Ontology Engineering Revisited: An Iterative Case Study*

Christoph Tempich[2], H.Sofia Pinto[1], and Steffen Staab[3]

[1] Dep. de Engenharia Informática, Instituto Superior Técnico, Lisboa, Portugal
sofia.pinto@dei.ist.utl.pt
[2] Institute AIFB, University of Karlsruhe (TH), 76128 Karlsruhe, Germany
tempich@aifb.uni-karlsruhe.de
[3] ISWeb, University of Koblenz Landau, 56016 Koblenz, Germany
staab@uni-koblenz.de

Abstract. Existing mature ontology engineering approaches are based on some basic assumptions that are often violated in practice, in particular in the Semantic Web. Ontologies often need to be built in a *decentralized* way, ontologies must be given to a community in a way such that individuals have *partial autonomy* over them and ontologies have a life cycle that involves an *iteration* back and forth between construction/modification and use. While recently there have been some initial proposals to consider these issues, they lack the appropriate rigor of mature approaches. i.e. these recent proposals lack the appropriate depth of methodological description, which makes the methodology usable, and they lack a proof of concept by a long-lived case study. In this paper, we revisit mature and new ontology engineering methodologies. We provide an elaborate methodology that takes decentralization, partial autonomy and iteration into account and we demonstrate its proof-of-concept in a real-world cross-organizational case study.

1 Introduction and Motivation

Ontologies are used in order to improve the quality of communication between computers, between humans and computers as well as between humans. An ontology is an agreement supporting such communication and this agreement must be constructed in a comprehensive ontology engineering process. There are several mature methodologies that have been proposed to structure this process and thus to facilitate it (cf. [1, 2, 3]) and their success has been demonstrated in a number of applications.

Nevertheless, these methodologies make some basic assumptions about the ontology engineering process and about the way the resulting ontologies are used. In practice, we observe that these methodologies neglect some important issues:

Decentralization: The methodologies do not take into account that even a medium sized group of stakeholders of an ontology is often quite *distributed* and does not necessarily meet often or easily.

* Research reported in this paper has been financed by EU in the IST projects SEKT (IST-2003-506826) and aceMedia (IST-FP6-001765).

Y. Sure and J. Domingue (Eds.): ESWC 2006, LNCS 4011, pp. 110–124, 2006.

Partial Autonomy: Users of an ontology are typically forced to use an ontology as is or to forget about it. A typical situation that we have encountered was that people want to retain a part of the shared ontology and *modify it locally*, i.e. personalize it.

Iteration: The methodologies mention the problem of evolving the ontology, but the cases that support the methodologies are typically cases where the construction phase of the ontology strictly precedes the usage phase of the ontology while we often see the need for interleaving ontology construction and use. Moreover, there is a lack of case studies that support hypothesis about how to iterate in the ontology *evolution* process.

These issues arise naturally for many ontologies and one might claim for all ontologies in the Semantic Web! Recently a number of approaches that touch these issues have been proposed [4, 5, 6] — among them our own, DILIGENT. However, **none** of these approaches *elaborated* their methodological description or *tested* their proposals in a case study with regard to Decentralization, Partial Autonomy and Iteration between the definition and the use of an ontology.

In this paper, we present our approach, DILIGENT. It is based on the process model described in [4]. We add substance to existing proposals by, *(i)*, specifying the internal structure of methodology stages[1] (i.e. their input, output, decision points, actions to be taken in each stage, and available tool support) and, *(ii)*, by providing a comprehensive case study that takes Decentralization, Partial Autonomy as well as Iteration between ontology construction/modification and usage seriously in a real-world case study of 3 months duration, where the ontology was the driving factor of a cross-organizations peer-to-peer knowledge management platform.

In the following, we first revisit ontology engineering methodologies to describe our starting point (Section 2). In Section 3, we survey our way of arriving at the methodology described here. Then we describe the refinements, adaptations and extensions we made to related methodologies in Section 4. Because of space restrictions, this description can only highlight some of our methodological improvements. For the full description we refer the reader to a technical report [7]. We evaluate DILIGENT by comparing it in detail to other methodologies (Section 5) and by validating it through a case study (Section 6) that shows a concrete instantiation of our methodology including two full iterations of the ontology life cycle.

2 Related Work

In the past, there have been OE case studies involving dispersed teams, such as $(KA)^2$ ontology [6] or [8]. However, they usually involved tight control of the ontology, of its development process, and of a small team of ontology engineering experts that could cope with the lack of precise guidelines.

Established methodologies for ontology engineering summarized in [1, 2, 3], focus on the centralized development of static ontologies, i.e. they do not consider iteration between construction/modification and use. **METHONTOLOGY** [1] and the **OTK**

[1] To facilitate reading, we introduce here a convention to refer to parts of processes. We call a larger part a ‘process stage’ or ‘stage’ and a smaller part, which is also a part of a stage, a ‘process action’ or ‘action’.

methodology [2] are good examples for this approach. They offer guidance for building ontologies either from scratch, reusing other ontologies as they are, or re-engineering them. They divide OE processes into several stages which produce an evaluated ontology for a specific domain. **Holsapple et al.** [9] focus their methodology on the collaborative aspects of ontology engineering but still aim at a static ontology. A knowledge engineer defines an initial ontology which is extended and modified based on the feedback from a panel of domain experts. **HCOME** is a methodology which integrates argumentation and ontology engineering in a distributed setting [5]. It supports the development of ontologies in a decentralized setting and allows for ontology evolution. It introduces three different spaces in which ontologies can be stored: In the *Personal Space* users can create and merge ontologies, control ontology versions, map terms and word senses to concepts and consult the top ontology. The evolving personal ontologies can be shared in the *Shared Space*. The *Shared Space* can be accessed by all participants. In the shared space users can discuss ontological decisions. After some discussion and agreement, the ontology is moved into the *Agreed space*. However, they have neither reported that their methodology had been applied in a case study nor do they provide any detailed description of the defined process stages.

3 Developing the New DILIGENT OE Methodology

In order to arrive at a sound OE methodology we have proceeded in five steps to develop DILIGENT. First, we built on [8] to conceive our initial DILIGENT framework. Second, this framework contained a step for initially constructing a core ontology. With regard to this step, we decided not to develop a new methodology but to adopt the OTK methodology. Third, in order to validate the combined methodology, we analyzed its potential for the past (and ongoing) development process that has led to the biological taxonomy of living species and we conducted a lab experiment case study (cf. [10]). Fourth, we started a real-life case study and reported about its initial state and supporting means in [4]. Fifth, by the sum of these initial methodologies, cases and experiments, we arrived at the new and refined DILIGENT methodology that we present here. The focus of the refinement has been on decentralization, iteration and partial autonomy as well as on guiding users who were not ontology engineering experts. The methodology has been validated by the iterative case study presented in Section 6. Thus, we could repeatedly switch between hypothesis formulation and validation in order to present the result of step five and its validation in the remainder of this paper.

4 The DILIGENT Methodology

In order to give the necessary context for the detailed process description as depicted in Fig. 1 we start by summarizing the overall DILIGENT process model.

4.1 General Process

The **DILIGENT** process [4] supports its participants, in collaboratively building one shared ontology. The process comprises five main activities: (I) **build**, (II) **local adaptation**, (III) **analysis**, (IV) **revision**, (V) **local update**. The process starts by having *domain experts*, *users*, *knowledge engineers* and *ontology engineers* **build**ing an initial ontology. It proposes that the team involved in building the initial ontology should be relatively small, in order to more easily find a small and consensual first version of the shared ontology. At this point, it is not required to arrive at an initial ontology that would cover the complete domain. Once the initial ontology is made available, users can start using it and **locally adapting** it for their own purposes. Typically, due to new business requirements, or user and organization changes, their local ontologies evolve. In their local environment they are free to change the reused shared ontology. However, they are not allowed to directly change the ontology shared by all users.

A board of ontology stakeholders **analyzes** the local ontologies and the users' requests and tries to identify similarities in their ontologies. At this point it is not intended to merge all local ontologies. Rather changes to local ontologies will be analysed by the board in order to decide which changes introduced or requested by the users will be introduced. Therefore, a crucial activity of the board is deciding which changes are going to be introduced in the next version of the shared ontology. A balanced decision that takes into account the different needs of user's evolving requirements has to be found. The board should regularly **revise** the shared ontology, so that the parts of the local ontologies overlapping the domain of the shared ontology do not diverge too far from it. Therefore, the board should have a well-balanced and representative participation of the different kinds of participants involved in the process, which includes *ontology engineers*, *domain experts* and *users*.[2] Once a new version of the shared ontology is released, users can **update** their own **local** ontologies to better use the knowledge represented in the new version. The last four stages of the process are performed in a cyclic manner: when a new common ontology is available a new round starts again.

4.2 DILIGENT Process Stages

In order to facilitate the application of ontology engineering processes in real settings, DILIGENT had to be detailed to provide guidance to its participants. For this purpose, we have analyzed the different process stages in detail. For each stage we have identified (i) major roles, (ii) input, (iii) decisions, (iv) actions, (v) available tools, and (vi) output information. One should stress that this elaboration is rather a recipe or check list than an algorithm or integrated tool set. In different contexts it may have to be adapted or further refined to fit particular needs and settings. Tools may need to be integrated or customized to match requirements of the application context. In Fig. 1 we sketch our results, which are presented in the following. For lack of space we refer the reader to a technical report that includes a more detailed description of all items depicted in Fig. 1 [7]. In this paper we consider 'Local Adaptation', 'Revision', and 'Local Update' at an abstract level, zoom only into the 'Analysis' stage in an exemplary fashion and we omit the 'Build' stage, which is well-covered by existing methodologies [1, 2].

Local Adaptation

Roles: The actors involved in the local adaptation step are users of the ontology. They use the ontology e.g. to retrieve documents which are related to certain topics modelled in the ontology or more structured data like the projects an employee was involved in.

[2] These are roles that may overlap.

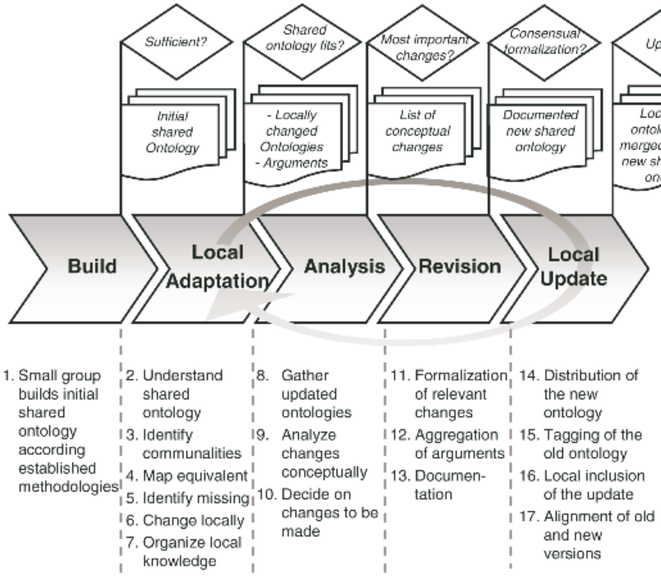

Sufficient?
Shared ontology fits?
Most important changes?
Consensual formalization?
Update?
Initial shared Ontology
- Locally changed Ontologies
- Arguments
List of conceptual changes
Documented new shared ontology
Local ontology merged with new shared one
Build
Local Adaptation
Analysis
Revision
Local Update
1. Small group builds initial shared ontology according established methodologies
2. Understand shared ontology
3. Identify communalities
4. Map equivalent
5. Identify missing
6. Change locally
7. Organize local knowledge
8. Gather updated ontologies
9. Analyze changes conceptually
10. Decide on changes to be made
11. Formalization of relevant changes
12. Aggregation of arguments
13. Documentation
14. Distribution of the new ontology
15. Tagging of the old ontology
16. Local inclusion of the update
17. Alignment of old and new versions

Fig. 1. Process stages (1-5), actions (1-17) and structures

Input: Besides the common shared ontology, in the local adaptation step the information available in the local information space is used. This can be existing databases, ontologies or folder structures and documents.

Decisions: The actors must decide which changes they want to make to their local ontology. Hence, they must decide if and where new concepts are needed and which relations a concept should have. They should provide reasons for their decisions.

Actions: To achieve the desired output the user performs different actions namely (2) *Understand shared ontology*, (3) *Identify communalities between own and shared conceptualization*, (4) *Map equivalent conceptualizations of different actors*, (5) *Identify missing conceptualizations*, (6) *Change conceptualization* and finally (7) *Organize local knowledge according to the conceptualization*.

The last three actions of the process step are performed in a cyclic manner until a new common ontology is available and the entire process step starts again. The single actions performed manually would require a grounded understanding of ontologies and their underlying formal representation. We cannot expect such knowledge from all actors participating in the process. The process should rather be integrated seamlessly in the environment the user works in. Hence we now indicate for each of the actions some available technology to support the actors.

Tool support: Building is supported by existing ontology editors like [11]. In [4] we describe how existing structure on local machines can be utilized to facilitate the creation of ontologies. The tool supports thus actions (3) and (5). We have further integrated ontology mapping to support step (4). (6) is a manual step. (7) is currently a manual step, too, but it could be supported by semi automatic classification.

Output: The output of the process step is a locally changed ontology which better reflects the user's needs. Each change is supported by arguments explaining the reasons for a change. At this point changes are not propagated to the shared ontology. Only in the *analysis step* the board gathers all ontology change requests and their corresponding arguments to be able to evolve the common shared ontology in the *revision step*.

Analysis

In this stage, in the middle of the overall ontology engineering process, the board (cf. the description of DILIGENT in Sec. 2) analyzes incoming requests and observations of changes. The frequency of this analysis is determined based on the frequency and volume of changes to the local ontologies.

Roles: In the analysis stage we can distinguish three roles played by board members: (i) The domain expert decides which changes to the common ontology are relevant for the domain and which are relevant for smaller communities only. (ii) Representatives of the users explain different requirements from the usability perspective. At this stage, work is conducted at a conceptual level. (iii) The ontology engineers analyze the proposed changes from a knowledge representation point of view foreseeing whether the requested changes could later be formalized and implemented.[3]

Input: The analysis stage takes as input the ontology changes proposed and/or made by the participating actors. To be able to understand the change requests, users should provide their reasons for each request. Both manual and automated methods can be used in the previous stages. Besides of arguments by ontology stakeholders, one may here consider rationales generated by automated methods, e.g. ontology learning. The arguments underlying the proposed changes constitute important input for the board to achieve a well balanced decision about which changes to adopt.

Decisions: The board must decide which changes to introduce into the new shared ontology at the conceptual level. Metrics to support this decision are (i) the number of users who introduced a change in proportion to all users who made changes. (ii) The number of queries including certain concepts. (iii) The number of concepts adapted by the users from previous rounds.

Actions: To achieve the desired output the board takes different actions namely (8) *Gather locally updated ontologies and corresponding arguments*, (9) *Analyze the introduced changes* and (10) *Identify changes presumably relevant for a significant share of all actors*.

Tool support: In [4] we present an extension to an ontology editor, which supports actions (8) and (9) and (10). (8) Ontologies can be collected from the users in a peer-to-peer system. Different sorting and grouping mechanisms help the board to analyze the introduced changes systematically. The identification of relevant changes is in the end a community process. Here we support decision making by structured argumentation support as described in [12].

Output: The result is a list of major changes to be introduced that were agreed by the board. All changes which should not be introduced into the shared ontology are filtered. At this stage it is not required to decide on the final modelling of the shared ontology.

[3] In the revision stage.

We now detail each one of the proposed actions:[4]

(8) Gather locally updated ontologies and corresponding arguments: Depending on the deployed application the gathering of the locally updated ontologies can be more or less difficult. It is important that the board has access to the local changes from users to be able to analyze them. It might also be interesting not only to analyze the final user ontology, but also its evolution. However, with an increasing number of participants this in-depth analysis might be unfeasible. Since analysis takes place at the conceptual level, reverse engineering is usually an important technique to get the conceptual model from the formalized model [1]. To support users providing their reasons, an argumentation framework that focuses the user on the relevant arguments was developed *cf.* [12].

(9) Analyze introduced changes: The number of change requests may be large and also contradictory. First the board must identify the different areas in which changes took place. Within analysis the board should bear in mind that changes of concepts should be analyzed before changes of relations and these before changes of axioms. Good indicators for changes relevant to the users are (i) overlapping changes and (ii) their frequency. Furthermore, the board should analyze (iii) the queries made to the ontology. This should help to find out which parts of the ontology are more often used. Since actors instantiate the ontology locally, (iv) the number of instances for the different proposed changes can also be used to determine the relevance of certain adaptations.

(10) Identify changes presumably relevant for a significant share of all actors: Having analyzed the changes and having grouped them according to the different parts of the ontology they belong to, the board has to identify the most relevant changes. Based on the provided arguments the board must decide which changes should be introduced. Depending on the quality of the arguments the board itself might argue about different changes. For instance, the board may decide to introduce a new concept that better abstracts several specific concepts introduced by users, and connect it to the several specific ones. Therefore, the final decisions entail some form of evaluation from a domain and a usage point of view. The outcome of this action must be a reduced and structured list of changes that are to be accomplished in the shared ontology.

Revision

Roles: The ontology engineers from the board judge the changes from an ontological perspective more exactly at a formalization level. Some changes may be relevant for the common ontology, but may not be correctly formulated by the users. The domain experts from the board should judge and decide wether new concepts/relations should be introduced into the common ontology even so they were not requested by the users (who may be domain experts or not).

Input: The input for the revision phase is a list of changes at a conceptual level which should be included into the ontology.

Decisions: The main decisions in the revision phase are formal ones. All intended changes identified during the analysis phase should be included into the common ontology. In the revision phase the ontology engineer decides how the requested changes

[4] Such a detailed description is available for all actions, but mostly omitted for sake of brevity.

should be formalized. Evaluation of the decisions is performed by comparing the changes on the conceptual level with the final formal decisions. The differences between the original formalization by the users and the final formalization in the shared ontology should be minimal.

Actions: To achieve the desired output the members of the board, mainly its ontology engineers, perform different actions namely (11) *Formalization of the decided changes*, (12) *Aggregation of arguments* and (13) *Documentation*. Judging entails *Evaluation* of proposed changes from a knowledge representation/ontological point of view.

Tool support: For the revision phase we do not envision any special tool support beyond the one provided by classical ontology engineering environments.

Output: The revision phase ends when all changes are formalized and well documented in the common ontology.

Local Update

Roles: The local update phase involves only the users. They perform different actions to include the new common ontology into their local system before they start a new round of local adaptation.

Input: The formalized ontology including the most relevant change request is the input for this step. We also require as an input the documentation of the changes. For a better understanding the user can request a delta to the original version.

Decisions: The user must decide which changes he will introduce locally. This depends on the differences between the own and the new shared conceptualization. The user does not need to update his entire ontology. This stage interferes a lot with the next local adaptation stage. We do not exclude the possibility of conflicts and/or ambiguity between local and shared ontologies, which may entail reduced precision if the ontology is being used in IR applications.[5]

Actions: To achieve the desired output the user takes different actions namely (14) *Distribution of the new ontology to all actors*, (15) *Tagging of the old ontology* to allow for a roll back, (16) *Local inclusion of the updated version* and (17) *Alignment of old and new versions*.

Tool support: The Local update stage is very critical from a usability point of view. Changes cannot be introduced without the user's agreement. Further he should not be bothered too often. In case of equivalent but local conceptualizations it must be possible to change to the common conceptualization. From a technical point of view this stage is supported by tools like KAON *cf.* [13].

Output: The output of the local update phase is an updated local ontology which includes all changes made to the common ontology. However, we do not require the users to perform all changes proposed by the board. The output is not mandatory, since the actors could change the new ontology back to the old one in the local adaptation stage.

[5] Ideally one should be able to blacken out the ambiguous parts like in multilevel databases. This has not been transferred to OE yet.

5 Comparison with Related Methodologies

In table 1 we compare DILIGENT to other well known methodologies. We have adapted the categorization of [1] separating *Management of the OE process activities*,[6] *Ontology development oriented activities* and *Ontology support activities*. To the original classification we have added the aspects of **Evolution**, different **Knowledge acquisition** modes and stages during **Documentation**.

Table 1. Summary of ontology engineering methodologies adapted from [1]

Feature			METHON-TOLOGY	On-To-Knowledge (OTK)	HCOME	DILIGENT
Management of OE process activities	Scheduling		Proposed	Described	NP	from OTK
	Control		Proposed	Described	NP	from OTK
	Quality assurance		NP	Described	NP	from OTK
Ontology development oriented activities	Pre development processes	Environment study	NP	Proposed	NP	from OTK
		Feasibility study	NP	Described	NP	from OTK
	Development processes	Specification	Descr. in detail	Descr. in detail	Proposed	Described
		Conceptualization	Descr. in detail	Proposed	Proposed	Descr. in detail
		Formalization	Described	Described	Proposed	Descr. in detail
		Implementation	Descr. in detail	Described	Proposed	Described
	Post development processes	Maintenance	Proposed	Proposed	Described	Descr. in detail
		Use	NP	Proposed	Described	Described
		Evolution	NP	NP	Proposed	Descr. in detail
Ontology support activities	Knowledge acquisition		Descr. in detail	Described	NP	Proposed
	■ Distributed know. acquisition		NP	NP	Proposed	Described
	■ Partial autonomy		NP	NP	NP	Described
	Evaluation		Descr. in detail	Proposed	NP	Proposed
	Integration		Proposed	Proposed	NP	Proposed
	Configuration management		Described	Described	NP	from OTK
	Documentation		Descr. in detail	Proposed	Described	from OTK
	■ Results		Descr. in detail	Proposed	Described	from OTK
	■ Decision process		NP	NP	Proposed	Descr. in detail
	Merging and Alignment		NP	NP	Proposed	Proposed

The comparison reveals that DILIGENT is well suited for ontology engineering tasks where distributiveness and change/evolution are of major concern. Further it is the first methodology which formalizes the argumentation taking place in an ontology engineering discussion. Hence, DILIGENT should be used in cases where tracing the engineering decisions is important. This allows future users to understand the different reasons which lead to the conceptualization. We think that these aspects are very important in the context of the semantic web. DILIGENT does not itself support management of OE process activities and Pre development activities, since these are already well supported by other mature methodologies.

6 Case Study Evaluation

The case study described in the following helped us to validate the previously defined methodology and to refine it in a few specific places. To this end, case study evaluation has incorporated the clients and practitioners to help us *understand the diversity* of the process. Before we describe how the DILIGENT ontology engineering process took place in our case study, we describe its organizational setting.

[6] Formerly named Ontology Management activities.

6.1 Organizational Setting

A DILIGENT process has been performed in a case study within the domain of tourism service providers of the Balearic Islands. To collaborate on regional issues some organizations set out to collect and share information about *indicators (SDI)* reflecting the impact of growing population and tourist fluxes in the islands, their environment and their infrastructures. For instance, organizations that require *Quality & Hospitality management (QHM)* use the information to better plan, *e.g.*, their marketing campaigns.

Due to the different working areas and goals of the collaborating organizations, it proved impossible to build a centralized ontology satisfying all user requirements. The users emphasized the need for local control over their ontologies. They asked explicitly for a system without a central server, where knowledge sharing was integrated into the normal work, but where different kinds of information, like files, emails, bookmarks and addresses could be shared with others. To this end a generic platform was built that would allow for satisfying the information sharing needs just elaborated using local ontologies, which were linked to a shared ontology. A case study was set up involving both hierarchical and loose organizations. The case study lasted for 3 months.

In this case study most of the tools were being developed at the same time as the process was taking place. Therefore, the administrator had a major role in bridging the gap between our real users and the weaknesses of the tools, for instance by doing the local adaptations for the users since the tools were not error-proof.

6.2 Instantiated DILIGENT Process

We now describe the initial building phase, and the two rounds following the DILIGENT process focusing on the analysis phase.

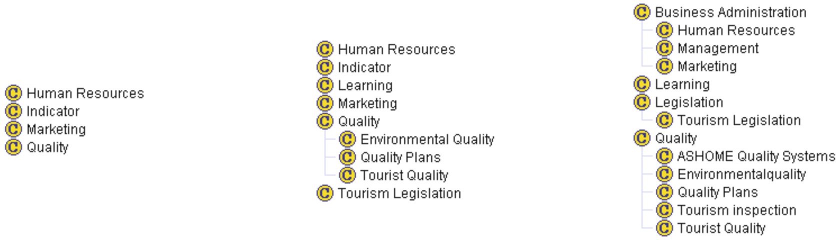
Build

(1)[7] To build the first version of the shared ontology two domain experts with the help of two knowledge/ontology engineers were involved. In this case, domain experts were also knowledge providers and users.

The OE process started by identifying the main concepts of the ontology through the analysis of competency questions and their answers. The most frequent queries and answers exchanged by users were analyzed. The main objective of the ontology was to categorize documents. The concepts identified were divided into three main modules: "Sustainable Development Indicators (SDI)", "New Technologies (NT)" and "Quality&Hospitality Management (QHM)". From the competency questions the board quickly derived a first ontology with 20 concepts and 7 relations for the "SDI" ontology. For "NT" the board identified 15 concepts and 8 relations and for "QHM" 8 concepts and 5 relations. Between the modules 8 cross module relations were introduced. A part of the result of the initial building stage is visualized in Fig. 2(a).

The first round of our OE process started with the distribution of the three modules of the common ontology to all users. In both rounds, users - during the local adaptation stage - and the board - in the revision stage - could perform ontology change operations. They could *introduce* concepts/relations/instances, *delete* concepts/relations/instances,

[7] The numbering here and in the following corresponds to the number of the actions in Fig. 1.



- Human Resources
- Indicator
- Marketing
- Quality

(a) First version of the common ontology

- Human Resources
- Indicator
- Learning
- Marketing
- Quality
  - Environmental Quality
  - Quality Plans
  - Tourist Quality
- Tourism Legislation

(b) Second version of the common ontology

- Business Administration
  - Human Resources
  - Management
  - Marketing
- Learning
- Legislation
  - Tourism Legislation
- Quality
  - ASHOME Quality Systems
  - Environmentalquality
  - Quality Plans
  - Tourism inspection
  - Tourist Quality

(c) Third version of the common ontology

Fig. 2. Excerpts from the common ontology evolution

or combine these operations arbitrarily, thus *extend* or *restructure* the ontology. Most frequently the concept hierarchy was changed.

6.3 First Round

The first month of the case study, corresponded to the first round of the DILIGENT process. One organization with seven peers participated. This organization can be classified as a rather loose one.

Local Adaptation

The users in our case study had no OE background. Therefore, they initially regarded the ontology mainly as a classification hierarchy for documents. Consequently they compared their existing folder structures with the common ontology to *(3) identify communalities between their own and the shared conceptualization*. *(5) Identification of missing conceptualizations* was thus based on mismatches between the common ontology and their local folder structures. Users *(6) changed the common conceptualization* accordingly. This entailed that the *local* documents stored in the folders were *(7) organized according to that conceptualization*. In this organization most of the users were very active and did local adaptations to best serve their own needs.

Analysis

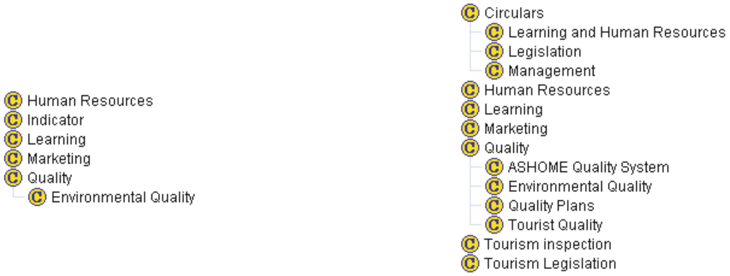
Roles: The board consisted of two ontology engineers and two domain experts/users.

Input: The local adaptations from seven users were collected. Additionally the board had access to the folder structures of those users.

Decisions: All changes introduced were motivated by the users, since they all made sense and were not contradictory on the conceptual level.

Actions:

(8) Gather locally updated ontologies and corresponding arguments: In the first round the board, through the administrator (i) directly accessed the formal local changes on the different peers and (ii) some change requests on the conceptual level. At this stage the board also used (iii) the folder structures as indication for the requirements on the ontology, and it used (iv) the number of documents related to the concepts of the ontology as an indicator for its usage. Additionally, the board received new background knowledge which led to many additions in the "NT" module. The "SDI" module was



(a) Example of a user extension to the first version of the common ontology

(b) Example of a user extension to the second version of the common ontology

Fig. 3. Examples of user extensions to the common ontology

changed based on the formal changes collected electronically. Although the number of changes varied between the different modules the kinds of changes were the same. Therefore, we subsequently focus on the changes introduced to the "QHM" module which are partly visualized in Fig. 2.

(9) Analyze introduced changes: The board analyzed the changes introduced by the users at a conceptual level. They can be categorized as follows:

Elaboration. The elaboration of the ontology was the most often observed action. The board could identify elaborations in three different ways. (i) The users correctly requested either formally or informally to add sub concepts to existing concepts to specialize them. (ii) The users incorrectly added new top level concepts, which were specializations of existing concepts. (iii) Finally they incorrectly refined the wrong concepts. In this way users elaborated the "NT" module with 15 concepts, the "SDI" module with 3 concepts and the "QHM" module also with 3 concepts.

Extension. The board regarded a change as an extension whenever users requested new concepts on the top level. Again, users could not distinguish wether a required concept was an elaboration or an extension. Users extended the "NT" module with 2 concepts and the "QHM" module also with 2 concepts. The "SDI" module was not extended.

Renaming. In two cases the users liked the way the board had conceptualized the domain, but did not agree with the names of the concepts.

Usage. Usage behavior of single concepts in the common ontology was analyzed. This included (i) the number of queries posed to the system containing a specific concept, (ii) the number of documents related to that concept and (iii) the elaboration of a concept. Most of the users did not delete any concepts or ask explicitly to remove concepts. Nevertheless the board concluded, that a concept which was never used should be removed.

(10) Decide on changes to be made: The board decided to introduce all change requests into the common ontology since all were supported by at least two users either through usage or extension/elaborations. Moreover, the domain expert could provide reasonable arguments for the introduction of all changes. Thus, the division of the ontology into 3 modules generated a consensual group of users, already.

Output: The analysis of the local adaptations resulted in 27 changes for the "NT" module, 10 changes for the "QHM" module, and 5 for the "SDI" module.

Revision

After modelling the conceptual changes, the second version of the common ontology contained 54 concepts and 13 relations (figure 2(b)).

Local update

(14) The extensions to the core ontology were distributed to the users. (17) The users were able to align their local ontologies with the new version of the shared and thus the feedback of the users was in general positive.

6.4 Second Round

In the second round the case study was extended to 4 organizations with 21 peers. The users participating in the first round had more experience. One of these organizations was very hierarchical.

Local Adaptation

As in the first round participants (6) changed and (7) used the common ontology according to their needs. Due to the larger number of participants more modifications were introduced. In particular the module "QHM" evolved (*cf.* Fig. 3(b)). In the hierarchical organization not all actors changed the ontology. They delegated the responsability to adapt the ontology to their hierarchical superior according to their organizational needs.

Analysis

Roles: In the second round the board consisted of one domain expert and two ontology engineers. Additionally two users were invited to answer questions to clarify the changes they introduced.

Input: The 21 local ontologies of the users were the input to the second round. Some of the users did not change the common ontology at all. Instead their supervisor was responsible to make all needed modifications for them. In very hierarchical and well defined organizations one single ontology could be adopted by all peers (Fig. 3(b)).

Decisions: In this round the board had to perform reverse engineering on the formal local ontologies from users in order to get conceptual models from them.

Actions:

(8) Gather locally updated ontologies and corresponding arguments: As in the first round the updated ontologies were retrieved electronically. Some of the modification requests were collected interviewing the participants.

(9) Analyze introduced changes: Similar to the first round the modifications did not follow good ontology building practices. With respect to the conceptual modelling decisions the board observed that this time the users modified the ontology on a deeper level than in the first round. Renaming was a bigger issue in this round due to political changes, which required the adoption of new naming conventions. Moreover, generalization took place in two cases. Users introduced concepts which were more abstract than existing ones. The board moved one concept "Indicator" to another module of the ontology, since there the users elaborated it extensively.

In Fig. 3(b) we observe that a user has extended the local version of the common ontology with concepts for Circulars. With help by the domain expert, and taking also into account other local updates, the knowledge engineers inferred on the conceptual level that the module lacked concepts for Business Administration. Hence, the board did not only introduce new concepts, but also generalize existing ones 2(c). To exemplify an argumentation thread in favor or against a modelling decision, one may consider the local extension of Circulars performed by one user: Legislation was introduced as a subclass of Circulars. The argumentation for a different way of modelling was straightforward, because the board found a Counter Example in the form of a document dealing with Legislation, which was not a Circular. Here, users were requesting an elaboration of Circulars for which the board found a countradicting example. The most convincing arguments were selected and emphasized for documentation purposes. In [12] we present the argumentation framework that allows such argument formalization.

(10) Decide on changes to be made: As in the first round the board included all change requests from users. Again, as in the first round, only few of the concepts in the common ontology were never used.

Output: The board identified 3 changes in the "NT" module, 28 modifications for the "QHM" module and 15 for the "SDI" module.

Revision

The third version of the common ontology contained 95 concepts and 15 relations (Fig. 2(c)).

Local update

(14) As in the first round the new version was distributed to the participants. (16) Updating to the new version is still a problem, since some instances of the ontology might have to be newly annotated to the new concepts of the shared ontology. In our case documents needed a new classification. (17) Partly this problem can be overcome with the help of technology *cf.* [13].

7 Discussion and Conclusions

Decentralization can take different forms. One can have more loose or more hierarchical organizations. We observed and supported both kinds of organizations in this case study. Therefore, the first finding is the fact that this process can be adapted both to hierarchical and to more loose organizations. DILIGENT processes cover both traditional OE processes and more Semantic Web-oriented OE processes, that is with strong decentralization and partial autonomy requirements.

The process helped non OE-expert users to conceptualize, specialize and refine their domain. The agreement met with the formalized ontology was high, as shown by people willing to change their folder structures to better use the improved domain conceptualization. In spite of the technical challenges, user feedback was very positive.

The DILIGENT process proved to be a natural way to have different people from different organizations collaborate and change the shared ontology. The set-up phase for DILIGENT was rather fast, and users could profit from their own proposals (local adaptations) immediately. The result was much closer to the user's own requirements. Moreover, other users profited from them in a longer term. Finally, this case study clearly has shown the need for evolution. Users performed changes and adaptations.

The development of ontologies in centralized settings is well studied and there are established methodologies. However, current experiences from projects suggest that ontology engineering should be subject to continuous improvement rather than a one-time effort and that ontologies promise the most benefits in decentralized rather than centralized systems. To this end we have conceived the DILIGENT methodology. DILIGENT supports domain experts, users, knowledge engineers and ontology engineers in collaboratively building a shared ontology in a distributed setting. Moreover, the methodology guides the participants in a fine grained way through the ontology evolution process, allowing for personalization. We have demonstrated the applicability of our process model in a cross-organizational case study in the realm of tourism industry. Real users were using the ontology to satisfy their information needs for an extended period of time. Two rounds following our methodology were observed and have been described here. To our knowledge, this is the first case study combining all the above mentioned features described in the literature.

References

1. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. Springer (2003)
2. Staab, S., Schnurr, H.P., Studer, R., Sure, Y.: Knowledge processes and ontologies. IEEE Intelligent Systems **16** (2001)
3. Uschold, M., King, M.: Towards a methodology for building ontologies. In: Proc. of IJCAI95 WS, Montreal, Canada (1995)
4. Pinto, H.S., Staab, S., Sure, Y., Tempich, C.: OntoEdit empowering SWAP: a case study in supporting DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies (DILIGENT). In: 1. Euro. Semantic Web Symposium, ESWS 2004, Springer (2004)
5. Kotis, K., Vouros, G.A., Alonso, J.P.: HCOME: tool-supported methodology for collaboratively devising living ontologies. In: SWDB'04: 2. Int. Workshop on Semantic Web and Databases. (2004)
6. Benjamins, V.R., Fensel, D., Decker, S., Gómez-Pérez, A.: $(KA)^2$: Building ontologies for the internet. International Journal of Human-Computer Studies (IJHCS) **51** (1999) 687–712
7. Sure, Y., Tempich, C., Vrandečić, Z.: D7.1.1. SEKT methodolgoy: Survey and initial framework. SEKT deliverable 7.1.1, Institute AIFB, University of Karlsruhe (2004)
8. Pinto, H.S., Martins, J.: Evolving Ontologies in Distributed and Dynamic Settings. In: Proc. of the 8th Int. Conf. on Princ. of Knowledge Representation & Reasoning (KR2002). (2002)
9. Holsapple, C.W., Joshi, K.D.: A collaborative approach to ontology design. Commun. ACM **45** (2002) 42–47
10. Pinto, H.S., Staab, S., Tempich, C.: 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). (2004)
11. Noy, N., Fergerson, R., Musen, M.: The knowledge model of Protégé-2000: Combining interoperability and flexibility. In: Proc. of the 12th Int. Conf. on Knowledge Engineering and Knowledge Management: Methods, Models, and Tools (EKAW 2000), Springer (2000)
12. Tempich, C., Pinto, H.S., Sure, Y., Staab, S.: An argumentation ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT). In: Second European Semantic Web Conference, ESWC 2005, Springer (2005)
13. Maedche, A., Motik, B., Stojanovic, L.: Managing multiple and distributed ontologies on the semantic web. The VLDB Journal **12** (2003) 286–302