A Tool for DILIGENT Argumentation: Experiences, Requirements and Design

Michael Engler
BT Research and Venturing
Ipswich, UK
michael.engler@bt.com

Denny Vrandečić
AIFB, University of Karlsruhe
Karlsruhe, Germany
vrandecic@aifb.uka.de

York Sure
AIFB, University of Karlsruhe
Karlsruhe, Germany
sure@aifb.uka.de

Abstract

Ontologies are defined as "the formal specification of a shared conceptualization of a domain of interest" [7]. Unlike most other ontology engineering methodologies, the DILIGENT methodology aims specifically at the collaborative and distributed building of ontologies, providing a rich argumentation framework in order to quickly proceed with building the ontology and tracking all relevant discussions about the conceptualization.

Based upon first experience with DILIGENT, we designed a tool to support engineers and domain experts to follow the DILIGENT processes. In this paper we present the experiences and requirements our work is based on, and present the resulting design and state of the ongoing implementation.

1 Introduction

The main value of an ontology comes from its "sharedness": the bigger the group of people that commit to the shared conceptualization formalized in an ontology, the more useful the ontology will render itself. Furthermore, ontologies should be ideally developed by domain experts with profound knowledge in ontology engineering - alas, such people are rare or sometimes even non-existent.

Ontology engineering methodologies need to cover the collaboration between domain experts and ontology engineers. As both are highly sophisticated experts in their own fields, the communication between them often can become harder than expected. It is crucial for the success of the ontology engineering project to follow a methodology that allows for the easy collaboration among heterogeneous and geographically distributed groups.

The DILIGENT methodology [16] for the *DIstributed, Loosely-controlled and evolvInG ENineering of oNTologies* fulfills the described requirements. Especially it provides an argumentation framework [12] that allows to capture and track the ongoing discussion related to engineering an ontology, which has proofed very useful within the case studies where DILIGENT was applied [9, 2, 3].

In order to follow DILIGENT more easily, we designed a tool to facilitate the application of the methodology. We followed [4] for the interaction design. The tool must be usable by domain experts, who may not have deep background on ontological issues, and still allow communication with ontology engineering experts. The tool covers both the DILIGENT process for evolving ontologies and the DILIGENT argumentation framework for discussing changes in an ontology.

We first take a look at related work in distributed ontology engineering (Sect. 2) and offer a short description of DILIGENT (Sect. 3). Then we write about the previous experiences with tools for DILIGENT, and the personas and scenarios we created (Sect. 4) and used in order to derive the tool's requirements (Sect. 5). Based on this we design a work model and the DILIGENT tool (Sect. 6), which is now being implemented (Sect. 7). We end with an outlook to future work and open research issues (Sect. 8).

2 Related work

In this work we aim at amending existing ontology editors with the facility to discuss changes in an ontology according to [9]; it is not our goal to provide a full-fledged ontology editor or even ontology engineering environment [10]. Whereas most ontology editors include some support for collaborative editing [11], almost none of them offer to track the changes and discuss them, with the notable exception of SWOOP[1]. SWOOP allows to send changes to other users and to annotate those changes. Our implementation, which is based on OntoStudio[2] progresses

[1] `http://www.mindswap.org/2004/SWOOP/`
[2] `http://www.ontoprise.de/content/e3/e43/index_eng.html`

from the approach implemented in SWOOP, taking into account methodological results from the research and evaluation done on DILIGENT.

3 Preliminaries

As the tool design described in this work aims at supporting the DILIGENT methodology, in this section we briefly introduce the parts relevant for our work. As shown in Figure 1 it comprises five main activities: (1) **build**, (2) **local adaptation**, (3) **analysis**, (4) **revision**, and (5) **local update**. In contrast to known ontology engineering methodologies [5, 6, 8, 14] our focus is on distributed ontology engineering. The process starts with *domain experts*, *users*, *knowledge engineers*, and *ontology engineers* who **build** an initial ontology.

The *build* phase aims at creating an initial version of the ontology quickly, so that the stakeholder can start using the ontology soon. During the next phase users *locally adapt* the ontology according to their own needs, while the ontology is in use e.g. to organise knowledge. The *analysis* phase requires the ontology control board to evaluate the changes suggested by the stakeholders. Then the board *revises* the ontology by deciding, which changes are applied to the ontology. In the last step the stakeholders *update* their local ontologies based upon the revised version of the ontology. For a detailed description of DILIGENT, please refer to [16].

We believe our tool will be most helpful in the *build*, *analysis*, and *revision* phase. In the build phase it will help to capture the intention, why a conceptualisation has been chosen. During *analysis* phase the tool helps the board members to understand the different conceptualisations and argumentations users have brought forward. When the board actually decides in the *revision* phase which changes will be applied to the next version of the ontology our tool helps to capture the pro and contra arguments regarding certain conceptualisations and helps to reach a decision

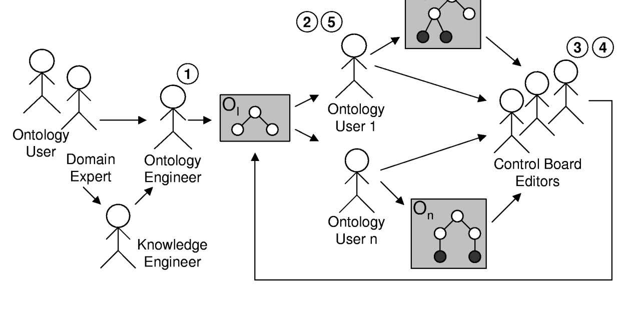


Figure 1. DILIGENT overview

through a voting functionality.

The other part of the DILIGENT methodology that our work is based on is the DILIGENT argumentation framework. Derived from previous work and experiments, the most effective argument types for ontology engineering decisions where identified, and formalized in an ontology. Here we give an overview of the relevant terms, as we define it for our work. **Issues** are high-level problems that need to be solved in order to create a useful ontology. An **idea** is a possible ontology change operations that resolves an issue. It is not the actual ontology change operation itself – this will only be implemented when an idea is agreed on, i.e. in the *revision* step. Idea sets capture different conceptualizations: they occur in the application if the discussion comes to a point where it is necessary to introduce conflicting ways of modelling parts of the ontology, i.e. the solution to an issue, and they collect a number of ideas. An **argument** supports or disagrees with an idea or idea set, or even an issue as a whole. With arguments users express opinions about the modelling of the ontology. The following argument types are introduced by the argumentation framework: *Alternative*, *Counterexample*, *Example*, and *Evaluation* (i.e. a *Justification*). For details of the argumentation framework, please refer to [12].

4 Experiences

We applied distributed ontology engineering and evolution within several case studies in the SEKT project. The PROTON ontology[3] [13], an upper level ontology, and several domain ontologies evolved during the course of the project. The evolution of all these ontologies involved geographically distributed partners, scattered around Europe, which strongly suggested the use of DILIGENT as ontology engineering process.

While designing these ontologies we found several difficulties, which heavily influenced the design of the DILIGENT tool. In a first attempt to support the ontology engineering a wiki was used to create a space for discussing the design of the ontology. The use of a wiki allowed easy task tracking and to trace the discussions of users about ontological entities, which proofed extremely valuable.

A problem with the wiki was the lack of ontology visualisation facilities. Users undertook great efforts to circumvent this: they created a visualisation in an ontology editor, took screen shots and posted these pictures in the wiki.

Also the lack of a notification mechanism in the used wiki software led to the continued use of an existing mailing list for discussing the ontology, rather than the wiki, because it seemed easier and more efficient to the users.

Based on this experience, we decided to develop a tool

[3]`http://proton.semanticweb.org`

to support the DILIGENT methodology. To ensure a high-quality tool design, which captures all important requirements from the users point of view we used several methods for interaction design as suggested in [4]. More specifically we created personas, wrote scenarios, and designed low-fidelity screen mock-ups (see Figure 2) to test the coherence of our interaction design. Afterwards the results were summarised in a Style Guide [17] describing the application design. The main persona used for the application design describes a domain expert, who is neither skilled in ontology design nor in computer science. The main scenario we explored concerned *Arguing over an ontology in an online-setting*, which focuses on how geographically distributed users can collaboratively create an ontology. We also explored more briefly the scenario *Arguing over an ontology in a meeting*, where users started out with a kick-off meeting, but as time runs out move the discussion into an online-setting. For more details and further scenarios refer to [17].

The main design requirements from these scenarios were the necessity to track tasks, notify users about new tasks, and handle the sources from which the ontology is derived. Most important though is a functionality to organise different possible conceptualisations, which can then be discussed by users with the help of an elaborate discussion mechanism. A user needs also to be able to express her opinion very quickly by voting on conceptualisations and thus, upon reaching consensus, change the ontology.

We have previously experimented with wikis for the ontology engineering task, but consider the text interface of a wiki not to be adequate for the development of an expressive ontology by a novice user. Semantic wikis [15] aim at the easy population of knowledge bases, but only consider a very small fragment of current web ontology languages.

5 Requirements

From our above described experiences and the carefully crafted scenarios, we came up with the following requirements for our application.

1. The application needs to support ontology design for geographically distributed participants.
2. To avoid conflicting changes to the discussion and the ontology, both need to be propagated to users in near real-time.
3. The requirements for the ontology need to be traceable throughout the application.
4. The discussion about the ontology design decisions needs to be traceable throughout the whole life cycle of an ontology.
5. In line with the DILIGENT Argumentation model issues, ideas, and elaborations need to be supported as means for guiding and tracing the discussion.
6. Arguments need to be classifiable according to the DILIGENT argumentation framework as "Evaluation", "Alternative", "Example", or "Counterexample" to focus the discussion on relevant arguments.
7. The DILIGENT Argumentation Ontology [12] needs to be employed to annotate the discussion about the ontology.
8. For expressing which ontology change operations are acceptable to team members, a voting mechanism is required.
9. Users need to be able to track which sources, issues, ideas, and discussions they have and have not dealt with.
10. Users need to be notified in an unobtrusive way of new developments in the discussion and the ontology.
11. The discussed ontology needs to be visualised to allow users an easier understanding of the ontological entities they are discussing.
12. Sources, issues, and ideas need to be sortable in a coherent way.
13. The application needs to be easy to learn rather than easy to use to allow none-technical domain experts to take part in the ontology engineering.

The bottom line is that the application needs to be easier to use than the alternatively used tools such as a wiki, a mailing list, or a forum.

6 Work Model and Application Design

This section focuses on the work model and the application design. The work model describes in an abstract way how a user interacts with the software. Afterwards other important aspects of the application design are discussed. Generally our thoughts on the design were guided by the requirement for ease of learning (see requirement 13). Whenever we had to decide between ease of use and ease of learning, we decided for ease of learning. Furthermore we looked at how users could learn how to use the application in the scenario *Learning the application* (see [17]).

The applications work model roughly consists of the steps *importing sources*, *raising issues and creating ideas*, *review and discussion*, and *voting*. The order of listing the steps corresponds to the actual order in the workflow, even though the order is not inherently fixed. All of the described steps are managed by tasks being assigned to participants of the ontology engineering process.

Before working with the application, the requirements and relevant *sources* for the ontology need to be identified. Sources can be, e.g. *competency questions*, which define what kind of questions the knowledge in the ontology is able to answer. Competency questions and other documents are referenced in the application by an URI allowing to manage them from within the application, and enabling to track ontological requirements in later stages of the ontology engineering process (see requirement 3).

After the import of the sources, *tasks* are created and assigned to the participants. This helps to ensure that all sources are utilized for the ontology design. From the sources the users create *issues*. Issues describe a problem or piece of knowledge which needs to be taken into account during the design of the ontology. When a fair set of issues has been created, the users start developing *ideas* on how to conceptualise the domain based on the created issues. Every idea is a possible ontology change operation or, in other words, an ontological modelling primitive, which is possibly added or changed in the ontology. This means, users express their ideas by provisionally adding modelling primitives to the ontology. If ideas are contradicting, the user needs to create a new *idea set*. In an idea set a user is able to remove some modelling primitives and replace them by other modelling primitives (see requirement 5). These idea sets allow to visualise different conceptualisations of the ontology, of which one will be selected to be the agreed conceptualisation of the domain.

The participants of the ontology engineering process are then able to exchange *arguments* about the ideas on how to design the ontology. The application captures these arguments, which allows to trace the discussion about design decisions (see requirement 4). In order to ensure an efficient discussion about how to design the ontology, the types of the arguments users are allowed to bring forward are restricted. We only allow the argument types "Evaluation", "Contrast", "Example", and "Counterexample", as these were evaluated to be the most effective ones [12]. Every argument is classified by the user (see requirement 6). The actual decision if modelling primitives are included into the ontology is determined either by "democratic" voting of the participants or by the ontology owner, who approves of an idea or idea set (see requirement 8). The classification of the users' arguments is stored by the application in an Argumentation Ontology as described in [12]. This allows to integrate the system with automatic ontology learning or ontology evaluation systems, as they can easily use the formalized argumentation framework. It also allows us to preserve the existing discussion and use it as an input for future revisions of the ontology (see requirement 7).

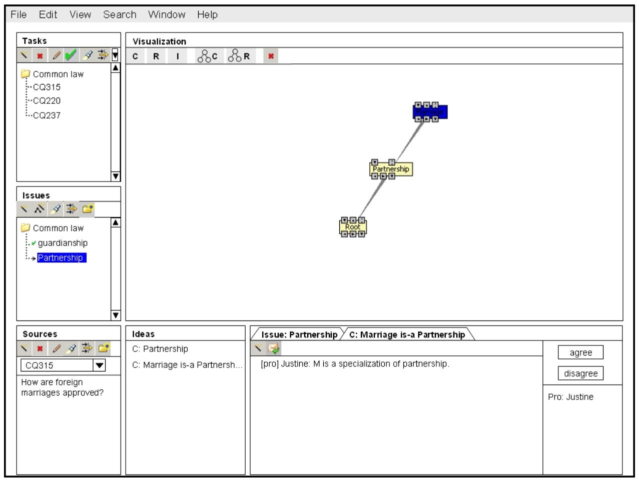
Ontology development can proceed in parallel with several users. This means, users are able to create sources, tasks, issues, ideas, and arguments simultaneously. This can lead to conflicts e.g. when two team members reply to an argument more or less simultaneously not being able to see the argument of the other team member. This might create contradicting overlapping arguments. In the application we refrain from locking argumentations on issues and ideas while they are edited by another person because we expect the argumentation to regulate itself in this respect. It might turn out though that a locking mechanism is required to avoid conflicts (see requirement 2).

The order of the events e.g. when sources, tasks, issues, ideas, and arguments are created, is not strictly set. An issue may arise rather late in the design process. This implies that ideas, which are already approved, can be revised if necessary. If a new issue, idea, or argument is created, all other team members except for the creator are assigned a new task to review the new idea or argument on the already agreed part of the ontology.

After having described above the work model of the application, we will continue with describing some other important parts of the application design. An obvious design choice is the usage of a server which stores and controls the ontology and which can be accessed from an internet or intranet connection. The server stores the ontology and propagates changes in the ontology in near real-time to other users so that the likelihood of the above described conflicting edits is low. Therefore users are required to always have a network connection to the server to receive updates on the ontology and the discussion. With the chosen design the application supports the requirements 1 and 2 regarding geographical distribution of the application users' as well as the avoidance of conflicting changes to the argumentation.

The application design shows a sophisticated notification mechanism allowing users to follow the discussions while not logged on to the server. The user has the choice to get notified by mail (either for each individual task she is assigned to or by a daily digest). Another possibility is to be notified by instant messaging. All notifications are stored as tasks within the application, so they cannot be lost due to a communication failure. Whenever appropriate the application creates these tasks automatically (e.g. when a new argument is used). A user can also create new tasks and assign them to himself and other users. Tasks are generally a way of bringing something to the attention of other users. The task construct and the way tasks bring changes to the attention of other participants fulfill the requirements 10 and 9. As we regard the ontology TBox in the application we expect that the amount of tasks will be rather small, as TBoxes are generally more stable and smaller than ABox ontologies, i.e. knowledge bases.

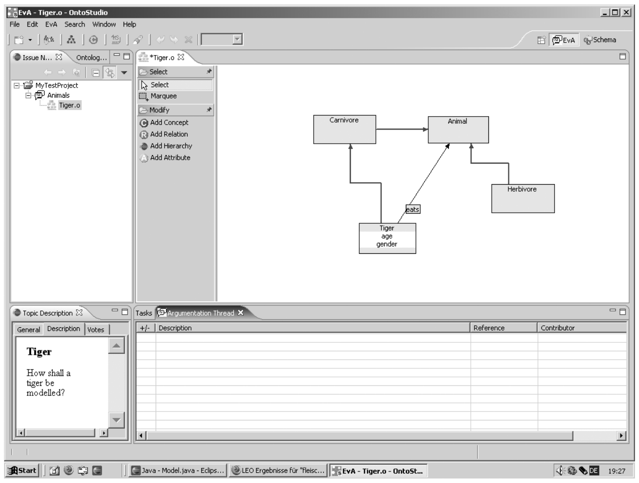
As of now we are not sure how effective voting will be for gathering consensus. Therefore we are carefully encap-



How are foreign marriages approved?

[pro] Justine: M is a specialization of partnership.

Figure 2. Low-fidelity screen design.



Tiger

How shall a tiger be modelled?

Figure 3. EvA screenshot

sulating the voting mechanism so that it can be changed easily in later versions of the tool. At the moment we envision functionality which allows to set a threshold on how many of the participants of the ontology engineering process have to agree to an ontology change operation. In some cases it might also be desirable to give only the owner of the ontology the power to approve or discard ideas. She would use the voting as an indicator to gather the opinion of the participants.

According to requirement 11 the application allows to visualise the initial ontology as well as different idea sets, i.e. other, contradicting, proposed ways to model parts of the ontology. The visualisation shows if the participants voted in favour or against an idea or idea set. When an issue is selected ideas and idea sets are highlighted so that they are clearly distinguishable from the surrounding modelling primitives. To compare different idea sets the user can switch between idea sets and see how they fit into the surrounding modelling primitives. Especially the requirement for the visualisation of the ontology suggests to use a rich client to cope with a sophisticated, flexible visualisation.

For organising hundreds or thousands of sources, tasks and issues we use a hierarchical folder structure. This allows to cluster the different sources according to different general topics, which need to be dealt with in the ontology design. All issues and tasks, which are related to a source are stored in a folder with the same name (see requirement 12). We expect this to be sufficient because otherwise the structuring of the sources and issues would already anticipate the design of the ontology. This would be problematic as this is to be made explicit and discussed by the users during the design process and not already by sorting sources. Having stated this, it might be useful though to introduce reference links between sources as well as issues.

7 Implementation

After the review of the tool design by potential users, i.e. case study partners of the SEKT project, an implementation called *EvA – Evolution Annotater* is ongoing. It builds upon OntoStudio, an Eclipse-based ontology engineering environment. EvA especially uses Eclipses Graphical Editor Framework GEF for the persistent visualisation of ideas, and the Argumentation Ontology [12] to safe the discussion about the ontology.

Figure 2 displays a screen of the low-fidelty prototype we used in validating the applications design. This prototype was shown to a small number of potential users. Their feedback went straight back into the design. In Figure 3 a screen shot from the on-going implementation is shown. Both figures show the main window of the application with views on sources, tasks, ideas, issues, the argumentation about ideas and issues, as well as the main window for the manipulation of the ontology.

8 Conclusions and Future Work

We have described the initial need of an ontology engineering environment to support the remote engineering and maintenance of ontologies, which became apparent during the SEKT project. We have shown that off the shelf tools such as wikis, mailing lists, forums are not sufficient for ontology engineering mainly due to the lack of visualisation and notification. Furthermore we have designed a specialised application to address the above described problems by employing an interaction design process e.g. suggested in [4]. This should improve the usability of the tool for none-technical experts such as domain experts.

An open issue is the voting mechanism. It is unclear yet in which environment which threshold (e.g. a simple majority, all participants need to agree, etc.) needs to be reached to accept a conceptualisation. Another open issue is the possibility to automatically classify arguments of the ontology engineering process according to the DILIGENT argumentation model. Letting agents take part in the discussion e.g. for evaluation of the ontology or for suggesting new concept by the use of ontology learning algorithm is yet open to research.

The design is being implemented right now, and we hope to test its value in a case study soon. Therefore we look for further reviews of the design from expert reviewers and especially the actual application of the tool in real world use cases.

Acknowledgements

Research reported in this paper was supported by the IST Programme of the European Community under the SEKT project, Semantically Enabled Knowledge Technologies (`http://www.sekt-project.com`, IST-1-506826-IP). We want to thank our colleagues for fruitful discussions and inspirations, especially Christoph Tempich, H. Sofia Pinto, Núria Casellas-Caralt and Daniel Politze. This publication only reflects the authors' views.

References

[1] C. Bussler, J. Davies, D. Fensel, and R. Studer, editors. *The Semantic Web: Research and Applications*, volume 3053 of *LNCS*, Heraklion, Crete, Greece, May 2004. Springer.

[2] P. Casanovas, N. Casellas, M. Poblet, J.-J. Vallbé, Y. Sure, and D. Vrandečić. Iuriservice ii ontology development. In P. Casanovas, P. Noriega, D. Bourcier, and V.R.Benjamins, editors, *XXII World Congress of Philosophy of Law and Social Philosophy - Workshop on Artificial Intelligence and Law: The regulation of electronic social systems. Law and the Semantic Web*, number B4 in Special Workshop, pages 327–328, Granada, Spain, MAY 2005. International Association for Philosophy of Law and Social Philosophy, University of Granada. Artificial Intelligence and Law.

[3] N. Casellas, M. Blázquez, A. Kiryakov, P. Casanovas, and R. Benjamins. OPJK into PROTON: legal domain ontology integration into an upper-level ontology. In R. M. et al., editor, *OTM Workshops 2005, LNCS 3762*, pages 846–855. Springer-Verlag Berlin Heidelberg, 2005.

[4] A. Cooper and R. M. Reimann. *About Face 2.0: The Essentials of Interaction Design*. Wiley and Sons, 2003.

[5] A. Gangemi, D. Pisanelli, and G. Steve. Ontology integration: Experiences with medical terminologies. In N. Guarino, editor, *Formal Ontology in Information Systems*, pages 163–178, Amsterdam, 1998. IOS Press.

[6] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Advanced Information and Knowlege Processing. Springer, 2003.

[7] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[8] H. S. Pinto and J. P. Martins. A methodology for ontology integration. In *Proceedings of the First International Conference on Knowledge Capture (K-CAP2001)*, pages 131–138, New York, 2001. ACM Press.

[9] H. S. Pinto, S. Staab, Y. Sure, and C. Tempich. OntoEdit empowering SWAP: a case study in supporting DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies (DILIGENT). In Bussler et al. [1], pages 16–30.

[10] Y. Sure and J. Angele, editors. *Proceedings of the First International Workshop on Evaluation of Ontology based Tools (EON 2002)*, volume 62 of *CEUR Workshop Proceedings*, Siguenza, Spain, 2002. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-62/.

[11] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In I. Horrocks and J. A. Hendler, editors, *Proceedings of the First International Semantic Web Conference: The Semantic Web (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science (LNCS)*, pages 221–235, Sardinia, Italy, 2002. Springer.

[12] C. Tempich, H. S. Pinto, Y. Sure, and S. Staab. An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (diligent). In *Second European Semantic Web Conference, ESWC 2005*, volume 3532 of *LNCS*, pages 241–256, 2005.

[13] I. Terziev, A. Kiryakov, and D. Manov. Base upper-level ontology (bulo) guidance. SEKT deliverable 1.8.1, Ontotext Lab, Sirma AI EAD (Ltd.), 2004.

[14] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, Montreal, Canada, 1995.

[15] M. Völkel, M. Krötzsch, D. Vrandečić, H. Haller, and R. Studer. Semantic wikipedia. In *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, May 23-26, 2006*, MAY 2006.

[16] D. Vrandečić, H. S. Pinto, Y. Sure, and C. Tempich. The diligent knowledge processes. *Journal of Knowledge Management*, 9(5):85–96, Oct 2005.

[17] D. Vrandečić, Y. Sure, C. Tempich, and M. Engler. Sekt methodology: Initial best practices and lessons learned in case studies. SEKT formal deliverable 7.2.1, Institute AIFB, University of Karlsruhe, DEC 2005.