How Much Does It Cost? Applying ONTOCOM to DILIGENT

Elena Paslaru Bontas[1], Christoph Tempich[2]
[1]Free University of Berlin, Takustr. 9, 14195 Berlin, Germany
paslaru@inf.fu-berlin.de
[2]Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
tempich@aifb.uni-karlsruhe.de

October 27, 2005

Abstract

Ontology Engineering is currently advancing from a pure research topic to real applications. This state of the art is emphasized by the wide range of European projects with major industry involvement and, in the same time, by the ever-growing interest of small and medium size enterprizes asking for consultancy in this domain. A core requirement in all of these efforts is, however, the availability of proved and tested methods which allow an *efficient* engineering of high-quality ontologies, be that by reuse, new building or automatic extraction methods. Several elaborated methodologies, which aid the development of ontologies for particular application requirements, emerged in the last decades. Nevertheless, in order for ontologies to be built and deployed at a large scale, beyond the boundaries of the academic community, one needs not only technologies and tools to assist the engineering process, but also means to *estimate and control its overall costs*. These issues are addressed only marginally by current engineering approaches though their importance is well recognized in the community. Different approaches exist to estimate costs for engineering processes. We will present the parametric cost estimation model ONTOCOM and its alignment with the DILIGENT engineering methodology. Based on the resulting cost function some analytical evaluations of application scenarios for the DILIGENT model are provided.

Contents

List of Figures

List of Tables

1 Introduction

Ontology Engineering is currently advancing from a pure research topic to real applications. This state of the art is emphasized by the wide range of European projects with major industry involvement and, in the same time, by the ever-growing interest of small and medium size enterprizes asking for consultancy in this domain. A core requirement in all of these efforts is, however, the availability of proved and tested methods which allow an *efficient* engineering of high-quality ontologies, be that by reuse, new building or automatic extraction methods. Several elaborated methodologies, which aid the development of ontologies for particular application requirements, emerged in the last decades. Nevertheless, in order for ontologies to be built and deployed at a large scale, beyond the boundaries of the academic community, one needs not only technologies and tools to assist the engineering process, but also means to *estimate and control its overall costs*. These issues are addressed only marginally by current engineering approaches though their importance is well recognized in the community.

A first attempt to bridge this gap has been made with the ONTOCOM approach [PBM05b, PBM05a], which is intended to be used as a method to estimate the efforts involved in building, reusing and maintaining ontologies. Likewise in the adjacent field of Software Engineering, a discipline in which cost prediction models belong to standard development environments, ONTOCOM proposes a top-down, parametric cost estimation method on the basis of pre-defined process stages and cost drivers, whose impact has been derived in advance by taking into account research and historical data from previous projects.

ONTOCOM distinguishes among three top-level phases of an ontology engineering process: 1). the development of a new ontology from scratch in conjunction with 2). reusing existing ontological sources and 3). the maintenance of ontologies in form of insertions, deletions and modifications of the initial content. For these categories cost drivers influencing the required development effort (in terms of person months) have been identified on the basis of a comprehensive analysis of current engineering methodologies and related case studies. Every cost driver is associated with effort multipliers (from very high to very low), depending on the individual characteristics of the application. A first estimation of the importance factors was performed based on ex post analysis of different ontology engineering efforts and preliminary expert validations with very promising results. Nevertheless more empirical data as well as an in-depth model validation and refinement are essential requirements for the elaboration of a reliable cost estimation method. The validation of the model [PBM05a] is based on the quality framework for cost prediction by Boehm [Boe81], which is a list of required and desirable features for these category of models, showing many similarities with established frameworks for evaluating the quality of general purpose information or data models [Epp01].

In this report we describe our experiences during the application of ONTOCOM to ontology engineering projects following the DILIGENT methodology. This attempt was motivated by the results of the above mentioned preliminary validation of the cost model, which made clear that the usability of the model would be significantly improved if we directly associate the cost drivers to more specific sub-tasks of the ontology engineering process, because this alignment would enable the definition of a

more precise and in the same time more reliable data extraction procedure. DILIGENT offers a fine-grained description of steps and activities required to collaboratively develop ontologies in application scenarios which have to cope with both distributed ontology usage and evolution, and volatile requirements arising during its life cycle. For this purpose it covers the most important process stages and activities which are well-recognized to be part of every typical ontology engineering scenario, thus being an excellent candidate for the refinement of the (yet very high-level) cost estimation methodology. On the other hand, DILIGENT currently lacks any support to estimate the effort required by the corresponding engineering process. The decision to exit a certain stage and enter a new cycle of the process model is not supported adequately. For this reason, a second goal of aligning ONTOCOM to the engineering methodology was to extend the latter with a cost estimation dimension in order to examine the ways costs could be involved as a decision support criteria to justify transitions between various process stages.

The remaining of this document is organized as follows: after introducing the setting of our work in Sections 2 and 3, we describe the refinement of the cost model towards applying it to the DILIGENT process model and the changes triggered by this mapping in each of the models (Section 4). The results of this mapping are used to specify the cost prediction procedure for concrete DILIGENT case studies in Sections 5 and 6. An overview of related and future work are given in Sections 7 and 8, respectively.

2 The ONTOCOM Cost Model

ONTOCOM is a parametric cost estimation model for ontologies which aims at predicting the effort invested in building, maintaining and reusing ontologies on the basis of pre-defined cost drivers. For the definition of the relevant factors, we adopted a combination of three general-purpose cost estimation methodologies [Boe81], which are in our opinion applicable to Ontology Engineering according to the current state of the art in the field (see [PBM05a] for an overview of the examined methods). We started with a top-down approach, by identifying upper-level sub-tasks of the ontology engineering process and defining the associated costs using a parametric method in correlation with a human-driven first validation. Expert judgement was used to evaluate the set of cost drivers associated to each process stage and to specify their start values in the a-priori model. Initially, we distinguished among three areas, whose costs were to be defined separately:

Ontology Building includes the typical stages of an ontology engineering process[FL99]: domain analysis (result: requirements specification), the conceptualization (result: conceptual model), the implementation (result: specification of the conceptual model in the selected representation language) and the ontology population i.e. the generation of instances and their alignment to the model (result: instantiated ontology).[1]

[1] At this point we restrict to manual ontology building activities. Automatic ontology generation methods

Ontology Maintenance includes getting familiar with and modifying the ontology (insert or delete new ontological primitives, re-model parts of the ontology etc.)

Ontology Reuse involves costs for the discovery and re-usage of existing (source) ontologies in order to generate a new (target) ontology. The latter includes understanding, evaluating and adapting the former ones to the requirements of the latter.

This upper-level distribution is of course subject of future refinements in order to increase the usability of the estimation method in real-world engineering projects. In particular, the ontology building area should be elaborated in the same top-down manner in order to partition this tedious and complex process down to a level in which the associated efforts can be reliably predicted. In this case, the cost drivers relevant the overall ontology building process (see below) are to be aligned (or even re-defined) to the corresponding sub-phases and activities. In particular, this issue will be further discussed in Section 4, where we describe the revision of the cost drivers related to building and reusing ontologies in conjunction with the DILIGENT framework.

Starting from a typical ontology engineering scenario, in which an ontology is developed—from scratch, by adapting existing knowledge sources, or both—and deployed/maintained by its users, ONTOCOM calculates the necessary person-months effort using the following equation:

$$PM \quad = \quad PM_B + PM_M + PM_R \tag{1}$$

PM_B , PM_M and PM_R represent the person months associated to building, maintaining and reusing ontologies, respectively and are calculated as:

$$PM_X = Size_X * \prod CD_{Xi} \tag{2}$$

Each of the three development phases is associated with *specific* cost factors. Experiences in related engineering areas [Kem87, Boe81] let us assume that the most significant factor is the *size of the ontology* involved in the corresponding process or process stage. In the formula above the size parameter $Size_X$ is expressed in thousands of ontological primitives – concept, relations, axioms and instances.[2] $Size_B$ corresponds to the size of the newly built ontology i.e. the number of primitives which are expected to result from the conceptualization phase. In case of ontology maintenance the size of the ontology ($Size_M$) depends on the expected number of modified items. For reuse purposes the relevant factor $Size_R$ is the (total) size of the original source(s) after being tailored to the present application setting. In particular this involves the parts of the source ontologies which have to be translated to the final representation language, the ones whose content has to be adapted to the target scope and the fragments directly integrated. The *cost drivers* CD_{Xi}—where X stays for B, R and M, respectively—have a rating level (from very low to very high) that expresses their impact on the development effort. For the purpose of a quantitative analysis, each rating level of each cost driver is associated to a weight (effort multiplier - EM). In the a-priori cost model a

as those proposed in the area of ontology learning are not considered in this work yet.

[2]For example for an ontology with 1000 concepts and 100 relations $Size$ will have the value 1.1.

team of 3 ontology engineering experts assigned start values between 0.7 and 1.9 to the effort multipliers, depending on the perceived contribution of the corresponding cost driver to the overall development costs.[3] These values are subject of further calibration on the basis of the statistical analysis of real-world project data. Additionally the values of the a-priori model (i.e. containing non-calibrated values) will be included in the expert validation process, which will is currently being performed as part of the mapping between ONTOCOM and the engineering methodology DILIGENT.

In the following we turn to a brief description of the cost drivers in ONTOCOM.[4] These parameters were derived after surveying recent literature and from empirical findings of various case studies in the ontology engineering field. For each cost driver we specified in detail the decision criteria which are relevant for the model user in order for him to determine the concrete value of the driver in a particular situation. For example for the cost driver LEXP—accounting for costs produced by the level of experience of the engineering team w.r.t. ontology representation languages—we predefined the meaning of the effort multipliers as depicted in Table 1. The values of the corresponding effort multipliers, which have been specified by human experts, are as follows: 1.60 (Very Low), 1.30 (Low), 1 (Nominal), 0.90 (High) and 0.80 (Very High) [PBM05a]. The suitable value should be selected during the cost estimation procedure and used as a multiplier in equation 2.

	Very Low	**Low**	**Nominal**	**High**	**Very High**
LEXP	2 months	6 months	1 year	3 years	6 years
Effort Multipliers	1.60	1.30	1.0	0.90	0.80

Table 1: Cost Driver LEXP (Language Experience)

In several cases the decision criteria associated with a cost driver are more complex than in the previous example and might be sub-divided into further sub-categories, whose impact is aggregated to the final value of the corresponding cost driver by means of weights.

2.1 Cost Drivers for Ontology Building

For the ontology building area we defined a list of cost drivers, which are, similar to [B. 97], divided into three groups:

- **Product-related** cost drivers account for the influence of ontology characteristics on the overall costs: i) Instance (DATA) to capture the effects that the instance data requirements have on the overall process, ii) Ontology Complexity (OCPLX) to express those ontology features which increase the complexity of the engineering outcomes, iii) Required Reusability (REUSE) to capture the additional effort associated with the development of a reusable ontology, and iv) Documentation match to life-cycle needs (DOCU) to state for the additional costs caused by very detailed documentation requirements.

[3] A list of the values is available in [PBM05a].

[4] See [PBM05a] for a detailed explanation of the approach.

- **Project-related** cost drivers relate the dimensions of the engineering process and its characteristics to the overall costs: i) Support tools for Ontology Engineering (TOOL) to measure the effects of using ontology management tools in the engineering process, and ii) Multisite Development (SITE) to mirror the usage of the communication support tools in a location-distributed team.

- **Personnel-related** cost drivers emphasize the role of team experience, ability and continuity w.r.t. the effort invested in the process: i) Ontologist/Domain Expert Capability (OCAP/DECAP) to account the perceived ability and efficiency of the single actors involved in the process (ontologist and domain expert) as well as their teamwork capabilities, ii) Ontologist/Domain Expert Experience (OEXP/DEEXP) to measure the level of experience of the engineering team w.r.t. performing ontology engineering activities, iii) Language/Tool Experience (LEXP/TEXP) to measure the level experience of the project team w.r.t. the representation language and the ontology management tools, and iv) Personnel Continuity (PCON) to mirror the frequency of the changes in the project team.

2.2 Cost Drivers for Ontology Reuse and Maintenance

Additionally to project and personnel cost drivers (as described in Section 2.1) we defined a set of further 4 cost drivers to deal with the characteristics of ontology reuse and maintenance, as reported by recent case studies in these areas [PBMT05, UHW+98, RVMS99, UCH+98]:

- **Ontology Understanding(OU)** accounts for the efforts required to get familiar with the ontologies to be used, a task which is a pre-condition to ontology evaluation and maintenance. It depends on ontology properties such as representation language or size and on the level of experience of the ontology engineer w.r.t. this ontology[PBM05a].

- **Ontology Evaluation(OE)** accounts for the additional efforts related to the evaluation phase given a satisfactory ontology understanding level (e.g. for testing the source ontologies against a specific set of requirements or for documenting the approach).

- **Ontology Modification/Translation(OM/OT)** are factors reflecting the costs involved by adapting the source ontologies to the new setting (e.g. inserting, deleting ontology concepts) and by translating to a target representation language, respectively.

2.3 Evaluation of ONTOCOM

The parametric approach described in this report is currently being validated towards a reliable method for estimating the costs of ontology engineering. The most important evaluation criterium is of course the reliability of its predictions, which however depends on the amount of accurate project data used to calibrate the model (i.e. adjust the values of the modifiers and identify eventual correlated cost drivers). On the other

hand, a comprehensive evaluation of the model should go beyond the evaluation of its functionality (i.e. the accuracy of its estimations) and also address issues related to its usability in typical ontology engineering scenarios, as suggested in common quality frameworks for information systems (such as [PS04, MSBS03, HLW99, KLS95]; see [Epp01] for a comprehensive survey on this topic).

For a comprehensive evaluation of the model we rely on the quality framework for cost models by Boehm, which was adapted to the particularities of ONTOCOM and Ontology Engineering. Parts of this framework are used in to assess the quality of the a-priori and the a-posteriori cost models, respectively (see below). According to this differentiation, the evaluation of the cost model is performed in two steps: in the first one we evaluate the relevance of the mentioned cost drivers for the purpose of predicting costs arisen in ontology engineering projects; the remaining aspects of the framework relate to its capability of reliably fulfilling its goal (i.e. that of estimating engineering efforts) and will be applied in a second step on the a-posteriori model resulting from the calibration of the preliminary one.

The original quality framework by Boehm [Boe81] consisted of the 10 features, which we divided into two categories, depending on their relevance to the a-priori and the a-posteriori model, respectively. The meaning of the quality criteria has been adapted to the scope of ONTOCOM.

A-priori evaluation :

definition : has the model clearly defined the costs it is estimating and the costs it is excluding? Does the estimate include the cost of management, training, domain analysis, conceptualization, implementation, testing, maintenance? Does the model clearly define the decision criteria used to specify the ratings of the cost drivers? Does the model use intuitive and non-ambiguous terms to denominate the cost drivers it involves?

objectivity : does the model avoid allocating most of the cost variance to poorly calibrated subjective factors? Are the cost drivers defined using objective decision criteria, which allow an accurate assignment of the corresponding cost driver ratings?

constructiveness can a user tell why the model gives the estimates it does?

detail : does the model easily accommodate the estimation of new process models or is it conceived for a particular ontology engineering process? Does it give accurate phase and activity breakdowns? Does the model take into consideration factors related to the main tasks of the engineering process? Do these sub-tasks correspond to the process model applied in your engineering process? Which phases should be further covered by the model in order to increase its usability?

stability : do small differences in inputs produce small differences in output cost estimates?

scope : does the model cover the class of projects whose costs you want to estimate? Is it representative for a wide class of ontology engineering projects?

ease of use : are the model inputs and options easy to understand and specify? Is it easy for you to assess a rating to a cost driver based on the associated decision criteria?

prospectiveness : does the model avoid the user of information which will not be well known until the project is complete? Can the model be applied in early phases of the project as well?

parsimony : does the model avoid the use of highly redundant factors or factors which make no appreciable contribution to the results?

A-posteriori evaluation :

all items of the former category, plus

fidelity , since this requirement will definitely not be fulfilled after collecting reliable data from previous projects used to refine the values of the cost drivers and to discover eventual correlations between them.

The features assigned to the first category will be used as criteria for the refinements resulting from the application of the cost model to DILIGENT, as described in Section 4.

2.4 Current Limitations

As mentioned above the approach we described in this section is intended as a first draft towards a reliable cost estimation method for ontology engineering projects. The relevance of the cost drivers and their quality in terms of scope and granularity will be assessed during the first step of the model evaluation procedure. In particular the factors describing the development of new ontologies should be further refined in order to realistically reflect the efforts invested in ontology building, which is well-recognized as a tedious and challenging process. Mapping these cost drivers to a fine-grained process model such as that proposed by the DILIGENT methodology is an excellent opportunity to prove their expedience in the described context. Finally the usability of the model is directly related to the collection of real-world project data, which are of course indispensable for the calibration of the values required for the parametric prediction equation. The case studies currently being carried out according to the DILIGENT methodology will provide valuable information w.r.t. this second issue.

3 The DILIGENT methodology

This section summarizes the current status of the DILIGENT process model (Fig. 1), which gives the necessary context for the detailed mapping between the engineering activities involved in the process and the ONTOCOM cost factors (Section 4).

Scenario In a *distributed* ontology development scenario there are several experts, with different and complementary skills, involved in collaboratively building the same ontology. Virtual Organizations, Open Source and Standardization efforts are typical

examples of such distributed scenarios, in which geographically dispersed experts belonging to different competing organizations come together for the achievement of a common goal. In these cases, developers are typically also users and, although some users are not directly involved in changing the ontology, they indirectly participate at the development process by using the ontology. This collaboration can take place in many forms – for instance in our particular case we used a peer-to-peer platform to support collaboration. While not fully substituting the human contributions, automated agents or tools can be used in parts of the process – ontology learning tools can, for instance, aid the ontology developers by extracting potential ontological information in terms of concepts, taxonomies relations or instances from designated texts.

3.1 General process

The **DILIGENT** process [PSST04] supports its participants, in collaboratively building one shared ontology. The process consists of 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 suggests 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. It does not require completeness of the initial shared ontology w.r.t. the domain. Once the first prototype is made available, users can start using and **locally adapting** it for their own purposes. Typically, due to new business requirements, as well as user and organizational changes, their local ontologies evolve.

A central board of ontology stakeholders **analyzes** the local ontologies and the requests and tries to identify similarities in users' ontologies. The board should regularly **revise** the shared ontology, so that local ontologies do not diverge too far from the shared ontology. Therefore, the board should have a well-balanced and representative participation of the different kinds of participants involved in the process, knowledge providers, domain experts, ontology engineers and users. 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, while a new cycle is triggered by the release of a new shared ontology.

3.2 DILIGENT process stages

In order to facilitate the ontology engineering process we analyzed the different process stages in detail, identifying the (i) major roles, (ii) input, (iii) decisions, (iv) actions (v) output information (vi) and tools that occur in each stage. In Fig. 1 we sketch our results as before the alignment with the ONTOCOM cost model. A more detailed description of all items depicted in Fig. 1 can be found in [STV04]. However, as the activities have undergone a major restructuring due to the alignment with the ONTOCOM cost model we present a detailed description of the changes occurred at the activity level in Section 4.3.

3.2.1 Build

The building stage is based on existing ontology engineering methodologies [GPFLC03, SS02].

Roles, Input, Output, Decisions, Actions: as in existing engineering methodologies [GPFLC03, SS02].

Tool support: Building is supported by existing ontology editors like [SEA+02, NFM00].

Once an initial ontology is (1) *built* and released, users will start to adapt it locally for their own purposes.

3.2.2 Local Adaptation

Roles: The actors involved in the local adaptation step are the users of the ontology. They use the ontology e.g., to retrieve documents – or even data with a more complex structures such as entire projects – related to certain topics modeled in the ontology. The main objective of the actors involved in this process stage is not necessarily information gathering, but fulfilling setting-specific tasks by means of the ontology.

Input: The local information space in form of existing databases, ontologies or folder structures and documents, as well as the current version of the shared ontology can be used as input for the local adaptation phase.

Output: The output of the process step is a locally changed ontology, which better reflects needs of the local users. Each change is supported by arguments explaining

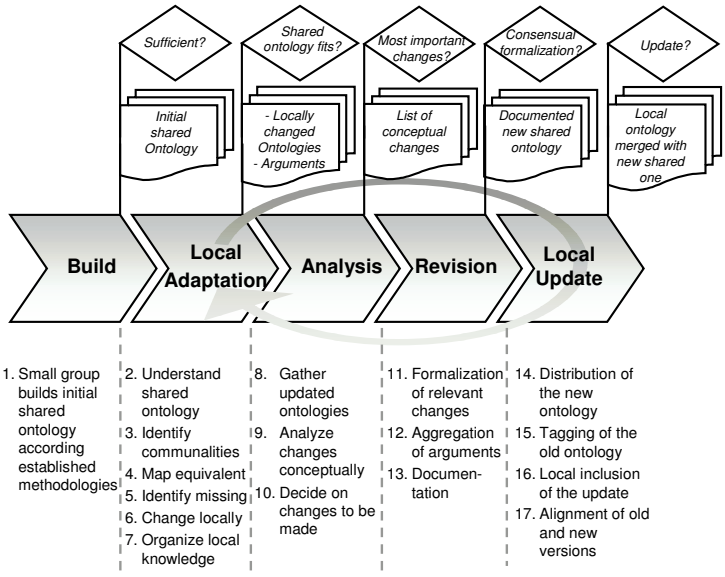

Figure 1: Process stages (1-5), actions (1-17) and structures

the reasons for performing it. Note that changes are not propagated to the shared ontology. The change requests are submitted to the central board, which examines them in the *analysis step* and, on the basis of the provided arguments propagates the most significant one in the *revision step*.

Decisions: The actors should decide which changes they want to make to their ontology. Hence, they should decide if and where new concepts are needed and which relations a concept should have. They should further provide reasons why they made certain decisions. To evaluate the decisions we propose to calculate the ratio between available information and the information which can be classified according to the adapted ontology. The proportion should ideally be high. Further classifications should be specific. Local concepts which could not be aligned with shared concepts should be introduced as local adaptations.

Actions: To achieve the desired output the user takes different actions namely (2) *Understand the shared ontology*, (3) *Identify the 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 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 the available technology to support the actors.

Tool support: Building is supported by existing ontology editors like [SEA+02, NFM00]. In [LET04] we described how existing structures 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) [ES04]. (6) is a manual step. (7) is currently a manual step, too, but it could be supported by semi-automatic classification *cf. e.g.* [HSC02].

3.2.3 Analysis

In this stage the board (cf. the description of DILIGENT in Sec. 7) analyzes incoming requests and observations of changes. The frequency of consecutive analysis stages could be determined based on the frequency and volume of changes to the local ontologies.

Roles: In the analysis stage we can distinguish among 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 (in the revision step, see below).

Decisions: The board must decide which changes to introduce into the new shared ontology at the conceptual level. Metrics to support this decision could be (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.

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 consider rationales generated by automated methods, e.g. ontology learning. The arguments underlying the proposed changes constitute an important input for the board to achieve a well-balanced decision about which changes to adopt.

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*.

Output: The result is a list of the major changes to be introduced that were agreed by the board. Hence, all changes which should not be introduced into the shared ontology are filtered. In this stage it is not required to decide upon the final modeling of the shared ontology.

Tool support: In [PSST04] we presented an extension to an ontology editor, which provides support for 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 [TPSS05].

3.2.4 Revision

Roles: The ontology engineer judges 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 should judge and decide wether new concepts/relations should be introduced into the common ontology even so they were not requested by the users.

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 of formal nature. 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 should be formalized. Evaluation of the decisions is performed by comparing the changes on 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 user takes different actions namely (11)

Formalization of the requested changes, (12) *Aggregation of arguments* and (13) *Documentation*. Judging entails *Evaluation* of the proposed changes from a knowledge representation/ontological point of view.

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

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

3.2.5 Local Update

Roles: The local update phase involves only the ontology 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.

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.

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.

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: Changes cannot be introduced without the agreement of the users. From a usability point of view multiple releases of the shared ontologies occurring with high frequency might be perceived as disturbing by certain users, especially in the case in which they are not actively involved in the evolution of the shared ontology. In case of equivalent but local conceptualizations it must be possible to change to the common one. From a technical point of view this stage is supported by tools like KAON *cf.* [MMS03]. We do not exclude the possibility of conflicts. Unresolved conflicts, though, will reduce the utility of the ontology, as this is globally inconsistent. Dealing with this sensitive issue is an ongoing research topic in the Semantic Web community, while first solutions have already emerged *cf.* [HvHH+05].

4 Bridging ONTOCOM and DILIGENT

The main objective of this work is to integrate the mentioned cost model ONTOCOM to ontology engineering processes performed according to the DILIGENT engineering

methodologies. Mapping the two models is mainly aimed at validating and refining the existing cost estimation strategy on the basis of real-world ontology development case studies. In order to increase the usability of the cost model and to collect high quality data necessary for its calibration we examined the relevance of the pre-defined cost drivers w.r.t. the process sub-tasks and actions and precisely define the ways the data is to be collected. This task resulted in an adaptation of the estimation model and the process model since missing parts could be identified. Further on we adapted the high-level parametric effort estimation equation from ONTOCOM to the DILIGENT scenario in order to be able to apply the model to the case studies currently being performed in the basis of this engineering methodology. The necessary data is to be collected from the case study partners in the near future and will provide first estimations of the envisioned total development costs and revise the cost model. The calibrated cost estimation model will allow its users to estimate in each process stage the future effort necessary to build and maintain the ontology. Cost-related information might be a significant decision factor for obtaining a feasible trade-off between start-up/maintenance efforts and ontology utility. The last part of this section reports on our experiences in trying to determine optimal relation between the initial effort, the utility and the maintenance effort implied by a concrete DILIGENT-based ontology development project.

To summarize, bridging the two models was performed according to the following steps, which will be further elaborated in the remaining of this section:

1. Alignment of the cost drivers with the DILIGENT process stages
2. Definition of the cost function and of data collection procedure in terms of the process model
3. Examination of the potential process decision support in terms of costs

The data collection and the subsequent calibration of the cost model according to the outcomes of this task will be performed in the near future in terms of the procedure described below.

4.1 Mapping the activities in DILIGENT to the cost drivers in ONTOCOM

In order to realistically estimate the costs induced by particular DILIGENT processes the general cost model introduced in 2 has to be customized to the activities and the stages of the engineering methodology. As introduced in section 3 the DILIGENT process is divided into five main stages, where each stage contains a number of sequential or parallel activities. The ONTOCOM to DILIGENT mapping assigns product, personnel and project cost drivers to the corresponding engineering activities. Tables 2 to 5 summarize the results of this mapping (for simplification purposes the costs associated to the analysis and revision phases were joined to a single overview table). Note that the changes occurred at the cost model level as a result of its alignment to the engineering methodology have been printed in the tables below in italics.

4.1.1 Cost drivers relevant for the Centralized Building stage

DILIGENT process		**Cost factor**		
DILIGENT phase	DILIGENT activity	Product Factors	Personal	Project
Centralized Building	Domain analysis	Complexity of the domain analysis	OCAP, DECAP OEXP, DEEXP TEXP, PCON	TOOL
	Conceptualization and implementation of shared ontology	*Complexity of the conceptualization (DOCPLX)*, REUSE, OU, OE, OT, OM, *Ontology Integration (OI)*	OCAP, DECAP OEXP, DEEXP, LEXP, TEXP PCON	TOOL
	Evaluation of shared ontology	*Ontology evaluation (OE)*	OCAP, DECAP OEXP, DEEXP, LEXP, TEXP, PCON	TOOL
	Documentation	DOCU	OCAP, DECAP OEXP, DEEXP, TEXP, PCON	TOOL

Table 2: Mapping DILIGENT to ONTOCOM: Centralized Building

The calculation of the expected costs for the centralized building stage (*cf.* table 2) depends on all cost drivers except the ones for Ontology Maintenance and Required Development Schedule (SCED). We can leave out the cost driver for maintenance as it is part of subsequent stages to keep the ontology updated. The cost driver SCED is left out since we are currently only interested in the estimation of person month rather than the complete project duration. This applies also for the subsequent process stages.

The cost drivers for ontology reuse should be considered only if the board decides to utilize existing ontologies to build the initial shared ontology.

We found out that the initial proposal in the DILIGENT process model to reduce the centralized building to only one activity was too coarse. Hence, we introduced four activities to capture this phase more accurately. Splitting up the building process into those activities led us to the conclusion that the cost driver for ontology complexity was too broadly defined. As a consequence, this cost driver was divided into three separate new ones, namely one for the complexity of the domain analysis, one for the complexity of the conceptualization and finally one for the implementation complexity.

Similarly, the cost model so far did not define any cost driver for the integration of different ontologies. From a process point of view integrating ontologies with each other is a rather time consuming task. Furthermore the cost driver for tool support covered only the tools available to formalize the ontology. Experience suggests, that for each activity different supporting tools are available. The cost driver for tool support should therefore be interpreted as an average of the quality of the available tools for each activity. Finally, the evaluation process for the resulting ontology was not covered

by any cost driver.

Even though the process defines a separate activity for the provision of arguments, we concluded that no extra costs driver is needed to calculated the respective effort, because it is already covered by the documentation cost driver.

4.1.2 Cost drivers relevant for the Local Adaptation stage

The cost drivers relevant to estimate the costs of the local adaptation stage (*cf.* table 3) are to a large extent the same as those associated to the previous stage. While the personnel factors relate to the capabilities of the domain experts, the main difference of this stages compared to the Centralized Building one is the application of the cost drivers to a multitude of sites and ontologies, while different users belonging to different sites are involved in different activities performed on the shared, the locally adapted and the externally adapted ontologies respectively. The sizes of these ontologies are put into relation with different groups of cost drivers when considering the calculation of the resulting costs. All users follow the process defined in the Local Adaptation stage, therefore each of them incurs some effort to understand, adapt and use the ontology. As the usage and customization of the shared ontology is theoretically inconceivable without it being previously understood by its users, one major cost category of the Local Adaptation phase is of course related to the Ontology Understanding effort multiplier in conjunction with the size of the currently shared ontology. Further on the modification of the shared ontology depends on the number of changes performed together with the corresponding effort multipliers for ontology maintenance in ONTOCOM. Additionally to the original cost model DILIGENT foresees an activity in which new requirements are specified, which is mapped to a cost driver in the ontology building category. If the users decide to adopt externally developed ontologies to conceptualize their local changes (i.e. instead of newly implementing them in their local ontology) the corresponding activities imply additional costs, as described in the ontology reuse category in ONTOCOM. Note that there are no costs for the translation of the ontologies to be reused since it is assumed that all ontologies in the application scenario are formalized in the same representation language. When calculating the cost estimation for the adaptation phase we recommend to use average values for the aforementioned effort multipliers instead of indicating separate values for each site.

In the previous version of the DILIGENT process model six activities were defined for the Local Adaptation stage. We decided to homogenize the granularity of the activities foreseen for this process stage in accordance with existing ontology building methodologies. The scope of *Understand shared ontology* was therefore broadened by adding evaluation aspects and is now called `Local analysis of shared ontology` emphasizing the different location between creation and usage of the ontology. The activities *Identify communalities* and *Identify missing* were merged and renamed to `Specification of new requirements`. The activity *Map equivalent* is now part of the `Ontology instantiation` and the `Integration of reused local ontologies to the shared ontology`. *Organize local knowledge* is also included in the new `Ontology instantiation` activity. The use of the ontology is not an explicit activity in the current version of the methodology, but since it is a major source for the detection of new requirements we introduced it as

DILIGENT process		**Cost factor**		
DILIGENT phase	DILIGENT activity	Product Factors	Personal	Project
Local adaptation	Local analysis of shared ontology	OU, OE	DECAP, DEEXP, LEXP, TEXP, PCON	TOOL
	Specification of new requirements	*Complexity of the domain analysis (DOCPLX)*	DECAP, DEEXP, LEXP, TEXP, PCON	TOOL
	Ontology utilization		DECAP, DEEXP, LEXP, TEXP	TOOL
	Ontology instantiation	DATA	OEXP, DEEXP, OCAP, DECAP, TEXP, LEXP, PCON	TOOL
	Local analysis of additional (local) ontologies	OU, OE, number of sites	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Customization of relevant local ontologies	OM, number of sites	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Integration of reused local ontologies to the shared ontology	*Ontology integration (OI)*	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Modification of shared ontology Argument provision	OM	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Evaluation of new local ontology	*Ontology evaluation (OE)*	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Documentation	DOCU	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL

Table 3: Mapping DILIGENT to ONTOCOM: Local Adaptation

`Ontology utilization`.

The local adaptation of the shared ontology, originally performed in the *Change locally* activity, was spread along four new activities `Local analysis of additional ontologies`, `Customization of relevant ontologies`, `Integration of reused local ontologies to the shared ontology` and `Modification of shared ontology` to account for the different ways of realizing it. In case ontologies from other users are reused we now use the terminology established in the literature for these purposes. Note that this is a different activity than the modification of the shared ontology, since it implies for instance the comprehension and evaluation of ontologies developed at different sites in the local context.

In compliance with established ontology engineering methodologies we further introduced the activities `Argument provision`, `Evaluation of new local`

ontology and Documentation, which were initially not supported by DILIGENT to a satisfactory extent.

4.1.3 Cost drivers relevant for the Centralized Analysis and Revision stage

From an cost estimation point of view the centralized analysis and revision stage (*cf.* table 4) resembles the building stage, while the former involves both the currently shared and the locally used ontologies. Each of these ontologies is associated with different categories of cost drivers. As the original shared ontology is already known to the board, there are no additional costs to understand it. The board examines in turn the changes submitted by the users. The number of SITEs, as well as the average number of changes introduced by the users will thus influence the costs. In the analysis and revision stage all requirements derive from user requests, thus we do not incur costs for domain analysis. The local ontologies from the users are available and no search effort is needed to obtain them. The costs for ontology modification depend on the actual changes introduced by the board.

W.r.t. the personnel cost drivers we need to distinguish between the average experience of the users and the experience of the board, while the latter is rated higher than the former when it comes to ontology engineering.

In the original version of the process we defined six activities for the centralized analysis. The activity *Gather updated ontologies* is now broader and includes the collection of all relevant information from the users, such as arguments and informal change requests. It is called Information collection from users. Similarly we renamed the activity *Analyze changes conceptually* to Analysis of obtained information accounting for the different ways change requests can be made. We have further introduced a controlling activity, Control of previously shared ontology, in order to establish a feedback loop in our process.

The activity *Decide on changes to be made* is better described as Specification of new requirements. Existing methodologies describe the activity *Formalization of relevant changes* as four separate activities: Customization of relevant local ontologies, Integration of reused local ontologies, Integration of reused local ontologies to the shared ontology and Modification of shared ontology. On the one hand the board will reuse the local ontologies from the users to extend the shared ontology on the other hand user can request changes informally which than lead to modifications of the shared ontology.

We further introduced explicitly the activity for Argument provision and for the Evaluation of new shared ontology.

The activities *Aggregation of arguments* and *Documentation* are unchanged, though they were renamed to Argumentation aggregation and Documentation, respectively.

In order to separate activities performed centrally or locally the last activity of this stage is the Distribution of new shared ontology which was previously an activity of the Local Update stage.

DILIGENT process		Cost factor		
DILIGENT phase	DILIGENT activity	Product Factors	Personal	Project
Centralized Analysis and Revision	Information collection from users			SITE
	Analysis of obtained information	OU, OE, number of sites	OEXP, OCAP, LEXP, TEXP, PCON	TOOL
	Control of previously shared ontology		OEXP, OCAP, TEXP, PCON	TOOL
	Specification of new requirements		OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	Customization of relevant local ontologies	OM, number of sites	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Integration of reused local ontologies to the shared ontology	*Ontology integration (OI)*	DEEXP, DECAP, TEXP, LEXP, PCON	TOOL
	Modification of shared ontology	, OM, REUSE	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	Argument provision			
	Argumentation aggregation			
	Evaluation of new shared ontology	*Ontology evaluation (OE)*	OCAP, OEXP, LEXP, TEXP, PCON	TOOL
	Documentation	DOCU	OCAP, OEXP, TEXP, PCON	TOOL
	Distribution of new shared ontology			SITE

Table 4: Mapping DILIGENT to ONTOCOM: Centralized Analysis and Revision

4.1.4 Cost drivers relevant for the Local Update stage

For the last stage in the DILIGENT process, the Local Update (*cf.* table 5) we need less cost drivers than for the previous stages. The relevant sizes are the number of changes introduced by the board and the number of sites. As in previous stages the personnel and management cost drivers remain unchanged. We have identified the three product cost drivers Ontology Understanding, Ontology Evaluation and Ontology integration as relevant for this stage.

The activity *Distribution of the new ontology* was moved to the previous stage. The activities *Tagging of the old ontology* and *Local inclusion of the update* in the old version of the process model are technically motivated. Though technically necessary, methodologically they are rather part of the `Integration of new and old version` activity as defined in the new version. This covers also the *Alignment of old and new versions*. Before the actual integration of the new version with the old

DILIGENT process		Cost factor		
DILIGENT phase	DILIGENT activity	Product Factors	Personal	Project
Local update	Control of new shared ontology		DECAP, DEEXP, TEXP	TOOL
	Local analysis of changes in the new shared ontology	OU, OE	DECAP, DEEXP, LEXP, TEXP, PCON	TOOL
	Integration of new and old version	*Ontology integration (OI)*	DECAP, DEEXP, LEXP, TEXP, PCON	TOOL

Table 5: Mapping DILIGENT to ONTOCOM: Local Update

version can be processed we introduced a `Control new shared ontology` activity to allow for feedback to the users. Furthermore we left the decision whether to update to the new version explicitly to the user. The `Local analysis of changes in the new shared ontology` is the basis for this decision.

4.2 Changes in the Cost Model

Applying the pre-defined ONTOCOM cost drivers to the DILIGENT engineering process model revealed several shortcomings of the former w.r.t. two dimensions: the insufficient coverage of some of the available cost drivers and the lack of support for tasks such as ontology merging and integration. Additionally to the adjustment of the model according to these findings, the expert values used by the a-priori cost estimation formula were adapted on the basis of the expertise provided by the DILIGENT engineering team.

We now turn to a detailed description of the model refinements arisen during the integration of the two models.

4.2.1 The Cost Driver OCPLX (Ontology Complexity)

The mapping between the DILIGENT process stages and the available product-based cost drivers made clear that the original cost driver coping with the complexity of the product to be developed (i.e. the ontology) was not covering all aspects of the ontology building process to a satisfactory extent. As already assumed by the authors, from a product perspective, distinguishing between three high-level phases (building, reuse and maintenance) has proved to be insufficient for the needs of real-world settings. Though the OCLPX cost driver, which was initially intended to cover the costs arisen from this activity, already mentioned some of the most important factors in this field, the DILIGENT ontology engineers evaluated its impact in the current form as too limited compared to the impact of other, less relevant and less complex cost drivers such as REUSE or DOCU (see [PBM05a] for a detailed description of the cost drivers). Consequently we divided the original OCPLX cost driver into three new parameters partially covering the overall complexity of the target ontology. The division in three complex-

ity areas was performed at the process level. This design decision is justified through the assumption that the complexity of the final ontology is implicitly related the complexity of the underlying building process, in particular the phases domain analysis, conceptualization and implementation, given a certain competence level of the personnel and sufficient project experience. The decision criteria for assigning a specific rating level to the new cost drivers were mainly derived from the ones originally proposed for the OCLPX driver. Additionally new criteria related to the availability of useful knowledge sources during the domain analysis phase and to the ontology implementation were introduced (see below). We now turn to a description of the new complexity parameters:

- Domain complexity (DCLPX)
- Conceptual complexity (CCPLX)
- Implementation complexity (ICPLX)

The domain complexity driver states for the efforts additionally arisen in the engineering project by the particularities of the ontology domain and its analysis during ontology building. The decision which concepts will be included and in which form they will be represented in an ontology depends not only on the intrinsic domain to be modeled (e.g., tourism), but rather on the application domain. The latter also involves the technical setting and the characteristics of the application in which the ontology is designed to be integrated to. As a third decision field we introduced the sources which could be eventually used as additional domain descriptions and thus as an aid for the domain analysis and the subsequent conceptualization. The global value for the DCLPX driver is a weighted sum of the aforementioned areas, which are depicted in Table 6.

In order to realistically classify the complexity of the domain analysis phase in terms of the pre-defined ratings we identified characteristics of the three areas which usually influence this measure. For the domain category, we considered the scope (narrow, moderate, wide), the commonality of the knowledge (be that common-sense knowledge or expert knowledge) and the connectivity of the domain. The latter is expressed in the number of interdependencies between domain concepts with ranges again among three levels (low, moderate and high), while the scope is a feature which is related to the generality, but also to the perceived amount of knowledge comprised per default in a certain domain. For example a domain such as some department of an organization is considered narrower than a domain describing a university, while the scope of the economics domain is of course classified as wide. The three criteria are prioritized according to common practices in the ontology engineering area, so that the connectivity of the domain is considered decisive for establishing the rating of this cost factor.

The complexity of the requirements which are to be taken into consideration when building an ontology is characterized here by the total number of requirements available in conjunction with the rate of conflicting ones and the rate of usability requirements, since the latter are seen as a fundamental source of complexity for the building

Rating Scale	DOMAIN
Very Low	narrow scope, common-sense knowledge, low connectivity
Low	narrow to moderate scope, common-sense or expert knowledge, low connectivity
Nominal	moderate to wide scope, common-sense or expert knowledge, moderate connectivity
High	moderate to wide scope, common-sense or expert knowledge, high connectivity
Very High	wide scope, expert knowledge, high connectivity
	REQUIREMENTS
Very Low	few, simple req.
Low	small number of non-conflicting req.
Nominal	moderate number of req., with few conflicts, few usability req.
High	high number of usability req., few conflicting req.
Very High	very high number of req. with a high conflicting degree, high number of usability req.
	INFORMATION SOURCES
Very Low	high number of sources in various forms
Low	competency questions and text documents available
Nominal	some text documents available
High	some unstructured information sources available
Very High	none

Table 6: The Domain Complexity Cost Driver DCLPX

process.[5]

Finally the availability of information sources guiding the engineering team during the building process or offering valuable insights in the domain to be modeled can be a major success factor in ontology engineering. When deciding upon the impact of the information sources on the effort required to perform the domain analysis activity we suggest considering the number, the type and the form of the sources.

The conceptualization complexity accounts for the impact of the structure of the conceptual ontology (taxonomy, conceptual graph etc.) and of help techniques such as modeling patterns on the overall engineering costs. On the other side, the existence of certain naming and modeling constraints might cause cost increases (see Table 7).

As mentioned in [PBM05a] one of the basic assumptions in ONTOCOM is that the most significant factor for estimating the costs of ontology engineering projects is the size of the conceptual model, while the implementation issue is regarded to be a matter of tools, since a manual encoding of a conceptualization in a particular formal representation language is not common practice. However the original ONTOCOM model did not pay any attention to the semantical differences between the conceptual and the implementation level, differences which might appear in situations in which the usage of a specific representation language is mandatory. In this case the implementation of the ontology requires a non-trivial mapping between the knowledge level of the conceptualization and the paradigms beyond the used representation language. The costs

[5]Usability requirements express the constraints imposed by a particular characteristics of the ontology user community w.r.t. its content or content representation.

Rating Scale	Conceptualization
Very Low	concept list
Low	taxonomy, high number of patterns, no constraints
Nominal	properties, general pattern available, some constraints
High	axioms, few modeling pattern, considerable number of constraints
Very High	instances, no patterns, considerable number of constraints

Table 7: The Conceptualization Complexity Cost Driver CCPLX

Rating Scale	Implementation
Low	The semantics of the conceptualization compatible to the one of the impl. lang.
Nominal	Minor differences between the two
High	Major differences between the two

Table 8: The implementation complexity cost driver ICPLX

arisen during this mapping are stated in the driver ICPX (implementation complexity), whose ratings are illustrated in Table 8. For simplification reasons we restricted the range of the ratings to 3 (from low to high).

To summarize the complexity of the target ontology in ONTOCOM is taken into account by means of three cost drivers, associated with the efforts arisen in the domain analysis, conceptualization and implementation phase. We analyzed features which are responsible for cost increases in these fields – independently of the size of the final ontology, the competence of the team involved or the setting of the current project – and aligned them to ratings from very low to very high for quantification purposes.

4.2.2 The Cost Driver DOCU (Documentation Needs)

The DOCU measure is intended to state the additional costs caused by detailed documentation requirements. Likewise COCOMOII we differentiate among 5 values from very low (many lifecycle needs uncovered) to very high (very excessive for lifecycle needs) as illustrated in Table 9. In the original cost model DOCU was defined as

	Very Low	Low	Nominal	High	Very High
DOCU	many LC needs uncovered	some LC needs uncovered	right-sized to LC needs	excessive for LC needs	very excessive for LC needs

Table 9: Ratings for Documentation Costs

a building cost driver. During the model evaluation during its integration to the DILIGENT framework this driver was found to be relevant to top-level phases distinguished by the model i.e. also for reuse and maintenance.

4.2.3 The Cost Driver OE (Ontology Evaluation)

In the current cost model ontology evaluation is only regarded as part of the reuse phase. Our mapping to the DILIGENT methodology revealed that ontology evaluation is in fact performed before reusing external ontologies, but also after building a new ontology. Hence we broadened the the scope of the evaluation factor to building and reuse while keeping most of the original meaning (Table 10). While in a reuse situation the effort required for the evaluation of an ontology was monitored separately as the one implied for its comprehension, in the building case the level of the cost driver is determined autonomously of other cost factors by considering the level of activity required to test a preliminary ontology against its requirements specification document and for documentation purposes.

Rating Scale	Ontology Evaluation
Very Low	small number of tests, easily generated and reviewed
Low	moderate number of tests
Nominal	high number of tests
High	considerable tests, easy to moderate to generate and review
Very High	extensive testing, difficult to generate and review

Table 10: The Ontology Evaluation Cost Driver OE

4.2.4 The Cost Driver OI (Ontology Integration)

The most important ONTOCOM revision arisen as a result of the mapping to DILIGENT was the definition of cost driver for ontology reuse processes, which measures the costs produced by integrating different ontologies to a common framework. The integration step is assumed to be performed on ontologies sharing the same representation language – the efforts required for this activity are covered by the OT (Ontology Translation) cost driver [PBM05a]. As criteria influencing its complexity we identified the following:

- overlapping degree among ontologies to be integrated: it is assumed that this issue is proportional to the effort required by the integration, since it is directly related to the number of mappings between ontological entities.
- type of mappings between ontological primitives: 1 to 1 mappings are more easily discovered than multiple one (1 to n or n to m)
- integration quality, in terms of precision (rate of correct mappings) and recall (rate of mappings discovered): higher quality requirements imply automatically increased efforts to perform the integration task.
- number of ontologies: it is clear that the integration effort is directly proportional to the number of sources to be integrated

According to these considerations the ratings for the OI cost drivers were defined as depicted in Table 11 below.

Rating Scale	Ontology Integration
Very Low	1-1 mappings, approx. 50% precision and recall required, barely overlapping, 2 ontologies
Low	1-1 mappings, approx. 60% precision and recall required, barely overlapping, 2 ontologies
Nominal	1-n mappings, approx. 70% precision and recall required, some overlapping, 2 ontologies
High	1-n mappings, approx. 80% precision and recall required, high overlapping, more than 2 ontologies
Very High	n-m mappings, approx. 95% precision and recall required, high overlapping, more than 2 ontologies

Table 11: The Ontology Integration Cost Driver OI

Rating Scale	TOOL
Very Low	High quality tool support, no manual intervention needed
Low	Few manual processing required
Nominal	Basic manual intervention needed
High	Some tool support
Very High	Minimal tool support, mostly manual processing

Table 12: The Tool Support Cost Driver TOOL

4.2.5 The Cost Driver TOOL (Tool Support)

In the current cost model tool support is limited to the support of the reasoning, building and maintenance activities in the process. However, product factors like domain analysis, integration and others can and should also be supported by tools. Instead of pre-defining the types of tools which are usually involved in an engineering project independently of the process phases in which these tools come into operation, we take account of the different levels of tool support for the different phases by one general-purpose cost driver and calculate the final value as the average tool support across the entire process.

Therefore the ratings for tool support are re-defined on a more general level, as shown in Table 12 below.

The rating of the cost driver should be specified for each of the most prominent process phases, while the importance of the corresponding phase is expressed in terms of weights. The global TOOL value for a specific project is calculated as a normalized sum of the weighted local values. For the DILIGENT methodology one should specify the tool support level for the following sub-tasks: domain analysis, conceptualization, implementation, ontology understanding and evaluation, ontology instantiation, ontology modification, ontology translation, ontology integration and documentation.

4.3 Changes in the DILIGENT process

The mapping between the cost model and the DILIGENT process revealed some problems w.r.t. the current definition of the process activities (Section 4.1). The process activities were not defined at the same conceptual or granularity level. For instance, the previous model contained activities such as *Understand shared ontology (2)* and *Map equivalent conceptualizations of different actors (4)*. While the first one requires a deep conceptual understanding of the ontology, the second activity describes the technical conclusions from the previous one. We decided to define all activities on the same conceptual level and provide technical requirements of solutions within those activities. The roles, input and output factors and decisions were not changed. The new process model is depicted in figure 2.

4.3.1 Changes to Centralized Build

As aforementioned the build phase in the DILIGENTprocess model was so far underspecified, as we assumed that this step would be performed according to established engineering methodologies. In order to achieve a homogeneous description of the process activities from a granularity point of view we decided to further specify this phase at the same level of detail as the subsequent ones. As summarized in [GPFLC03]

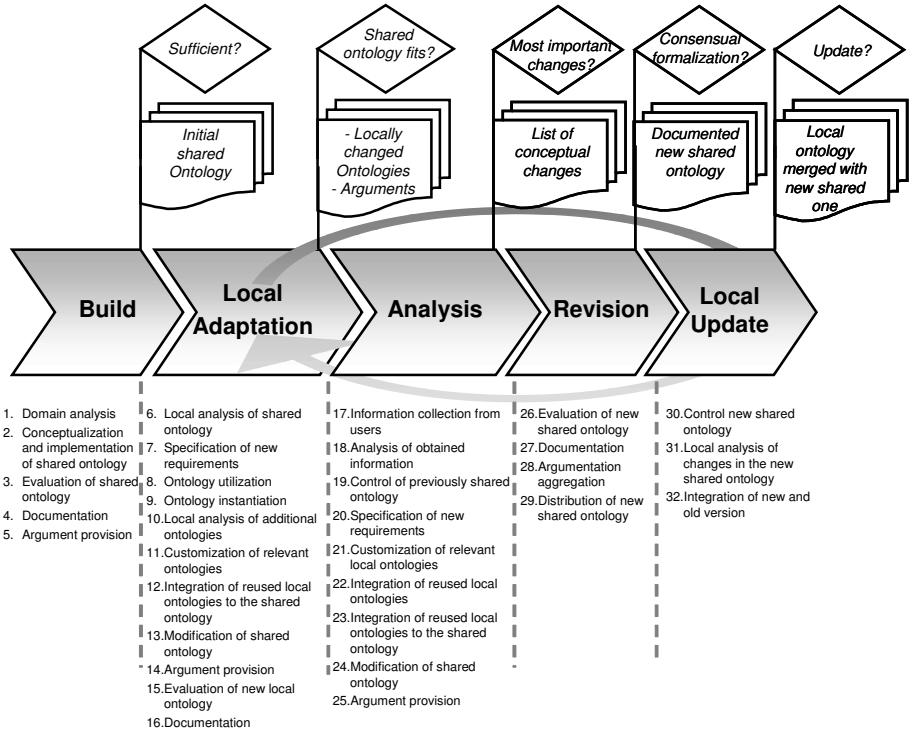

Figure 2: Process stages (1-5), activities (1-32) and structures

the available methodologies separate the building process in different activities, while each of the methodologies focuses on different aspects. For our purposes and for the alignment with the cost model we separated ontology building into the four activities *Domain analysis*, *Conceptualization and implementation of shared ontology*, *Evaluation of shared ontology*, *Argument provision* and *Documentation*.

Domain analysis In our case the domain analysis covers all tasks necessary to conceptualize the ontology. We thereby assume that the result of the feasibility study for the project was positive. A detailed definition on the execution of a feasibility study can be found in the OTK methodology [SAB+03, SS02]. The domain analysis itself covers tasks as defined in the specification phase in METHONTOLOGY [GPFLC03]. In the OTK methodology this activity is performed in the kickoff phase. We emphasize that the domain analysis includes the definition of modules covering sub domains in order to keep the building trackable. Competency questions are a good possibility to capture detailed requirements. An interesting special case occurs when existing ontologies should be reused. Here, domain analysis involves also the identification and selection of the reusable ontologies [PM00]. METHONTOLOGY introduces also management activities supporting the building process. These are of course necessary during the entire building process. However, as our process explicitly assumes only loose control we leave the level of management to the participants and their particular requirements. Knowledge acquisition is foreseen as a parallel activity to the building process in METHONTOLOGY. We acknowledge that knowledge acquisition is performed in all stage of the process but rather regard it as an implicit activity rather than an explicit one.

Conceptualization and implementation of shared ontology In contrast to other methodologies we subsume conceptualization and implementation of the ontology as one activity. As most building processes are supported by ontology editors, the actual implementation is mostly done automatically. For the applicant of the methodology the separation seems thus artificial. Conceptualization and implementation covers the tasks essential to build the shared ontology.

Argument provision In existing ontology building methodologies the capturing of the argumentation during the construction was neglected. In the field of Software Engineering capturing of arguments during the requirements analysis has been well-researched in the last decade. We have introduced an argumentation model tailored for the specific requirements of ontology building and maintenance [TPSS05]. In our process the explanation of the design principles underlying the conceptual model is of particular interest, since not all users are necessarily involved in the first building process but are allowed to change the ontology in later stages. Therefore, the arguments in favor and against certain design decisions must be captured to allow the users of the ontology to extend and change the shared ontology adequately.

Evaluation of shared ontology Ontology evaluation is not a mature area of ontology engineering yet. However several approaches proposes partial solutions for the evalu-

ation issue, from a general-purpose or a usage-related perspective. Approaches in the first category introduce methods to evaluate ontology content, i.e. the quality of the conceptual model [GP01, GW02, UHW+98]. Assessing the usability of an ontology in a target application context is addressed for example in OntoMetric [LTGP04], a framework for selecting appropriate ontologies. Additionally to these categories, we find approaches aiming at evaluating the a-posteriori usage of an ontology for a specific task such as semantic annotation of texts.

Documentation Ideally the documentation step should be performed in parallel to the aforementioned activities, in order to ensure an efficient process quality control and improve the reusability of the prototypical ontology, which is subject of further changes on the basis on its usage in a variety of environments in the distributed setting. Likewise Software Engineering it is crucial that the documentation is performed not only at the implementation level, but during the entire building process, including domain analysis and evaluation. Additionally to standard documentation procedures, in our process it is helpful to capture the participants in the original building process, who were involved in the initial design decisions.

4.3.2 Changes to Local Adaptation

New shared ontologies are made available to users either by push or pull mechanisms. The users deciding to utilize a new version of the shared ontology first get familiar with

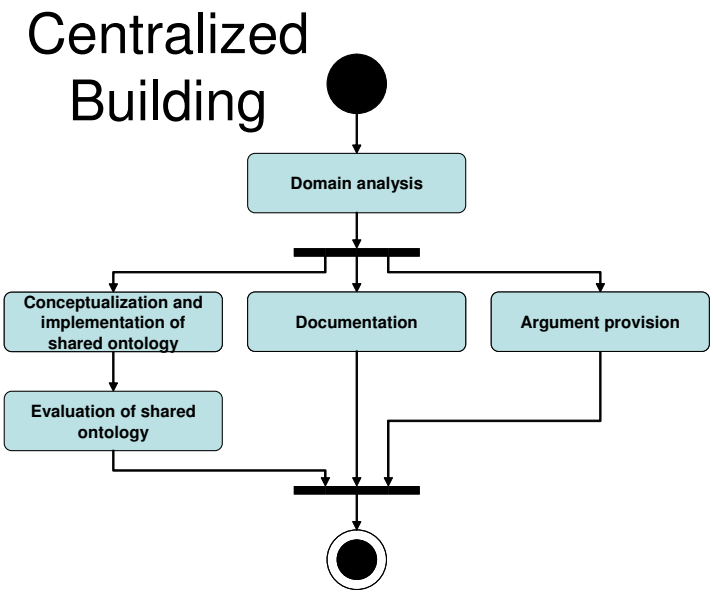

Figure 3: Centralized Building: Activity Diagram

the shared ontology in order to be able to use it correctly. In the next step they may interact with the ontology in parallel in a threefold manner, depending on the concrete application setting. Some users will use the ontology only for retrieving information either locally or from other participants. Others will also actively instantiate the ontology with their own information. Performing both activities the user may detect missing conceptualizations in the shared ontology. This and the analysis of shared ontology can result in the local definition of new requirements for the shared ontology.

Conceptualizing the new requirements incorporates activities as they are known from classical ontology engineering. Ontology users might decide to integrate existing ontologies (originating from different local parties involved in the process) to the local one or to conceptualize the desired changes from scratch. As other users, in particular the board, should be able to understand these changes, argument provision becomes crucial. Documentation is the last activity in the process. We acknowledge though that documentation was rarely done in most of our case studies. Figure 4 visualizes the dependencies between the single activities.

To achieve the desired output the user takes different activities namely *Local analysis of shared ontology Specification of new requirements Ontology utilization Ontology instantiation Local analysis of additional (local) ontologies Customization of relevant (local) ontologies Integration of reused (local) ontologies to the shared ontology Modification of shared ontology Argument provision Evaluation of new local ontology Documentation*

The evaluation of the adapted shared ontology might suggest that not all local requirements are met yet, thus we define a cyclic behavior inside the process stage. The single activities 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.

Local analysis of shared ontology The goal of this activity is to understand the ontology. An ontology is a conceptualization of the real world. It should furthermore be the result of a common agreement w.r.t. modeling issues. A completely shared ontology can never be engineered, since different people have varying interpretations of the real world. Therefore it is necessary as a first action to relate the own interpretation of the world to the shared conceptual model. Thus the user should learn where the different concepts are located in the ontology and how they are interrelated with other concepts. The ontology can be very complex, thus comprehension of the ontology depends mainly on its presentation. Different technologies can be used to provide the user with a context-sensitive view on the ontology to reduce complexity.

After completing this activity the user is able to instantiate the ontology and to use it to answer her information needs. It is not necessary that the user understands the entire ontology immediately. The analysis can be performed gradually. However for the parts the user modifies, instantiates or uses in subsequent activities she must first understand what these parts are about.

In order to understand the shared ontology the user should first look at the different modules (implicitly) defined in the ontology. She can then consult the definitions

of the concepts and read the available documentation. The arguments underlying the conceptualization may also be helpful.

As a guideline the user should start to understand the concept hierarchy before looking at the relations between the concepts. Defined axioms and inference rules are likely to be regarded last, because their understandability assumes a satisfactory comprehension level of the concepts, taxonomy and interconceptual relations. The meaning of the instances defined in the shared ontology may serve as good examples.

The analysis of the shared ontology is followed by three alternative activities. The

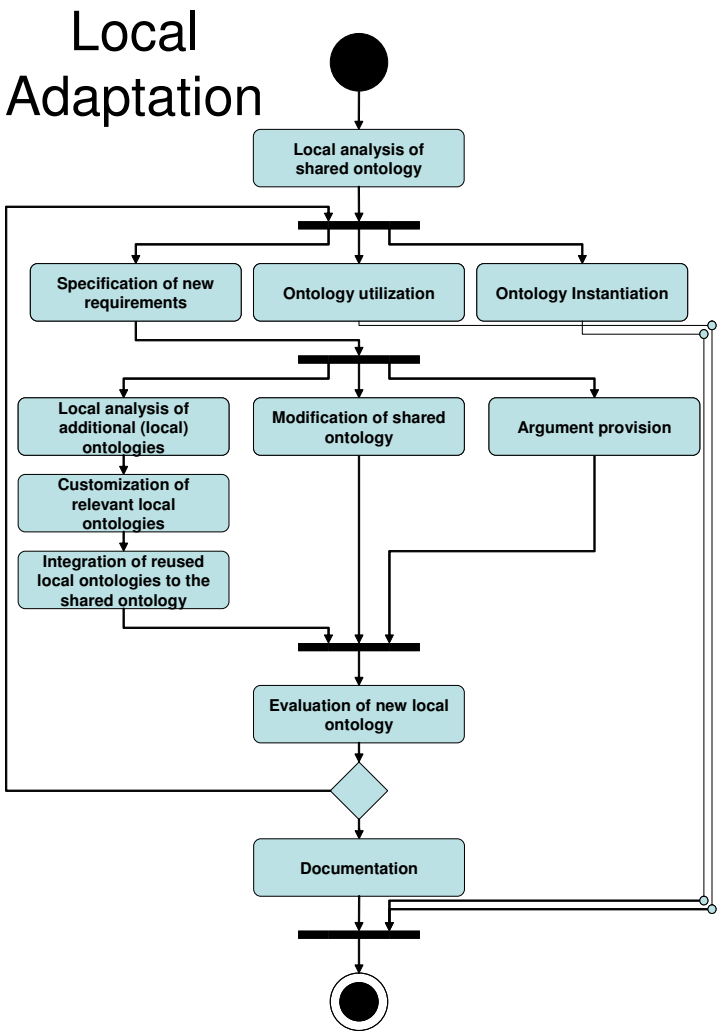

Figure 4: Local Adaptation: Activity Diagram

user might organize her local knowledge according to the ontology, modify it in order to improve its usability in the local context or use it to retrieve knowledge from the system. Hence, the underlying ontology can actually change depending wether the user has modified it in any of the parallel activities.

Ontology utilization This activity summarizes the general utilization of the ontology as a means to query and gather local and remote information. Depending on the restrictions imposed by privacy laws, the utilization can be automatically monitored in order to detect usage patterns. Frequency of use of certain entities in the ontology can enable the board to enhance the quality of the conceptual model. In case users miss concepts when formulating their information needs new requirements emerge.

The activity continues during the entire local adaptation step and is not proceeded by other activities.

Ontology instantiation Ontology instantiation also known as ontology population is the activity of classifying the available knowledge in terms of the ontology. The local information is one source of available knowledge[6] Other sources are the participants in the process and external providers. With this activity all participants contribute to the collective knowledge available in the system. Users also integrate knowledge they have retrieved from third parties into their own knowledge base. Relations between own and other conceptualizations beyond the already shared ontology might be detected during the integration of external knowledge. This activity includes also the provision of mappings between equivalent conceptualizations from different users.

The local and remote information may contain knowledge which cannot be formalized in the shared ontology and thus can lead to the specification of new local requirements. Depending on the ontology and the objective of system, information can be organized thematically. In our case studies the ontology comprised a topic hierarchy. Which knowledge is finally encoded in the ontology depends on the design of the ontology and the user needs. We have observed also different instantiation styles. Some users are very enthusiastic and instantiate the ontology very quickly other are more reluctant and shy away from the initial extra effort.

The activity continues during the entire local adaptation step and is not proceeded by other activities.

Specification of new requirements During the local adaptation phase, the more important activities from an ontology engineering perspective occur when the user realizes that the shared ontology does not conform to the requirements of the local application setting and should thus be modified towards these additional requirements. The objective of this activity is to identify and specify the requirements which lead to modification of the ontology.

The task(s) the ontology is involved in play(s) an important role in identifying the requirements. The possibilities are wide-ranging. As already mentioned in the build phase trained ontology engineers can use ORSD documents to capture the requirements in this phase [Sur03]. Other methods to identify requirements have also been

[6] In our case studies the local information consisted of documents, contacts, emails, Web bookmarks etc.

mentioned in conjunction with traditional ontology building efforts. [7] Requirements can be derived from competency questions as it was suggested in [GF95]. Existing ontologies might be the driver for new requirements as well. In our case study the identification of requirements through the analysis of existing folder structures was particularly useful. It is important to note that requirements can also come from other users – one participant in the process captures the requirements of other users as a representative.

The user must decide weather to implement the requirements consulting other users, adapting the local ontology on his own or to submit the requirements to a representative.

Local analysis of additional (local) ontologies Depending on the ontology development scenario the user might have access to other parties' ontologies and integrate their local adaptations in his own application ontology as a resource saving alternative to a new build. This activity is related to the activities defined in ontology building by reuse. As described above in the reuse setting ontology engineers must thoroughly examine the candidate ontologies. The ontology engineer should for example consider which conceptualizations must be change, removed, relocated, which definitions and documentations must be changed etc. Note that the translation of the ontologies involved in this reuse attempt is not an issue in DILIGENT, which assumes the usage of a unique representation language for the shared and local ontologies in the distributed scenario.

Reusing existing ontologies is a feasible alternative for both technically versed ontology engineers, which follow established reuse methodologies to fulfill this task, and eventually less experienced ontology users. The users should analyze the external ontologies in a straightforward manner. The shared ontology allows for examining only certain parts of the ontology and consider only them for reuse. Furthermore, users reuse other ontologies although they might not exactly fit their requirements, since they do not agree building it on their own. If small modifications of the shared ontology are needed to meet the user requirements then users will not consider other local ontologies.

In any case the result of the activity is the decision from whom to reuse which parts of the remote local ontologies.

Customization of relevant local ontologies The goal and the content of this activity depend on the result of evaluating external ontologies w.r.t. their local suitability. In the extreme fusion case the reused ontology serves only as an input to be completely reorganized. In the case of integration parts of the remote ontologies can be reused unchanged. Depending on the reuse level the remote ontologies might be subject of more or less radical customization measures. However, a lower customization effort will help the board to find reuse patterns.

[7] The users in our case study were mainly missing topics to be able to classify their documents in a enough fine granular manner. In this case the requirements analysis was less structured and mainly driven by the users experiences during the instantiation of the ontology.

The result of this activity is an ontology which can be integrated with the local shared ontology.

Integration of reused local ontologies to the shared ontology Finally the reused remote ontologies should be locally aligned to the shared ontology. This may result in the modification in their shared ontology. Again generalization or refinements can be necessary in order to integrate the reused ontologies.

Additionally to the traditional reuse activity the integration involves also a mapping task. The users keep a direct reference to the reused conceptualizations, with the beneficial consequences that they can access the remote users data and that the board can recognize communalities between different users, and utilize this knowledge when changing the shared ontology.

The result of this activity is an locally adapted shared ontology with adaptations based on remote users ontologies. The origin of the adaptations is stored in mappings between the reused and local ontologies.

Modification of shared ontology The modification of the shared ontology is another option to adapt it to the local requirements. Modifications range from small changes *e.g.* adding a new concept or relation to a complete restructuring of the shared ontology. The local requirements can also implicate that an entire new module should be integrated into the shared ontology.

The user should decide which parts of the ontology should be changed and how to implement the desired changes. The changes the user introduces may point to missing abstractions in the existing model.

Furthermore this activity starts from similar assumptions as they were described in the Sensus methodology [SPKR96]. The shared ontology represents far more knowledge than the user actually needs. This implies that the user will remove a number of conceptualizations from the shared ontology and maintain only a smaller part. In this case the board has the challenging task to decide which parts of the shared ontology should be removed because they are not needed and which parts are e.g. to general to be used but helpful for inter-user communication.

This activity includes the conceptualization as well as the implementation of the required changes.

As a result of this activity the local ontology fulfills the local requirements.

Argument provision As far as possible the user should track the reasons why certain modeling decisions were performed in a certain way. We propose the provision of arguments according to a specific model [TPSS05, PTS04] to capture these decisions. The model defines the process of providing arguments and several kinds of arguments, and aids decision making. While the latter is particularly relevant to collaborative ontology engineering, the first two aspects will help the board to understand the users decisions.

Arguments can range from simple usage examples (e.g., some document could not be classified using the ontology, some query could not be answered by the ontology to a satisfactory extent) to twisted argumentations trading-off the pro's and con's of a

decision. The more expressive the argumentation is, the easier it will be for the board to understand the reasons for the decisions and to integrate the newly submitted change requests to the shared ontology. Additionally, users intending to reuse the conceptualization – as aforementioned in the previous, reuse-oriented activities – are provided considerable support to comprehend and use the corresponding ontology correctly.

Evaluation of new local ontology The evaluation procedure is divided into three categories. The syntactic, semantic and pragmatic evaluation of the new local ontology. As the user utilizes the ontology mainly to organize his own knowledge the pragmatic evaluation is predominant in this activity. She can quickly realize wether the proposed way of organizing her knowledge in the shared ontology is sufficient to capture her local knowledge. The user requirements change with time, hence a sufficient local ontology can become insufficient after some time. In this case the user will start the process again and capture the emerging requirements.

Documentation As far as possible the user should document the changes introduced into the shared ontology. Documentation includes the meta data provision like, when a change was performed, who has performed the change, if the change was done on request from an other user etc. Furthermore, brief description of the added conceptualizations will facilitate the boards task.

4.3.3 Changes to Analysis and Revision

The board meets regularly in order to include emerging requirements into the shared ontology. To achieve the desired output the board takes different activities namely *Information collection from users Analysis of obtained information Control of previously shared ontology Specification of new requirements Customization of relevant local ontologies Integration of reused local ontologies to the shared ontology Modification of shared ontology Argument provision Argumentation aggregation Evaluation of new shared ontology Documentation* and *Distribution of new shared ontology*.

We now detail each one of the proposed activities:

Information collection from users Before the board can identify new requirements it collects the local ontologies from all participants, the respective argumentation, change requests provided by other means, usage information and finally mapping information.

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.

This activity reminds of the classical reuse approach in which candidate ontologies must be gathered. In contrast to the classical reuse approach the ontologies which must be integrated into the shared ontology is given. Furthermore the domain and application scenario are already defined. Usage information for the ontology is available, hence the relevance for the shared ontology is easier to determine. No translation must be performed in order to integrate the ontologies.

Control of previously shared ontology The goal of this activity is to examine the changes introduced in the last cycle. Specifically the board checks how many users have integrated the proposed changes and the tasks the shared ontology was used for. The board can detect if the users accept the common conceptualizations, if the analysis methods are appropriate and if the users understand and agree with the view of the board.

Specification of new requirements New requirements for the shared ontology can be obtained by analyzing the change requests, the changes in the local ontologies and the arguments provided by the users.

Analysis of obtained information 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 [GPFLC03].

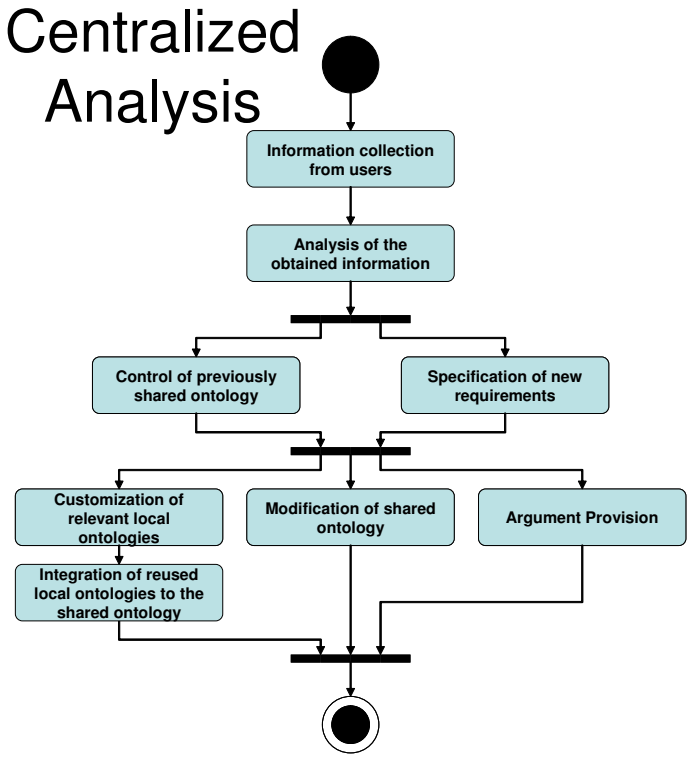

Figure 5: Analysis: Activity Diagram

The number of change requests may be huge 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 frequently 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.

Customization of relevant local ontologies After analyzing the changes and assigning them according to the concrete ontology modules they address, 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.

Integration of reused local ontologies to the shared ontology The customized reused local ontologies must be integrated with shared ontology. Here again it might be necessary to include abstractions or refinements into the shared ontology in order to be able to integrate the reused ontologies adequately.

Modification of shared ontology Similar to established methodologies the requested changes must be formalized with respect to the expressivity of the ontology. We will not go into detail with this step since it is already described in methodologies referred to in the related work section.

Argument provision As described above we have conceived an argumentation framework to support the discussion taking place in collaborative ontology engineering. The board is also supported in making decisions.

Evaluation of new shared ontology The board will evaluate the shared ontology from an syntactic and semantic perspective.

Argumentation aggregation As arguments play a major role in the decision process we expect that the changes which are eventually included into the common ontology are supported by many arguments. One of the reasons for keeping track of the arguments is to enable users to better understand why certain decisions have been made with respect to the ontology. Hence, the user should be able to retrieve the most convincing arguments made to introduce a certain change. Here the board aggregates the arguments exchanged during their discussion and makes them more accessible.

Documentation With the help of the arguments, the introduced changes are already well documented. However, we assume that some arguments might only be understandable for the domain expert and not for the users. Hence, we expect that the changes should be document to a certain level. In particular it should be documented who supported certain conceptualizations and the alternatives.

Distribution of new shared ontology Analogously to step 3.2.3 the shared ontology must be distributed to the different participants. Depending on the overall system architecture different methods can be applied here.

4.3.4 Changes to Local Update

As a result of the revision stage the participants in the process are aware of the new version of the shared ontology. They must now decide which parts - if any - they use from the new shared ontology. Switching from one ontology to an update incurs effort for understanding the new parts and partly reorganizing the local knowledge

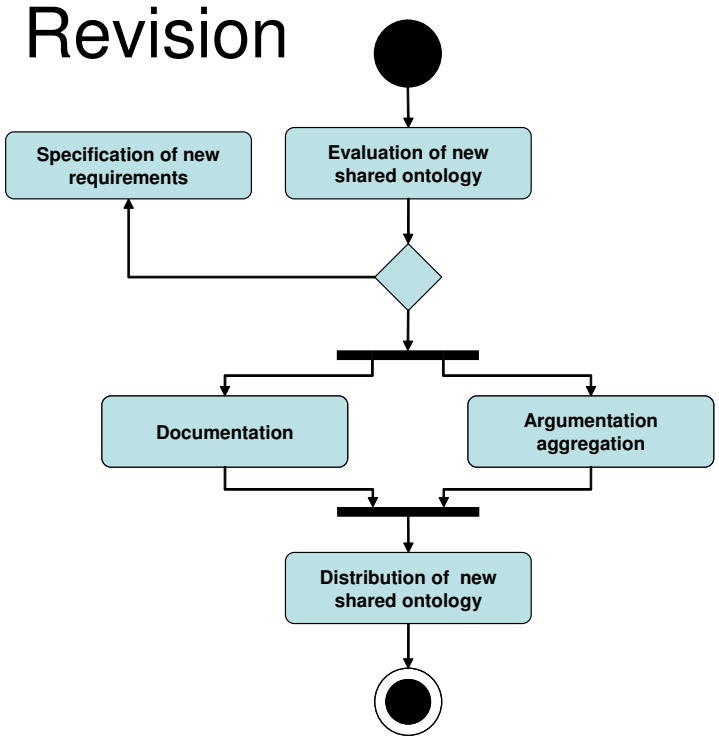

Figure 6: Revision: Activity Diagram

base. The gains of updating are lower communication effort and actual information. The incentives for the user to update are higher the more change requests to the shared conceptualizations are included in the shared ontology. Thus the user controls how many of the own proposals are included in the new version and in which way they are implemented. Furthermore the user analyzes all changes to the shared ontology and decides whether to finally integrate the new version with her local ontology.

This stage can be divided into the three activities *Control of new shared ontology* *Local analysis of changes in the new shared ontology* and *Integration of new and old version*.

Control of new shared ontology Likewise the board controlling the acceptance of the shared ontology the user controls the implementation of her own proposals. The user controls whether the proposed changes are implemented in the new shared ontology at all, conceptually or as proposed. This allows the user to judge which of her proposal are interesting for the community. Furthermore she learns how the board translates the proposals into conceptualizations in the shared ontology.

Local analysis of changes in the new shared ontology The user changes locally to the new shared ontology only if her benefits predominate the effort of updating. The analysis of the introduced changes inform her wether the changes effect her or not.

Technically this step requires the construction of an delta view on the ontology.

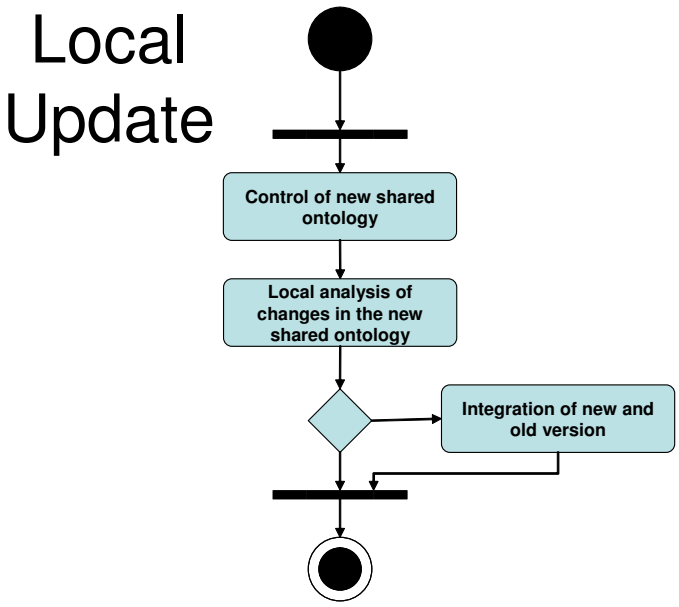

Figure 7: Local Update: Activity Diagram

Integration of new and old version The result of the analysis is the decision to integrate completely or partially the new shared ontology with the existing local ontology. The new shared ontology may contain refinements of existing model. In this case the user should consider to adapt her instantiations with respect to the refinements. The outcome of the controlling activity allows the user to decide which restructuring she must perform in order to stay in line with the new model. The new version can also be a model for knowledge which was previously not covered by the shared ontology. In this case the existing local knowledge can be the source for the population of the ontology in the next stage.

From a technical point of view we could identify several requirements from the case study. Acceptance and usability of the process model largely depends on the ease of translation from old to new versions. As in other systems the possibility to switch automatically between the different versions of the ontology enhances user experience. The system must support the user to easily integrate the new version into his local system. It must be guaranteed that all annotations made for the old version of the ontology are available in the new version.

The user should be enabled to use from now on the shared model instead of his own identical model. Furthermore, the board might have included a change based on arguments the user was bringing forward, but has drawn different conclusions. Here the user can decide whether he prefers the shared interpretation. Other option might emerge in the course of the case studies.

To ensure user satisfaction, the system must enable the user to return to his old version of the ontology at any time. The user might realize that the new updated version of the common ontology does not represent his needs anymore and thus want to leave the update cycle out. To reach a better acceptance this must be possible and is foreseen in the methodology. The user can always balance between the advantages of using a shared ontology or using his own conceptual model.

5 A Cost Function for DILIGENT Processes

In Section 4 we described the mapping between the ONTOCOM cost model and the DILIGENT methodology, which aimed at defining the role the cost drivers listed in the former play w.r.t. the efforts invested in individual phases and activities of the latter. On the basis of this mapping and the changes triggered by this task in both models we customized the general person month equation in ONTOCOM to the particularities of DILIGENT processes. The resulting function was further simplified in order to allow the elaboration of optimization criteria in DILIGENT, which we assumed to be useful as decision support for state transitions in the incremental engineering cycle.

5.1 The complete cost function

The general-purpose ONTOCOM equations 3 and 4

$$PM = PM_B + PM_M + PM_R \quad (3)$$

$$PM_x = Size_x * \prod CD_{xi} \quad (4)$$

which assume a linear engineering process, in which an ontology is built from scratch, by reuse or both and is further maintained by its users, were adapted to the cyclic model of DILIGENT as in equation 5 below:

$$PM = PM_{CB} + \sum_{i=1}^{n} (PM_{LA_i} * m_i + PM_{CAR_i} + PM_{LU_i} * m_i) * p^i, \quad (5)$$

where PM_{CB}, PM_{LA_i}, PM_{CAR_i} and PM_{LU_i} are the person months necessary for the initial building phase and for the local adaptations, centralized analysis and revision and local updates in cycle i, respectively. Note that i iterates over the number of cycles n and that in every cycle the number of sites participating at the process is considered through the variable m_i. Finally, we introduced the parameter p ($p > 0$) as a learning rate between consecutive cycles in the process model. Usually we can assume that $p \leq 1$, which means that the team involved in the project improves its experience level and is able to solve the same tasks more efficiently (i.e. with less costs) from one building cycle to another. However, while the positive learning rate is intended to reflect the changes occurring on the effort multiplier level between consecutive cycles, the size of the resulting ontologies (i.e. the size of the locally modified ontologies in the local adaptation phase, the size of the shared ontology obtained after a new board meeting and the size of the final locally updated one) vary from development cycle to development cycle. This observation justifies the usage of the index i in the second part of equation 5 for the person months variables PM_{LA_i}, PM_{CAR_i} and PM_{LU_i}, since their values are vary with the size of the ontologies involved.

We elaborated the detailed cost functions for each of the 4 enumerated process stages: centralized building, local adaptation, centralized analysis and revision and local updates.

5.1.1 The costs of the Centralized Building phase

$$\begin{aligned} PM_{CB} = {} & Size_{CBB} * \prod PROD_{CBB} * \prod PERS * TOOL + \\ & Size_{CBR} * \prod PROD_{CBR} * \prod PERS * TOOL \end{aligned} \quad (6)$$

The efforts required by the centralized building phase are divided into the ones invested in building a new ontology and the ones invested in reusing external ones. The product effort multipliers are as follows:

$$\begin{aligned} \prod PROD_{CBB} = {} & DCPLX * CCPLX * ICPLX * REUSE * \\ & DOCU * OE * OI \end{aligned} \quad (7)$$

$$\prod PROD_{CBR} = OU * OE * OI * OT * OM * DOCU \quad (8)$$

In case of the personnel factors the multiplier values is computed as in 9.

$$\begin{aligned} \prod PERS &= OCAP * DECAP * OEXP * DEEXP * \\ &\quad PCON * LEXP * TEXP \end{aligned} \quad (9)$$

Note that the reused size $Size_{CBR}$ should be calculated as the sum over all single ontologies reused, while the reuse equation contained in 6 was simplified in comparison to the original one in ONTOCOM [PBM05a].

5.1.2 The costs of the Local Adaptation phase

$$\begin{aligned} PM_{LA} = &\ Size_{LAS} * \prod PROD_{LAS} * \prod PERS_{LAS} * TOOL + \\ &\ Size_{LAM} * \prod PROD_{LAM} * \prod PERS_{LAM} * TOOL + \\ &\ Size_{LAR} * \prod PROD_{LAR} * \prod PERS_{LAR} * TOOL \end{aligned} \quad (10)$$

The first part of the equation calculates the effort required to evaluate and use the shared ontology ($Size_{LAS}$ is the size of the shared ontology). The rest accounts for the additional efforts arisen if external ontologies (i.e. developed locally at different sites) are analyzed for being reused in the local context or if the shared ontology needs to be modified ($Size_{LAM}$ is the modified size).

$$\begin{aligned} \prod PROD_{LAS} &= DATA * OU * OE * OI * DOCU \\ \prod PERS_{LAS} &= DEXP * DECAP * LEXP * TEXP * PCON \end{aligned} \quad (11)$$

Note that in the product equation we include the DATA driver to measure the costs of the instantiation of the ontology, while the personnel equation incorporates exclusively factors related to domain experts.

If the shared ontologies need to be refined in order to fulfill local needs, the ontology users may decide between reusing existing ontologies, which have been developed by other users in the network, or by performing the desired modifications themselves. For the first case the effort multipliers are listed in equation 12. The second one is addressed by 13.

$$\begin{aligned} \prod PROD_{LAR} &= OU * OE * OM * OI * DOCU \\ \prod PERS_{LAR} &= DEXP * DECAP * LEXP * TEXP * PCON \end{aligned} \quad (12)$$

Again the parameter $Size_{LAR}$ is understood as the total size of the reused ontologies. Integration costs arise, of course only in case external ontologies are reused (instead of separately modifying the shared ontology, the local adaptation may resort to existing modifications fulfilling the same requirements).

$$\begin{aligned} \prod PROD_{LAM} &= OM * OI * DOCU \\ \prod PERS_{LAM} &= DEXP * DECAP * LEXP * TEXP * PCON \end{aligned} \quad (13)$$

5.1.3 The costs of the Centralized Analysis and Revision phase

$$\begin{aligned} PM_{CAR} = \; & Size_{CARR} * \prod PROD_{CARR} * \prod PERS_{CARR} * TOOL * SITE + \\ & Size_{CARM} * \prod PROD_{CARM} * \prod PERS_{CARM} * TOOL * SITE \end{aligned} \quad (14)$$

In this equation the first part computes the efforts needed to evaluate the changes performed locally, while the second part of the formula states for the efforts invested in executing these modifications. Note that $Size_{CARR}$ is the total size of the local ontologies. The parameter SITE accounts for eventual additional costs produced by the distributed setting. Again we elaborate the product and personnel multipliers (Eq. 15 and 16).

$$\begin{aligned} \prod PROD_{CARR} &= OU * OE * OI * DOCU * OE \\ \prod PERS_{CARR} &= OEXP * OCAP * LEXP * TEXP * PCON \end{aligned} \quad (15)$$

$$\begin{aligned} \prod PROD_{CARM} &= OM * OI * DOCU * OE * REUSE \\ \prod PERS_{CARM} &= OEXP * OCAP * LEXP * TEXP * PCON \end{aligned} \quad (16)$$

In the equations above the Ontology Evaluation (OE) multiplier appears repeatedly, since the board necessitates an evaluation of the submitted ontologies and a final evaluation of the new shared ontology.

5.1.4 The costs of the Local Update phase

$$\begin{aligned} PM_{LU} = \; & Size_{LUR} * \prod PROD_{LUR} * \prod PERS_{LUR} * TOOL + \\ & Size_{LUI} * OI * DOCU * \prod PERS_{LUI} \end{aligned} \quad (17)$$

The parameter $Size_{LUR}$ designates the size of the "incoming" shared ontology, which is merged with the previous local one. For this reason, $Size_{LUI}$ is the sum of

the two sizes involved in the merging process, the size of the new shared ontology plus the one of the existing local ontology.

$$\begin{aligned} \prod PROD_{LUR} &= OU * OI * OE * DOCU \\ \prod PERS_{LUR} &= = DEXP * DECAP * LEXP * TEXP * PCON \end{aligned} \quad (18)$$

$$\prod PERS_{LUI} = = DEXP * DECAP * LEXP * TEXP * PCON \quad (19)$$

Equations 5 to 18 offer the parametric setting necessary for estimating the person month effort invested in arbitrary DILIGENT processes. However, as aforementioned, the costs' dimension might be additionally used as a decision support factor on achieving an optimal distribution between centralized and local building phases during the ontology life cycle. In order to achieve this goal we need a reduced cost function – on the basis of the one elaborated in this section – which allows us to identify the most important dependencies between the major parameters of the process, as these dependencies are likely to be responsible for the realization of an optimal configuration of central and local building phases. This configuration is to be discovered in terms of the free parameters of the reduced formula.

5.2 The reduced cost function

For the derivation of the reduced formula we start with the general DILIGENT cost equation 5 and consider the first level formulae for the corresponding person months calculations.

$$\begin{aligned} PM &= x_{new} * E_{new} + x_{reused} * E_{reused} + \\ &+ \sum_{i=1}^{n} (m_i * x_{las_i} * E_{las} + m_i * x_{lam_i} * E_{lam} + m_i * x_{lar_i} * E_{lar} + \\ &+ x_{carr_i} * E_{carr} + x_{carm_i} * E_{carm} + \\ &+ m_i * x_{lur_i} * E_{lur} + m_i * x_{lui_i} * E_{lui}) * p^i \end{aligned} \quad (20)$$

where the x_ks represent the sizes of the corresponding and the E_ks the multipliers. Note that the variation of the ontologies' size in each cycle is captured by the i indexation, while the variation of the cost driver values is modeled through the exponentially growing learning rate.

Let a be the average number of local changes submitted in every cycle $i = 1 \ldots n$, b be the average number of changes initiated by the board pro cycle and c the number of locally accepted changes. Further on let x_{la_i} be the size of a local ontology, which is submitted to the board after cycle i, x_{s_i} the size of the shared ontology obtained in the same cycle, and x_{lu_i} the size of the local ontology at the end of the cycle. The dependencies between the three sizes are as follows:

$$
\begin{aligned}
x_{la_i} &= x_{s_{i-1}} + a, \forall i = 1 \ldots n, x_{s_0} = x \\
x_{s_i} &= x_{s_{i-1}} + b, \forall i = 1 \ldots n, x_{s_0} = x \\
x_{lu_i} &= x_{s_i} + c, \forall i = 1 \ldots n
\end{aligned} \tag{21}
$$

If x is the size of the shared ontology in the first cycle ($x = x_{new} + x_{reused}$), then

$$
\begin{aligned}
x_{la_i} &= x + (i-1) * b + a, \forall i = 1 \ldots n, x_{s_0} = x \\
x_{s_i} &= x + i * b, \forall i = 1 \ldots n \\
x_{lu_i} &= x + i * b + c, \forall i = 1 \ldots n
\end{aligned} \tag{22}
$$

A simplification of the DILIGENTcost function is achieved if we reduce the effort multipliers related to the central board and to the user communities to single parameter with average values. That is, we abandon the difference between the effort multipliers involved in single activities of each process stage (e.g. the local analysis activity in the local adaptation phase), we obtain the formula below, in which E is again responsible for the centralized setting, while F represents average the local settings:

$$
\begin{aligned}
PM &= x * E + M * F * \sum_{i=1}^{n} (x + (i-1) * b + M * a) * p^i + \\
&+ M * F * \sum_{i=1}^{n} (x + i * b + c) * p^i + \\
&+ E * \sum_{i=1}^{n} (M * (x + (i-1) * b + a) + b) * p^i
\end{aligned} \tag{23}
$$

In formula 23 we also assume a constant number of sites $m_i = M$ and an initial size of the ontology x.

If $n \to \infty$ then the formula 23 is further transformed to

$$
\begin{aligned}
PM &\approx x * E + M * F * (2x + M * a + c) * \frac{1}{1-p} + \\
&+ M * F * b * \frac{p^2 + p}{(1-p)^2} + \\
&+ E * (M * (x + a) + b) * \frac{1}{1-p} + \\
&+ E * M * b * \frac{p^2}{(1-p)^2}
\end{aligned} \tag{24}
$$

5.3 Applications of the reduced cost function

In order to come up with an approximation of the cost function corresponding to DILIGENT engineering processes we start with equation 24 and isolate the terms depending on the involved cost drivers E and F:

$$\begin{aligned} PM &\approx E * (x + \frac{1}{1-p} * (M * (x+a) + b) + M * \frac{p^2}{(1-p)^2} * b) + \\ &+ F * M * ((2x + M * a + c) * \frac{1}{1-p} + b * \frac{p^2+p}{(1-p)^2}) \end{aligned} \quad (25)$$

The formula above is used to analytically describe alternative engineering scenarios in DILIGENT. We illustrate the usage of the cost function as an objective means for decision support for the following tasks:

- the identification of a specific engineering strategy: The DILIGENTmethodology foresees a two-step engineering approach in which a first part of the shared ontology is jointly developed by domain experts and engineers, while the rest of the ontology evolves according to the needs of its users. Cost information might be useful to identify the sweet spot between the effort invested in centralized building and the remaining phases. A second engineering decision relates to the possibility of taking into consideration external parties' ontologies (i.e. local versions of the shared ontology available at external sites) while performing modifications. A first possibility is to modify the shared ontology according to the local requirements and submit potentially locally relevant change requests independently of the requirements of other user communities across the network. In the second, reuse-oriented scenario the users first try to map their own requirements to local ontologies emerging across the network and to reuse these local ontologies instead of introducing new change requests. The decision on one of the alternatives could be documented by means of cost information.
- the identification of the optimal meeting frequency: For an optimal process execution one needs decision criteria to estimate the frequency of the board meetings and implicitly a rate for the amount of submitted and approved changes to the shared ontology. Since every new board meeting is related to (basic) costs for the centralized analysis and the local updates too frequent meetings are expected to produce an overload both on the side of the ontology engineers and of the users.

In order to analytically describe the aforementioned scenarios, we consider a simplified version of the DILIGENT process, in which the costs of the local updates are negligible. In this case, equation 25 is transformed to

$$\begin{aligned} PM &\approx E * (x + \frac{1}{1-p} * (M * (x+a) + b) + M * \frac{p^2}{(1-p)^2} * b) + \\ &+ F * M * ((x + M * a) * \frac{1}{1-p} + b * \frac{p^2}{(1-p)^2}) \end{aligned} \quad (26)$$

5.3.1 1. Scenario: The size of the initial ontology

In order to analyze the impact of the initial size of the ontology on the overall costs required to create an ontology of a given final size we compare the costs defined in equation 26 with the ones implied by an initial ontology of size $x + \alpha$. In this situation, we assume that the number of changes required by the users decrease with the size of the initial ontology with a parameter β. A large initial ontology is profitable if

$$E * M * (\alpha - \beta) + F * M * (\alpha - M * \beta) - \alpha * F * M > 0 \tag{27}$$

Inequality 27 is equivalent to $E * (\alpha - \beta) > F * M * \beta$, which means that increasing the size of the ontology is profitable as long as the costs arisen by this activity for analysis and evaluation do not overcome the costs required to originally build the additional concepts. The last inequality will be not fulfilled for a sufficiently high number of sites M, which implies that we need to start with a small shared ontology in case we need to handle a high number of sites. In case the number of sites is sufficiently low, the satisfiability of the inequality is influenced by the ratio between the productivity of the board and of the ontology users.

5.3.2 2. Scenario: Reuse-oriented vs isolated building

In order to detect the impact of local ontology reuse on the overall costs, we proceed in a similar manner as in the first scenario by comparing the difference arising from modifying a number of α concepts instead of trying to reuse them from external sources. By replacing the number of changes a in 26 with $a + \alpha$ we obtain that reusing existing conceptualizations is profitable only if $E * M * \alpha + F * M * \alpha > F * M^2 * \alpha$. Again, reuse is more feasible for scenarios with a relatively low number of sites. If M is sufficiently high, the inequality above is not satisfied anymore.

5.3.3 3. Scenario: Frequency of board meetings

In order to analyze the optimal frequency of board meetings, which influence the number of submitted and accepted changes (the frequenter the meetings are, the less changes are submitted pro cycle). If α is the difference between the number of submitted changes for a higher number of development cycles, β is the difference at the level of approved changes – the costs for local updates are ignored – then the difference implied by these three parameters w.r.t. the person months efforts in two consecutive cycles is determined from equation 23 as:

$$\begin{aligned} PM_i * P - PM_{i+1} &= M * F * ((i-1) * \beta + M * \alpha - b_{i+1} + \\ &+ E * (M * (i-1) * \beta - M * b_{i+1} + M * \alpha + \beta) \end{aligned} \tag{28}$$

In this case $PM_i * P - PM_{i+1} > 0$ if $(i-1) * \beta + \alpha > b_{i+1}$ and $M * \alpha + (i-1) * \beta > b_{i+1}$. The latter is satisfied for a sufficiently high number of sites M, while the former depends on the parameter i. If $\beta + \alpha > b_{i+1}$ increasing the number of meetings is feasible independent on the number of sites or on the learning rate.

6 Data collection and model calibration

In this report we demonstrated the ways the generic cost estimation model ONTOCOM was applied to the ontology engineering methodology DILIGENT. The alignment of the model to this particular methodology also revealed the limitations of the model w.r.t. a complete coverage of ontology engineering aspects. As a consequence the estimation model was refined with cost drivers such as "Ontology Integration". In the same time, the alignment can be seen as a significant step towards the final validation of ONTOCOM, which is performed according to the quality framework described in Section 2. However, the usability of the model in real-world settings is primarily dependent on the accuracy of its results, achieved after calibrating the a-priori parameter values on the basis of historical project data. For this purpose, we analyzed the technical means which can be used for the calibration and produced an online questionnaire for the collection of the data. These two issues are described in the remaining sections.

6.1 Technical realization of the data collection

For the realization of an online tool for data collection we made use of the Open Source survey software PhpESP (available at `http://sourceforge.net/projects/phpesp/`), which offers basic functionality for the generation of public surveys. The data collection procedures is foreseen as a set of questions by which the user is required to provide introductory information about a specific project (i.e. an ontology) and to specify the values of the parameters included in the cost model. As a result of the survey, the data is stored in a relational database and is exported to a statistical component in order to be used for the calibration of the model.

Figure 8 depicts the introductory section of the data collection survey, while Figure 9 shows an excerpt related to the cost drivers "DATA" and "DCPLX". For each cost driver[PBM05a] we provide a short explanation of the scope and associated decision criteria, which are intended to be used to aid the data provider in specifying the rating value of the driver. Finally we can export the collected data as shown in figure 10 and calibrate the model.

A current version of the survey is available online at `http://kompass.mi.fu-berlin.de/phpESP/public/survey.php?name=ontocom2final_260905`.

6.2 Calibration method

The data collected in the survey is used to calibrate the ONTOCOM cost estimation model. Before we explain the methods used to calibrate the model, we need to emphasize the restrictions of any calibration. (1) Due to the number of cost drivers we need a high number of observations to calibrate the model in a statistically significant way. Any calibration with less than approximately 300 data points will not be significant from a statistical point of view. Thus any calibration can only be seen as an indication of the direction. (2) Experience in other fields with cost estimation models suggests, that a calibration for a particular company or project team yields more accurate estimations than a general purpose calibration. Our calibration can therefore only serve

as an example for the calibration process, rather than an accurate model calibration. Nevertheless, the calibration is useful, as project teams can compare their estimations against a general average as provided by us. (3) A calibration uses historical data to estimate future outcomes. Although the average and the variation observed in the historical data may also be observed in future projects, any specific project can still require

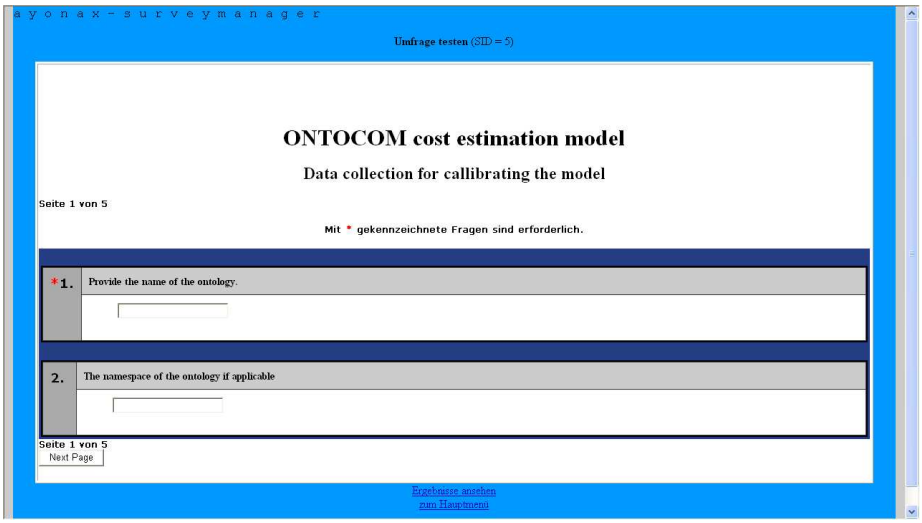

Figure 8: ONTOCOM data collection: introductory questions

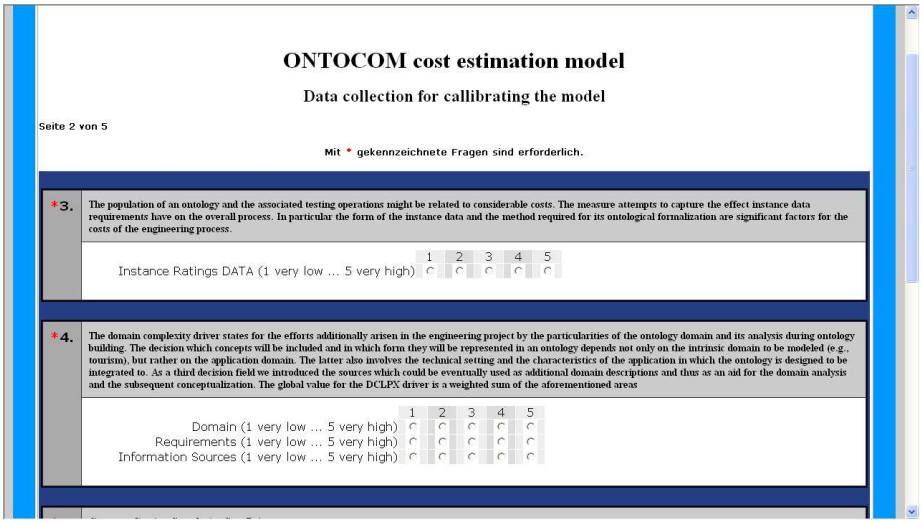

Figure 9: ONTOCOM data collection: cost drivers

Parameter	Description
A	adjustment parameter
$Size$	The size of the ontology
$DCPLX(CD_{X1})$	Effort multiplier for domain complexity
$OE(CD_{X2})$	Effort multiplier for final ontology evaluation complexity
$REUSESize$	Size of the reused ontology
OU	Reused ontology understandability

Table 13: Simplified cost model factors

significantly more or less effort to build the ontology than the predicted one.

The number of cost drivers defined in ONTOCOM is too high, to work through a conclusive example. In order to explain the calibration method to refine the ONTOCOM cost model we introduce a very simple cost model. For the simplified cost model we provide a complete example.

Our simplified cost model consist of six factors as listed in table 13.

$$AdSize_X = Size_X - (1 - \%reuse) * REUSESize * OU \tag{29}$$

$$PM_X = A * AdSize_X * \prod_{i=1}^{2} CD_{Xi} \tag{30}$$

Figure 10: Data export from phpESP

Rating	DCPLX			OE			OU		
	E1	E2	Av.	E1	E2	Av.	E1	E2	Av.
very low	0,6	0,8	0,7	0,6	0,8	0,7	0,6	0,8	0,7
low	0,7	0,9	0,8	0,7	0,9	0,8	0,7	0,9	0,8
nominal	1	1	1	1	1	1	1	1	1
high	1,1	1,3	1,2	1,1	1,3	1,2	1,1	1,3	1,2
very high	1,8	2,0	1,9	1,8	2,0	1,9	1,8	2,0	1,9

Table 14: Delphi result

For our simplified cost model the Delphi method resulted in the following expert estimations for our effort multipliers (in table 14, the estimation of the experts are abbreviated with $E1$ and $E2$, respectively. Average values are termed by $AV.$).

We collected data from six ontology building projects. The results are summarized in table 15 ($RSIZE$ is the size of the reused ontologies, while the DCPLX columns correspond to the three decision components defined for the cost driver Domain Analysis Complexity, as introduced in Section 4).

				DCPLX					
Ontology	SIZE	PM	% newly build	req.	con.	info.	RSIZE	OU	OE
swpatho1	1300	5	20	5	1	4	1040	2	5
opjk	700	2,6	100	5	4	5			5
ArguOnto	200	2	100	4	2	4			2
COS	75	2,5	100	5	3	3			5
OMV	300	2,5	100	3	2	4			3
VDO	1400	0,5	100	4	4	4			1

Number	Rating
1	very low (VL)
2	low (L)
3	nominal (N)
4	high (H)
5	very high (VH)

Table 15: Data collection

The collected data is than adjusted in order to apply the calibration. In this step the ratings for the domain complexity are averaged and the size of the reused ontology and the final size are combined 16.

The data collected from real projects can now be used to calibrate our model. Linear regression is the adequate method to find the adjusted parameters. We reformulate equation 30 in order to apply later on linear regression and introduce a parameter β_i as an exponent for the cost drivers. β_i is a scaling factor, by which the existing parameters should be scaled in order to fit the model. We recall that α is factor to represent a

Ontology	AdSize	PM	DCPLX	OE
swpatho1	635	5	3	5
opjk	700	2,6	5	5
ArguOnto	200	2	3	2
COS	75	2,5	4	5
OMV	300	2,5	3	3
VDO	1400	0,5	4	1

Table 16: Adjusted collected data

learning rate, in other case also used to model economies of scale.

$$PM_X = A * AdSize_X^{\alpha} * \prod_{i=1}^{2} CD_{Xi}^{\beta_i} \tag{31}$$

We apply the logarithm to equation 31 and can now apply a classical linear regression to our data to estimate β_i. This step is only possible if our data is distributed exponentially, thus we have significantly more data points with a low number of entities than with a high number of entities. We omit this test for our example, but will do so for the final calibration.

$$\ln(PM_X) = \ln(A) + \alpha * \ln(AdSize_X) + \sum_{i=1}^{2} \beta_i * ln(CD_{Xi}) \tag{32}$$

The resulting matrix of data points can be used to calculate the covariance matrix and the correlation matrix. In particular the correlation matrix is helpful, to identify cost drivers which are highly correlated and can thus be integrated into one. For the sack of completeness we have listed the results of the correlation analysis in table 17. The analysis of the correlation matrix reveals that for our limited data set the total effort for building the ontology is highly correlated with the extend of the ontology evaluation activity. Furthermore domain complexity and ontology evaluation are correlated. Surprisingly, the size of the ontology and the required effort to build it are inversely correlated, which implies the larger the ontology becomes the less effort one has to spend building it. This result shows, that an estimation model, calibrated only based on historical data could in fact be misleading. We survey several methods to overcome this problem later.

In table 18 we have summarized the results of the linear regression. Applying the results to the parameters we get new parameters according to table 19 ($Av.$ means "Average").

We can now compare the predicted effort according to our model before and after its calibration. As recognized before some of the results of the linear regression are counter intuitive. Different options exist to overcome this problem.

	AdSize	DCPLX	OE	PM
AdSize	1,00			
DCPLX	0,27	1,00		
OE	-0,25	0,41	1,00	
PM	-0,40	-0,09	0,74	1,00

Table 17: Correlation matrix for our example

	A	DCPLX	OE	α
m_i	0,84	-1,37	1,48	-0,03
$s(m_i)$	1,79	1,34	0,70	0,30

Table 18: Results of the linear regression

6.2.1 Linear combination

The expert ratings found in the Delphi experiment are a-priori estimations for our parameters and incorporate knowledge about the underlying activities. Using only historical data to calibrate the model would thus waste this knowledge. A natural solution is to combine the values estimated by the experts with the parameters found from the historical data. In the literature a combination which weights the expert values with 90% and the values from the linear regression with 10% is proposed.

6.2.2 Bayesian Linear Models

The linear combination of expert estimations and historical data is not optimal. The combination should take into account the number of data points used for the linear regression and the variance observed in the expert rating as well as in the data points. A factor which all experts have given the same rating, while the linear regression results in a high variance should be influenced less by the data than by the experts. Bayesian analysis is a way to achieve the desired outcome. [DC99] provides an exhaustive explanation of the application of Bayesian analysis for cost estimation models. As Bayesian analysis requires methods which go beyond the standard statistical functions offered by eg. Microsoft Excel software packages such as produced in the Bayesian inference Using Gibbs Sampling (BUGS) project `http:`

Rating	DCPLX			OE			A			α		
	Delphi	Data	Av.	Delphi	Data	Av.	Delphi	Data	Av.	Delphi	Data	Av.
VL	0,7	1,63	0	0,7	0,59	0	1	2,31		0,98	-0,03	
L	0,8	1,36	0	0,8	0,72	0	1	2,31		0,98	-0,03	
N	1	1	1	1	1	1	1	2,31		0,98	-0,03	
H	1,2	0,78	0	1,2	1,31	0	1	2,31		0,98	-0,03	
VH	1,9	0,41	0	1,9	2,59	0	1	2,31		0,98	-0,03	

Table 19: Parameter estimation from experts and based on the data

//www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml must be used. An Excel package http://www.jstatsoft.org/v14/i05/v14i05.pdf for the BUGS software exists[8].

Later on we consider to offer an online service which calibrates the model automatically when new information is available. In this case a PHP based service might be useful[9].

For our purposes we can weight the parameters based on the variance of the linear regression parameters m_i. In this case the weight of parameter is calculated as in equation 33.

$$CD_{Xi}^{new} = CD_{Xi}^{expert} * (1 - \frac{1}{s(m_i)^2 + 1}) + CD_{Xi}^{data} * \frac{1}{s(m_i)^2 + 1} \tag{33}$$

As the Bayesian analysis is a very sophisticated method, our data however is still very limited we opt for the linear combination of expert and estimated parameters. In table 20 we compare the accuracy of our estimations w.r.t. different parameter settings. As accuracy we define the percentage of estimations, which lie within a certain range of the actuals.

Name of the Ontology	PM actual	Expert		Linear 90% expert		Linear 20% expert		Linear 0% expert		Simple Bayesian	
		Accuracy									
		30%	50%	30%	50%	30%	50%	30%	50%	30%	50%
		0,17	0,17	0,0	0,67	0,33	1,0	0,17	1,0	0,33	0,83
		Estimations									
swpatho1	5	1,2		3,6		4,5		4,8		4,0	
opjk	2,6	2,5		6,4		3,2		2		3,7	
ArguOnto	2	0,2		1,5		1,4		1,4		1,5	
COS	2,5	0,2		4,5		4,2		4		4,0	
OMV	2,5	0,3		1,9		1,9		1,9		1,9	
VDO	0,5	1,2		1,4		1,0		0,8		1,1	

Table 20: Effort estimation based on expert estimation and historical data

Alternative calculation

As we have already observed from the correlation matrix, size and effort are invers correlated. This has unintended effects on the estimated learning rate. In order to overcome this problem from the beginning, we can assume a learning rate of 1, thus

[8]Further examples can be found at http://www.biostat.umn.edu/~sudiptob/pubh5485/BayesianLinearModelText.pdf

[9]http://www.devshed.com/c/a/PHP/Implement-Bayesian-inference-using-PHP-Part-1/

	A	DCPLX	OE
m_i	-5,20	-3,29	2,48
$s(m_i)$	0,68	2,60	1,37

Table 21: Results of the linear regression - alternative

we do not assume any learning. This changes eq. 31 slightly and thus 32 as shown in eq. 34

$$\ln(\frac{PM_X}{AdSize_X}) = \ln(A) + \sum_{i=1}^{2} \beta_i * ln(CD_{Xi}) \tag{34}$$

In this case the results of the linear regression are as shown in table 21

Future path

We have demonstrated with a simplified example the process of calibration for the ONTOCOM model. As we are now gathering more data of real world ontology building efforts we will soon be in a position to calibrate the complete model. The calibration will probably result in an adaption of the cost drivers. Some cost drivers might be highly correlated and can thus be joined. Others might have such a big impact on the final estimation that they can be divided into more than one cost driver. As soon as more accurate date is available we will continue and report the results of the calibration.

7 Related work

As mentioned in the introduction of this report estimating costs is a fundamental requirement for a wide-scale dissemination of ontologies in business contexts. However, though the importance of cost issues is well-recognized in the community, no cost estimation model for ontology engineering is available so far. Cost estimation methods have a long-standing tradition in more mature engineering disciplines such as software engineering or in industrial production. Approaches in these areas[Boe81, Kem87, Ste95] offered us valuable information about methods which can be applied to define and evaluate ONTOCOM.

Established methodologies for ontology engineering focus on the centralized development of static ontologies, i.e. they consider the iteration between ontology construction/modification and utilization only in passing (see for instance [GPFLC03, SSSS01] for a review of the methods). Further on, these methodologies do not adress economical aspects of engineering processes, such as cost management, cost reduction or cost benefit analysis. Methodologies such as **METHONTOLOGY**[GPFLC03] or the **OTK Methodology**[SS02] can be considered representative for the current state of the

art in this field. They offer guidance for building ontologies either from scratch, for reusing other ontologies as they are, or for re-engineering them. They divide ontology engineering processes into several stages which produce an evaluated ontology for a specific domain. **Holsapple et al.** [HJ02] focus their methodology on the collaborative aspects of ontology engineering while still aiming 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 [KVA04]. 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 not reported that their methodology had been applied in a case study neither do they provide any detailed description of the defined process stages.

8 Conclusions

In the last couple of years we witness a change of focus in the area of ontologies and ontology-based information systems: while the application of ontologies was restricted for a long time to academia projects, in the last ten years ontologies have become increasingly relevant for commercial applications as well. A first prerequisite for the successful introduction of ontologies in the latter setting is the availability of proved and tested Ontology Engineering methodologies, which break down the complexity of typical engineering processes and offer guidelines to monitor it. Existing methodologies have proven to fulfill these requirements. A further prerequisite is, however, the availability of cost information for the ontology building effort so that the project initiator can compare the estimated costs against the prospected utility of the ontology. Research in this field is not very advanced yet, but ONTOCOM is an initial attempt in this direction.

In this report we have described the ONTOCOM cost estimation model and the DILIGENT ontology engineering methodology. ONTOCOM is a parametric cost estimation model, which assumes a linear relation between the size of an ontology and a serie of cost drivers which are determined according to the project setting. The model can be applied in the early project phases (such as the feasibility study) in order to compute an estimation of the effort (expressed in person months) arisen by building, reusing or maintaining ontologies. DILIGENT is a methodology for the distributed, loosely controlled and evolving engineering of ontologies,

We have shown that the cost model ONTOCOM can be aligned to a specific ontology engineering process such as DILIGENT, which covers all major phases of ontology engineering, such as building, reuse and maintenance. The alignment process was beneficial for both models as missing cost drivers were identified within ONTOCOM and activities in the DILIGENT process stages could be generalized and extended. As a

result ONTOCOM incorporates now 25 cost drivers which cover efforts for ontology building, reuse and maintenance, divided into three categories: product, personnel and project cost drivers. The five DILIGENT process stages are now defined at a finer and more homogeneously grained level and are subdivided into 32 activities.

The cost function specific for DILIGENT process was defined according to the ONTOCOM model and the alignment of the cost drivers to the activities they have an impact on. The cost function was then simplified in order to enable the usage of cost information as decision support for three engineering scenarios, which were specified during this work: 1.) finding the optimal size of the initial ontology, 2). the extension of reuse at the local sites, and 3). the optimal frequency of board meetings. In summary, our analytical investigations on the basis of the simplified cost function revealed that the decisions should depend on the number of sites the ontology is used at, and the capabilities of the ontology engineers and its users respectively.

In order to calibrate the ONTOCOM model, thus to find the right parameters for the different cost factors, we set-up an online survey to collect data from existing ontology engineering projects. So far the survey was utilized to capture data from 22 projects. To minimize misinterpretations of the cost drivers and their ratings, the authors of this report interviewed members of the corresponding engineering teams and entered the data themselves. Up-to-now the data collection covers historival projects at our institutes (Free University of Berlin, University of Karlsruhe) and the EU project SEKT. Data collection from the Knowledge Web project and other organizations will follow.

Although the number of collected data points is still insignificant, we can already draw some preliminary conclusions and point to future research issues. The interviewed persons could describe their experiences during the ontology building effort with the proposed cost drivers, which demonstrates the usability of the cost model for the intended class of engineering projects. The alignment to the elaborated DILIGENT methodology definitely contributed to a large extent to this satisfactory coverage. Nevertheless future alignments with other ontology engineering methodologies are expected to help us complete and refine ONTOCOM's list of cost drivers.
Further on our experiences so far suggest that the complexity of ontology evaluation has a significant impact on the overall project costs. Consequently, this indicates that better support for ontology evaluation could yield important benefits. Another potential cost-relevant parameter appears to be the number of domain experts from different domains, building a single shared ontology. The extension of the model with this coordinate, which is not supported by the current version, is subject of future investigations.

Beyond cost estimation, the list of cost drivers was found helpful for ontology engineers to breakdown activities related to a specific ontology building process during the feasibility study.

With this T-REX exchange we could not only improve the existing models for cost estimation and ontology building but also identify new research questions. Furthermore with the integration of the results in the two European projects SEKT and Knowledge Web we can also guarantee that the models are evaluated in real world case studies and reach an audience beyond the research community.

Aknowledgements This work has been partially supported by the European Network of Excellence "KnowledgeWeb-Realizing the Semantic Web" (FP6-507482), as part of the KnowledgeWeb researcher exchange program **T-REX**, and by the European project "Sekt-Semantically-Enabled Knowledge Technologies"(EU IST IP 2003-506826). We want to thank all the people who agreed to contribute to the ONTOCOM data collection.

References

[B. 97] B. W. Boehm, C. Abts, B. Clark and S. Devnani-Chulani. COCOMO II Model Definition Manual, 1997.

[Boe81] B. W. Boehm. *Software Engineering Economics*. Prentice-Hall, 1981.

[DC99] Sunita Devnani-Chulani. *BAYESIAN ANALYSIS OF SOFTWARE COST AND QUALITY MODELS*. PhD thesis, FACULTY OF THE GRADUATE SCHOOL UNIVERSITY OF SOUTHERN CALIFORNIA, 1999. `http://sunset.usc.edu/publications/dissertations/SChulani.pdf`.

[Epp01] M. J. Eppler. The Concept of Information Quality: An Interdisciplinary Evaluation of Recent Information Quality Frameworks. *Studies in Communication Sciences*, 1:167–182, 2001.

[ES04] Marc Ehrig and Steffen Staab. QOM - quick ontology mapping. In *Proc. of the 3rd ISWC*, 2004.

[FL99] M. Fernández-López. Overview of methodologies for building ontologies. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*. CEUR Publications, 1999.

[GF95] M. Grueninger and M. Fox. Methodology for the design and evaluation of ontologies, 1995.

[GP01] A. Gmez-Prez. Evaluation of ontologies. *International Journal of Intelligent Systems*, 16(3), 2001.

[GPFLC03] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, 2003.

[GW02] N. Guarino and C. Welty. Evaluating Ontological Decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, 2002.

[HJ02] C. W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Communications of the ACM*, 45(2):42–47, 2002.

[HLW99] K. T. Huang, Y. W. Lee, and R. Y. Wang. *uality Information and Knowledge*. Prentice Hall, 1999.

[HSC02] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-CREAM – Semi-automatic CREAtion of Metadata. *Expert Update, Special Issue - Intelligent Services for The Knowledge Lifecycle*, 2002.

[HvHH+05] Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure. A framework for handling inconsistency in changing ontologies. In *Proceedings of the Fourth International Semantic Web Conference (ISWC2005)*, NOV 2005.

[Kem87] C. F. Kemerer. An Empirical Validation of Software Cost Estimation Models. *Communications of the ACM*, 30(5), 1987.

[KLS95] J. Krogstie, O. I. Lindland, and G. Sindre. Defining Quality Aspects for Conceptual Models. In *Proceedings of the IFIP8.1 working conference on Information Systems Concepts ISCO03: Towards a Consolidation of Views*, 1995.

[KVA04] K. Kotis, G. A. Vouros, and J. Padilla Alonso. HCOME: tool-supported methodology for collaboratively devising living ontologies. In *SWDB'04: 2. Int. Workshop on Semantic Web and Databases*, 2004.

[LET04] Steffen Lamparter, Marc Ehrig, and Christoph Tempich. Knowledge extraction from classification schemata. In *Proc. of the Int. Conf. on Ontologies, Databases and Applications of SEmantics (ODBASE)*. Springer, 2004.

[LTGP04] A. Lozano-Tello and A. Gomez-Perez. ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, 15(2), 2004.

[MMS03] A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies on the semantic web. *The VLDB Journal*, 12(4):286–302, Nov 2003.

[MSBS03] D. L. Moody, G. Sindre, T. Brasethvik, and A. Solvberg. Evaluating the quality of information models: empirical testing of a conceptual model quality framework. In *Proceedings of the 25th International Conference on Software Engineering ICSE03*, 2003.

[NFM00] N. Noy, R. Fergerson, and M. Musen. 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.

[PBM05a] E. Paslaru Bontas and M. Mochol. A cost model for ontology engineering. Technical Report TR-B-05-03, Free University of Berlin, April 2005.

[PBM05b] E. Paslaru Bontas and M. Mochol. Towards a Cost Estimation Model for Ontology Engineering. In *Proceedings of the Berliner XML Days Conference*, 2005.

[PBMT05] E. Paslaru Bontas, M. Mochol, and R. Tolksdorf. Case Studies in Ontology Reuse. In *Proceedings of the 5th International Conference on Knowledge Management IKNOW05*, 2005.

[PM00] H. S. Pinto and J. Martins. Reusing ontologies. In *AAAI 2000 Spring Symposium on Bringing Knowledge to Business Processes*, pages 77–84, 2000.

[PS04] R. Price and G. Shanks. A Semiotic Information Quality Framework. In *Proceedings of the International Conference on Decision Support Systems DSS04*, 2004.

[PSST04] 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 *1st European Semantic Web Symposium, ESWS 2004*. Springer, May 2004.

[PTS04] H. Sofia Pinto, Christoph Tempich, and Steffen Staab. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engingeering of ontologies. In Ramon Lopez de Mantaras and Lorenza Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), August 22nd - 27th*, pages 393–397, Valencia, Spain, AUG 2004. IOS Press.

[RVMS99] T. Russ, A. Valente, R. MacGregor, and W. Swartout. Practical Experiences in Trading Off Ontology Usability and Reusability. In *Proceedings of the Knowledge Acquisition Workshop KAW99*, 1999.

[SAB+03] Y. Sure, H. Akkermans, J. Broekstra, J. Davies, Y. Ding, A. Duke, R. Engels, D. Fensel, I. Horrocks, V. Iosif, A. Kampman, A. Kiryakov, M. Klein, T. Lau, D. Ognyanov, U. Reimer, K. Simov, R. Studer, J. van der Meer, and F. van Harmelen. *On-To-Knowledge: Semantic Web–Enabled Knowledge Management*, chapter 13, pages 278–301. Springer-Verlag", 2003.

[SEA+02] York Sure, Michael Erdmann, Juergen Angele, Steffen Staab, Rudi Studer, and Dirk Wenke. OntoEdit: Collaborative Ontology Development for the Semantic Web. In *Proc. of the 1st ISWC*, 2002.

[SPKR96] B. Swartout, R. Patil, K. Knight, and T. Russ. Toward distributed use of large-scale ontologies. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW'96)*, Banff, Canada, November 1996.

[SS02] Y. Sure and R. Studer. On-To-Knowledge methodology. In *On-To-Knowledge: Semantic Web enabled Knowledge Management*. J. Wiley and Sons, 2002.

[SSSS01] S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1), 2001.

[Ste95] Stewart, R. D. and Wyskida, R. M. and Johannes, J. D. *Cost Estimator's Reference Manual*. Wiley, 1995.

[STV04] Y. Sure, C. Tempich, and Z. Vrandečić. D7.1.1. SEKT methodology: Survey and initial framework. SEKT deliverable 7.1.1, Institute AIFB, University of Karlsruhe, 2004.

[Sur03] Y. Sure. *Methodology, Tools and Case Studies for Ontology based Knowledge Management*. PhD thesis, University of Karlsruhe, 2003.

[TPSS05] 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*. Springer, 2005.

[UCH+98] M. Uschold, P. Clark, M. Healy, K. Williamson, and S. Woods. An Experiment in Ontology Reuse. In *Proceedings of the 11th Knowledge Acquisition Workshop KAW98*, 1998.

[UHW+98] M. Uschold, M. Healy, K. Williamson, P. Clark, and S. Woods. Ontology Reuse and Application. In *Proceedings of the International Conference on Formal Ontology and Information Systems FOIS98*, pages 179–192, 1998.