REUSING A TIME ONTOLOGY

Duarte Nuno Peralta and H. Sofia Pinto
Grupo de Inteligência Artificial, Departamento de Engenharia Informática
Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa, Portugal
Email: {duarte,sofia}@gia.ist.utl.pt

Nuno J. Mamede
L^2F INESC-ID/IST - Spoken Language Systems Laboratory
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
Email: Nuno.Mamede@inesc-id.pt

Key words: Ontology Engineering, Ontology Building, Evolving Prototyping, Ontology Reuse, Knowledge Representation Translation, Reverse Engineering, Technical Evaluation, Knowledge Acquisition.

Abstract: Ontologies are becoming crucial in several disparate areas, such as the Semantic Web or Knowledge Management. Ontology building is still more of an art than an engineering task. None of the available methodologies to build ontologies from scratch has been widely accepted. One cost effective way of building ontologies is by means of reuse. In this article we describe the development of an ontology of Time by means of reuse, following an evolving prototyping life cycle. This process involved several complex subprocesses: knowledge acquisition and requirement specification using Natural Language techniques, reverse engineering, knowledge representation translation, technical evaluation. As far as we know, this is the first time that all these processes have been combined together. We describe the techniques and best practices that were successfully used.

1 INTRODUCTION AND MOTIVATION

Nowadays ontologies are becoming crucial in several disparate areas, such as the Semantic Web (Berners-Lee et al., 2001) or in Knowledge Management. In the Semantic Web, they will be one of its building blocks, improving conventional information retrieval technologies. In Knowledge Management, they support interoperability (easing knowledge sharing) and organization of information.

Ontology building is still more of an art than an engineering task. None of the available methodologies to build ontologies from scratch has been widely accepted. In the Ontology Engineering area several processes to ease the time consuming and complex task of ontology building have been proposed.

In this article we describe the ongoing ontology building process of ONTO-SD. This ontology will be organized in several sub-ontologies in different domains related to traveling, for instance commercial transactions (buying and selling), geographical information and time. The emphasis of this article is the development of the Time sub-ontology.

Reuse processes are a cost effective way of building ontologies. The Time ontology was built by means of reuse, following an evolving prototyping life cycle. The process described in this article involved several complex subprocesses: knowledge acquisition and requirement specifi cation using Natural Language techniques, reverse engineering, knowledge representation translation, technical evaluation. As far as we know, this is the fi rst time that all these processes have been combined together. We describe the techniques and best practices that were successfully used.

We start by explaining the context of our experiment, Section 2. Then, we describe the ontology building process, Section 3. We discuss the most interesting features of our experiment, Section 4. Finally, we analyze related work, Section 5, and present our conclusions and future work, Section 6.

2 CONTEXT OF THE EXPERIMENT

We are involved in the development of ontologies to be used in a Natural Language dialogue system. This dialogue system will 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, lan-

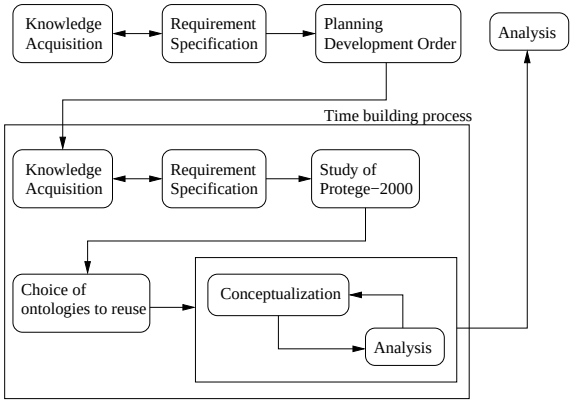

Figure 1: Building process of ONTO-SD

guage independent,[1] the terms used to refer to those concepts are not).

We began the development of this ontology with the sub-ontology of time. At the moment, the several modules of the final ontology have been identified and the sub-ontology of time has been developed.

3 THE ONTOLOGY BUILDING PROCESS

When compared with typical software engineering processes, ontology building processes are less straight-forward. Typically, an ontology building process involves several iterations, most of them not a-priori planned. That is, ontology building processes follow an evolving prototyping life cycle.

In this section we describe the ONTO-SD building process performed so far (Pinto et al., 2002; Peralta, 2002), Fig. 1. Some parts of this process are rather complex and can be considered subprocesses on their own. We describe them in detail in appropriate subsections. Not only do we describe the activities that were performed - in *italics* - but also how they were performed.

Knowledge acquisition We began the whole process by going to a bus terminal in Lisbon and taping the dialogues that took place between users and the ticket booth employee. We will further describe this part of the process in Section 3.1.

Requirement specification/Conceptualization First we looked for the most common terms in the transcription of the dialogues. With this information we identified the several sub-domains

[1]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.

User:	To Oporto, at **17 hours**, in front.
Employee:	17 ?
User:	Yes.
Employee:	12 Euro.
User:	And for the boy ?
Employee:	8.

Figure 2: Example of real (translated) dialogue

that are involved, and the relations between them. Then, we structured the ONTO-SD ontology. We will further describe this part of the process in Section 3.1.

Planning development order Based on the structure found for ONTO-SD, an order was established for the development of the sub-ontologies. We will further describe this part of the process in Section 3.2.

Building the Time sub-ontology Building this sub-ontology is a sub-process of the overall building process. It was divided into:

Knowledge acquisition To represent and process the knowledge in the dialogues one needs a large corpus of knowledge, ontologies. We found that most of the terms in the dialogues require implicit knowledge in order to be represented. We *searched* for sources of knowledge, mostly ontologies, that contain the knowledge that cannot be extracted from the blind analysis of the texts corresponding to the transcription of the dialogues. The *chosen* source was Allen's time theory (Allen, 1984).

Requirement specification In this stage we *refined* the requirement specification stage of ONTO-SD for this particular sub-ontology. Mainly, we *identified* all terms related to time in the dialogues. One of the shortest and easiest dialogues that took place is shown in Fig. 2. It must be possible to represent the concepts referred by these terms using only the `Time` sub-ontology. This list of terms allows us to evaluate whether the final ontology meets its requirements.

Study of Protégé-2000 At this stage we started to *study* Protégé-2000 (Noy et al., 2001), since it was the chosen development tool. However, the learning process was continuous, and it stretched throughout the entire process. This tool was chosen because (1) it is locally installed, which is one of the a-priori requirements that was imposed, (2) it has a large community of users and (3) it has a very good and efficient technical support.

Choice of ontologies to reuse We *looked for* ontologies that could be reused both in libraries,

such as the Ontolingua Server[2] (Farquhar et al., 1996), and in large ontologies, in particular WordNet[3], upperCyc[4] and SUMO.[5] We did a *preliminary analysis* and found out that most of the available ontologies about time were implementations of Allen's time theory. We decided to reuse the Simple-Time ontology, which is kept in the Ontolingua Server. The reasons for this *choice* were:

- This ontology is an implementation of the chosen time theory.
- Protégé-2000 has a plug-in to import ontologies kept in the Ontolingua Server, which eases the translation task.
- In large ontologies, such as WordNet, SUMO and upperCyc, knowledge about time is scattered throughout the ontology. Since these large ontologies are not organized in subontologies it is difficult to extract all the knowledge about a specific domain.
- In the Ontolingua Server the Simple-Time ontology is one ontology, an unique object, with well defined boundaries, so it is easier to reuse.

Conceptualization/Analysis/Revision Cycle During this stage of the process, the subontology of time went through several iterations. This cycle consisted in: (1) make an initial conceptualization, (2) analyze the ontology and driven by this analysis, (3) revise the ontology, changing the conceptualization. We will further describe this part of the process in Section 3.3.

3.1 Knowledge Acquisition & Requirement Specification/Initial Conceptualization of ONTO-SD

We decided not to separate the descriptions of the knowledge acquisition and requirement specification/conceptualization stages, because we were constantly interchanging between the two. The knowledge acquisition and requirement specification/conceptualization stages of ONTO-SD can be divided into three steps:

- **Collect dialogues** We started by *taping* real dialogues in a bus terminal in Lisbon. We left the recorder in a ticket booth for one hour. The users were not warned of the presence of the recorder, so the dialogues were not biased by its presence.

- **Statistical analysis of dialogues** Using the *transcription* of these dialogues, we used a simple computer program[6] to count the number of occurrences of each term. In the first frequency analysis of the dialogues only individual terms were considered. However, we found that in some cases we should consider groups of terms instead of individual terms. For instance, we should consider the expression "Cartão Jovem" (Youth Card), and not "Cartão" and "Jovem". Therefore, we changed the transcription of the dialogues, so that those groups of terms would appear as one term only. Part of those results is presented in Fig. 3.

  This simple *analysis* can only help us in: (1) getting an idea of the most important concepts that will have to be appropriately represented[7] and (2) identifying the domains involved.

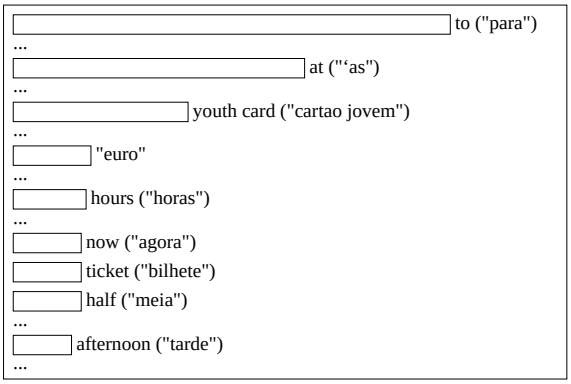

Figure 3: Frequency analysis of terms in the dialogues

- **Identification of domains** Using the results of the statistical analysis, we *identified* the most important domains that are a part of our ontology. To identify the relations between the several domains, we *analyzed* the most complex dialogues, that involved all domains.

  The most common term used is "para",[8] Fig. 3. This means that the users identify the tickets they want to buy by stating their traveling destination. Another set of words often used is "Cartão Jovem", which refers to one available discount. This means the final ontology needs to have knowledge about possible discounts. There are also many words related to time. Sometimes a simple term analysis is not enough. We have to analyze the dialogues

[2] `http://ontolingua.stanford.edu:5915`

[3] `http://www.cogsci.princeton.edu/~wn/`

[4] `http://www.cyc.com/cyc-2-1/cover.html`

[5] `http://ontology.teknowledge.com`

[6] Program `WordCount` available at `http://www.intellij.com` as a sample of IDEA IDE for JAVA, from IntelliJ.

[7] Its aim is not to provide us with the required `Time` ontology.

[8] "Para" is the Portuguese word for "to", as in "ticket *to* the Algarve".

in detail and use abstraction to extract the concepts to be represented in the ontology. For instance, we will need to represent the concept `destination`, although the dialogues only refer specific locations.

All the domains and relations between them were identified, and the *structure* of ONTO-SD was established. This structure is shown in Fig. 4. There are five modules. The `Time` module defines concepts that are strictly related to time, for instance "hour". The module `Time-Order`, will define concepts like "next", "first", etc. The reason why we chose not to represent these concepts also in the Time module is that not only they are used in the dialogues to express a relation order between time points and time ranges, but also they can be used to express order between other concepts, in a different domain. For instance, related to trips, we have the concepts of "first bus stop", "last bus stop", etc. By separating the modules, we can later reuse the Time-Order module to represent those concepts. Another module is `Spatial-Information`. It will define concepts such as origin, destination, stop, all possible stops for all buses, and other spatial concepts, such as the position of seats inside a particular bus, or the exact location inside the bus terminal from which the bus will depart. The `Travel` module will define the concepts related to traveling, for instance, "bus stop", or "bus ticket". The `Commerce` module will define the concepts related to buying and selling, for instance, "Euro".

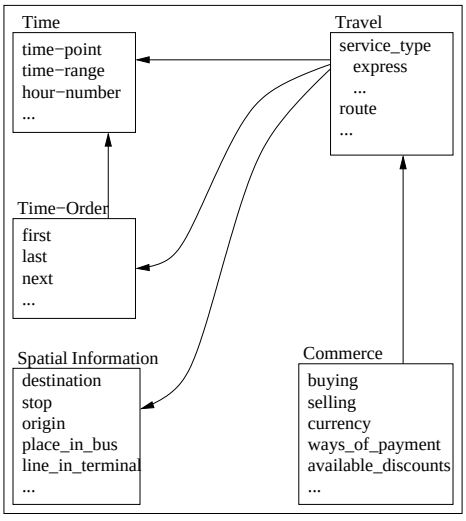

Figure 4: ONTO-SD structure

Trying to find the structure of the ontology and the relations between these modules is already a part of conceptualization activities.

3.2 Planning the Development Order

Before knowledge acquisition and requirement specification stages we had already decided to start the process by the sub-ontology of Time. Empirically, we could foresee that ONTO-SD would have a sub-ontology about time and this module would be independent from the rest of the ontology. Having established the final structure, our idea was confirmed. There are two modules in ONTO-SD that do not include any others. These modules are `Time` and `Spatial-Information`. The `Time` module is a better place to start than the `Spatial-Information` module because (1) it involves common sense knowledge about time, and (2) we would not need an expert in the domain to perform acquisition.

The established development *order* is: (1) `Time`, (2) `Time-Order`, (3) `Spatial-Information`, (4) `Travel` and (5) `Commerce`.

3.3 Conceptualization/Analysis/Revision Cycle of Time ontology

In this subprocess, Fig. 5, the ontology went through several iterations:

- **Reverse Engineering of Simple-Time** At this stage we *obtained a possible conceptual model* for the Simple-Time ontology. Part of this conceptual model is shown in Fig. 6. This subprocess was divided into three stages:
  - **Identify classes** We analyze the source code, Fig. 7, searching for definitions of classes. We link each class (in **bold**) to its super/subclasses establishing the taxonomy of classes for the ontology.
  - **Identify instances** We analyze the source code one more time searching for definitions of instances and link them (in *italics*) to the appropriate classes in the conceptual model.
  - **Identify relations and functions** We analyze the source code searching for definitions of relations and functions, and add them to the conceptual model. In the case of binary relations we link the appropriate origin and target concepts. The label of these links is the name of the relation.
- **Import from Ontolingua** To *import* the Simple-Time ontology from the Ontolingua Server into Protégé-2000 we used the OKBC (Chaudri et al., 1998) tab plug-in.[9]

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

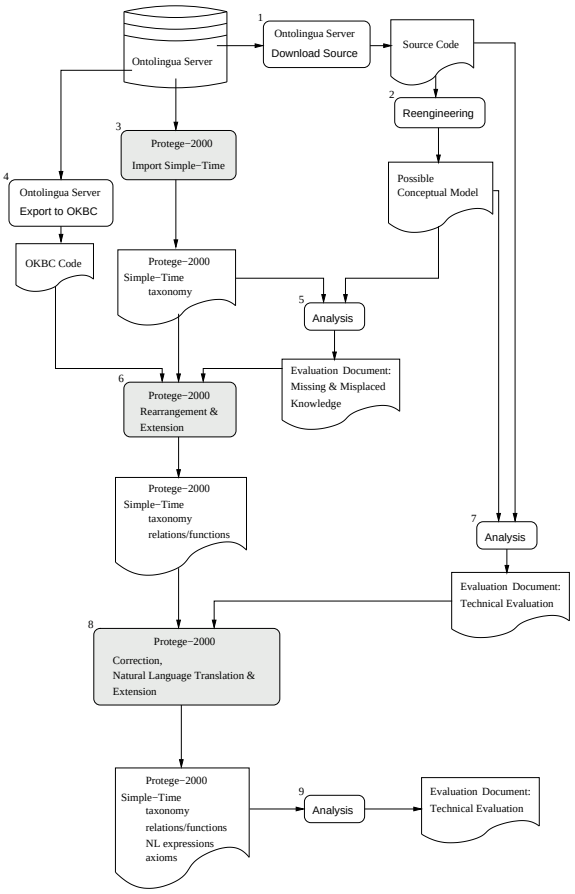

Figure 5: Conceptualization/Analysis/Revision cycle of Time ontology

- **Analysis - identification of missing and misplaced knowledge** At this stage, we *compared* the conceptual model obtained earlier for the Simple-Time ontology with the translated version of the ontology. We *found* that the taxonomy was correctly imported. However, all relations and some functions were lost. All the axiomatic definitions of these relations were also lost. Although relations and axioms are outside to OKBC knowledge model, the OKBC export translator of the Ontolingua Server translated these relations as slots.[10] 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.

[10]The OKBC knowledge model only allows binary relations, which are represented as slots.

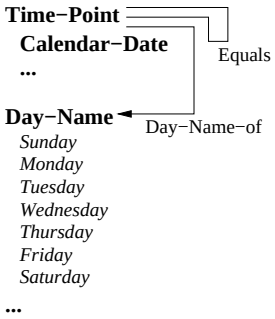

Figure 6: Part of Simple-Time conceptual model

- **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 (Gómez-Pérez et al., 1995). 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[11] (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.[12]

- **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). The axioms were only introduced at this stage because in the documentation of the tool it is recommended that the axioms should be written only after the taxonomy is defined. Moreover, some of the *concepts* that we found during the requirement specification stage of `Time`, were *added*. For instance, half an hour.

- **Analysis - technical evaluation of Time ontology** Finally, we *technically evaluated* our ontology, again using the criteria proposed in (Gómez-Pérez et al., 1995), and also *comparing* the final result with the original Simple-Time ontology.

  One of the differences is that the `Time` ontology introduces a new concept, `representacao-`

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

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

```
(Define-Class Calendar-Date (?T)
"a specification of a point in absolute calendar time,
 at the resolution of one day."
:Def (And (Time-Point ?T)
          (Has-One ?T Day-Of)
          (Has-One ?T Month-Of)
          (Has-One ?T Year-Of))) ...

(Define-Frame Time-Point
:Own-Slots
 ((Arity 1)
  (Documentation
      "A time-point is a point in real, historical time
       (on earth).  It is independent of observer and
       context.  A time-point is not a measurement of
       time, nor is it a specification of time.
       It is the point in time. The time-points at which
       events occur can be known with various degrees of
       precision and approximation, but conceptually
       time-points are point-like and not interval-like.
       That is, it doesn't make sense to talk about what
       happens during a time-point, or how long the
       time-point lasts.")
  (Domain-Of Day-Of Minutes-Of Month-Of Seconds-Of
   Year-Of)
  (Instance-Of Class)
  (Subclass-Of Individual))) ...

(Define-Relation Equals (?Time-Point-1 ?Time-Point-2)
 " a time point ?time-point-1 is equal to a time point
   ?time-point-2. a time range ?time-range-1 is
   identical to a time range ?time-range-2."
:Axiom-Def
    ((=> (And (Time-Point ?Time-Point-1)
              (Time-Point ?Time-Point-2))
         (<=> (Equals ?Time-Point-1 ?Time-Point-2)
              (And (= (Year-Of ?Time-Point-1)
                      (Year-Of ?Time-Point-2))
                   (= (Month-Of ?Time-Point-1)
                      (Month-Of ?Time-Point-2))
                   (= (Day-Of ?Time-Point-1)
                      (Day-Of ?Time-Point-2))
                   (= (Hour-Of ?Time-Point-1)
                      (Hour-Of ?Time-Point-2))
                   (= (Minute-Of ?Time-Point-1)
                      (Minute-Of ?Time-Point-2))
                   (= (Second-Of ?Time-Point-1)
                      (Second-Of ?Time-Point-2)))))

     (=> (And (Time-Range ?Time-Range-1)
              (Time-Range ?Time-Range-2))
         (<=> (Equals ?Time-Range-1 ?Time-Range-2)
              (And (Equals (Start-Time-Of ?Time-Range-1)
                   (Start-Time-Of ?Time-Range-2))
                   (Equals (End-Time-Of ?Time-Range-1)
                   (End-Time-Of ?Time-Range-2)))))) ...
```

Figure 7: Example of Ontolingua source code

`-numerica`,[13] which is a function. It relates each temporal entity with the corresponding integer which is normally used to represent it.[14] It also introduces the concept `numero-tempo`,[15] which is a superclass of all classes that can be characterized using the `representacao-numerica` slot.

In order to *test the expressivity requirements* of the ontology, we also *represented* the concepts identified in the dialogues during the requirement specification stage. We *found* that some of these concepts cannot be represented directly in the ontology. For instance, the concept of `amanha` ("tomorrow") depends on the specific date on which the knowledge base is queried, so it cannot be statically represented in the ontology.

[13] "Representacao-numerica" is the Portuguese word for "numeric representation".

[14] For instance, the numeric representation of the instance `1-the-month-number` of the class `month-number` would be the `Integer` 1.

[15] "numero-tempo" is the Portuguese word for "time-number".

4 DISCUSSION

There are some unique features in this experience. Technical evaluation of the chosen source ontology was only performed after importing the ontology. If we had performed it before actually reusing the ontology, the logical step would be to correct the errors in the source ontology. However, in our case: (1) we could not change the source ontology, since it is available at the library of the Ontolingua Server, and (2) changing the source code outside the Ontolingua Server would prevent us from using Protégé-2000 to automatically perform the import procedure.

In what concerns the amount of effort, there two facts that contributed in easing the process: (1) the reuse of an existing ontology on the same domain and (2) the use of an ontology development tool.

Reusing an ontology was helpful, because it provided a conceptualization of the domain, which reduced the effort of knowledge acquisition.

The use of an ontology development tool helped (1) when all terms related to the concepts defined in the ontology were translated into Portuguese, because each knowledge piece only had to be translated once and all references to it were automatically changed, (2) when misplaced knowledge was rearranged in Protégé-2000, since we used drag&drop operations, and (3) when the Simple-Time ontology was automatically imported from the Ontolingua Server. Even with the conceptual model, which is the main advantage of reusing the Simple-Time ontology, if we had manually inserted the whole ontology from scratch into Protégé-2000, the effort would be much greater.[16]

5 RELATED WORK

There are several ways of building ontologies. Ontologies can either be built from scratch or by means of reuse. There are two different kinds of reuse processes: merge and integration (Pinto et al., 1999). The process described in this article is a typical integration process.

Although there are several methodologies proposed in the literature to build from scratch, none has been

[16] The Simple-Time ontology defines 14 classes, 209 instances, 17 relations and 14 functions.

accepted as standard. Some ontology reuse processes are described in the literature, but there are very few methodologies proposed for each kind of reuse process.[17]

Reuse processes are described in (Gangemi et al., 1998; Chaudhri et al., 2000; Pinto, 1999). The process described in (Gangemi et al., 1998) is a typical merge process. The one described in (Chaudhri et al., 2000) combines both merge and integration subprocesses. This particular reuse process combines automatic translation, analysis, slicing, reformulation, merging and extension. Comparing this process with our experience, both involved automatic translation. Moreover, they both have involved OKBC translators. Both processes involved analysis, although at different levels. While we have extensively and successively performed analysis activities, to choose the adequate ontologies, to structure the final ontology and to perform quality verification, the analysis described in the merge/integration experience was for comprehension purposes only. Slicing "involves selecting a portion of an input ontology for use in a new application", because one may not need the whole ontology. For instance, if we had chosen to reuse WordNet we would just need the concepts related to time. In our experience we could avoid slicing because the reused ontology is a unique object with well defined boundaries. In our case, we did not need to perform reformulation since the Protégé-2000 knowledge model is OKBC compliant (which was not the case in the merge/integration experience where a more expressive final knowledge model was used). Both processes involved extension activities. In our case, we performed extension to both (1) recover knowledge that was lost in the automatic translation and (2) to introduce knowledge missing in the source ontology that was found important for our application. In the merge/integration experience knowledge was introduced because it was missing.

Finally, the processes described in (Pinto, 1999) are also integration experiences that have led to a methodology (Pinto and Martins, 2001). One of the aims of our experiment was to test how natural this methodology is. For that, we deliberately prevented the actual person performing this process from knowing beforehand the actual details of this methodology. Comparing both experiences, we can see that our process closely follows the proposed methodology. However, there are some differences. We performed evaluation of the reused ontology after importing, while the methodology proposes that evaluation should be performed while choosing the source ontology to be reused. Another difference, was the fact that the actual experiments described in (Pinto, 1999) have not involved such a large variety of subprocesses, namely they avoided translation (although this subprocess is referred in the methodology).

A manual reengineering process is described is (Blázquez et al., 1998). This process involves three steps: (1) abstract a conceptual model of an ontology given its implementation, (2) modify the conceptual model according to some criteria, and (3) implement the new conceptual model. In our case, we only extracted (step 1) the conceptual model of the Simple-Time ontology kept in the Ontolingua Server because we wanted to compare it with the conceptual model of the ontology obtained after the knowledge representation translation subprocess.

A translation process involving the Simple-Time ontology in the Ontolingua Server is described in (Russ et al., 1999). In this case, the export translator of Ontolingua into Loom was used. The authors identified two problems that make the automatic translation process difficult: (1) a mismatch of modeling styles between Ontolingua and Loom and (2) an inference engine bias. In our case, there was no mismatch in modeling style, but there was an inference engine bias, since PAL is a constraint checking mechanism rather than a theorem proving mechanism.

There is an evaluation methodology, OntoClean (Welty and Guarino, 2001), and criteria for evaluation (Gómez-Pérez et al., 1995). OntoClean is a method to correct taxonomies according to notions such as *rigidity*, *identity* and *unity*. These are philosophical notions that are difficult to understand at first. Therefore, in our experience, we performed two technical evaluations according to the criteria proposed by Goméz-Pérez. There are already some tools which automate the evaluation activity, like WebODE (Fernández-Lopéz and Goméz-Pérez, 2002) and the Analyzer of the Ontolingua Server. However, we did not use WebODE because it is not freely available and we did not use the Analyzer of Ontolingua because it only performs a syntactical analysis.

6 CONCLUSIONS AND FUTURE WORK

Ontology building is still an open area of Ontology Engineering. We followed an integration methodology that is general enough to be used in different integration processes. In our experience, we (1) used Natural Language techniques to help knowledge acquisition, (2) used tools to perform automatic translation of an ontology, (3) performed part of a manual reengineering process and (4) performed manual ontology evaluation. Although in the literature there are descriptions of the several subprocesses that were performed in this experience, they have never been all

[17] As far as we know, there is only one methodology proposed for each one of the reuse processes.

combined in one particular ontology building process. One of the aims of this article is to show how they can all be combined. Even the more complex reuse experiences described in the literature only involve some of these subprocesses.

The reuse of an ontology and the use of a development tool were crucial in reducing the amount of effort required to build the Time sub-ontology. The main advantage of reusing an existing ontology was that it gave us an initial conceptualization, which eased the knowledge acquisition stage. As for the tool, the main advantage was the automation of some of the subprocesses, namely knowledge representation translation.

At the moment only one of the modules of ONTO-SD is complete. We plan to proceed with the development of ONTO-SD. We are also going to improve Protégé-2000 OKBC tab plug-in.

REFERENCES

Allen, J. (1984). Towards a General Theory of Action and Time. *Artificial Intelligence*, 23:123–154.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284.

Bl´azquez, M., Fern´andez, M., Garc´ıa-Pinar, J. M., and Gom´ez-P´erez, A. (1998). Building Ontologies at the Knowledge Level Using the Ontology Design Environment. In *Proceedings of the Knowledge Acquisition Workshop, KAW98*.

Chaudhri, V., Stickel, M., Thomere, J., and Waldinger, R. (2000). Using Prior Knowledge: Problems and Solutions. In *AAAI2000 Proceedings*, pages 436–442. AAAI Press.

Chaudri, V., Farquhar, A., Fikes, R., Karp, P., and Rice, J. (1998). Open Knowledge Base Connectivity 2.0.3. Technical Report KSL-98-06, Knowledge Systems Laboratory, Stanford University.

Farquhar, A., Fikes, R., and Rice, J. (1996). The Ontolingua Server: A Tool for Collaborative Ontology Construction. In *Proceedings of the Knowledge Acquisition Workshop, KAW96*.

Fern´andez-Lop´ez, M. and Gom´ez-P´erez, A. (2002). The integration of OntoClean in WebODE. In *Proceedings of EKAW2002 Workshop on Evaluation of Ontology-based Tools*. also alvailable as CEUR Proceedings, Volume 62, `http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-62/`.

Gangemi, A., Pisanelli, D. M., and Steve, G. (1998). Ontology Integration: Experiences with Medical Terminologies. In Guarino, N., editor, *Formal Ontology in Information Systems*, pages 163–178. IOS Press.

G´omez-P´erez, A., Juristo, N., and Pazos, J. (1995). Evaluation and Assessment of the Knowledge Sharing Technology. In Mars, N., editor, *Towards Very Large Knowledge Bases*, pages 289–296. IOS Press.

Noy, N., Sintek, M., Decker, S., Crub´ezy, M., Fergerson, R., and Musen, M. (2001). Creating Semantic Web Contents with Prot´eg´e-2000. *IEEE Intelligent Systems*, 48.

Peralta, D. N. (2002). Desenvolvimento de ontologias. Trabalho Final de Curso, Departamento de Eng. Inform´atica, Instituto Superior T´ecnico.

Pinto, H. S. (1999). Towards Ontology Reuse. In *Proceedings of AAAI99's Workshop on Ontology Management, WS-99-13*, pages 67–73. AAAI Press.

Pinto, H. S., G´omez-P´erez, A., and Martins, J. P. (1999). Some Issues on Ontology Integration. In *Proceedings of IJCAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends*, pages 7.1–7.12.

Pinto, H. S. and Martins, J. P. (2001). A Methodology for Ontology Integration. In *Proceedings of the First International Conference on Knowledge Capture, K-CAP'01*, pages 131–138. ACM Press.

Pinto, H. S., Peralta, D. N., and Mamede, N. J. (2002). Using Prot´eg´e-2000 in Reuse Processes. In *EON2002, Evaluation of Ontology-based Tools*, pages 15–26. CEUR-WS.

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

Welty, C. and Guarino, N. (2001). Supporting Ontological Analysis of Taxonomic Relationships. *Data and Knowledge Engineering, September 2001*.