Using Protégé-2000 in Reuse Processes

H. Sofia Pinto[1], Duarte Nuno Peralta[1], and Nuno J. Mamede[2]

[1] Grupo de Inteligência Artificial, Departamento de Eng. Informática
Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa, Portugal
{sofia,duarte}@gia.ist.utl.pt
[2] L[2]F INESC-ID/IST - Spoken Language Systems Laboratory
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{Nuno.Mamede}@inesc-id.pt

Abstract. There is already a considerable number of ontology-based tools. In order to better choose the appropriate tool and to better use its capabilities, tools need to be compared and the experiences in using them should be shared. We have been using Protégé-2000 in an ontology reuse experience. In this case, we reused an ontology kept in the Ontolingua Server library. In this article, we report our experience in using the import translators for OKBC and the support provided by the tool in revision/extension processes. The analysis provided in this article is from an user point of view.

1 Introduction and Motivation

There is already a considerable number of ontology-based tools. At this moment these tools are attracting an increasing number of new users of all kinds, from naive users to power users. As more tools become available, the problem of interoperability between different tools becomes more important. Different tools will be combined in different ways to support varied and increasingly more complex ontology related processes. For instance, one may import an ontology available in the library of a particular ontology building environment into another ontology building environment and extend the imported ontology with more knowledge. The resulting ontology may then be translated into another knowledge representation language and introduced in an evaluation tool.

In order to better choose the appropriate tool and to better use its capabilities, tools need to be compared and the experiences in using them should be shared. This analysis has got to be performed in much more detail than a brochure-like collection of features, specially for power users. That is, we must know the limits of available technology.

We have been using Protégé-2000 [8] to build an ontology. In this case, the ontology was built through a reuse process. We reused an ontology kept in the Ontolingua Server library [4]. In this article, we report our experience in using the import translators for OKBC [2] of Protégé-2000 and the support provided by this tool in reuse processes.

In this article, we begin by referring existing evaluation studies about ontology-based tools and their shortcomings. Then we describe the context of our experience and the actual reuse process that was performed using Protégé-2000. We describe the problems and strong points of Protégé-2000 in our case-study. Finally, we explain the reasons underlying these problems and analyze related work.

2 Evaluation of Ontology-Based Tools

There are already some studies evaluating ontology-based tools in the literature, like WonderTools [3] and the survey on ontology-based tools of OntoWeb [9].

In [3] an evaluation framework is proposed and a comparative study of several ontology-building tools was made. This was the first systematic evaluation/comparison study of ontology-based tools, more precisely of ontology building tools. However, one important dimension that is not analyzed by this framework is the study of the interoperability between the different tools. For instance, the existence of import/export translators is not analyzed.

In [9] a general survey of ontology-based tools is presented, namely for ontology building, merge, evaluation, annotation, and storage and query tools. This survey proposes an evaluation framework for each kind of tool, and compares each tool against the corresponding framework. Although this survey provides an evaluation of each kind of tool, the interoperability between different tools is not analyzed in detail. For instance, the limitations of current translators are not analyzed.

3 Context of the Experiment

We are involved in the development of ontologies to be used in a Natural Language dialogue system. This dialogue system is to be placed in a bus terminal, as a ticket-vending/information machine. An important requirement of this application is that the language must be Portuguese (although the concepts represented in an ontology are, in general, language independent,[3] the terms used to refer to those concepts are not). The construction of this ontology will involve several subontologies in different domains related to traveling, for instance commercial transactions (buying and selling), geographical information and time.

We began the development of this ontology with the subontology of time. One of the requirements that was a-priori imposed was that the ontologies should be developed in a locally installed tool. Therefore, the tool that was chosen was Protégé-2000.

[3] In Portuguese there is a concept that finds no parallel in other languages: "Saudade", which is a kind of nostalgia, home sick, is a genuine Portuguese concept.

4 A Reuse Process in Protégé-2000

The building process of the ontology of time has already went through several stages. In Fig. 1 we represent part of this process. The activities performed with Protégé-2000 are shadowed.

Import from Ontolingua To build our ontology, we reused the Simple-Time ontology[4] from the Ontolingua Server. We used the OKBC tab plug-in[5] to import the ontology into Protégé-2000.

Analysis - identification of missing and misplaced knowledge Before performing the analysis, we had already applied a manual reengineering [1] process to the source code. The result of this process was one possible conceptual model for Ontolingua's Simple-Time ontology. At this stage, we compared this model with the translated version of the ontology. We found that the translation process provided the taxonomic hierarchy of concepts (classes+instances). However, it did not translate all Ontolingua functions and it did not translate any relation or axiomatic definition. Moreover, some functions were misplaced.

Revision - rearrangement and extension of knowledge The knowledge we found misplaced in the ontology was relocated. In what concerns missing knowledge, we introduced the relations and functions. Following the guidelines provided in the documentation of Protégé-2000, we left the introduction of axioms for a later stage.

Analysis - technical evaluation of source ontology Having the taxonomy, relations and functions, we evaluated the Simple-Time ontology in the Ontolingua Server according to the criteria proposed in [6]. For that we used both the source code and the conceptual model. The reasons why we used both were: (1) if we only use the conceptual model we are not able to analyze the syntactical errors[6] (language conformity) and (2) if we only use the source code we loose the overall perspective of the ontology which is crucial to perform a thorough analysis.[7]

Revision - Correction, Natural Language Translation and Extension The problems found in the source ontology were corrected in the version kept in Protégé-2000. Then we translated all the names of the knowledge pieces represented in the ontology into Portuguese. Finally, the axioms were added to our ontology using Protégé Axiom Language (PAL). Moreover, some concepts, that are needed to describe the time domain for our particular application, were added.[8] For instance, half an hour.

---

[4] It defines 14 classes, 209 instances, 17 relations and 14 functions.

[5] `http://protege.stanford.edu/plugins/okbctab/okbc_tab.html`

[6] For instance, the `equals` relation in Ontolingua is defined for both `time-point`s and `time-range`s (polymorphic refinement). However, there is an error, since the definition for `time-range`s introduces two variables that are not declared.

[7] For instance, without the conceptual model, we could miss the fact that functions `month-of` and `month-name-of` are the same function.

[8] The description of the requirement specification performed in this development process is out of the scope of this paper.

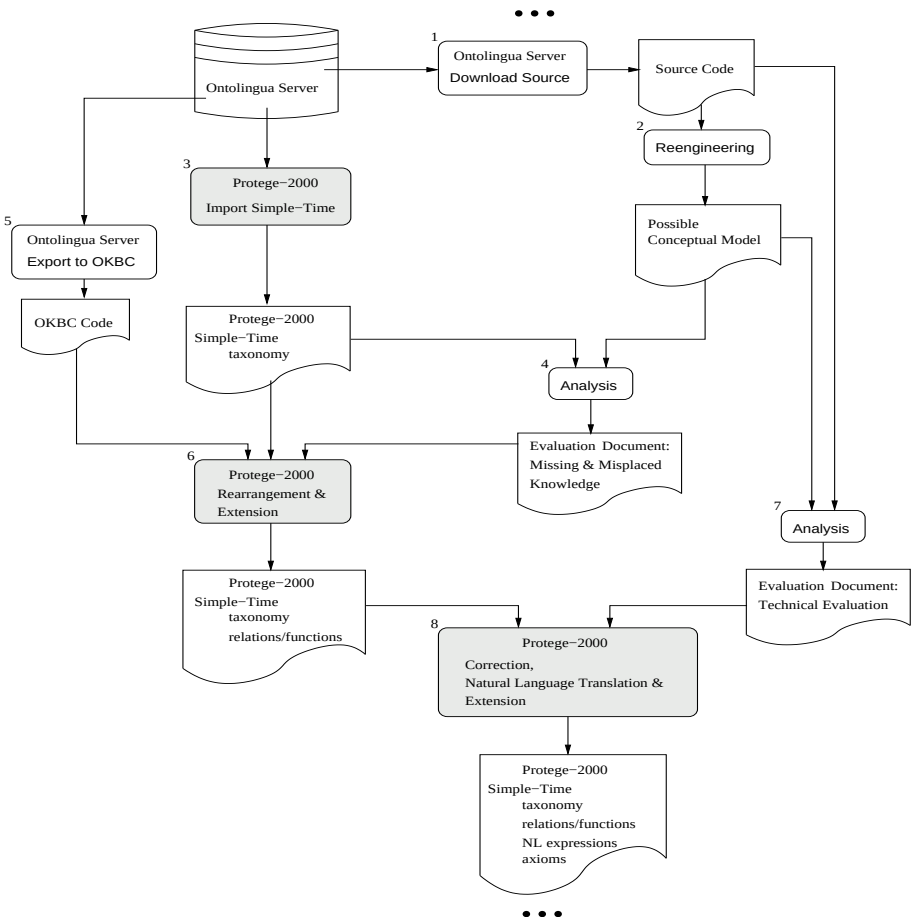

Fig. 1. Building process

In the following sections we analyze the use of Protégé-2000 to perform this process. The focus is placed on the advantages and disadvantages of using Protégé-2000 to build our time ontology. Since the tool was not used in the analyses stages of this process, we will not address them in this paper.

4.1 Problems after Importing from Ontolingua

To import the Simple-Time ontology from the Ontolingua Server, the OKBC tab plug-in was used. We found that part of the knowledge represented in the source ontology and imported using this translator was lost.

Comparing the definitions of the same concepts after using Ontolingua's export translator and after using Protégé-2000 import translator we can see that there are important differences. Take, for instance, the definition of the `Meets`

```
(define-relation Meets (?time-range-1 ?time-range-2)
"a time range ?time-range-1 ends at the same time a time range ?time-range-2 starts."
    :iff-def (Equals (End-Time-Of ?time-range-1)
                     (Start-Time-Of ?time-range-2)))
```

Fig. 2. Definition of relation `Meets` in Ontolingua

relation in Ontolingua, Fig. 2. The Ontolingua OKBC export translator represented this binary relation as a slot, Fig. 3. However, the OKBC import translator of Protégé-2000 lost this relation. Since axioms are outside the OKBC model, the axioms defining the relations were not translated, for instance the axioms defining the relation `Meets`.

Although one can argue that relations (and functions) are also outside the OKBC knowledge model,[9] the same problem seemed to affect the translation of slots. For instance, we tried to import the Agents ontology from the Ontolingua Server. The `agent` class is defined as a frame with a template slot `name`. After importing this ontology into Protégé-2000, all slots were lost.

Some of the functions were imported, but not all. For instance, the function `+` involving `time-points`, `time-ranges` and `durations` was lost. Moreover, the functions that were imported, were not translated as expected. For instance, the function `year-of` is defined in the source ontology with domain `time-point`. However, after the translation, the slot that was created was not attached to the `time-point` class. It was only attached to the classes `calendar-year`, `calendar--date` and `universal-time-spec`. These concepts were characterized in their Ontolingua [7] definitions by having one (`has-one`) `year-of`. Moreover, the minimum cardinality constraints of the slots corresponding to these functions were lost.

Summarizing, the problems we found in the translation of the Simple-Time ontology were:

- All knowledge represented using the `define-relation` primitive was not translated, for instance the relation `Meets`.
- Some functions were imported, but most were lost. The only slots that were created in Protégé-2000 correspond to functions that appear in the definition of classes using the `has-one` primitive. Although the slot was created, it was attached to the incorrect class (when compared to the source code) and the minimum cardinality constraints were lost.

4.2 Rearrangement and Extension of Knowledge

In the beginning of this stage we had just identified what knowledge had been lost and misplaced. At this time, we decided to rearrange misplaced knowledge and only add the missing relations and functions.

[9] Classes, instances, slots and facets.

```
(define-okbc-frame Meets
                   :frame-type :slot
                   :direct-types (Relation Binary-Relation)
                   :own-slots ((Arity 2))
                   :primitive-p common-lisp:nil
                   :sentences
                   (...
                    (=> (Meets ?time-range-1 ?time-range-2)
                     (Equals (End-Time-Of ?time-range-1)
                             (Start-Time-Of ?time-range-2)))
                    ...))
```

Fig. 3. Definition of relation `Meets` in OKBC using Ontolingua's export translator

Part of the rearrangement and extension done at this stage is shown in Fig. 4. For example, with the translated version of the ontology, it was not possible to characterize the day of a given `time-point`. Since the class of `time-points` lost all its attributes, we could not partially characterize specific `time-point` instances. This can be done in the Simple-Time ontology at the Ontolingua Server. The only classes for which we could do this were `calendar-year`, `calendar-date` and `universal-time-spec`.

The tool proved to be a good help, since relocating knowledge simply corresponds to a drag&drop action of the mouse. Like in a regular frame-based system, another way of relocating a slot is removing the slot from the class where it is wrongly associated and attach it to the correct class. In fact, the intuitive use of the tool is one of its main features.

To manually introduce the missing slots, we used the OKBC code produced by Ontolingua's export translator as a guide, Fig. 3. We simply created the slots that were defined in this translation and attached them to the appropriate classes. All relations (17) and lost functions (9) were introduced.

4.3 Correction, Natural Language Translation, and Extension

In the beginning of this stage we had a translated version of the Simple-Time ontology, consisting of the whole taxonomy, the relations and functions. We also had the technical evaluation document. The problems found in the analysis of the source ontology were corrected (Correction), the names in the ontology were translated into Portuguese (Natural Language Translation) and the axioms that were lost during the import translation from Ontolingua were added (Extension).

Correction consisted mainly in changing the ranges and domains of functions and relations. In Protégé-2000, this means relocating slots (change of domain) or changing the value of the `allowed-classes` facet of the slot (change of range).

After correcting the taxonomy and before starting to write axioms, the ontology was translated into Portuguese, because axiom definitions refer to names of frames in the ontology, and changes to these names are not propagated to the textual definitions of the axioms. One major advantage of using this tool to perform this task was that once we change the name of a frame, this change is immediately propagated through the entire taxonomy.

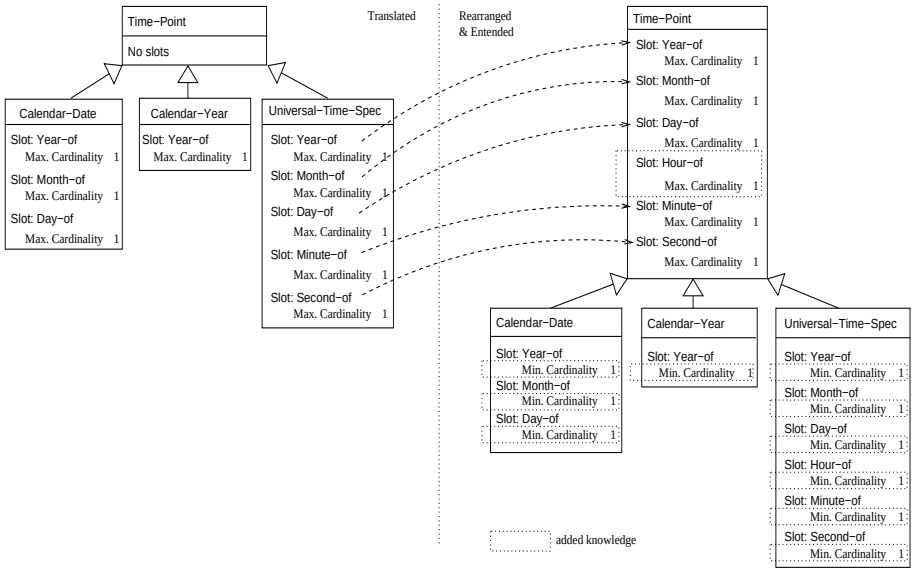

Fig. 4. Rearrangement and Extension of `time-point` and its subclasses

Finally, we added knowledge that was completely lost in the translation process: axioms. To write axioms in Protégé-2000 the PAL constraints tab plug-in[10] was used. PAL constraints are part of the PAL toolset plug-in for Protégé-2000. The idea is to allow the user to write constraints over the possible values of instances that cannot be represented using only classes, slots and pre-defined facets.[11] PAL is a limited predicate logic extension of Protégé-2000. Its syntax is similar to KIF [5]. However, statements like `defrelation` and `deffunction` are not supported. The PAL language is completely integrated with the Protégé-2000 knowledge model. Constraints are instances of the `:pal-constraint` class. When writing a PAL constraint, we can use any slot as a predicate, for instance `(start-time-of (time-range x) (time-point y))`. If the slot has a `maximum-cardinality` of 1, it can also be used as a function. In this case, `(start-time-of (time-range x))` represents `(time-point y)`. The PAL constraint checking mechanism can be called by the user, to show which constraints are violated and by which instances. It can also be called programmatically by an application that uses the Protégé-2000 API.

The PAL constraint checking mechanism also has a trace mechanism that allows us to follow the evaluation of a given constraint. This is very useful when writing constraints, since it helps to understand why they are not working as we

[10] `http://protege.stanford.edu/plugins/paltabs/PAL_tabs.html`

[11] For instance, `value-type`, minimum and maximum `cardinalities`, `minimum` and `maximum` values allowed.

```
(defrange ?time-range-1 :FRAME TIME-RANGE)
(defrange ?time-range-2 :FRAME TIME-RANGE MEETS)
(forall ?time-range-1
        (forall ?time-range-2
                (=> (and (MEETS ?time-range-1 ?time-range-2)
                         (own-slot-not-null END-TIME-OF ?time-range-1)
                         (own-slot-not-null START-TIME-OF ?time-range-2))
                    (= (END-TIME-OF ?time-range-1)
                       (START-TIME-OF ?time-range-2)))))
```

Fig. 5. Axiom written in PAL

think they should. However, this tracing mechanism only traces predicates and functions predefined in PAL language, for instance < for numbers, and only one at a time. It would be also useful to provide trace-ability of user-defined slots (treated in PAL as predicates or functions).

Although the PAL constraint checking mechanism is useful, it is not very easy to use at first. One of the reasons is because knowledge represented in KIF axioms that is not supported by PAL has to be transformed before it can be incorporated. PAL axioms consist on a set of variable range definitions and a predicate that must hold over those variables. An example of an axiom written in PAL is shown in Fig. 5.

This first user's reaction to PAL was not a very good one. This could be easily changed with a more detailed documentation. We found some examples on how to write a new constraint in the documentation, but some examples on how to transform an axiom written in KIF (or any other language) into a PAL constraint would also be useful.[12]

5 Translator Analysis

We analyzed the OKBC-tab plug-in code, in order to better understand why knowledge was being lost or misplaced. The plug-in starts by connecting to an OKBC compliant server (in our case, the Ontolingua Server) and then uses the OKBC protocol to get information about the ontology being imported. The main import procedure, `getClassDetails`, starts with an initial set of classes and then goes down the `is-a` hierarchy getting for each class its name, documentation, instances, template slots,[13] and for each template slot the value of the facets `value-type`, `maximum-cardinality` and `minimum-cardinality`. When reaching instances it gets their names, slots and slot values. This procedure leaves out any knowledge that is not explicitly represented in the definition of a class or instance frame. Moreover, any knowledge that is represented as an own slot in a class frame is also lost.

[12] Since our goal was to reuse the definitions implemented in KIF (or Ontolingua).
[13] The knowledge model of Protégé-2000 does not include own slots attached to classes.

In the case of the Simple-Time ontology all relations are represented in OKBC as slot frames and are not explicitly referred in the definition of any class frame. Therefore, all relations were lost.

However, this still did not explain why in the Agents ontology the slot `name` associated to the `agent` class was not translated. We found that the OKBC plug-in does not import frames whose names are keywords and `name` is a keyword in OKBC.

We also discovered the causes of misplaced knowledge. For instance, the `time-point` class is defined in Ontolingua as being the `domain-of` of the function `year-of`. Since the definition of the class `time-point` in OKBC only makes reference to the `year-of` function in the own-slot `domain-of`, the corresponding slot is not created. However, the class `calendar-date` is defined in Ontolingua as having one (`has-one`) `year-of`. In this case, the OKBC definition of `calendar--date` explicitly mentions that `year-of` is a template slot of this class. Therefore, the slot is created in Protégé-2000 and attached to the `calendar-date` class.

Knowledge about a slot is explicitly represented in the definition of a class frame in OKBC if in the Ontolingua definition of that class some specific primitives are used. If we use the `define-class` primitive we can use the `has-one` or `has-some` primitives to constraint cardinalities. If we use the `define-frame` primitive to define a class we can specify one of the following items: (1) the name of the template slot, (2) its `maximum-cardinality`, (3) `minimum-cardinality` or (4) `value-type`. Regarding cardinalities, Protégé-2000 only uses the values of cardinalities to assess whether the slot is multiple-valued. However, there are some problems. The maximum cardinality is always one and the minimum cardinality is not used at all.

To summarize:

- The import translator of Protégé-2000 does not import knowledge that is not explicitly represented in the definition of a specific class or of an instance in the Ontolingua Server's OKBC code.
- Frames (classes, instances and slots) whose name is a keyword of the OKBC protocol are not imported.
- All knowledge represented in own-slots of classes is not imported.
- Not all Ontolingua primitives can be used to explicit knowledge about template slots in the definition of a class.
- All facets, except `value-type`, are lost. Although the translator looks for the facets `maximum-cardinality` and `minimum-cardinality`, Protégé-2000 only uses that information to assess the multiplicity of the slot.

6 Related Work

A translation process involving the Simple-Time ontology in the Ontolingua Server is described in [10]. In this case the export translator of Ontolingua into Loom was used. At that time, the conclusions were that the automatic translators were still at draft level. Although they were useful to provide initial versions,

considerable human interaction was needed to improve the automatic versions. The two problems found at that time were:

Mismatch of modeling styles The way knowledge is modeled in Ontolingua is different from the way it is usually modeled in Loom. The constructs provided by each language form a representation ontology that is different for each language. A translator between two languages must start by mapping between the two representation ontologies. In general, building a translator is easier if these ontologies are similar.

Inference engine bias Even if the knowledge is modeled without a specific application in mind, it is usually modeled considering certain inferences. For instance, in Loom, knowledge is usually modeled considering that it is going to be used by its built-in inference engine.

In our case study we found that:

- Both the Ontolingua Server and Protégé-2000 are OKBC-compatible, so there is no mismatch in modeling styles.
- However, the inference engine bias is very strong in our case. For instance, when it comes to axioms, the PAL language has a rather different style from KIF. The PAL toolset is a constraint-checking rather than theorem-proving mechanism. This means that PAL only checks constraints based on the instances in the ontology. So, axioms written in PAL have to make strong closed-world assumptions about the knowledge that is being modeled. Moreover, in order to simplify the axioms, we added a slot `representacao-numerica` (number-representation) to all classes that are subclasses of integers (`hour-number`, `year-number`, etc.). This was done because in PAL the relations `<`, `>` and `=` are already defined for integers.

7 Conclusions and Future Work

In this article we report a reuse experience that involved translation, rearrangement, correction and extension of an ontology using Protégé-2000. We tried to evaluate the support provided by this tool in reuse processes. We have found that the OKBC import translator of Protégé-2000, although useful for providing an initial version, is not ready to be used in a fully automatic translation process. We analyzed the source code of the OKBC-tab plug-in and discovered the source of the problems that we had identified.

Regarding usability, we have found the tool to be intuitive and easy to use. The tool eased our reuse process. In particular, we have found that it was more cost effective (time+effort) to use the tool rather than building the whole time ontology from scratch in Protégé-2000.

In future we plan to build other ontologies by means of reuse, namely the other subontologies needed for the natural language dialogue system, using the support of available tools. We also plan to improve the OKBC import translator of Protégé-2000.

8 Acknowledgements

We thank the anonymous reviewers for their usefull comments. We also thank the support provided by the technical support teams of Protégé-2000 and Ontolingua Server. We thank FCT.

References

1. Blázquez, M., Fernández, M., García-Pinar, J. M., Goméz-Pérez, A.: Building Ontologies at the Knowledge Level Using the Ontology Design Environment. In: *Proceedings of the Knowledge Acquisition Workshop* (KAW98), Banff, Alberta, 1998.
2. Chaudri, V., Farquhar, A., Fikes, R., Karp, P., Rice, J.: *Open Knowledge Base Connectivity 2.0.3.* Knowledge Systems Laboratory, KSL-98-06, Stanford University, 1998.
3. Duineveld, A. J., Stoter, R., Weiden, M. R., Kenepa, B., Benjamins, V. R.: Wondertools? A comparative study of ontological engineering tools. In: *Proceedings of the Knowledge Acquisition Workshop* (KAW99), Banff, Alberta, 1999.
4. Farquhar, A., Fikes, R., Rice, J.: The Ontolingua Server: A Tool for Collaborative Ontology Construction. In: *Proceedings of the Knowledge Acquisition Workshop* (KAW96), Banff, Alberta, 1996.
5. Genesereth, M.: Knowledge Interchange Format. In:J. Allen and R. Fikes and E. Sandewall (eds.): *KR91 Proceedings*, Morgan Kaufmann: 599–600, 1991.
6. Gómez-Pérez, A., Juristo, N., Pazos, J.: Evaluation and Assessment of the Knowledge Sharing Technology. In: N.J.I. Mars (eds.): *Towards Very Large Knowledge Bases*, IOS Press: 289–296, 1995.
7. Gruber, T.: A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, **5**: 199–220, 1993.
8. Noy, N.F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R.W., Musen, M.: Creating Semantic Web Contents with Protégé-2000. *IEEE Intelligent Systems* **48**(2): 60–71, 2001.
9. OntoWeb: *Deliverable 1.3: A survey on ontology tools*, 2002.
10. Russ, T., Valente, A., MacGregor, R., Swartout, W.: Practical Experiences in Trading Off Ontology Usability and Reusability. In: *Proceedings of the Knowledge Acquisition Workshop* (KAW99), Banff, Alberta, 1999.