Ontology Engineering Cost Estimation with ONTOCOM

Elena Paslaru Bontas, Malgorzata Mochol
AG Netzbasierte Informationssysteme
paslaru@inf.fu-berlin.de
mochol@inf.fu-berlin.de

March 24, 2006

Technical Report B 06-01

Abstract

Techniques for reliably estimating development efforts are a fundamental requirement for a wide-scale dissemination of ontologies in business contexts. In this report we account for the similarities and differences between software and ontology engineering in order to establish the appropriateness of applying software cost models to ontologies. We present a parametric approach to cost estimation for ontology development – ONTOCOM – and analyze various cost factors implied in the ontology engineering process.

Executive Summary

No	Item	Author	Date	Description
1	Version 1	Mochol	Januar 2006	Description of the ONTOCOM with modelling aspects and updated cost drivers
2	Version 1.1	Mochol	March 2006	Small changes in the cost factors and example

Contents

1 Introduction

Ontologies are targeted at providing means to formally specify a commonly agreed understanding of a domain of interest in terms of concepts, relationships, and axioms [17]. With applications in fields such as Knowledge Management, Information Retrieval, Natural Language Processing, eCommerce, Information Integration or the emerging Semantic Web, ontologies are part of a new approach to building intelligent information systems [11] : they are intended to provide knowledge engineers with reusable pieces of declarative knowledge, which can be—together with problem-solving methods and reasoning services—easily assembled to high-quality and cost-effective systems [27]. Though ontologies and associated ontology management tools have become increasingly popular in the last decades—in particular in conjunction with the emergence of Semantic Web technologies—the dissemination of ontology-driven technologies in real-world business contexts is inconceivable without fine-grained methodologies which are able to deal with both technical and economical challenges of ontology engineering.

Besides feasible tool support, a major requirement for ontologies to be built and deployed at large scale is the availability of methods which address the business-oriented aspects of ontology development and deployment. A wide range of ontology engineering methodologies have been elaborated in the Semantic Web community ([13, 18, 12, 38, 34]). According to these methodologies, engineering ontologies is defined as an iterative process, which shows major similarities with established models from the neighbored research field of Software Engineering. However existing methodologies, unlike adjacent engineering disciplines, do not cover a crucial aspect of the engineering process, which has gained significant attention in adjacent engineering areas because of its importance in real-world business contexts. Issues like costs estimation (using pre-defined cost models), quality assurance procedures (w.r.t. both end product i.e. ontologies and associated engineering process) or means to monitor the business value or the impact of these technologies in organizational context have yet been marginally exploited by the Ontology Engineering research community. In order to precisely estimate the costs related to the ontology engineering process, there is a need for empirically tested cost models which exploit the results already achieved w.r.t. this issue in related engineering fields. In the same time a cost model for ontologies should take into account the critical factors and particularities of the ontology engineering process. In this report we propose ONTOCOM (ONTOlogy COst Model), a parametric cost model for the estimation of the efforts involved in building, reusing and maintaining ontologies in information systems.

2 Ontology vs. Software Engineering

The overall process of developing a cost model for ontology development shares commonalities with existing approaches in adjacent engineering disciplines. In

particular, the similarities between ontology engineering and software engineering processes, as well as between ontology and software modelling make software cost estimation methods a reasonable starting point for the development of a cost estimation methodology for ontologies. In the following we account for the similarities and differences between these disciplines in order to establish the appropriateness of applying software cost models to the ontology engineering. software engineering and ontology engineering belong to the general area of engineering, a field which is usually defined by the usage of scientific principles and methods to construct artifacts. Software engineering, one of the core disciplines in Computer Science, provides systematic methods to handle the complexity of designing, implementing and maintaining software systems. One of the basic actions of the software engineering is modelling. The process of building a model—a concrete or mental image of an existing thing or a paradigm of an original which can be created—supplies [16]:

- an inventory of fundamental languages and methods to describe and analyze specific classes of problems
- fundamental knowledge of problem solving methods
- methodologies for the building of software artifacts, as well as
- a deep understanding of the nature of these artifacts

Several decades of research and development in software engineering showed that software projects can not be reduced to software implementation: they should equally focus on user requirements, design, testing, documentation, deployment and maintenance. Modelling is applied in many of these activities for various purposes:

- requirements as models of the problem definition,
- architectures as models of the solution,
- test regulations as models of correct functioning of the code, and
- software itself as model of cutout of a real world [15].

Likewise software engineering, ontology engineering includes scientific methods to create, deploy and maintain ontologies (see Table 1). Though ontology engineering and software engineering share the same final goal—the generation of high-quality artifacts—and a similar development process [25], ontology engineering methodologies available so far do not address every phase of the underlying process at the same level of detail and do not provide a fully-fledged tool environment to aid the engineering team in their attempt to create, maintain or use ontologies.

The main difference between the two engineering processes focuses on their outcomes: software vs. ontologies. A software system is a collection of computer programs and associated data and computer equipment that performs a

Models	Software Engineering	Ontology Engineering
Model of the problem	System analysis (requirements specification, reuse)	Domain analysis (requirements specification, knowledge acquisition)
Model of solution	System design (architecture)	Conceptualization (conceptual model)
Model of solution	Implementation (program code)	Implementation (specification of the conceptual model)
Model of tests	Test data generation	Ontology population
Model of correct functioning of the code	System testing (refinement)	Ontology evaluation (refinement)
Model for the evolution of the system	System maintenance	Ontology maintenance

Table 1: Software vs. Ontology Engineering

specific function or purpose. It consists of interrelated software components which exchange data by means of interfaces and produce a certain result or behavior given user-defined inputs. An ontology specifies a commonly agreed conceptualization of a given universe of discourse. It is a collection of ontological primitives, usually concepts or classes connected to each other by relations and constrained by axioms, which may belong to specific sub-ontologies. Consequently the way to measure the size of a software system, usually expressed in lines of code or function/object points, can not be directly applied to ontologies. Due to the fact that the implementation of an ontology is mostly realized using tools (i.e. ontology editors), the main size factor to express the complexity of an ontology is not given by the actual size of an implementation in a specific representation language, but by the number of ontological primitives contained by conceptual model. Further on, modelling different types of ontological primitives is associated to significantly different amounts of efforts: the conceptualization of classes and their classification in a subclass-hierarchy is recognized to imply considerably less efforts than modelling relationships or axioms. In comparison to a software system, an ontology may be used in a twofold manner: it may be embedded in a software system for a specific purpose (e.g. to index domain-specific documents) or it may be used directly by domain experts without any mediator in form of some computer program (e.g. as a commonly agreed vocabulary). Nevertheless an ontology does not show any behavior-it does not produce an output for a given input as for a software system. This aspect has implications in the way software and ontologies are evaluated. While the evaluation of software (in the sense of software testing) is a mature discipline despite the difficulties in achieving a commonly agreed, general purpose software quality model, evaluating ontologies is still an open issue both from a technical and a user-centered point of view. Another important difference

between software engineering and ontology engineering is related to engineering process themselves: the ontology population/instantiation, a particularity of ontology engineering vs. the related software field, is often associated with significant efforts depending on the characteristics of the data to be aligned to the ontology. The project team involved in the construction of an ontology is relatively small compared to the common team size in software development. As a consequence of the importance of the conceptualization phase, the role of the user/domain expert in the ontology construction and his experience in the domain to be modelled are crucial factors for the success of ontology engineering projects. Finally, as for the knowledge of the authors, given the present state of the art of the ontology engineering area, the duration of ontology engineering projects is significantly shorter than the typical duration of software projects. Despite these differences a cost estimation model for ontologies should benefit from the similar properties of the two engineering fields. The efforts associated with the engineering process can still be assumed to depend on the complexity of the resulting artifact: the size of the ontology to be modelled and in its particularities, and in the size and functionality of the software product respectively [5]. A cost model for ontology engineering should take into account the results achieved by the software engineering community and customize and extend them in order to cover the particularities of the ontology development task. Consequently, we now turn to a survey of some of the most relevant methodologies for costs estimation in software engineering for the purpose of assessing their suitability for current ontology engineering processes.

3 ONTOlogy COst Model - ONTOCOM

For the development of a ontology cost model – ONTOCOM – we adopt a combination of three estimation methodologies, which are in our opinion applicable to ontology engineering according to the current state of the art in the field:

Top-Down Method This method can be applied to predict the costs associated with different upper-level stages of the engineering process. Despite the young nature of the ontology engineering discipline, the Semantic Web community has made significant progresses in analyzing the ontology engineering process with its phases, their particularities and associated activities [13]. This is why we expect a top-down method to be appropriate to construct an ontology cost model. For this purpose, one needs techniques to evaluate the costs associated with the stages of the engineering process: the domain analysis, conceptualization, implementation etc.

Delphi Method The expert judgement method seems to be appropriate for our goals since large amount of expert knowledge w.r.t. ontologies is already available in the Semantic Web community, while the costs of the related engineering efforts are not. Experts' opinion on this topic can be used to compliment the results of other estimation methods.

Parametric Method Apart from the lack of costs-related information which should be used to calibrate cost estimation formula for ontologies, the analysis of the main cost drivers affecting the ontology engineering process can be performed on the basis of existing case studies on ontology building, representing an important step toward the elaboration of a predictable cost estimation strategy for ontology engineering processes. The resulting cost model has to be constantly refined and customized when cost information becomes available. Nevertheless the definition of a fixed spectrum of cost factors is important for a controlled collection of existing real-world project data, a task which is fundamental for the subsequent model calibration.

Apart from expert judgement, we start with a top level approach, by identifying upper-level sub-tasks of the ontology engineering process and define the associated costs using a parametric method. We distinguish among three areas:

Ontology Building includes the sub-tasks like: specification, conceptualization, implementation, instantiation and evaluation.

Ontology Maintenance involves costs related to getting familiar and updating the ontology.

Ontology Reuse accounts for the efforts related to the re-usage of existing (source) ontologies for the generation of new (target) ontologies and involves costs related to finding, evaluating and adapting the former ones to the requirements of the latter.

As a consequence, estimating the effort (in person months) related to ontology engineering is reduced to a sum of the costs arising in the building (with or without reuse) and maintaining ontologies:

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

where PM_B , PM_M and PM_R represent the effort associated to building, maintaining and reusing ontologies, respectively.

The partial costs are calculated in ONTOCOM as:

$$PM_x \quad = \quad A * (Size_x)^B * \prod CD_{xi} \tag{2}$$

Each of the three development phases is associated with *specific* cost factors. Experiences in related engineering areas[22, 5] let us assume that the most significant one is the size of the ontology involved in the corresponding process. In the formula above the size parameter $Size_x$ is expressed in thousands of ontological primitives – concept, relations, axioms and instances. For example for an ontology with 1000 concepts and 100 relations $Size$ will have the value 1.1.

$Size_b$ corresponds to the size of the newly built ontology i.e. the number of primitives which are expected to result from the conceptualization phase.

$Size_m$, in case of ontology maintenance, depends on the expected number of modified items.

$Size_r$, for reuse purpose, is the size of the original source 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 possibility of a non-linear behavior of the model w.r.t. the size of the ontology is covered by parameter B. Further on, start-up costs, which are not proportional to the size of a project, are intended to be counterbalanced by the constant A.

The core parts of the ONTOCOM-formula are the cost drivers CD_{xi}, which 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). The average EM assigned to a cost driver is 1.0 (nominal weight). If a rating level causes more development effort, its corresponding EM is above 1.0. If the rating level reduces the effort then the corresponding EM is less than the nominal value. For each cost driver we specified in detail the decision criteria which are relevant when assigning the corresponding effort multipliers. In the a-priori cost model a team of 3 ontology engineering experts assigned start values between 0.1 and 2 to the effort multipliers, depending on the contribution of the corresponding cost driver to the overall development costs. These parameters were derived after surveying recent literature and from empirical findings of various case studies in the ontology engineering field (such as [28, 37, 31, 2, 1, 35, 36, 24, 4, 6, 33, 29, 14, 39]). These values are subject of further calibration on the basis of the statistical analysis of real-world project data. A list of the initial input values for the cost drivers (the so-called "a-priori cost model") is depicted in Appendix A.

4 ONTOCOM: Product Cost Drivers

The product category accounts for the influence of product properties on the overall costs.

4.1 Cost Drivers for Ontology Building

4.1.1 Complexity of the Domain Analysis: DCPLX

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 modelled (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 2, 3, 4.

Rating	Rating Scale
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

Table 2: Domain Complexity

Rating	Rating Scale
Very Low	few, simple requirements
Low	small number of non-conflicting requirements
Nominal	moderate number of requirements, with few conflicts, few usability requirements
High	high number of usability requirements, few conflicting requirements
Very High	very high number of requirements with a high conflicting degree, high number of usability requirements

Table 3: Requirements complexity

Rating	Rating Scale
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 4: Information sources complexity

4.1.2 Complexity of the Conceptualization: CCPLX

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 process.1 Finally the availability of information sources guiding the engineering team during the building process or offering valuable insights in the domain to be modelled 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 modelling patterns on the overall engineering costs. On the other side, the existence of certain naming and modelling constraints might cause cost increases (see Table 5).

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

Table 5: Conceptualization Complexity

4.1.3 Complexity of the Implementation: ICPLX

As mentioned in 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 semantic 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 arisen during this mapping are stated in the driver ICPX (implementation complexity), whose ratings are illustrated in Table 6. For simplification reasons we restricted the range of the ratings to 3 (from low to high).

Rating	Rating Scale
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 6: Complexity of the Implementation

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.

4.1.4 Complexity of the Instantiation: DATA

The population of an ontology and the associated testing operations might be related to considerable costs[7, 8, 10, 20, 32]. The measure attempts to capture the effect instance data requirements have on the overall process. In particular the form of the instance data and the method required for its ontological formalization are significant factors for the costs of the engineering process (Table 7).

On the basis of a survey of ontology population and learning approaches, we assume that the population of an ontology with available instance data with an unambiguous semantics can be performed more cost-effective than the processing of relational tables or XML-structured data. Further on, the extraction of ontology instances from poorly structured sources like natural language documents is assigned the highest value magnitude, due to the complexity of the task itself and of the pre-processing and post-processing activities.

The rating does not take into consideration any costs related to eventual mapping operations which might be required to integrate data from external

Rating	Rating Scale
Very Low	structured data, same repr. language
Low	structured data with formal semantics
Nominal	semi-structured data e.g. databases, XML
High	semi-structured data in natural language, e. g. similar web pages
Very High	unstructured data in natural langauge, free form

Table 7: Instance Ratings DATA

resources. For example, if the data is provided as instances of a second ontology, be that in the same representation language as the one at hand or not, the estimation of the DATA cost driver should account for the efforts implied by defining a mapping between the source and the target ontology as well. In this case, the parameter is to be multiplied with an increment M (Mapping), as depicted in Table 8 below.

M Increment for DATA	**Ontology Mapping**
0.0	no mapping necessary
0.2	direct mapping
0.4	concept mapping
0.6	taxonomy mapping
0.8	relation mapping
1.0	axiom mapping

Table 8: M Increment for DATA

The M factor increments the effect of the DATA measure: an 1.0 M increment causes a 100% increase of the DATA measure while an 0.0 one does not have any influence on the final value of DATA.

4.1.5 Required Reusability: REUSE

The measure attempts to capture the effort associated with the development of a reusable ontology. Reusability is a major issue in the ontology engineering community, due to the inherent nature of ontologies, as artifacts for knowledge sharing and reuse. Currently there is no commonly agreed understanding of the criteria required by an ontology in order to increase its reusability. Usually reusability is mentioned in the context of application-independency, in that it is assumed that application-dependent ontologies are likely to imply significant customization costs if reused. Additionally several types of ontologies are often presumed to endue an increased reusability: core ontologies and upper-level ontologies describing general aspects of the world are often used in alignment tasks in order to ensure high-level ontological correctness. The Formal Ontological Analysis of Guarino[19] also mentions 3 levels of generality, which might be

associated with different reusability degrees: upper-level ontologies are used as ontological commitment for general purpose domain and task ontologies, while the latter two are combined to realize so-called application ontologies, which are used for particular tasks in information systems. According to these considerations the rating for the REUSE measure is depicted in Table 9.

Rating	Rating Scale
Very Low	for this application
Low	for this application type
Nominal	application independent domain ontology
High	core ontology
Very High	upper level ontology

Table 9: Required Reusability

4.1.6 Documentation Needs: DOCU

The DOCU measure is intended to state the additional costs caused by detailed documentation requirements. Likewise COCOMOII [3] we differentiate among 5 values from very low (many lifecycle needs uncovered) to very high (very excessive for lifecycle needs) as illustrated in Table 10.

Rating	Rating Scale
Very Low	many lifecycle needs uncovered
Low	some lifecycle needs uncovered
Nominal	right-sized to lifecycle needs uncovered
High	excessive for lifecycle needs
Very High	very excessive for lifecycle needs

Table 10: Ratings for Documentation Needs

4.1.7 Complexity of the Ontology Integration: OI

This cost drivers 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 (see below) . 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.

Rating	Rating Scale
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 high overlapping, more than 2 ontologies and recall required,
Very High	n-m mappings, approx. 95% precision and recall required, high overlapping, more than 2 ontologies

Table 11: Ratings for Complexity of the Ontology Integration

4.1.8 Complexity of the Ontology Evaluation: OE

The cost drivers captures the effort invested in evaluating ontologies, be that testing, reviewing, usability or ontological evaluation. 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	Rating Scale
Very Low	many lifecycle needs uncovered
Low	some lifecycle needs uncovered
Nominal	right-sized to lifecycle needs uncovered
High	excessive for lifecycle needs
Very High	very excessive for lifecycle needs

Table 12: Ratings for Complexity of the Ontology Evaluation

4.2 Cost Drivers for Reuse and Maintenance

Though there is yet no fine-grained methodology to reuse existing ontologies in the Semantic Web community, the main steps and the associated challenges involved in the process are well-accepted by current ontology-based projects. This process is, however, related to significant costs and efforts, which may currently outweigh its benefits. First, as in other engineering disciplines, reusing some existing component implies costs to find, get familiar with, adapt and update the necessary modules in a new context. Second building a new ontology means partially translating between different representation schemes or performing scheme matching or both. For our cost estimation model we assume that relevant ontologies are available to the engineering team and, according to the mentioned top-level approach and to some case studies in ontology reuse we examine the following two phases of the reuse process w.r.t. the corresponding cost drivers:

- ontology evaluation: get familiar with the ontology and assess its relevance for the target ontology
- ontology customization: translate the sources to a desired format, eventually extract relevant sub-ontologies and finally integrate them to the target ontology

For the evaluation phase the engineering team is supposed to assess the relevance of a given ontology to particular application requirements. The success of the evaluation depends crucially on the extent to which the ontology is familiar to the assessment team. The customization phase implies the identification/extraction of sub-ontologies which are to be integrated in a direct, translated and modified form, respectively. In the first categories sub-ontologies are included directly to the target ontology. The re-usage of the second category is conditioned by the availability and the appropriate costs of knowledge representation translators, while the last category involves modifications of the original model in form of insertions, deletions or updates at the ontological primitives level.

4.2.1 Complexity of the Ontology Evaluation: OE

This measure accounts for the real effort needed to evaluate the ontology for reuse purposes (see Table 13). The measure assumes a satisfactory ontology understanding level and is associated solely with the efforts needed in order to decide whether a given ontology satisfies a particular set of requirements and to integrate its description into the overall product description.

4.2.2 Complexity of the Ontology Modifications: OM

This measure reflects the complexity of the modifications required by the reuse process after the evaluation phase has been completed (see Table 14).

Rating	Rating Scale
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 13: Ontology Evaluation Cost Driver

Rating	Rating Scale
Very Low	few, simple modifications
Low	some, simple modifications
Nominal	some, moderate modifications
High	considerable modifications
Very High	excessive modifications

Table 14: Complexity of the Ontology Modifications Cost Driver

4.2.3 Ontology Translation: OT

Translating between knowledge representation languages is an essential part of a reuse process. Depending on the compatibility of the source and target representation languages, as well as on the availability and performance of the translating tools (amount of pre- and post-processing required), we differentiate among 5 values as depicted in Table 15.

Rating	Rating Scale
Very Low	direct
Low	low manual effort
Nominal	some manual effort
High	considerable manual effort
Very High	manual effort

Table 15: Cost Driver for Ontology Translation

4.2.4 Ontology Understandability: OU

Reusing an ontology and the associated efforts depend significantly on the ability of the ontologists and domain experts to understand the ontology, which is influenced by two categories of factors: the complexity of the conceptual model and the self-descriptiveness or the clarity of the conceptual model. Additionally, in case of the ontology engineer the comprehensiveness of an ontology depends on his domain experience, while domain experts are assumed to provide this know-how by definition. Factors contributing to the complexity of the model are the size and expressivity of the ontology and the number of imported models

together with the complexity of the import dependency graph. The clarity of the model is mainly influenced by the human-perceived readability.

Rating	Rating Scale
Very Low	complex dependency graph large domain complex representation language no concept names
Very Low	taxonomic dependency graph large domain complex representation language concept names
Nominal	taxonomic dependency graph middle domain moderate representation language concept names
High	no imports middle domain simple representation language concept names
Very High	no imports small domain simple representation language concept names

Table 16: Cost Drivers for Complexity of the Ontology Understandability

4.2.5 Ontologist/Domain Expert Unfamiliarity: UNFM

The effort related to ontology maintenance decreases significantly in situations where the human user works frequently with the particular ontology. This measure accounts for this dependency and distinguishes among 6 levels as depicted in Table 18.

The UNFM factor increments the effect of the Ontology Understanding measure: an 1.0 UNFM increment causes a 100% increase of the OU measure while an 0.0 one does not have any influence on the final value of OU (see Table 18).

5 ONTOCOM: Personnel Cost Drivers

The personnel cost drivers emphasizes the role of team experience, ability and continuity for the effort invested in the process.

Rating	Rating Scale
Very Low	representation language know-how no comments in naturale language no metadata
Low	representation language know-how no comments in naturale language no metadata
Nominal	representation language tool 30% comments in naturale language no metadata
High	representation language tool 60% comments in naturale language no metadata
Very High	representation language tool 90% comments in naturale language metadata

Table 17: Cost Drivers for Clarity of the Ontology Understandability

Rating	Rating Scale
0.0	self built
0.2	team built
0.4	every day usage
0.6	occasional usage
0.8	little experience
1.0	completely unfamiliar

Table 18: Ratings for Ontologist/Domain Expert Unfamiliarity

5.1 Cost Drivers for Building, Reuse and Maintenance

The personnel cost drivers, in contrast to the product factors, do not differentiate between the three sub-tasks of ontology engineering: building, reuse and maintenance.

5.1.1 Ontologist/Domain Expert Capability: OCAP/DECAP

The development of an ontology requires the collaboration between a team of ontology engineers (ontologists), usually with an advanced technical background, and a team of domain experts that provide the necessary know-how in the field to be ontologically modeled. These cost drivers account the perceived ability and efficiency of the single actors involved in the process, as well as their teamwork capabilities.

Rating	Rating Scale
Very Low	15%
Low	35%
Nominal	55%
High	75%
Very High	95%

Table 19: Capability of the Engineering Team Cost Drivers

5.1.2 Ontologist/Domain Expert Experience: OEXP/DEEXP

These measures take into account the experience of the engineering team consisting of both ontologists and domain experts w.r.t. the ontology engineering process. They are not related to the abilities of single team members, but relate directly to the experience in constructing ontologies and in conceptualizing a specific domain respectively.

	Very Low	**Low**	**Nominal**	**High**	**Very High**
OEXP	2 months	6 months	1 year	1.5 years	3 years
DEEXP	6 months	1 year	3 years	5 years	7 years

Table 20: Ontologists and Domain Experts Experience Cost Drivers

5.1.3 Personnel Continuity: PCON

As in other engineering disciplines frequent changes in the project team are a major obstacle for the success of an ontology engineering process within given budget and time constraints. Due to the small size of the project teams we adapted the general ratings of the COCOMO model to a maximal team size of 10 (see Table 21).

	Very Low	**Low**	**Nominal**	**High**	**Very High**
PCON	2 months	6 months	1 year	3 years	6 years

Table 21: Personnel Continuity Cost Driver

5.1.4 Language and Tool Experience: LEXP/TEXP

The aim of these cost drivers is to measure the level experience of the project team constructing the ontology w.r.t. the conceptualization language and the ontology management tools respectively. The conceptualization phase requires the usage of knowledge representation languages with appropriate expressivity (such as Description Logics or Prolog), while the concrete implementation is addicted to support tools such as editors, validators and reasoners. The distinction among language and tool experience is justified by the fact that while

ontology languages rely on established knowledge representation languages from the Artificial Intelligence field and are thus possibly familiar to the ontology engineer, the tool experience implies explicitly the previous usage of typical ontology management tools and is not directly conditioned by the know-how of the engineering team in the KR field. The maximal time values for the tool experiences are adapted to the ontology management field and are thus lower than the corresponding language experience ratings (see Table 22).

	Very Low	**Low**	**Nominal**	**High**	**Very High**
LEXP	2 months	6 months	1 year	3 years	6 years
TEXP	2 months	6 months	1 year	1,5 years	3 years

Table 22: Language and Tool Experience Cost Drivers

6 ONTOCOM: Project Cost Drivers

The project category states the dimensions of the engineering process which are relevant for the cost estimation.

6.1 Cost Drivers for Building, Reuse and Maintenance

The project cost drivers as well as personnel cost drivers do not differentiate between the three sub-tasks of ontology engineering: building, reuse and maintenance.

6.1.1 Tool Support: TOOL

We take account of the different levels of tool support for the different phases of an ontology engineering process (domain analysis, conceptualization, implementation, ontology understanding and evaluation, ontology instantiation, ontology modification, ontology translation, ontology integration and documentation) by means of a single general-purpose cost driver and calculate the final value as the average tool support across the entire process. The ratings for tool support are defined at a general level, as shown in Table 23 below.

Rating	Rating Scale
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 23: Tool Support Cost Driver

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.

6.1.2 Multisite Development: SITE

Constructing an ontology requires intensive communication between ontology engineers and domain experts on one hand and between domain experts for consensus achievement purposes on the other hand. This measure involves the assessment of the communication support tools (see Table 24).

Rating	Rating Scale
Very Low	mail
Low	phone, fax
Nominal	email
High	teleconference, occasional meetings
Very High	frequent F2F meetings

Table 24: Multisite Ontology Development Cost Driver

6.1.3 Required Development Schedule: SCED

This cost driver takes into account the particularities of the engineering process given certain schedule constraints. Accelerated schedules (ratings below 100%, see Table 25) tend to produce more efforts in the refinement and evolution steps due to the lack of time required by an elaborated domain analysis and conceptualization. Stretch-out schedules (over 100%) generate more effort in the earlier phases of the process while the evolution and refinement tasks are best case neglectable.

	Very Low	**Low**	**Nominal**	**High**	**Very High**
SCED	75%	85%	100%	130%	160%

Table 25: Required Development Schedule Cost Driver

For example, a high SCED value of 130% (see Table 25) represents a stretch-out of the nominal schedule of 30% and thus more resources in the domain analysis and conceptualization.

7 Evaluation

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 [30, 26, 21, 23]; see [9] 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 [5] 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 evaluation of the a-priori model was performed by conducting interviews with two groups of experts in the area of Ontology Engineering, consisting of 4 and, respectively 8 participants. In each of the two phases of the evaluation, participants were given a one hour overview of the ONTOCOM approach, which was followed by individual interviews according to the aforementioned framework. During the interviews, the participants expressed their concerns w.r.t. a wide range of costs-related issues in ontology engineering projects and discussed about their experiences in building, reusing or deploying ontologies. The first phase of the evaluation resulted in major changes w.r.t. the definition of the cost drivers and the introduction of missing ones. The second phase of the evaluation produced minimal adaptations of the revised model, in particular w.r.t. a more precise scope of the model, and helped us identifying possible directions for further research. The model presented in the previous sections is the result of this first phase. The second phase of the evaluation is being performed, in parallel to the recently started data collection initiative. Empirical data from previous ontology engineering projects is collected from various organizations in the Semantic Web field, contributing to the continuous calibration of the model. Furthermore, the data collection procedure offers us valuable information about the usability of ONTOCOM and its cost drivers w.r.t. a wide range of ontology engineering scenarios (by now we collected data from 27 projects). The analysis of the empirical data collected so far indicates a well-balanced influence of the cost drivers on the model estimations. The reliability of the predictions is planed to be computed on a data set of at least 75 data points.

8 Conclusions and Future Work

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 the ontology engineering area. The model is intended to predict cost arisen during ontology engineering processes by analyzing the costs factors caused by the end product, the process itself and the engineering team. We evaluate the model a-priori and a-posterior.

In the future we will apply ONTOCOM to other ontology engineering methodologies. This can lead to the introduction of new cost drivers or the redefinition of exiting ones, or both. While definitely in an incipient phase, the a-posterior evaluation has already indicated a balanced influence of the cost factors. The collection of data is also a pre-requisite for a more accurate calibration of the model. Nevertheless our present experiences with the model are very promising: the evaluation of the a-priori model demonstrated the applicability of the model for ontology development processes, while the public data collection initiative was received favorably by the community and will continue in the future.

Acknowledgements This work is a result of the cooperation within the Semantic Web PhD-Network Berlin-Brandenburg and has been partially supported by the KnowledgeWeb - Network of Excellence, by the Project A Semantic Web for Pathology" funded by the DFG (German Research Foundation) and by 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.

A Non-calibrated Values of the Cost Drivers in ONTOCOM

The initial input values for the cost drivers in the ONTOCOM model are illustrated in Tables 26, 27 and 28.

Product Cost Drivers					
Building	Rating				
	Very Low	Low	Nominal	High	Very High
DCPLX	0,70	0,85	1	1,30	1,60
CCPLX	0,70	0,85	1	1,30	1,60
ICPLX	0,85		1	1,30	
DATA	0,80	0,90	1	1,30	1,60
REUSE	0,70	0,85	1	1,15	1,30
DOCU	0,70	0,85	1	1,15	1,30
OE	0,80	0,90	1	1,30	1,60
OI	0,80	0,90	1	1,30	1,60
Reuse and Maintenance	Very Low	Low	Nominal	High	Very High
OE	0,70	0,85	1	1,30	1,60
OM	0,80	0,90	1	1,20	1,40
OT	0,70	0,85	1	1,30	1,60
OU	1,80	1,40	1	0,90	0,80
UNFM	for ratings see Tab. 18				

Table 26: Product Cost Drivers and their ratings

Personnel Cost Drivers					
	Rating				
	Very Low	Low	Nominal	High	Very High
OCAP	1,30	1,15	1	0,85	0,70
DECAP	1,30	1,15	1	0,85	0,70
OEXP	1,30	1,15	1	0,85	0,70
DEEXP	1,30	1,15	1	0,85	0,70
PCON	1,30	1,15	1	0,85	0,70
LEXP	1,60	1,30	1	0,90	0,80
TEXP	1,50	1,25	1	0,90	0,80

Table 27: Personnel Cost Drivers and their ratings

Project Cost Drivers					
	Rating				
	Very Low	Low	Nominal	High	Very High
TOOL	1,60	1,30	1	0,90	0,80
SITE	1,30	1,15	1	0,85	0,70
SCED	1,30	1,15	1	0,85	0,70

Table 28: Project Cost Drivers and their ratings

B Using ONTOCOM: An Example

In this section we give a brief example on the usage of the ontology cost model described in this report. 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 domain 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. Ontological primitives are concepts, relations (including is-a), axioms and instances. Note that we do not consider the size of the data set which will be used to populate the ontology, but only the instances which are to be conceptualized manually during the conceptualization phase. For example, if we consider an ontology with 1000 concepts, 200 relations and 100 axioms, the size parameter of the estimation formula will be 1,3.

The next step is the specification of the cost driver ratings, corresponding to the information available at this point. Assuming that the ratings of the cost drivers are those depicted in Table 29 these ratings are replaced by numerical values (as illustrated in Tables 26, 27 and 28 in Appendix A).

The value of the DCPLX cost driver was computed as an equally weighted,

	Cost Driver	Rating (Value)
Product Factors	DCPLX	High (1,20)
	CCPLX	Nominal (1)
	ICPLX	Low (0,85)
	DATA	High (1,30)
	REUSE	Nominal (1)
	DOCU	Low (0,85)
	OE	Low (1)
	OI	Low (1)
Personnel Factors	OCAP	High (0,85)
	DECAP	Low (1,15)
	OEXP	High (0,85)
	DEEXP	Very Low (1,30)
	PCON	Very High (0,70)
	LEXP	Nominal (1)
	TEXP	Nominal (1)
Project Factors	TOOL	Very Low (1,60)
	SITE	Nominal (1)
	SCED	Nominal (1)

Table 29: Values of the Cost Drivers

averaged sum of a high-valued rating for the domain complexity (1,3) , a nominal rating for the requirements complexity (1,0) and a high effort multiplier for the information sources complexity (1,3):

$$DCPLX = \frac{1 * 1,3 + 1 * 1,0 + 1 * 1,3}{3} \quad (3)$$

According to the main formula (Formula 2) the estimated effort in person months would be amount to 1,77 PMs and be calculated as follows:

$$PM = 1 * (1,3)^1 * (1,2 * 1^8 * 0,85^4 * 1,30^2 * 1,15 * 0,7 * 1,60) \quad (4)$$

References

[1] A. Advani, S. Tu, and M. Musen. Domain Modeling with Integrated Ontologies: Principles for Reconciliation and Reuse. Technical Report SMI-97-0681, Stanford Medical Informatics, 1997.

[2] M. Annamalai and L. Sterling. Guidelines for Constructing Reusable Domain Ontologies. In *Proceedings of the OAS*, pages 71–74, 2003.

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

[4] A. Bernaras, I. Laresgoiti, and J. Corera. Building and Reusing Ontologies for Electrical Network Applications. In *Proceedings of the European Conference on Artificial Intelligence ECAI96*, 1996.

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

[6] W. N. Borst. Construction of Engineering Ontologies. Technical report, University of Tweenty, Eschede, 1997.

[7] P. Buitelaar, D. Olejnik, and M. Sintek. A Protege Plug-In for Ontology Extraction from Text Based on Linguisitc Analysis. In *Proceedings of the European Semantic Web Symposium ESWS04*, 2004.

[8] M. Dittenbach, H. Berger, and D. Merll. Improving domain ontologies by mining semantics from text. In *Proceedings of the 1st Asian-Pacific Conference on Conceptual Modelling*, pages 91–100, 2004.

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

[10] D. Faure and Poibeau T. First experiments of using semantic knowledge learned by ASIUM for information extraction task using INTEX. In *Proceedings of the Ontology Learning Workshop at the ECAI00*, 2000.

[11] D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer Verlag, 2001.

[12] M. Fernandez, A. Gomez-Perez, and N. Juristo. Methontology: From ontological art towards ontological engineering. In *Proceedings of the AAAI'97 Spring Symposium on Ontological Engineering*, 1997.

[13] M. Fernndez-Lpez and A. Gmez-Prez. Overview and analysis of methodologies for building ontologies. *Knowledge Engineering Review*, 17(2):129–156, 2002.

[14] M. Fernndez-Lpez, A. Gmez-Prez, A. Pazos-Sierra, and J. Pazos-Sierra. Building a Chemical Ontology Using METHONTOLOGY and the Ontology Design Environment. *IEEE Intelligent Systems and their Applications*, pages 37–46, January/February 1999.

[15] Glinz, M. Informatik IIa Modellierung. Folienskript. http://www.ifi.unizh.ch/req/courses/inf_II, 2004.

[16] Glinz, M. and Rumpe, B. Informatik Forsch. und Entwick. *Informatik Forsch. und Entwick.*, 18:103–104, 2004.

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

[18] M. Grüninger and M. Fox. Methodology for the Design and Evaluation of Ontologies. In *Proceedings of the IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[19] N. Guarino. Formal Ontology and Information Systems. In *Proceedings of the International Conference on Formal Ontology and Information Systems FOIS98*, pages 3–15, 1998.

[20] U. Hahn, M. Romacker, and K. Schnattinger. Automatic knowledge acquisition from medical text. In *Proceedings of the 1996 AMIA Annual Symposium*, pages 383 – 387, 1996.

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

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

[23] 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.

[24] D. B. Lenat and R. V. Guha. *Building large knowledge-based systems*. Addison-Wesley Publising Company, 1990.

[25] A. Maedche and S. Staab. Ontology Engineering beyond the Modeling of Concepts and Relations. In *Proc. of the ECAI'2000 Workshop on Applications of Ontologies and Problem-Solving Methods*, 2000.

[26] 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.

[27] R. Neches, R. E. Fikes, T. Finin, T. R. Gruber, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):35–56, 1991.

[28] 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.

[29] B. J. Peterson, W. A. Andersen, and J. Engel. Knowledge Bus: Generating Application-focused Databases from Large Ontologies. In *Proceedings of the KRDB*, 1998.

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

[31] 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.

[32] D. Schlangen, M. Stede, and E. Paslaru Bontas. Feeding OWL: Extracting and Representing the Content of Pathology Reports. In *Proceedings of the NLPXML04*, 2004.

[33] F. Sowa, A. Bremen, and S. Apke. Entwicklung der Kompetenz-Ontologie fr die Deutsche Montan Technologie GmbH. http://www.kowien.uni-essen.de/workshop/DMT_01102003.pdf, 2003.

[34] Y. Sure, S. Staab, and R. Studer. *On-To-Knowledge Methodology (OTKM)*, pages 117–132. Springer Verlag, 2004.

[35] B. Swartout, R. Patil, K. Knight, and T. Russ. Toward Distributed Use of Large-Scale Ontologies. In *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1996.

[36] 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.

[37] 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.

[38] M. Uschold and M. King. Towards a Methodology for Building Ontologies. In *Proceedings of the IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[39] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The Enterprise Ontology. *The Knowledge Engineering Review*, 13, 1998.