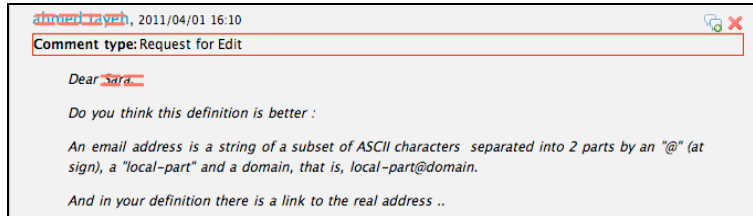


Fig. 9: Example of a request for edit, a user proposing a better definition and pointing to a problem in the existing definition.

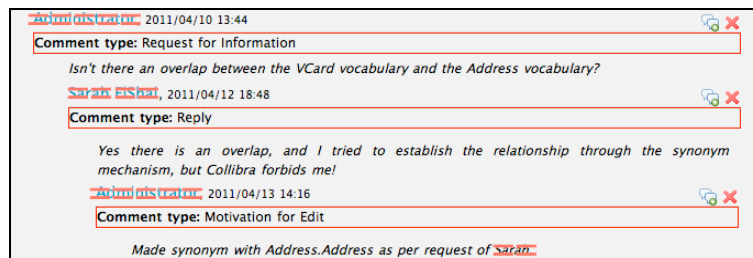


Fig. 10: Request for information. In this particular case, someone noted an overlap between two ontologies. Another member replied that an attempt was made to define synonyms, but was unable to do so. A third user then made the request instead.

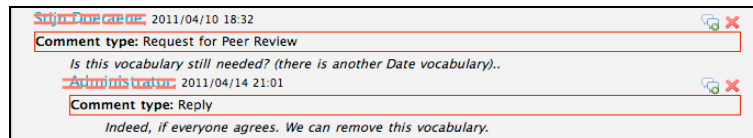


Fig. 11: Request for Peer Review. In the Business Semantics Glossary, ontologies are referred to as vocabularies, a term more “accessible” to users. In this particular case, a member of the community noticed that one of the ontologies was redundant.

4.2. Application commitments in the Feedback Loop

Commitments provide valuable information on which terms and lexons the different members of the community representing their organization commit to. This selection is exploited by informing those members when changes are requested (and occur) in the ontology as to stimulate discussion.

The mapping α in those commitments is furthermore used to delve into the annotated data in search for support or counterexamples for certain statements made by the community, e.g. to notify the community whether proposed constraint is true for all annotated information systems currently known in the community. This process will guide the community in its dialogue to achieve agreement. This is done by generating the necessary queries using the commitments of each of the applications, populating the lexons in the conceptual schema and then reason over the data in terms of lexon populations. This tool, called the Ω -DIPPER, has been described in [8] and the results will be reported elsewhere. **Fig. 12** extends **Fig. 4** and depicts the place of Ω -DIPPER in the feedback loop.

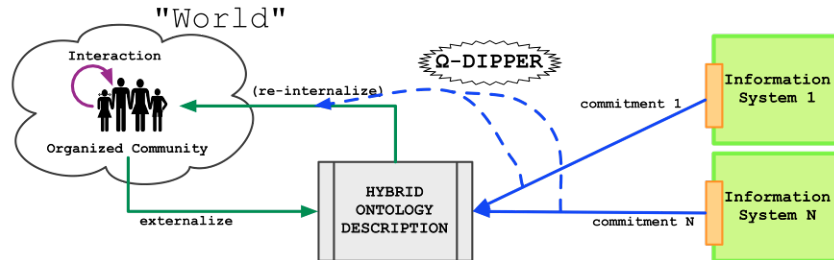


Fig. 12: Feedback loop from the ontologies to the community by not merely taking into account the lexons committed to by the application, but the data in the annotated organization information systems as well.

5. Tool

These results were implemented and tested in a web application supporting BSM called Business Semantics Glossary (BSG, see **Fig. 13**), also developed by Collibra nv/sa. BSG is based on the Wiki paradigm that is a proven technique for stakeholder collaboration. Governance models are built-in and user roles can be applied to distribute responsibilities and increase participation. **Fig. 13** also shows the lexons and gloss (here called “Description”) of that community around the concept of Home Address.

The interactions and resulting ontology operations were implemented using the flexible architecture provided by XWiki⁸, on which the BSG is based. The figures in the previous section depict some screen shots of the user interactions and resulting ontology evolution operators registered by the system. The tool was used by 45 users worked collaboratively on creating several ontologies concerning e-business (products, offers and requests for proposals). The 45 people represented 9 autonomously developed information systems: 4 request for proposals (RFP) systems and 5 vendor systems. The communities were: (i) a product community, which concerned everyone, (ii) an RFP community and (iii) a vendor community. As the experiment progressed, different communities evolved around general concepts such as Address, Dates, Contact Details, etc. in which some of the original 45 users are part of.

6. Conclusions

Ontologies are key in meaningful and efficient interoperation of autonomously developed and maintained information systems and come to be as a community effort. Those members in a community interact with each other, talking about the concepts and their relations in their natural languages. The result of those interactions is reflected in changes on the ontology. One important tool in agreeing on the meaning of concepts are glosses, natural language descriptions of those concepts.

In this paper, we presented a method for ontology engineering based on fact-oriented conceptual modeling techniques called DOGMA, in which the implementation details of an ontology has no importance at design time. This method returns

⁸ <http://www.xwiki.org/xwiki/bin/view/Main/>

ontology descriptions as an artifact, which will be used for the implementation of ontologies in languages such as RDF(S) and OWL in similar way database schema's were distilled from conceptual schemas in information systems modeling.

We extended those DOGMA ontology descriptions with hybrid aspects by giving the glosses a more prominent role in the agreement process and furthermore formalized the social processes involved in ontology engineering and how these are translated into ontology evolution operators. We proposed operators involving glosses that capture the agreement processes between members of a community and how glosses are used to identify the concepts. These describe flow from community to hybrid ontology description in the feedback loop. The use of the annotated information systems to provide feedback and steer the community in the engineering process will be reported elsewhere. These ontology evolution operators and the social interactions were implemented in a tool called Business Semantics Management, which supports the Business Semantic Management methodology built around DOGMA.

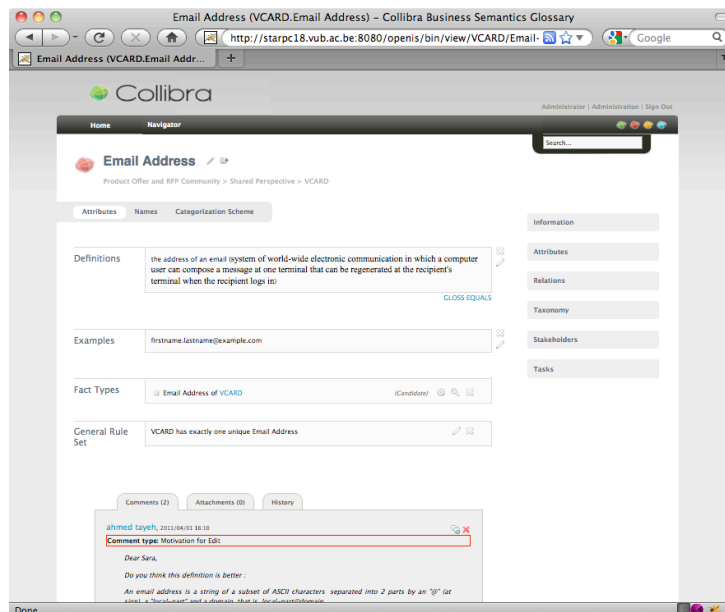


Fig. 13: Screenshot of the Business Semantics Glossary. Here we see the definition of Email Address in the VCARD ontology by the same community. Note the discussion between users around this concept.

7. References

1. Concepts and Terminology for the Conceptual Schema, Technical Report 9007, International Organization for Standardization, Geneva, Switzerland (1987)
2. OMG MOF, version 2.0. <http://www.omg.org/spec/MOF/2.0/> (2009)
3. OMG SBVR, version 1.0. <http://www.omg.org/spec/SBVR/1.0/> (2009)

4. De Leenheer, P., de Moor, A., Meersman, R.: Context dependency management in ontology engineering: a formal approach. *Journal of Data Semantics* 8 (2007) pp. 26–56 LNCS 4380, Springer.
5. De Leenheer, P., Christiaens, S., Meersman, R.: Business semantics management: A case study for competency-centric HRM. *Computers in Industry* 61(8) (2010) pp. 760–775
6. De Leenheer, P., Mens, T.: Ontology evolution. In: *Ontology Management: Hepp, M., De Leenheer, P., de Moor, A., Sure, Y. (eds.)*. Vol. 7 of *Semantic Web And Beyond Computing for Human Experience*. Springer (2008) pp. 131–176
7. De Leenheer, P., Debruyne C.: DOGMA-MESS: A Tool for Fact-Oriented Collaborative Ontology Evolution. In: *OTM Workshops: Meersman, R., Tari, Z. Herrero, P. (eds.)*. Vol. 5333 of LNCS, Springer (2008) pp. 797-806
8. Debruyne, C.: On the social dynamics of ontological commitments. In: *OTM Workshops: Meersman, R., Dillon, T., Herrero, P. (eds.)*. Vol. 6428 of LNCS, Springer (2010) pp. 682–686
9. Fellbaum, C.: *WordNet: An Electronic Lexical Database*. MIT Press (1998)
10. Guarino, N.: Formal ontology and information systems. In: *Int. Conf. On Formal Ontology In Information Systems FOIS'98, Trento, Italy, Amsterdam, IOS Press (1998)* pp. 3–15
11. Halpin, T.: *Information Modeling and Relational Databases*. Morgan Kaufmann, San Francisco, CA, USA (2008)
12. Hepp, M., De Leenheer, P., de Moor, A., Sure, Y. (eds.): *Ontology Management, Semantic Web, Semantic Web Services, and Business Applications*. Vol. 7 of *Semantic Web And Beyond Computing for Human Experience*. Springer (2008)
13. Jarrar, M.: Towards the notion of gloss, and the adoption of linguistic resources in formal ontology engineering. In: *Proc. of the 15th Int. Conf. on World Wide Web (WWW2006)*, ACM Press, New York, NY (2006) pp. 497–503
14. Meersman, R.: Semantic ontology tools in IS design. In: *ISMIS: Ras, Z.W., Skowron, A. (eds.)*. Vol. 1609 of LNCS, Springer (1999) pp. 30–45
15. Meersman, R.: Reusing certain database design principles, methods and techniques for ontology theory, construction and methodology. Technical report, VUB STARLab, Brussel (2001)
16. Meersman, R., Debruyne, C.: Hybrid ontologies and social semantics. In: *Proc. of 4th IEEE Int. Conf. on Digital Ecosystems and Technologies (DEST 2010)*, IEEE Press (2010)
17. Schneider, J., Passant, A., Breslin, J.: Enhancing mediawiki talk pages with semantics for better coordination - a proposal. In: *In The Fifth Workshop on Semantic Wikis: Linking Data and People Workshop at 7th Extended Semantic Web Conference (ESWC)*. (2010)
18. Spyns, P., Meersman, R., Jarrar, M.: Data modelling versus ontology engineering. *SIGMOD Record Special Issue* 31 (4) (2002) pp. 12–17
19. Trog, D., Tang, Y., Meersman, R.: Towards ontological commitments with Ω -RIDL markup language. In: *RuleML: Paschke, A., Biletskiy, Y. (eds.)*. Vol. 4824 of LNCS, Springer (2007) pp. 92–106
20. Verheyden, P., De Bo, J., Meersman, R.: Semantically unlocking database content through ontology-based mediation. In: *SWDB: Bussler, C., Tannen, V., Fundulaki, I. (eds.)*. Vol. 3372. (2004) pp. 109–126
21. Viegas, F. B., Wattenberg, M., Kriss, J., van Ham, F.: Talk before you type: Coordination in wikipedia. In: *Proc. of the 40th Annual Hawaii Int. Conf. on System Sciences. HICSS '07, Washington, DC, USA, IEEE Computer Society (2007)* pp. 78–87
22. Wintraecken, J.: *The NIAM Information Analysis Method, Theory and Practice*. Kluwer Academic Publishers (1990)