The Relation between a Framework for Collaborative Ontology Engineering and Nicola Guarino's Terminology and Ideas in "Formal Ontology and Information Systems"

Christophe Debruyne

Semantics Technology and Applications Research Lab
Vrije Universiteit Brussel
chrdebru@vub.ac.be

Abstract. In this paper, we investigate the relation between Guarino's seminal paper "Formal Ontology and Information Systems" and the DOGMA ontology-engineering framework. As DOGMA is geared towards the development of ontologies for semantic interoperation between autonomously developed and maintained information systems, it follows that the stakeholders in this project form a community and adds a social dimension to the ontology project. The goal of this exercise is to examine how the different terminologies and ideas relate to one and another, thus providing a reference for clarifying DOGMA's ideas and notation inside Guarino's framework.

Key words: Formal Ontology, Ontology-engineering Frameworks

1 Introduction

An ontology is commonly defined as: *"a [formal,] explicit specification of a [shared] conceptualization"* [?][1]. Ontologies constitute the key resources for realizing a semantic Web. The main difference between a conceptual schema and an ontology is that the first is intended for the development of one particular information system in one organization and the latter for reuse and therefore general for a particular domain [15].

Gruber's definition was based on the definition of Genesereth and Nilsson's notion of a conceptualization [6] that used an extensional notion for describing one particular state of affairs. Guarino and Gieretta in [9], however, have argued that a different intensional account of the notion of conceptualization has to be introduced for this definition in order for Gruber's definition to have some sense. Guarino then wrote his – currently – most influential work "Formal Ontology and Information Systems" in which he provided definitions for *conceptualization*, *ontological commitment* and *ontology.*

[1] Gruber's original definition is without the words "shared" and "formal", but are accepted by relevant scientific communities to describe more precisely the intention of ontologies.

Over the past years, quite a few (collaborative) ontology-engineering methods have been developed, each with their own characteristics; e.g., the formalism adopted, approach of agreement processes, application domain, etc. [4]. In this paper, we will take a closer look at one particular ontology-engineering framework – which served as the framework for collaborative methods such as BSM [1], DOGMA-MESS [3] and GOSPL [4] – and relate the concepts in this framework to that of Guarino's.

The goal of this paper is to investigate how the different terminologies relate to each other given the fact that Guarino's work aims to provide a definition for ontology in computer science and a collaborative ontology-engineering framework for establishing semantic interoperability between autonomously developed information systems. The paper is organized as follows: Section 2 provides a summary and explanation of Guarino's ideas and definitions, Section 3 describes the problem of semantic interoperability between autonomously developed and maintained information systems, Section 4 describes the notion of the fact-oriented collaborative ontology-engineering framework DOGMA, Section 5 discusses the relation between the two, followed by a conclusion in Section 6.

2 Formal Ontology and Information Systems

This section explains the terminology used in Guarino's most influential work [8]. The definition provided by Genesereth and Nilsson [6] and adopted by Gruber [**?**] is given below.

Definition 1 (Extensional notion of conceptualization [6]). *A conceptualization is defined as a structure $\langle D, R\rangle$, where D is a domain and R is a set of relevant relations on D.*

However, Guarino argued in [8] that this notion of conceptualization was to restrictive as it was referring to only one particular state of affairs: *"The problem with Genesereth and Nilsson's notion of conceptualization is that it refers to ordinary mathematical relations on D, i.e. extensional relations. These relations reflect a particular state of affairs: for instance, in the blocks world, they may reflect a particular arrangement of blocks on the table. We need instead to focus on the meaning of these relations, independently of a state of affairs: for instance, the meaning of the "above" relation lies in the way it refers to certain couples of blocks according to their spatial arrangement. We need therefore to speak of intensional relations: we shall call them conceptual relations, reserving the simple term "relation" to ordinary mathematical relations."*. He therefore introduced a notation of *domain space* that refers to a set of possible states of affairs for a particular domain and a notion of *conceptual relations* on such domain spaces:

Definition 2 (Domain Space [8]). *A domain space is a structure $\langle D, W\rangle$, where D is a domain and W is a set of maximal states of affairs of such a domain (also called* possible worlds*).*

Definition 3 (Conceptual relation [8]). *Given a domain space $\langle D, W \rangle$, a conceptual relation ρ^n of arity n on $\langle D, W \rangle$ is defined as a total function $\rho^n : W \rightarrow 2^{D^n}$ from W into the set of all n-ary relations on D.*

With the notions introduced by Guarino, the structure $\langle D, R \rangle$ introduced in [6] can now be regarded as referring to a particular state of affairs [8]. A conceptualization according to Guarino is then defined as follows:

Definition 4 (intensional notion of conceptualization [8]). *A conceptualization for D is defined as an ordered triple $C = \langle D, W, \mathcal{R} \rangle$, where $\mathcal{R}$ is a set of conceptual relations on the domain space $\langle D, R \rangle$.*

Given a logical language L with a vocabulary V, Guarino provided an extensional and an intensional interpretation of the language.

Definition 5 (Extensional interpretation of a language [8]). *A model for L is defined as a structure $\langle S, I \rangle$, where $S = \langle D, R \rangle$ is a world structure and $I : V \rightarrow D \cup R$ is an interpretation function assigning elements of D to constant symbols of V and elements of R to predicate symbols of V.*

Definition 6 (Intensional interpretation of a language [8]). *An intensional interpretation of a language is by means of a structure $\langle C, \mathcal{I} \rangle$, where $C = \langle D, W, \mathcal{R} \rangle$ is a conceptualization and $\mathcal{I} : V \rightarrow D \cup \mathcal{R}$ a function assigning elements of D to constant symbols of V and elements of $\mathcal{R}$ to predicate symbols of V. This intensional interpretation is called an ontological commitment for L. if $K = \langle C, \mathcal{I} \rangle$ is an ontological commitment for L, we say that L commits to C by means of K, while C is the underlying conceptualization of K.*

Definition 7 (The set of intended models of a language L according to a commitment K [8]). *The set $I_K(L)$ of all intended models of a language L that are compatible with commitment $K = \langle C, \mathcal{I} \rangle$ are all the models of L that are compatible with K. A model $\langle S, I \rangle$ is compatible with K if: $S \in \{S_{wC} | w \in W\}$, where $S_{wC} = \langle D, R_{wC} \rangle$ is the intended structure of w according to C and $R_{wC} = \{\rho(w) | \rho \in \mathcal{R}\}$ is the set of extensions relative to w of the elements of $\mathcal{R}$ (i.e., S is one of the intended world structures of C); (ii) for each constant c, $I(c) = \mathcal{I}(c)$; (iii) $\exists w \in W$ such that for each predicate symbol p, there exists a conceptual relation ρ such that $\mathcal{I}(p) = \rho \wedge \rho(w) = I(p)$.*

Guarino argued that the set of intended models only provide a weak characterization of a conceptualization as an intended model does not necessarily reflect a particular world and can even describe a situation common to most worlds. Indeed, if one would take for instance a domain space with a domain of the following persons $D = \langle Peter, Louis, Joe, Bonnie \rangle$ and where all words contain all possible marriage configurations as well as all persons classified according to their gender (Peter and Joe being male, Louis and Bonnie female). If one takes as set of conceptual relations only the unary predicates for being male and female, the intended model for each world – via an $\mathcal{I}$ an appropriate $\mathcal{I}$ – will be the same.

Because an intended model can describe a situation common to many worlds, it only rules out absurd situations. One can merely create a set of axioms for a commitment K and a language L such that the models of the ontology approximates as close as possible $I_K(L)$. This set of axioms is what Guarino calls an ontology. Ontologies thus indirectly reflect an ontological commitment by approximating the intended models of that commitment.

3 Closed vs. Open Information Systems

Now that we have presented Guarino's ideas and definitions, we will present the difference between closed and open information systems. As we will see, for both systems an ontology will be needed that will replace the real word and hence one can state that the "same" exercise is done at an enterprise level for the former and at a domain level for the latter. Developing ontologies for open information systems, however, will prove to be more challenging due to the different perceptions of reality and use of language (e.g., jargon) used by the stakeholders representing the different autonomously developed and maintained information systems.

An information system is a system containing information in a database for a given application context of a given organization. The application context defines the functionality of such a system, which is prescribed by the organizations' requirements. The development of an information system thus involves the creation of a requirements specification and an agreement on the design. Costing is an important factor here and will also influence the choice whether components will be selected for its implementation, outsourced, or even build from scratch. One should involve end-users during the requirements specification process for several reasons. Two of these reasons are the impedance mismatch between the jargon (used by end-users) and the business knowledge, and the end-users being experts on (their part of) the domain.

As shown in Fig. 1, domain experts and end-users observe the world. Domain experts try to abstract the world, whereas the end-users will interact with the world and are able to test the developed information systems by comparing the instances stored in the information systems with objects in the real world. Both domain experts and end-users will collaborate with a knowledge designer so that the latter can put the resulting agreements into a CASE tool to build a conceptual schema that described the business domain. The conceptual schema will – in turn – be used to generate parts of the processes, parts of the constraints and a data model of the information system.

The knowledge in such a conceptual schema is typically a mixture of domain-general knowledge and enterprise-specific knowledge derived from the requirements. Domain knowledge can contain constraints that are shared or generic. Enterprise-specific knowledge describes the applications and constraints local to the enterprise making use of the domain knowledge. Enterprise-specific knowledge will often constrain the domain-general knowledge even further. For example, the knowledge that an ISBN identifies a book is part of the domain knowl-

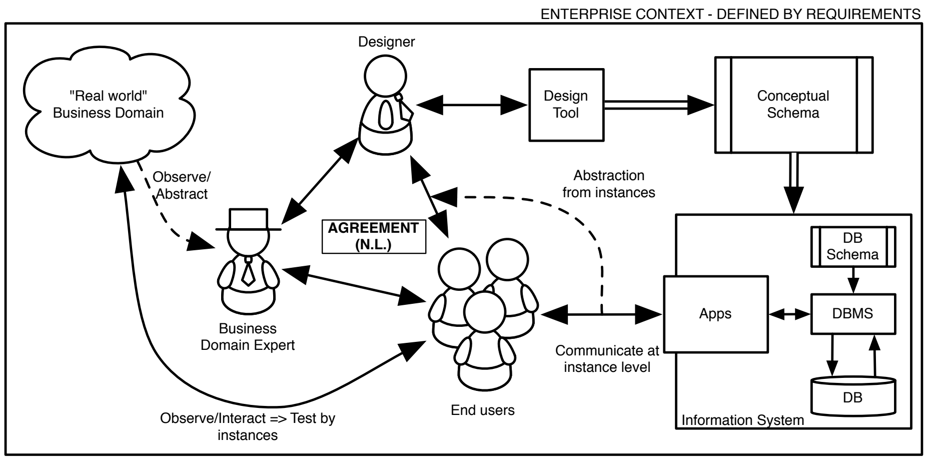
ENTERPRISE CONTEXT - DEFINED BY REQUIREMENTS
"Real world" Business Domain
Designer
Design Tool
Conceptual Schema
Observe/ Abstract
Abstraction from instances
AGREEMENT (N.L.)
Business Domain Expert
DB Schema
Apps
DBMS
DB
Communicate at instance level
End users
Observe/Interact => Test by instances
Information System

Fig. 1. Information Systems in an enterprise context.

edge. In an enterprise providing a movie rental service, however, the knowledge that a customer is only able to lend at most five movies at a time is part of the enterprise knowledge.

The conceptual schema, often in the shape of a formal (mathematical) construct, actually replaces the business domain, as the business domain is not accessible inside a computer. This is necessary in order to store and reason about semantics the business domain. The formal semantics of an information system is then the correspondence between this system and the conceptual schema, which represents the business domain as perceived by the domain expert and the end-users. Once the system is adequately designed and implemented, a statement output by the information system can be correctly interpreted by end-users in terms of objects in the business domain if and only if such statement is derived from stored instances of fact types described in the conceptual schema. Those stored instances are mapped by the intensional semantics to observed relationships among those same objects.

In Fig. 1, the conceptual schema corresponds with the ontology in Guarino's terminology (i.e., the intensional description of the concepts and relations) and the database provides the extensional account of a particular situation.

But what happens when two or more autonomously developed and maintained information systems need to interoperate and yet need to remain autonomous[2]? The business domains of each information systems relates to the shared domain knowledge of all humans involved. In order for these systems to interoperate meaningfully, agreements on the domain knowledge by the represen-

[2] In general, semantic interoperability is defined as the ability of two or more information systems or their (computerized) components to exchange data, knowledge or resources and to interpret the information in those systems [2].

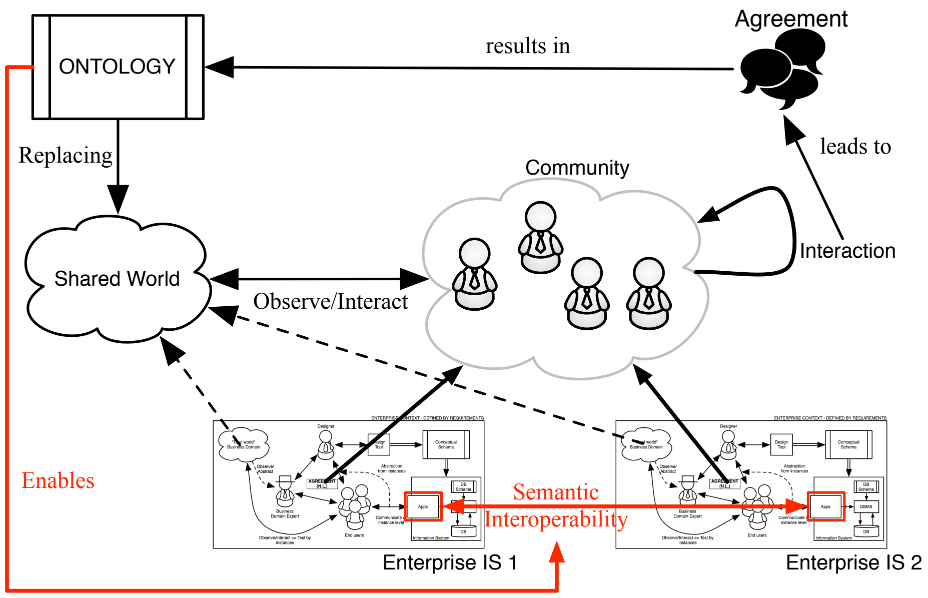

Fig. 2. Agreements leading to ontology for enabling semantic interoperability

tatives of these systems – which we will call a *community* – are needed. Again, as the world is not accessible in each one of those information systems, the shared domain needs to be replaced by another formal (mathematical) construct, called an *ontology*. As shown in Fig. 2, each autonomous information system has its own database which each provides an extensional interpretation of the conceptualization approximated by the ontology. In other words, each database of an information system can be regarded as a possible world. Fig. 2 also depicts that a community of stakeholders interacts to achieve agreements on what perceptions they share. The approximation of the shared reality that is shared among those stakeholders is then stored as an ontology, which will in turn be used the replace the shared world as the real world is not immediately available from the information systems.

4 Developing Ontology Guided Methods and Applications

In [16] and [14] a formalism and method for ontology development called Developing Ontology Guided Methods and Applications (DOGMA) was defined that illustrated and implemented these principles, now lifted to domain level from the mere enterprise system level. In the method and life-cycle 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.

Definition 8 (DOGMA Ontology Descriptions). *A DOGMA Ontology Description Ω is an ordered triple $\langle \Lambda, ci, \mathcal{K} \rangle$ 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. Here, the usual alphabets for constructing the elements of $T \cup R$ are omitted for simplicity. $ci : \Gamma \times T \to C$ is a function mapping pairs of context identifiers and terms to unique elements of C, a finite given set of concepts. $\mathcal{K}$ is a finite set of ontological[3] commitments. Each commitment is an ordered triple $\langle \sigma, \alpha, c \rangle$ where $\sigma \subset \Lambda$ is a selection of lexons from the DOGMA ontology description, $\alpha : \Sigma \to T$ is a mapping called an annotation from the set Σ of application (information, system, database) symbols to terms occurring in that selection, and c is a predicate over $T \cup R$ of that same selection expressed in a suitable first-order language.*

Context-identifiers are pointers to the origin of a lexon, and helps with the disambiguation of term- and role-labels. Within a context $\gamma \in \Gamma$ and $t \in T$, $ci(\gamma, t)$ is the definition itself of the concept agreed by all users.

We furthermore make a distinction between community commitments and application commitments [5]. The first is an engagement of the community members to commit to the lexons and constraints agreed upon by the community. The latter is a selection of lexons that are constrained (according to how the application uses these lexons) and a set of mappings from application symbols to terms and roles in that selection. A community commitment is motivated by the need for proper semantic interoperation between information systems. Depending on the goal of the ontology, instances shared across different autonomous information systems need to some degree to be compared for equivalence. One example is joining information about an instance across heterogeneous sources. In order to achieve this, the members of the community have to agree upon a series of attributes that uniquely, and totally identify the concepts they share. In other words, the conceptual reference structures. By sharing the same reference structures, the information systems are able to interpret information describing instances and find the corresponding instance in their data store (of that of a third system). Application commitments refer to community commitments and can contain additional lexons and constraints. For instance, lexons needed to annotate application specific symbols (e.g., artificial IDs, often found in relational databases) to ensure that instances of concepts are properly aligned (e.g., a proper annotation of the foreign keys in a join-table). Both community- and application commitments also store information about the agreements across communities.

[3] Not to be confused with Guarino's ontological commitment.

5 Relation between the two Formalisms

As noted by [13], DOGMA embraces the intensional notion of a conceptualization of Guarino, but arrived at it from a database-inspired perspective [14]. DOGMA, however, also pursues this idea to arrive at concrete software architectural and engineering conclusions [13]. Other than this statement in [13], there is no existing publication on the relationship between the work of Guarino and DOGMA.

Lexons are *plausible* fact types, which means that if one can think of (an) application(s) for such a lexon, they may be entered in the lexon base. The sets T and R for term- and role-labels in lexons correspond to the predicate symbols in V. The context-identifier γ provides an interpretation from terms to concepts. Since lexons are entered when they are plausible for one or more applications, the context-identifier γ actually corresponds to Guarino's interpretation function $\mathcal{I}$. In other words, if one selects in the lexon base all lexons holding in a particular context with context-identifier γ, one is able to reconstruct Guarino's interpretation function $\mathcal{I}$: all concepts x referred to by $ci(\gamma, t)$ (for each term t in those lexons) will refer to the interpretation of a unary predicate.

DOGMA's is based on ORM [12] and NIAM [17], which are fact-oriented modeling language. In fact-oriented formalisms, knowledge is expressed by means of fact types. A fact type is a generalization of a fact. For example: "Person is born on Date" is a fact type and "Christophe is born on 8 August 1984" a fact. Facts are thus instances of fact types. Fact types are elementary when they can be simplified without loss of meaning. Because of DOGMA's fact-oriented nature – it is based on ORM [12] and NIAM [17] – the use of the predicates denoted by the term- and role-labels are already constrained [11]. A binary fact type $\langle A, R, S, B \rangle$ is actually translated into the following first order logic statements [11]:

- $\forall x \forall y (R(x, y) \rightarrow (A(x) \wedge B(y))$
- $\forall x \forall y (R(x, y) \leftrightarrow S(y, x))$

The combination of the predicates with quantifications and connectives already reduces the set of all possible models with those satisfying above constraints. The fact-orientented formalism already provides some constraints that restrict the intended models of the language; i.e. it leaves out some absurd situations. In DOGMA, role is identified by its context, role label, term and co-term. The roles in each of the following lexons are thus different:

- ⟨*Person Context, Person, with, of, Name*⟩
- ⟨*Person Context, Dog, with, of, Name*⟩
- ⟨*Person Context, Person, with, of, Age*⟩
- ⟨*Project Context, Person, with, of, Name*⟩

Indeed, a group can argue that the first and second lexon in the example above are referring to the same notions of "having a name", but such an agreement would actually be a constraint on these lexons that will be captured by community commitments that we will describe later on. Also, the first and last lexon are deemed to be different since the term- and role-labels are residing in different contexts. Unless explicitly specified, the same labels in different contexts do not necessarily mean that those labels are referring to the same concepts.

A commitment $k \in \mathcal{K}$ of the DOGMA Ontology Description corresponds with an ontology from Guarino's framework. It is a selection of lexon from the lexon base that is constrained such that it approximates as good as possible the domain it aims to describe. Those constraints correspond with the notion of axioms and typically include notions such as: type- and role hierarchies, totality constraints, uniqueness constraints, value constraints, etc.

Value constraints are interesting to note that they limit domain elements for the interpretation of concept referred to by a term. There are two types of value constraints: exhaustive sets and descriptions of sets. We start with an example of the first type of value constraints: giving the term `Size` referring to the size of a drink at a fast food restaurant in the `Fast Food` context, one can constraint the instances of that concept to the following values {`"Small", "Medium", "Large"`}. These instances of values then would refer to the concepts of these different sizes in that context. The latter type – e.g. a regular expression – checks whether an instance complies with the condition and are therefore considered restraining the use of a unary predicate.

A community commitment further restrains all possible models of the lexons committed to. An application commitment will even further restrain those by providing additional lexons, constraints, and narrowing down all possible models by providing additional constants via the mappings.

Since the real world cannot be stored in a computer-based system, one needs to replace that real world with a database (and corresponding database schema) in order to reason about things in the real world. We assume that a database of an organization corresponds with one real world. The mappings provided by a commitment $k \in \mathcal{K}$ are used how the constant symbols in a database are related in aforementioned predicates. For instance: assuming that one has the lexon ⟨*Fast Food, Soda, with, of, Name*⟩ and a particular fast food organization has a table `SSS` containing records about different sodas with information such as the name in the field `NNN`, one can provide an extensional account for the predicates "Food", "Name", "with" and "of" by adding the following mapping: `MAP "SSS"."NNN" ON Name of Soda.` and all symbols corresponding to the SQL query generated with this mapping will be used to populate the predicates with constants (e.g., `"Pepsi", "Sprite,"` ...).

It is interesting to note that when relating DOGMA with Guarino's seminal work, the ontological commitment in DOGMA ontology descriptions does not allow constant symbols, but that the ontologies in the DOGMA ontology description do so for constraints and via the mappings.

In summary:

– Guarino's vocabulary V would correspond with the union of DOGMA's T, R, values in value constraints in each $\mathcal{K}$ and symbols from records of each database described in the mappings of each $\mathcal{K}$ via queries. This vocabulary is then used for the language L.

- Guarino's ontology corresponds with the constraints of one commitment $k \in \mathcal{K}$. In the case of a community commitment, the intended models are narrowed down with those constraints. In the case of an application commitment, the intended models are even further narrowed down. This is the case as the constant symbols of a database are then also used. Note that a database does not necessarily describe one world, but could even describe situations plausible for many worlds.

  We can even argue that the mappings provided in an application commitment are – albeit indirectly – and even further restriction of the intended models, i.e., the mappings for a constraint on their own.
- Guarino's seminal work is not particularly clear on the ontological language L. On one hand it seems to be richer than the mere use of the predicate and constant symbols of V, but on the other the constraints on those predicates are left to the ontological commitment such as demonstrated in

  [10]. The fact-orientation of DOGMA – for us – corresponds to Guarino's notion of a language and, as shown above, already provides some constraints on the intended models.

From above, it follows that one needs to decompose the commitments and combine pieces with the lexon base to reconstruct Guarino's ontological commitment. In other words, there is a high cohesion between ontological commitments and ontologies in the DOGMA ontology-engineering framework.

6 Conclusions

This paper presented a thought exercise on relating a framework for collaborative ontology engineering with Nicola Guarino's seminal work "Formal Ontology and Information Systems". This exercise allows us to provide documentation to the reader on the DOGMA framework and might even clarify some notions presented by Guarino. As the DOGMA framework is mainly used for establishing semantic interoperability between autonomously developed information systems, the relation of the symbols in databases to concepts denoted by labels in the ontology was "new" compared to Guarino's framework. The work presented in this paper will furthermore allow us to revise or clarify some of the definitions we presented in the past.

Acknowledgements

This work was partially funded by the Brussels Institute for Research and Innovation through the Open Semantic Cloud for Brussels Project.

References

1. 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
2. De Leenheer, P., Mens, T.: Ontology evolution: State of the art and future directions. 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
3. de Moor, A., De Leenheer, P., Meersman, R.: DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering. In: Proceedings of the 14th International Conference on Conceptual Structures (ICCS 2006). Vol. 4068 of LNCS., Springer (2006) pp. 189–203
4. Debruyne, C., Tran, T.K., Meersman, R.: Grounding ontologies with social processes and natural language (to appear). Journal of Data Semantics (2013)
5. Debruyne, C., Vasquez, C.: Exploiting natural language definitions and (legacy) data for facilitating agreement processes. In: SWQD: Winkler, D., Biffl, S., Bergsmann, J. (eds.). Vol. 133 of LNBIP., Springer (2013) pp. 244–258
6. Genesereth, M., Nilsson, N.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann, San Mateo, CA (1987)
7. Gruber, T.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies **43** (1993) pp. 907–928
8. Guarino, N.: Formal ontology and information systems. In: International Conference On Formal Ontology In Information Systems FOIS'98, Trento, Italy, Amsterdam, IOS Press (1998) pp. 3–15
9. Guarino, N., Giaretta, P.: Ontologies and Knowledge Bases: Towards a Terminological Clarification. Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing (1995) pp. 25–32
10. Guarino, N., Oberle, D., Staab, S.: What is an ontology? Handbook on Ontologies (2009) pp. 1–17
11. Halpin, T.A.: A Logical Analysis of Information Systems: static aspects of the data-oriented perspective. PhD thesis, University of Queensland (1989)
12. Halpin, T.A., Morgan, T.: Information Modeling and Relational Databases. Morgan Kaufmann, San Francisco, CA, USA (2008)
13. Jarrar, M., Meersman, R.: Ontology engineering – the DOGMA approach. In: Advances in Web Semantics I: Dillon, T.S., Chang, E., Meersman, R., Sycara, K. (eds.). Vol. 4891 of LNCS. Springer Berlin Heidelberg (2009) pp. 7–34
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.: The use of lexicons and other computer-linguistic tools in semantics, design and cooperation of database systems. In: The Proceedings of the Second International Symposium on Cooperative Database Systems for Advanced Applications (CODAS99): Zhang, Y., Rusinkiewicz, M., Kambayashi, Y. (eds.), Springer (1999) pp. 1–14
16. Meersman, R.: Reusing certain database design principles, methods and design techniques for ontology theory, construction and methodology. Technical report, VUB Starlab (2001)
17. Wintraecken, J.: The NIAM Information Analysis Method: Theory and Practice. Kluwer Academic Publishers (1990)