Pieter De Leenheer and Christophe Debruyne. DOGMA-MESS: A Tool for Fact-Oriented Collaborative Ontology Evolution. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, OTM Workshops, volume 5333 of LNCS, pages 797-806. Springer, 2008

DOGMA-MESS: A Tool for Fact-Oriented Collaborative Ontology Evolution *

Pieter De Leenheer and Christophe Debruyne

Semantics Technology and Applications Research Laboratory (STARLab)
Vrije Universiteit Brussel, Pleinlaan 2, Brussels 5, Belgium

Abstract. Ontologies being shared formal specifications of a domain, are an important lever for developing meaningful internet systems. However, the problem is not in what ontologies are, but how they become operationally relevant and sustainable over longer periods of time. Fact-oriented and layered approaches such as DOGMA have been successful in facilitating domain experts in representing and understanding semantically stable ontologies, while emphasising reusability and scalability. DOGMA-MESS, extending DOGMA, is a collaborative ontology evolution methodology that supports stakeholders in iteratively interpreting and modeling their common ontologies in their own terminology and context, and feeding back these results to the owning community. In this paper we extend DOGMA Studio with a set of collaborative ontology evolution support modules.

1 Introduction

Ontologies, being formal, computer-based specifications of shared conceptualisations of the worlds under discussion, are an important lever for developing meaningful communication between people and internet systems [10, 9]. However, the problem is not in what ontologies are, but how they become *community-grounded* resources of semantics, and at the same time be made operationally relevant and sustainable over longer periods of time. The state of the art in ontology evolution regards change as a pain that must be technically alleviated by presuming a project-like practice where ontologies are created and deployed in discrete steps [6]. The requirements for the "ontology project" are usually deduced from the technical web service requirements that were solo-designed by a single application developer, rather than collaboratively grounding them directly in the community. In the DOGMA framework [13], *fact-oriented approaches* such as NIAM/ORM [21, 11] have been proven useful for engineering ontologies. A key characteristic here is that the analysis of information is based on natural language facts. This brings the advantage that "layman" domain experts are facilitated in building, interpreting, and understanding attribute-free, hence semantically stable ontologies, using their own terminology. DOGMA-MESS is a teachable and repeatable *collaborative ontology evolution methodology* that supports stakeholders in interpreting and modeling

* We would like to thank Stijn Christiaens for his valuable comments on the usability of the tool. The research described in this paper was partially sponsored by the EC projects FP6 IST PROLIX (FP6-IST-027905).

their common ontologies in their own terminology and context, and feeding back these results to the owning community. In this paper we extend DOGMA Studio with a set of modules (Perspective Manager, Version Manager, and Community Manager) that support these fact-oriented collaborative ontology evolution processes.

2 DOGMA Ontology Engineering

The DOGMA[1] ontology approach and framework [15] is adopted with the intention to create flexible, reusable bounded semantics for very diverse computational needs in communities for an unlimited range of pragmatic purposes. DOGMA has some distinguishing characteristics that make it different from traditional approaches such as (i) its groundings in the linguistic representations of knowledge, (ii) the explicit separation of the conceptualisation (i.e., lexical representation of concepts and relationships) from its axiomatisation (i.e., semantic constraints) and (iii) its independence from a particular representation language. The goal of this separation, referred to as the *double articulation* principle [19], is to enhance the potential for reuse and design scalability.

Lexons are initially uninterpreted binary fact types, hence underspecified, which increases their potential for reusability across community perspectives or goals. Lexons are collected in a lexon base, a reusable pool of possible vocabularies, and represented as 5-tuples declaring either: a taxonomical relationship (*genus*): e.g., $\langle \gamma, manager, is\,a, subsumes, person \rangle$; or a non-taxonomical relationship (*differentia*): e.g., $\langle \gamma, manager, directs, directed\,by, company \rangle$, where γ is an abstract context identifier, lexically described by a string in some natural language, and is used to group lexons that are logically related to each other in the conceptualization of the domain.

Another distinguishing characteristic of DOGMA is the explicit *duality* (orthogonal to double articulation) in interpretation between the syntactic level and semantic level. The goal of this separation is primarily to disambiguate the syntactic representation of terms in a lexon into concept definitions, which are word senses taken from a *community glossary* such as WordNet[2]. The meaning of the terms in a lexon is dependent on the *context of elicitation* [4]. For example, a term "capital" elicited from a typewriter manual (read: context γ), it has a different meaning (read: concept definition) than when elicited from a book on marketing. Though ontologies can differ in syntax, semantics, and pragmatics, they all are built on this shared vocabulary, called the lexon base.

The *perspective commitment layer* mediates between the lexon base and its applications. Each such perspective defines a partial semantic account of an intended conceptualization [10]. 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 one on conceptualization [15]. E.g., the rule stating that each patient is a person who suffers from at least one disease may be too strong in some domains.

[1] Developing Ontology-Grounded Methods and Applications
[2] http://wordnet.princeton.edu/

3 Community Evolution

Community dynamics, as illustrated in Fig. 1, is characterised by Nonaka's [17] four modes of *knowledge conversion*: *socialisation, externalisation, combination*, and *internalisation*. At the heart of the community dynamics is the Ontology Server, that bridges the semiotic gap between the community system parts. It is embedded in a central ontology evolution support system we introduced [7] and validated [2] earlier. There are three types of *knowledge workers*: the *knowledge engineer*, the *core domain expert* (CDE), and the *domain expert* (DE). As we will show, in DOGMA-MESS, the involved ontology evolution processes (*community grounding, rendering, alignment, and commitment*) are inherently driven by the social knowledge conversion modes.

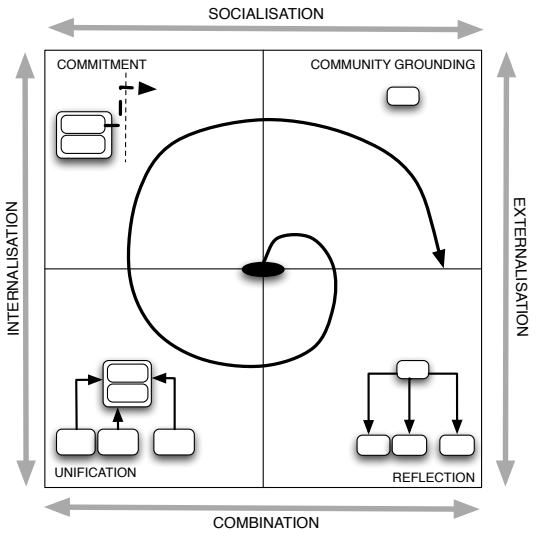

Fig. 1. DOGMA-MESS ontology evolution spiral model.

1. **Community Grounding:** In this phase, shared conceptions of the world under discussion that emerged from *socialisation* are analysed by the CDE. With the assistance of the KE, he identifies the key *conceptual patterns* that are relevant to be further externalised to the ontology. This results in a generalised *upper common ontology* (UCO) which represents the conceptualizations that are common to and accepted by the community.
2. **Perspective Rendering:** All participating stakeholders' DEs render their perspective on the UCO, by specialising the conceptual patterns, resulting in a set of diverging *stakeholder perspectives* (SPs). Doing so, ontology evolution is grounded (bottom-up) in the community, starting with the variety of terminologies found in

the community itself. This allows DEs to syntactically and semantically nuance their intensions in a more natural manner using their own vocabulary. In order to impose UCO reuse, different types of *perspective reuse policies* can be formalised, including articulation, specialisation, and application. A reuse policy is formalised by a set of applicable operations on a perspective (see [4].

3. **Perspective Unification:** In the *lower common ontology* (LCO), a new proposal for the next version of the common ontology is produced, *combining* relevant material from the UCO and various stakeholder perspectives. Basically, there is only a very simple rule: all (selected) definitions need to be full specializations of the conceptual patterns in the UCO. This, however, is overly simplified. In the ontology evolution process, despite reuse policies, the constructivist paradigm should allow to *override* the reuse policies, and hence new definitions to be created that are not (complete) specializations, but represent new insights for the CDE in preparing new evolution rounds, for example. This makes the alignment process far from trivial. This process is conducted collaboratively by all involved DEs, the CDE, and the KE.
4. **Perspective Version Commitment:** The part of the LCO that is aligned by the community forms the legitimate UCO for the next version of the common ontology. All participating organisations finally internalise and commit their instance bases to the new version.

In all phases, the views of all stakeholders are considered. This fourfold collaborative ontology evolution process is iteratively applied until an optimal balance of differences and commonalities between organisational and common perspectives are reached that meets the communication goals.

4 DOGMA Studio

DOGMA Studio contains both a *Workbench* and a *Server*. The Workbench is constructed according to the Eclipse plugin architecture. The loose coupling allows any arbitrary community to support its own ontology engineering method by customised ontology viewing, querying or editing plugins. The Server is an advanced J2EE application running in a JBoss server which efficiently stores Lexons and Commitments in a PostgreSQL Database. The manual input method uses the NORM notation, which is an adaptation of NIAM/ORM2, introduced by [20]. Workbench can also perform conversion to and from the following formats: (i) comma-separated files can be imported in (exported from) the Perspective Base; (ii) Ω-RIDL commitment files can be imported in (exported from) the Perspective Commitment Layer; and (iii) RDF(S)/OWL files can be imported in (exported from) the Perspective Base (see [1, 12]). Following is an overview of the three main Eclipse perspectives to support the community evolution processes, i.e. Version Manager, Community Manager, and Perspective Manager. Due to space limitations the overview is rather limited. For more demo material we refer to the DOGMA Studio website[3].

[3] http://www.starlab.vub.ac.be/website/dogmastudio (last access: 5 July 2008)

4.1 Version Manager

The Ontology Version manager provides basic plugins for Viewing and Editing Perspective and Pattern Versions.

Ontology Viewer plugin allows to explore the perspectives, patterns, perspective policies, and their versions that are currently stored in the Server. The tree directory view sorts the ontologies per domain, as shown in Fig. 2 (top). The first level enlists the domains (e.g., opensourceartefacts); the second level, within one domain, enlists the ontologies (e.g., sodocu); the third level, within one ontology, enlists the version history; the fourth level, within one ontology, enlists the available patterns (and their versions) (e.g., software_artefact); and finally, the fifth level, given a pattern, shows the perspective version history. To uniquely identify definition versions, we adopted a universe resource identifier (URI). E.g., domain/ontology/pattern;j#stakeholder,i identifies a perspective on a pattern, with version number j, rendered by stakeholder, with version number i.

Ontology Editor plugin has three panes for viewing/editing the taxonomy, the relations, or the whole pattern or perspective that was selected from the Ontology Viewer. The latter is illustrated by Fig. 2 (second line) [4]. The title bar shows the version URI. Each part of the perspective has a different colour: pattern parts are coloured blue, UCO parts are coloured grey, and perspective parts currently rendered by the DE are coloured in yellow. Figure 2 (third line) shows a fourth pane that displays the events and editing log, tracking each change operation made to the perspective. For each operation, the pre- and postconditions are shown indicating why the operation succeeded or failed. Change logs allows for semantic conflict analysis during *change-based merging* of parallel SPs. For a formalisation of this using *graph transformation theory* see e.g., [5]. Finally, the Concept Viewer plugin retrieves the concept definition from the Community Glossary when a term is selected.

4.2 Community Manager

The Community Manager provides plugins for managing *tickets* and conceptual patterns. During the community grounding phase, for each key concept of interest a ticket is created with attached a conceptual pattern, and sent to the relevant stakeholders in the community to render their perspective on it. The *Ticket Viewer* and the *Dialogue Box* are illustrated in Fig. 2 (bottom). A ticket has a title; informal information about the rendering task; a priority code (high, higher, normal, lower or low); the context in which the ticket is to be executed; the CDE who created the ticket; and finally, the stakeholding DEs receiving the ticket. A Dialogue Box is opened prompting the knowledge worker to open the ontology context in which the evolution task is to be performed. The *Ticket Maker* plugins are not illustrated due to space limitations.

[4] The examples are in Dutch as they are extracted from a realistic case study (Sect.4.4) that was conducted in the Netherlands.

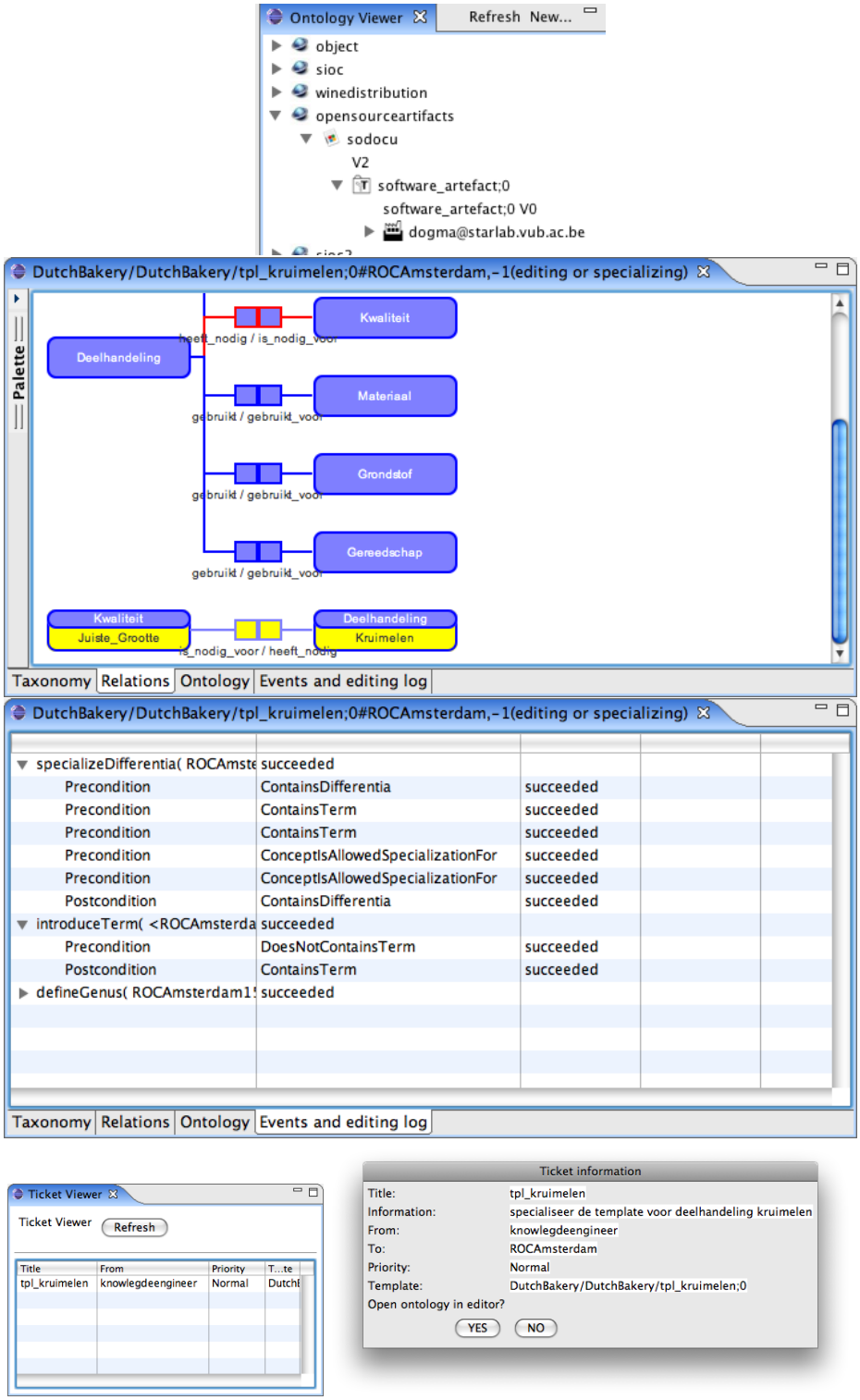

Fig. 2. From top to bottom: (1) the Ontology Viewer; (2) the integrated ontology viewing/editing pane of the Ontology Editor; (3) the events and editing log pane of the Ontology Editor; (4) the Ticket Viewer and its Dialogue Box.

4.3 Perspective Manager

The Perspective Manager provides a Conflict Viewer, a Conflict Browser, and finally, a Perspective Analyser.

Conflict Viewer plugin allows the exploration of conflicts against reuse policies in a graphical way. All conflicts have a conflict code that uniquely identifies the conflict type, and a conflict ID that facilitates retrieving more detailed information about the conflict in the Conflict Browser. From within the Conflict Viewer the stakeholder can check a perspective against all the different combinations of reuse policies that were defined in [4]. As a stakeholder can have multiple perspectives, he must first select the perspective he prefers to check.

Conflict Browser Conflict Browser provides detailed information about all conflicts in a tree structure. The following table shows the meaning of three of the in total eleven defined conflict codes (Ci, 11 in total):

- C3: NewlyDefinedRelationConflict: a new relation was specified;
- C6: IntroduceTermConflict: a new term was introduced;
- C8: DefinedGenusWithNewConceptConflict: a genus was defined consisting of one or more newly introduced concepts.

When expanding the line, description of the involved concepts and relationships, and a cause for the conflict is revealed. This allows the DE in finding related conflicts that caused this particular conflict, and how it could be solved. Conflicts can be sorted based on conflict id, type, or cause. As an example, consider Fig. 3. The conflicts concern a perspective that has a specialisation reuse policy with a pattern in the UCO. When a specialisation policy holds, the DE is restricted to operators that specialise relationships by reusing concepts that were already defined in the UCO [4]. The first conflict with ID 6 has code C3, and indicates that a new relation is defined <Deelhandeling[5], resulteert in, resultaat van, Product>. The proposed solution is to drop the relation. The second conflict with ID 3 and code C6 indicates that a term Rapport[6] is introduced in the perspective while introducing new terms is restricted by the policy. The third conflict with ID 5 and code C8 was caused by conflict 6 as the introduction of <Rapport, is a, subsumes, Product"> involves a newly introduced term Rapport, which is not allowed. The Conflict Browser also notes that conflict 5 would be automatically resolved if conflict 3 is resolved.

Perspective Analyser plugin allows the Knowledge Worker to explore and analyse the differences and similarities between divergent stakeholder perspectives that were concurrently rendered on one original pattern. The plugin consists of two views, both illustrated in Fig. 4. When selecting a concept or relationship, summarising statistics are shown. When clicking on a term in a pattern, the Analyser parses for each stakeholder perspective the change log, and shows how the term evolved, illustrated on the bottom of Fig. 4.

[5] Partial Activity
[6] Report

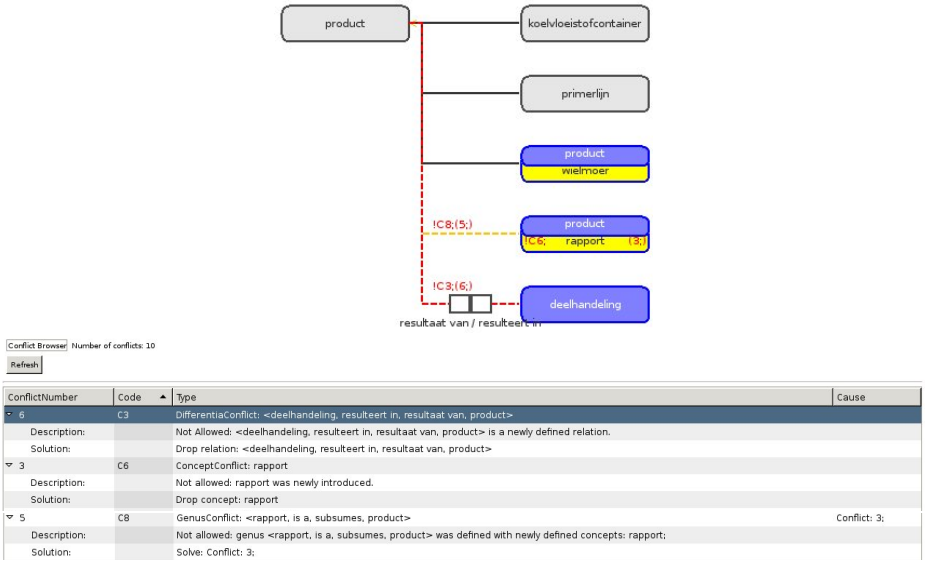

Fig. 3. The Perspective Manager Eclipse perspective.

4.4 Validation

We validated the DOGMA-MESS tools in the context of a realistic case study of the European CODRIVE[7] project. The CODRIVE project aims at contributing to a competency-driven vocational education by using state-of-the-art ontology methodology and infrastructure in order to develop a conceptual, shared and formal KR of competence domains. Domain stakeholders included educational institutes and public employment organisations from various European countries. The resulting shared "Vocational Competency Ontology" will be used by all partners to build interoperable competency models. All the samples in this paper were drawn from this case study. For an elaboration on the case study and design choices made for DOGMA-MESS, we ref to [2] and [3]

5 Discussion and Future Work

The current version of DOGMA Studio has a number of improvements on schedule. We plan to extend the colour palette in the Ontology Editor. E.g., red could be used for parts that are depricated. We also could use colour ranges to indicate the percentage of agreement on a certain concept in the UCO. The Community Manager will be further extended with an underlying community ontology that will give a semantic helicopter view on the current situation when creating tickets and patterns. First-class citizens of this ontology are inspired from related ontological frameworks. E.g., inspired by [8], a *community* a a special type of social system for which different directions and aims are

[7] CODRIVE is an EU Leonardo da Vinci Project (BE/04/B/F/PP-144.339).

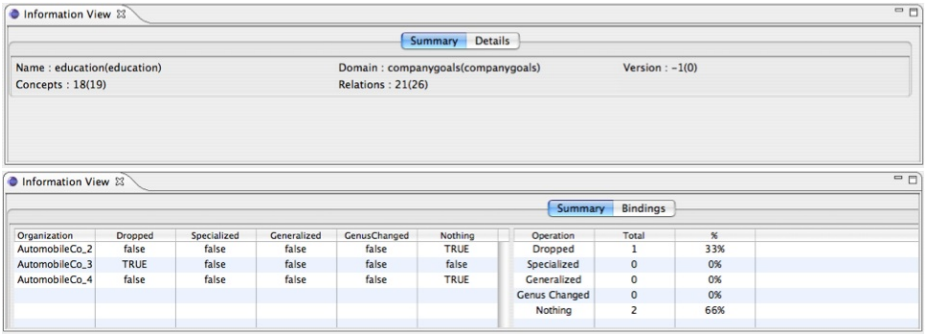

Fig. 4. Analysing Stakeholder Perspectives: the top view shows statsitics when selecting an arbitrary definition. The bottom view is shown when selecting a term in a pattern.

set, as well-established common *goals* towards which the community strives in order to create *added value* and which normally are accomplished by coherent (collaborative) *actions* that are performed by subscribed legitimate *stakeholders* and where these actions are aiming at changing the community state in a desired way. Other examples include SIOC[8]. In order to support the perspective unification process, we are currently implementing a meaning negotiation and argumentation module, that is inspired by related tools such as HCOME [14] and Diligent [18].

6 Conclusion

The problem in ontology engineering is not on what ontologies are, but how they become operationally relevant and sustainable over longer periods of time, and how proper methodology and tool support can be provided. DOGMA-MESS, extending the fact-oriented and layered ontology framework DOGMA, is a collaborative ontology evolution methodology that supports stakeholders in iteratively interpreting and modeling their common ontologies in their own terminology and context, and feeding back these results to the owning community. In this paper we extend DOGMA Studio with a set of MESS modules: Version Manager, Community Manager, and Perspective Manager.

References

1. D. Bach, R. Meersman, P. Spyns, and D. Trog. Mapping OWL-DL into ORM/RIDL. In Meersman et al. [16], pages 742–751.
2. S. Christiaens, P. De Leenheer, A. de Moor, and R. Meersman. Business use case: Ontologising competencies in an interorganisational setting. In M. Hepp, P. De Leenheer, A. de Moor, and Y. Sure, editors, *Ontology Management for the Semantic Web, Semantic Web Services, and Business Applications, from Semantic Web and Beyond: Computing for Human Experience*. Springer, 2008.

[8] htttp://www.sioc-project.org

3. P. De Leenheer. Meaningful competency-centric human resource management: a case study for Dogma Mess. In *Proc. of European Semantic Technology Conference 2008 (Vienna, Austria)*, 2008.
4. P. De Leenheer, A. de Moor, and R. Meersman. Context dependency management in ontology engineering: a formal approach. *LNCS Journal on Data Semantics*, 8:26–56, 2007.
5. P. De Leenheer and T. Mens. Using graph transformation formal collaborative ontology evolution. In *Proc. of Agtive (Kassel, Germany)*, volume 5088 of *LNCS*. Springer, 2007.
6. P. De Leenheer and T. Mens. Ontology evolution: State of the art and future directions. In M. Hepp, P. De Leenheer, A. de Moor, and Y. Sure, editors, *Ontology Management for the Semantic Web, Semantic Web Services, and Business Applications*. Springer, 2008.
7. A. de Moor, P. De Leenheer, and R. Meersman. DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering. In *In Proc. of the 14th Int'l Conference on Conceptual Structures (ICCS 2006) (Aalborg, Denmark)*, LNAI 4068, pages 189–203. Springer Verlag, 2006.
8. E.D. Falkenberg. FRISCO : A framework of information system concepts. Technical report, IFIP WG 8.1 Task Group, 1998.
9. T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
10. N. Guarino and P. Giaretta. Ontologies and knowledge bases. towards a terminological clarification. In N. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, 1995.
11. T. Halpin and T. Morgan. *Information Modeling and Relational Databases, Second Edition*. Morgan Kaufmann Publishers, 2008.
12. M. Jarrar. Mapping ORM into the SHOIN/OWL description logic. In Meersman et al. [16], pages 729–741.
13. M. Jarrar, J. Demey, and R. Meersman. On reusing conceptual data modeling for ontology engineering. *Journal on Data Semantics*, 1(1):185–207, 2003.
14. K. Kotis and G. Vouros. Human-centered ontology engineering: The HCOME methodology. *Knowledge and Information Systems*, 10:109–131, 2005.
15. R. Meersman. Semantic ontology tools in IS designs. In *In Proc. of the International Symposium on Methodologies for Intelligent Systems (ISMIS)*, pages 30–45, 1999.
16. R. Meersman, Z. Tari, and P. Herrero, editors. *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, OTM Academy Doctoral Consortium, MONET, OnToContent, ORM, PerSys, PPN, RDDS, SSWS, and SWWS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I*, volume 4805 of *Lecture Notes in Computer Science*. Springer, 2007.
17. I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company : How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, May 1995.
18. H. Pinto, S. Staab, and C. Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, Valencia, Spain, 2004.
19. P. Spyns, R. Meersman, and M. Jarrar. Data modelling versus ontology engineering. *SIGMOD Record*, 31(4):12–17, 2002.
20. D. Trog, J. Vereecken, S. Christiaens, P. De Leenheer, and R. Meersman. T-lex: A role-based ontology engineering tool. In R. Meersman, Z. Tari, and P. Herrero, editors, *OTM Workshops (2)*, volume 4278 of *Lecture Notes in Computer Science*, pages 1191–1200. Springer, 2006.
21. G. Verheijen and J. Van Bekkum. NIAM, an information analysis method. In *Proc. of the IFIP TC-8 Conference on Comparative Review of Information System Methodologies (CRIS 82)*. North-Holland, 1982.