Cost estimation for ontology development: applying the ONTOCOM model

Elena Paslaru Bontas Simperl[1], Christoph Tempich[2], and Malgorzata Mochol[3]

[1], [3]Free University of Berlin, Takustr. 9, 14195 Berlin, Germany
[1]paslaru,[3]mochol@inf.fu-berlin.de
[2]Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
[2]tempich@aifb.uni-karlsruhe.de

Abstract. Techniques for reliably estimating development efforts are a fundamental requirement for a wide-scale dissemination of ontologies in real-world application settings. The parametric cost estimation model ONTOCOM is a first attempt to cope with the lack of instruments for business measurements in the field of ontology engineering. This paper is intended as a user guide to ONTOCOM. It describes how ontology engineers can customize the general-purpose model to particular process models and subsequently apply it in various development phases to calculate the estimated efforts.

1 Introduction

A core requirement for the take-up of ontology-driven technologies at industry level is the availability of proved and tested methods which allow an efficient engineering of high-quality ontologies, be that by reuse, manual 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 (**Onto**logy **Co**st **M**odel) approach [10], which provides an instrument to estimate the efforts involved in building, reusing and maintaining ontologies. Just as 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.

ONTOCOM is based on a work breakdown structure which complies to the process model recommended by all established ontology engineering methodologies.[1] It differentiates between four main process stages: 1). **requirements analysis**; 2). **conceptualization**; 3). **implementation**; and 4). **evaluation**. For these categories, cost drivers influencing the required effort (in terms of person months) have been identified on the

[1] *cf.* [3] or [13] for recent surveys on ontology engineering methodologies.

basis of a comprehensive analysis of current engineering methodologies and related case studies. Their relevance to the cost estimation issue has been confirmed by experts in the field in a comprehensive evaluation study [10]. Every cost driver is associated with effort multipliers (from *Very High* to *Very Low*), depending on the characteristics of the corresponding project setting. A first estimation of the numerical values associated with these effort multipliers was performed based on ex post analysis of different ontology engineering efforts and preliminary expert validations with very promising results [10].

This paper explains how ONTOCOM can be applied to estimate the efforts related to ontology development in arbitrary projects. For this purpose we illustrate the usage of the general-purpose model with the help of a simple example, and then describe how this model can be further refined and customized for project settings which follow a different ontology engineering methodology. The last issue is exemplified in relation to the DILIGENTmethodology, which is targeted at the construction of rapidly changing ontologies in distributed settings [14].

The remaining of this paper is organized as follows. After introducing the ONTOCOM model in Section 2, we demonstrate how it can be applied to estimate the costs of ontology development in Section 3. Section 4 describes the steps required to adapt the generic model to new ontology engineering methodologies. This process is exemplified for the DILIGENT methodology in Section 5. Section 6 provides a brief overview of previous work which is related to ONTOCOM, while Section 7 summarizes the main contributions of this paper and outlines directions of future research and development.

2 The generic ONTOCOM model

In this section we introduce the generic ONTOCOM cost estimation model. The model is generic in that it assumes a sequential ontology life cycle, according to which an ontology is conceptualized, implemented and evaluated, after an initial analysis of the requirements it should fulfill (see below). By contrast ONTOCOM does not consider alternative engineering strategies such as rapid prototyping or agile methods, which are based on different life cycles.[2] This limitation is issued in Section 4, which describes how the generic model should be customized to suit such scenarios.

The cost estimation model is realized in three steps. First a *top-down* work breakdown structure for ontology engineering processes is defined in order to reduce the complexity of project budgetary planning and controlling operations down to more manageable units [1]. The associated costs are then elaborated using the *parametric* method. The result of the second step is a statistical prediction model (i.e. a parameterized mathematical formula). Its parameters are given start values in pre-defined intervals, but need to be calibrated on the basis of previous project data. This empirical information complemented by expert estimations is used to evaluate and revise the predictions of the initial *a-priori model*, thus creating a validated *a-posteriori model*.[3]

[2] *cf.* [3] for a discussion on the relation between this process model and the IEEE standards [5].
[3] *cf.* [1] for an overview of cost prediction methods for engineering projects.

2.1 The work breakdown structure

The top-level partitioning of a generic ontology engineering process can be realized by taking into account available process-driven methodologies in this field [3, 13] According to them ontology building consists of the following core steps (cf. Figure 1):
1) Requirements Analysis. The engineering team consisting of domain experts and ontology engineers performs a deep analysis of the project setting w.r.t. a set of predefined requirements. This step might also include **knowledge acquisition** activities in terms of the re-usage of existing ontological sources or by extracting domain information from text corpora, databases etc. If such techniques are being used to aid the engineering process, the resulting ontologies are to be subsequently customized to the application setting in the conceptualization/implementation phases. The result of this step is an ontology requirements specification document [12]. In particular this contains a set of competency questions describing the domain to be modeled by the prospected ontology, as well as information about its use cases, the expected size, the information sources used, the process participants and the engineering methodology.
2) Conceptualization. The application domain is modeled in terms of ontological primitives, e. g. concepts, relations, axioms.
3) Implementation. The conceptual model is implemented in a (formal) representation language, whose expressivity is appropriate for the richness of the conceptualization. If required reused ontologies and those generated from other information sources are translated to the target representation language and integrated to the final context.
4) Evaluation. The ontology is evaluated against the set of competency questions. The evaluation may be performed automatically, if the competency questions are represented formally, or semi-automatically, using specific heuristics or human judgement. The result of the evaluation is reflected in a set of modifications/refinements at the requirements, conceptualization or implementation level.

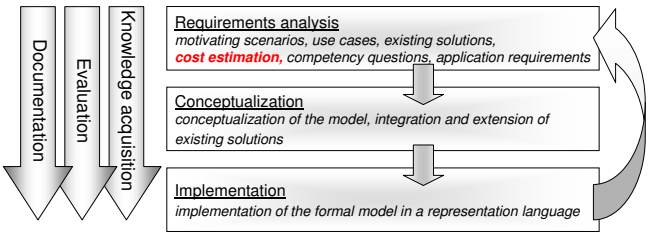

Fig. 1. Typical Ontology Engineering Process

Depending on the ontology life cycle underlying the process-driven methodology, the aforementioned four steps are to be seen as a sequential workflow or as parallel activities. Methontology [4], which applies prototypical engineering principles, considers **knowledge acquisition**, **evaluation** and **documentation** as being complementary *support activities* performed in parallel to the main development process. Other methodologies, usually following a classical waterfall model, consider these support activities

as part of a sequential engineering process. The OTK-Methodology [12] additionally introduces an initial **feasibility study** in order to assess the risks associated with an ontology building attempt. Other optional steps are **ontology population/instantiation** and **ontology evolution/maintenance**. The former deals with the alignment of concrete application data to the implemented ontology. The latter relates to modifications of the ontology performed according to new user requirements, updates of the reused sources or changes in the modeled domain. Further on, likewise related engineering disciplines, reusing existing knowledge sources—in particular ontologies—is a central topic of ontology development. In terms of the process model introduced above, **ontology reuse** is considered a **knowledge acquisition** task.

The parametric method integrates the efforts associated with each component of this work breakdown structure to a mathematical formula as described below.

2.2 The parametric equation

ONTOCOM calculates the necessary person-months effort using the following equation:

$$PM = A * Size^{\alpha} * \prod CD_i \tag{1}$$

According to the parametric method the total development efforts are associated with cost drivers specific for the ontology engineering process and its main activities. Experiences in related engineering areas [1, 6] let us assume that the most significant factor is the *size of the ontology* (in kilo entities) involved in the corresponding process or process phase. In Equation 1 the parameter $Size$ corresponds to the size of the ontology i.e. the number of primitives which are expected to result from the conceptualization phase (including fragments built by reuse or other knowledge acquisition methods). The possibility of a non-linear behavior of the model w.r.t. the size of the ontology is covered by parameter α. The constant A represents a baseline multiplicative calibration constant in person months, i.e. costs which occur "if everything is normal". The *cost drivers* CD_i 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_i). The *productivity range* PR_i of a cost driver (i.e. the ratio between the highest and the lowest effort multiplier of a cost driver $PR_i = \frac{max(EM_i)}{min(EM_i)}$) is an indicator for the relative importance of a cost driver for the effort estimation [1].

2.3 The ONTOCOM cost drivers

The ONTOCOM cost drivers, which are proved to have a direct impact on the total development efforts, can be roughly divided into three categories:
1) PRODUCT-RELATED COST DRIVERS account for the impact of the characteristics of the product to be engineered (i.e. the ontology) on the overall costs. The following cost drivers were identified for the task of ontology building:
• **Domain Analysis Complexity (DCPLX)** to account for those features of the application setting which influence the complexity of the engineering outcomes,

• **Conceptualization Complexity (CCPLX)** to account for the impact of a complex conceptual model on the overall costs,
• **Implementation Complexity (ICPLX)** to take into consideration the additional efforts arisen from the usage of a specific implementation language,
• **Instantiation Complexity (DATA)** to capture the effects that the instance data requirements have on the overall process,
• **Required Reusability (REUSE)** to capture the additional effort associated with the development of a reusable ontology,
• **Evaluation Complexity (OE)** to account for the additional efforts eventually invested in generating test cases and evaluating test results, and
• **Documentation Needs (DOCU)** to state for the additional costs caused by high documentation requirements.

2) PERSONNEL-RELATED COST DRIVERS emphasize the role of team experience, ability and continuity w.r.t. the effort invested in the engineering process:
• **Ontologist/Domain Expert Capability (OCAP/DECAP)** to account for the perceived ability and efficiency of the single actors involved in the process (ontologist and domain expert) as well as their teamwork capabilities,
• **Ontologist/Domain Expert Experience (OEXP/DEEXP)** to measure the level of experience of the engineering team w.r.t. performing ontology engineering activities,
• **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,
• **Personnel Continuity (PCON)** to mirror the frequency of the personnel changes in the team.

3) PROJECT-RELATED COST DRIVERS relate to overall characteristics of an ontology engineering process and their impact on the total costs:
• **Support tools for Ontology Engineering (TOOL)** to measure the effects of using ontology management tools in the engineering process, and
• **Multisite Development (SITE)** to mirror the usage of the communication support tools in a location-distributed team.

The ONTOCOM cost drivers were defined after extensively surveying recent ontology engineering literature and conducting expert interviews, and from empirical findings of numerous case studies in the field.[4] 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 rating of the driver in a particular situation. For example for the cost driver CCPLX—accounting for costs produced by a particularly complex conceptualization—we pre-defined the meaning of the rating levels as depicted in Table 1. The appropriate rating should be selected during the cost estimation procedure and used as a multiplier in equation 1. The concrete values of the effort multipliers have been determined during the calibration of the model, which is described in [10].

The decision criteria associated with a cost driver are typically more complex than in the previous example and might be sub-divided into further sub-categories, whose

[4] See [9] for a detailed explanation of the approach.

Rating Level	Effort multiplier	Description
Very Low	0.28	concept list
Low	0.64	taxonomy, high nr. of patterns, no constraints
Nominal	1.0	properties, general patterns available, some constraints
High	1.36	axioms, few modeling patterns, considerable nr. of constraints
Very High	1.72	instances, no patterns, considerable nr. of constraints

Table 1. The Conceptualization Complexity Cost Driver **CCPLX**

impact is aggregated to the final effort multiplier of the corresponding cost driver by means of normalized weights.[5]

3 Using the generic model

Starting from a typical ontology building scenario, in which a domain ontology is created from scratch by the engineering team, we simulate the cost estimation process according to the parametric method underlying ONTOCOM. Given the top-down nature of our approach this estimation can be realized in the early phases of a project, in particular after the requirements analysis has been accomplished and an initial prediction of the size of the target ontology is available. The first step of the cost estimation is the specification of the size of the ontology to be build, expressed in thousands of ontological primitives (concepts, relations, axioms and instances): if we consider an ontology with 1000 concepts, 200 relations (including `is-a`) and 100 axioms, the size parameter of the estimation formula will be calculated as follows:

$$Size = \frac{1000 + 200 + 100}{1000} = 1,3 \tag{2}$$

The next step is the specification of the cost driver ratings corresponding to the information available at this point (i.e. without reuse and maintenance factors, since the ontology is built manually from scratch). Depending on their impact on the overall development effort, if a particular activity increases the nominal efforts, then it would be rated with values such as *High* and *Very High*. Otherwise, if it causes a decrease of the nominal costs, then it would be rated with values such as *Low* and *Very Low*. Cost drivers which are not relevant for a particular scenario should be rated with the nominal value 1, which does not influence the result of the prediction equation.

Assuming that the ratings of the cost drivers are those depicted in Table 2 these ratings are replaced by numerical values. The value of the DCPLX cost driver was computed as an equally weighted, averaged sum of a high-valued rating for the domain complexity, a nominal rating for the requirements complexity and a high effort multiplier for the information sources complexity (for details of other rating values see [10]):

[5] Refer to `http://ontocom.ag-nbi.de` for a complete list of the cost drivers and the associated effort multipliers.

According to the formula 1 ($\alpha = 1$) the estimated effort in person months would be amount to 5,6 PMs and be calculated as follows (the A parameter was set to 2.92 as described in Section 2):

$$PM = 2,92 * 1,3^1 * (1,26 * 1^{10} * 1,15 * 1,11 * 0,93 * 1,11 * 0,89) \quad (3)$$

Cost driver	Effort	Value	Cost driver	Effort	Value
Product factors			**Personnel factors**		
DCPLX	High	1,26	OCAP	High	1,11
CCPLX	Nominal	1	DCAP	Low	0,93
ICPLX	Low	1,15	OEXP	High	1,11
DATA	High	1	DEEXP	Very Low	0,89
REUSE	Nominal	1	LEXP	Nominal	1
DOCU	Low	1	TEXP	Nominal	1
OE	Nominal	1	PCON	Very High	1
Project factors					
TOOL	Very Low	1	SITE	Nominal	1

Table 2. Values of the cost drivers

4 Adapting the generic model to other methodologies

ONTOCOM is intended to be applied in early stages of an ontology engineering process. In accordance to the process model introduced above the prediction of the arising costs can be performed during the feasibility study or, more reliably, during the requirements analysis. Many of the input parameters required to exercise the cost estimation are expected to be accurately approximated during this phase: the expected size of the ontology, the engineering team, the tools to be used, the implementation language etc.[6]

The high-level work breakdown structure foreseen by ONTOCOM can be further refined depending of the ontology development strategy applied in an organization or in a certain application scenario. As explained in Section 2.1 the generic ONTOCOM model assumes a sequential ontology life cycle which contains only the most important management, development and support activities (cf. [3]). In case the model is applied to a different setting, the relevant cost drivers are to be aligned (or even re-defined) to the new sub-phases and activities, while the parametric equation needs to be adapted to the new activity breakdown. A detailed example of how ONTOCOM can be applied to an ontology development methodology targeted at rapid prototyping in distributed scenarios is provided in the next section.

In order to create the cost formula for a specific ontology engineering methodology we propose the following process:

[6] Note that ontology engineering methodologies foresee this information to be collected in an ontology requirements document at the end of the requirements analysis phase [12].

1. Select ontology engineering methodology for which the cost estimation formula should be created
2. Identify the process stages within the methodology. The estimated effort for the entire life cycle of the ontology development (PM) is then the sum of the estimated efforts for each process stage (PM_i)

$$PM = \sum_{i=0}^{n} PM_i \tag{4}$$

   where n is the number of process stages.
   The definition of process stages might differ between methodologies. In the case of ONTOCOM a process stage should have a defined starting and end point. A process stage should result in a formalized ontology.[7]
3. For each process stage identify the relevant ontology engineering activities. Map them to the corresponding cost drivers defined in ONTOCOM. If this mapping can not be directly established, check whether the activity may be generalized or refined and map the activity to the corresponding cost drivers. If the latter is not possible, the quality of the predictions is likely to decrease.
4. In order to derive the cost formula for a specific process stage remove the non-relevant cost drivers from the equation 1. The size parameter is determined by the size of the ontology built in the specific process stage.
5. Calibrate the adapted model. The generic model has been calibrated on the basis of historical project data. However, the obtained values might not be significant for the particular setting for which ONTOCOM has been adapted. Therefore, it might be of benefit to rerun this task on a relevant set of data points, collected from projects which followed the same engineering strategy.

After this optional customization step the model can be utilized for cost predictions. For this purpose the engineering team needs to specify the rating levels associated with each cost driver, as described in the previous example in Section 3. The values correspond to parameters in equation 1. The results obtained for individual stages are summed up in equation 4.

We now turn to an example of adapting ONTOCOM to rapid prototyping methodology.

5 Adapting the generic model to DILIGENT

5.1 Step 1: Select ontology engineering methodology

DILIGENT stands for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies [14]. It addresses the requirements for ontology engineering in distributed knowledge management settings. For that it distinguishes two kinds of ontologies: **shared ontology** and **local ontologies**. A shared ontology is available to all users but they cannot change it directly. In order to change the shared ontology the participants obtain a copy of it and modify it in locally, obtaining a new local ontology.

[7] Note that according to this definition the generic cost model presented in Section 2 assumes an ontology engineering project consisting of a single development phase, at the end of which the prospected ontology is released.

5.2 Step 2: Identify process stages

A DILIGENT process comprises five main stages: (1) **build**, (2) **local adaptation**, (3) **central analysis**, (4) **central revision**, (5) **local update**. Due to space restrictions we do not provide a complete description of the methodology here but sketch the overall process and focus on a particular process stage only: *viz.* local adaptation. For a more detailed description we refer the interested reader to [14]. The detailed description gives a complete account of the different roles involved in the process, input and output factors, related activities and major decisions to be made by the actors.

Central Build The process starts by having *domain experts*, *ontology users*, and *ontology engineers* **build** an initial shared ontology.

Local Adaptation Users work with the current version of the shared ontology, adapting it to their local needs. They submit change requests to the central board in order to align their local revisions to the remotely built ontologies.

Central Analysis In order to update the shared ontology in accordance to the new user requirements a board **analyzes** the local ontologies and the associated change requests. The board tries to identify similarities in local ontologies so as to decide which changes are going to be introduced in the next version of the shared ontology.

Central Revision Once the board decides upon the changes to be made to the shared ontology, it revises and distributes it to the user community.

Local Update If a new version of the shared ontology is available, its users may **update** their own **local** ontologies accordingly. Updating may involve a re-organization of the local ontology, as only a part of the requested changes have been accepted by the engineering board. Nevertheless, using an updated version of the ontology is of benefit, as it increases the interoperability between local users.

General cost formula for DILIGENT DILIGENT is a iterative ontology engineering methodology. The general cost formula for DILIGENT can thus be defined as:

$$PM = PM_{CB} + \sum_{j=1}^{n}(PM_{LA_j} * m_j + PM_{CA_j} + PM_{CR_j} + PM_{LU_i} * m_j) \quad (5)$$

where PM_{CB}, PM_{LA_j}, PM_{CA_j}, PM_{CR_j} and PM_{LU_j} are the person months necessary for each process stage and j iterates over the total number of cycles n. The number of sites participating at the process in every cycle is captured by the variable m_j.

5.3 Step 3: Identify activities and define mappings

As aforementioned, we exemplify this step for a single process stage of the DILIGENT methodology, namely local adaptation. The remaining process stages can be handled analogously.

In the local adaptation stage the users perform different activities in order to obtained the desired output: *local analysis of shared ontology*, *local specification of new requirements* , *ontology use*, *local customization of local ontology*, *local integration of reused ontologies to the local ontology*, *local modification of the local ontology*, *argument provision*, *evaluation of new local ontology* [14]. The activities are repeated in the given order until a new shared ontology is available.

Local analysis of shared ontology In this activity the users get familiar with the shared ontology. The users learn where the different concepts are located in the ontology and how they are interrelated with other concepts.

Ontology use The ontology is used in the local environment.

Specification of new requirements The local usage of the shared ontology leads to the specification of new requirements, if it does not completely represent the knowledge required by the users.

Local customization of local ontology Building an ontology is a combination of the two approaches: building from scratch and building by reuse. The local customization activity defines therefore the two sub-activities *Local modification of local ontology* and *Local integration of reused ontologies to the local ontology*.

Local modification of local ontology The local modification of the shared ontology is one option to adapt it to the local requirements.

Local integration of reused ontologies to the local ontology The second possibility to meet new local requirements of the users is to reuse external ontologies.

Argument provision The users externalize the reasons for their modeling decisions using arguments.

Evaluation of new local ontology The user evaluates his local ontology w.r.t. his local requirements. He does not evaluate the entire ontology, but only the parts relevant to him.

Documentation The ontology user documents the changes introduced into the shared ontology.

Table 3 shows the cost drivers which are relevant to each of the listed activities.

DILIGENT process		**Cost factor**
DILIGENT phase	DILIGENT activity	Product; Personal; Project
Local adaptation	Local analysis of shared ontology	DCPLX, OE; DECAP, DEEXP, LEXP, TEXP, PCON; TOOL
	Ontology use	DATA; DECAP, DEEXP, LEXP, TEXP; TOOL
	Specification of new requirements	DCPLX; DECAP, DEEXP, LEXP, TEXP, PCON; TOOL
	Local customization of local ontology • Local modification of local ontology • Local integration of reused ontologies to the local ontology	CCPLX,ICPLX; DEEXP, DECAP, TEXP, LEXP, PCON ; TOOL
	Argument provision	
	Evaluation of new local ontology	OE; DEEXP, DECAP, TEXP, LEXP, PCON; TOOL
	Documentation	DOCU; DEEXP, DECAP, TEXP, LEXP, PCON; TOOL

Table 3. The cost drivers relevant for the local adaptation phase

5.4 Step 4: Derive the cost formula

Just as in the previous section we restrict the example to the local adaptation phase. The formula is derived from the generic ONTOCOM formula 1 adapted to comply with the mapping depicted in Table 3. The equation calculates the effort required to evaluate, adapt and use the shared ontology ($Size_{LA}$ is the size of the shared ontology plus the average number of changes introduced by the users).

$$PM_{LA} = A * Size_{LA}^{\alpha} * DCPLX * CCPLX * ICPLX * DATA * OE * DOCU * DECAP * DEEXP * LEXP * TEXP * PCON * TOOL \quad (6)$$

5.5 Step 5: Calibration

The start values of the cost drivers may be obtained from the general ONTOCOM model. However, in order to to improve the quality of the predictions, the adapted cost model should be calibrated. In [10] we present a statistically sound method to combine the estimations calculated with the generic model with newly obtained project data in order to improve the results. A detailed description of the calibration process is out of the scope of this paper.

6 Related Work

Cost estimation methods have a long-standing tradition in more mature engineering disciplines such as software engineering or industrial production [1, 6, 11]. Although the importance of cost issues is well-acknowledged in the community, as to the best knowledge of the authors, no cost estimation model for ontology engineering has been published so far. Analogue models for the development of knowledge-based systems (e.g., [2]) implicitly assume the availability of the underlying conceptual structures. [8] provides a qualitative analysis of the costs and benefits of ontology usage in application systems, but does not offer any model to estimate the efforts. [7] adjusts the cost drivers defined in a cost estimation model for Web applications w.r.t. the usage of ontologies. The cost drivers, however, are not adapted to the requirements of ontology engineering and no evaluation is provided. We present an evaluated cost estimation model, introducing cost drivers with a proved relevance for ontology engineering, which can be applied in the early stages of an ontology development process.

7 Conclusion

Reliable methods for cost estimation are 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. Starting from existing cost estimation methodologies applied across various engineering disciplines, we propose a parametric cost estimation model for ontologies by identifying relevant cost drivers having a direct impact on the effort invested in the main activities of the ontology life cycle. We explain how this

model can be adapted to other ontology engineering methodologies and exemplify this with the help of the rapid prototyping method DILIGENT. In the future we intend to continue the data collection procedure in order to improve the quality of the generic model and its customizations.

Acknowledgements 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**, the European project "Sekt-Semantically-Enabled Knowledge Technologies"(EU IP IST-2003-506826) and the "Knowledge Nets"' project, which is part of the InterVal - Berlin Research Centre for the Internet Economy, funded by the German Ministry of Research BMBF. Further information about ONTOCOM can be found under: `http://ontocom.ag-nbi.de`.

References

1. B. W. Boehm. *Software Engineering Economics*. Prentice-Hall, 1981.
2. A. Felfernig. Effort estimation for knowledge-based configuration systems. In *Proc. of the 16th Int. Conf. of Software Engineering and Knowledge Engineering SEKE04*, 2004.
3. A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, 2003.
4. A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering – with examples form the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Verlag, 2004.
5. IEEE Computer Society. IEEE Standard for Developing Software Life Cycle Processes. IEEE Std 1074-1995, 1996.
6. C. F. Kemerer. An Empirical Validation of Software Cost Estimation Models. *Communications of the ACM*, 30(5), 1987.
7. M. Korotkiy. On the effect of ontologies on web application development effort. In *Proc. of the Knowledge Engineering and Software Engineering Workshop*, 2005.
8. T. Menzies. Cost benefits of ontologies. *Intelligence*, 10(3):26–32, 1999.
9. E. Paslaru Bontas and C. Tempich. How Much Does It Cost? Applying ONTOCOM to DILIGENT. Technical Report TR-B-05-20, Free University of Berlin, October 2005.
10. E. Paslaru-Bontas Simperl, C. Tempich, and Y. Sure. Ontocom: A cost estimation model for ontology engineering. In *Proceedings of the 5th International Semantic Web Conference ISWC2006*, 2006. to appear.
11. R. D. Stewart, R. M. Wyskida, and J. D. Johannes. *Cost Estimator's Reference Manual*. Wiley, 1995.
12. Y. Sure, S. Staab, and R. Studer. Methodology for development and employment of ontology based knowledge management applications. *SIGMOD Record*, 31(4), 2002.
13. Y. Sure, C. Tempich, and D. Vrandecic. Ontology engineering methodologies. In *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. Wiley, UK, 2006.
14. Christoph Tempich, H. Sofia Pinto, and Steffen Staab. Ontology engineering revisited: an iterative case study with diligent. In *Proceedings of the 3rd European Semantic Web Conference, ESWC 2006*, pages 110–124, 2006.