Adapting general linguistic knowledge to applications in order to obtain friendly and efficient NL interfaces

M. Gatius and H. Rodríguez
Universitat Politècnica de Catalunya
e-mail:
gatius@lsi.upc.es, horacio@lsi.upc.es

Abstract:

This work, supported by the project ITEM[1] and the project BASURDE[2], proposes a mechanism for adapting general linguistic knowledge to applications in order to obtain automatically application-restricted NL interfaces. The knowledge involved in the process is represented in separate data structures: a conceptual ontology, a linguistic ontology, a general lexicon and a set of control rules. The general and application-dependent knowledge relevant to the communication tasks is represented in a conceptual ontology. The general linguistic information is represented in a linguistic ontology. The core of our approach consists of describing the application elements (both entities and operations) relevant to the communicative tasks in terms of the existing conceptual ontology. Lexical coverage of these elements has to be provided as well. A general basic set of control rules will then relate the application specifications to the linguistic ontology in order to obtain application-restricted NL interfaces. The system has been applied to provide NL interaction to SIREDOJ, an already existing expert system in the legal domain.

1 Introduction

Natural language (NL) is generally considered one of the most suitable modes of communication. It seems an appropriate mode especially for interaction with knowledge base systems (KBSs) due to the complexity and diversity of the concepts and actions involved in the communication process. However, few systems incorporate natural language interfaces (NLIs). The main reasons are the high cost of the development and maintenance of the linguistic knowledge sources needed, the large space and run-time requirements and the problem of the user lack of knowledge about the system capabilities. For dealing with the first two problems, an automatic procedure for adapting general linguistic knowledge to applications in order to obtain the NLIs is proposed. For dealing with the second, a restricted menu-based form of communication, guiding the user to introduce the correct utterances, is incorporated. The generated interfaces, once included in the overall application, could be responsible for all the communicative tasks between the user and the application.

The use of large coverage linguistic resources for specific domain-restricted applications (as NL interfaces) has proved generally unsatisfactory due to the space and run-time requirements. Application-restricted grammars improve efficiency in language processing but are expensive to develop and difficult to reuse. Several ways have been attempted to reduce the cost of creating application-dependent grammars. Most approaches adapt general linguistic resources to specific applications in order to obtain application-tuned grammars. The coverage of these grammars depends on the application domain and the complexity of the communicative tasks. The process of tuning general linguistic resources to applications can be performed by generating a specific application subgrammar or by providing the more general grammar of a dynamic mechanism to restrict the grammatical options at run-time. The cost of generating application-tuned subgrammars is reduced when done automatically, as is the case in some recent works, such as Henschel and Bateman [7]. Using dynamic mechanisms is also an efficient way to restrict grammars, as is proved in the dynamic rule pruning mechanism, described in [3]. This pruning mechanism uses information available at run-time to reduce the grammatical options that must be considered.

In this paper, a general mechanism for obtaining application-restricted grammars and lexicons from application specification and general linguistic resources is described. Both mentioned strategies in adapting general linguistic resources have been considered in order to obtain the specific grammar for each application. Application-tuned subgrammars and lexicons are generated from general sources and

[1] The development of the project ITEM (Textual Information Retrieval in a multilingual environment using NL Techniques) has been supported by the Interministerial Science Technology Comission (CICYT) and registered as TIC96-1234-C03-02.

[2] The development of the project BASURDE (Spontaneus-Speech Dialogue System In Limited Domains) has been supported by the CICYT and registered as TIC98-423-C06-06.

incorporate dynamic restriction mechanisms. The process of obtaining the subgrammars is based on the performance of a set of control rules relating the application knowledge to the general linguistic knowledge.

A study of an efficient representation of the different types of knowledge relevant in the communication and its reusability in different applications has been done. An architecture organizing this knowledge in four separate data structures has been designed. Two ontologies have been used to enable the conceptual and linguistic knowledge sharing and reuse. The general knowledge involved in the communication process is described in a conceptual ontology (CO). This general knowledge is the skeleton for anchoring the domain and functional application knowledge. General linguistic information needed to cover the linguistic realization of communication is represented in a linguistic ontology (LO) and a general lexicon (GL). Finally, the control information to relate application knowledge to linguistic knowledge in order to generate the application-restricted interfaces is described by a set of production rules.

Building an interface requires the representation of the application specifications in terms of the CO and the incorporation of the new application-dependent lexical entries to the general lexicon. The general control rules are then applied over the CO and the LO in order to generate the interface grammar and lexicon. Although there is a basic set of control rules providing the basic linguistic coverage, the user is able to extend this coverage adding (or modifying) the basic control set and enriching the LO. The process of obtaining an application-restricted interface is described in Figure 1.

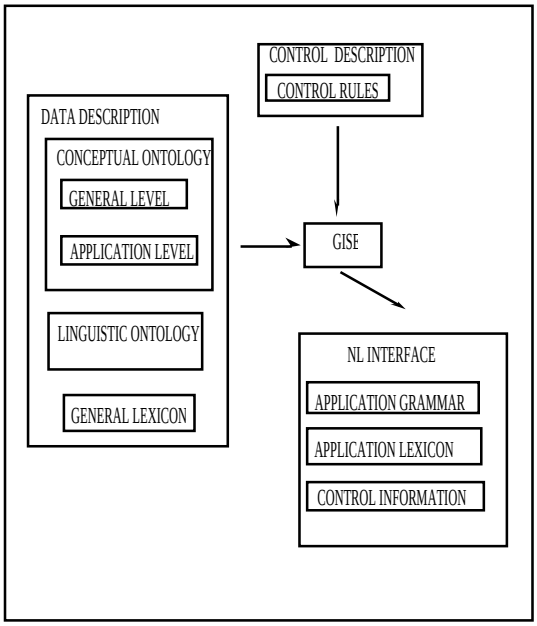

Figure 1: The process of obtaining an application-restricted interface

GISE (*Generador de Interfaces a Sistemas Expertos*), a system incorporating the proposed knowledge organization has been designed. This system generates domain-restricted task-guided NL-interfaces from expert systems specifications represented in the CO and the LO. The generated interfaces use an unambiguous and concise sublanguage adapted to the functionality of the expert systems (ESs) and regarding efficient guidance about system capabilities. A menu-based mode, similar to the used in [9], is also integrated into the interface. During the communication process, all NL available options are displayed in windows and the user chooses from among those options to incrementally construct a complete statement. A prototype of the system was implemented and applied to an expert system in law, SIREDOJ.

2 Representing the general conceptual and linguistic knowledge involved in the NL communication

As described in the introduction, in the proposed architecture the linguistic sources needed in the communication process are obtained from the application knowledge represented in the CO and the general linguistic knowledge represented in the LO. Only the concepts relevant to the communication process and their natural language expression are included in the conceptual and linguistic ontologies. This knowledge covers the concepts and relations describing the application and operations over these concepts.

The CO is organized on two levels: the general level and the application level. The general level describes the basic concepts, attributes and operations common to all applications. In the application level, application specifications are represented on the basis of the general level. The application information is organized, following [6] and [10], in domain ontology and process ontology. Two sublevels are distinguished on application level: the application description level, where all application information is represented, and the case level, where information about specific cases are represented. During the communication process, only the case level is modified.

The basic classes in the CO general level are concepts, attributes describing concepts and operations. These three basic objects are organized in separate taxonomies.

An important problem in ontologies for NL processing is the representation of the general relations between conceptual knowledge and its expression in NL. Many of the ontologies present in these systems are mixed ontologies, where additional specific linguistic information is incorporated into the conceptual representation. A major problem in mixed ontologies is that conceptually motivated and

linguistically motivated classes are mixed in a taxonomy. To avoid this problem, in the proposed CO, the concept classification is only conceptually motivated, and the attributes describing the concepts are classified according to their linguistic behaviour. Only the linguistic coverage of the operations of the ontology is considered. Because these operations are concept-based operations, considering the linguistic realization of the attributes describing the concept is very important.

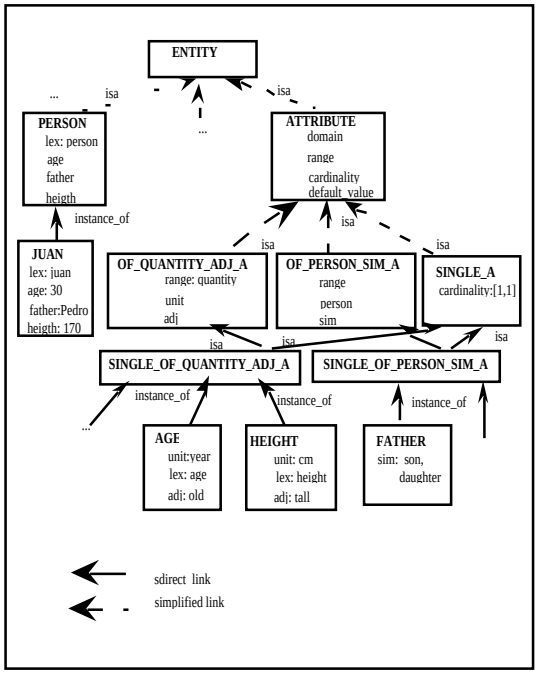

Figure 2: A fragment of the conceptual ontology

The attribute classification makes possible a variety of linguistic coverage for each attribute class. This classification is reusable. The basic attribute taxonomy was obtained by considering existing classifications of attributes of concepts, such as that by Perkins [8] and the Penman Upper Model [2], and by means of an empirical evaluation of the utterances used in different ES applications. All the attribute classes distinguished are necessary to reflect different surface realizations. These basic attribute classes are associated with grammatical roles: participants (**who_does**, **who_object**, **what_object)**, being (**is**, includes identifying and attributive), possession (**has**), circumstances and other relationships between two or more objects (**of)** and related processes (**does**). Subclasses are obtained from basic classes considering other information relevant for the linguistic realization of attributes. For example, the class **of_quantity**, a subclass of the class attribute **of**, is used to describe attributes referring to quantities. Attributes in this class always involve the use of a unit of measure. The class **of_quantity_adj**, a subclass of the class attribute **of_quantity**, represents attributes having an associated adjective (in Spanish these attributes have an associated verb instead of an adjective). Compound attributes are also considered. For instance, the class **is_subject,** is a subclass of the class **is** representing attributes composed of two words: the first word representing an entity describing the concept and the second representing a property or state of this entity.

The taxonomy of attributes is based on Spanish linguistic distinctions. However, it is intended to be reusable across several languages because most linguistic considerations in classifying attributes are relevant in other languages. An example adapted to English, given in Figure 2, shows how linguistic information is represented in the taxonomy of attributes. Only the relevant classes and instances for a particular example are represented in the figure. The plain arrows represent a link between a class (or instance) and its direct upper class. The dash arrows represent a link between a class and an upper class indicating that the direct upper class is not shown in the figure.

The values of the attributes of concepts are classify in closed values, open values, menu values (values consisting of a set of predefined options that will be displayed in the screen at run-time) and values representing instances of concepts.

All operations performed over the conceptual objects are represented as objects in the CO and organized in a taxonomy. Description of operations includes both the signature (number and class of the operands) and the conditions of applicability (preconditions). Two types of preconditions are considered: class preconditions, conditions to be applied during the generation phase and case preconditions, conditions to be applied at run-time.

The linguistic information covering the communication with different types of applications is represented in the LO. Only the sublanguage necessary to express the CO operations is considered. In LO, objects representing linguistic classes are assumed to be common to all applications. Objects representing the specific aspects of the information to be expressed for each application are represented as instances of the linguistic classes.

The basic linguistic classes in the LO were obtained from the Penman Upper Model ontology. Linguistic classes were organized depending on their rank. The top class is **rank**. Within the top class, three main subclasses were defined: the class **clause** (having a subject and a finite verb), the class **group** (having a head and a variable number of modifiers) and the class **word** (representing verbs, nouns, articles, etc). A fragment of the LO is represented in Figure 3.

The linguistic classes having only one constituent are classified as **terminal** classes, and those having more than one constituent as **non-terminal** classes. The instances of terminal classes represent lexical entries and the instances of non-terminal classes grammar rules. The terminal classes are further subclassified in **closed** and **open** classes. Closed

classes represent always the same lexical entries for all applications. Examples of these classes are the class **article** and the class **preposition**. Open classes, such as the class **noun**, represent a wide range of lexical entries, different for each application.

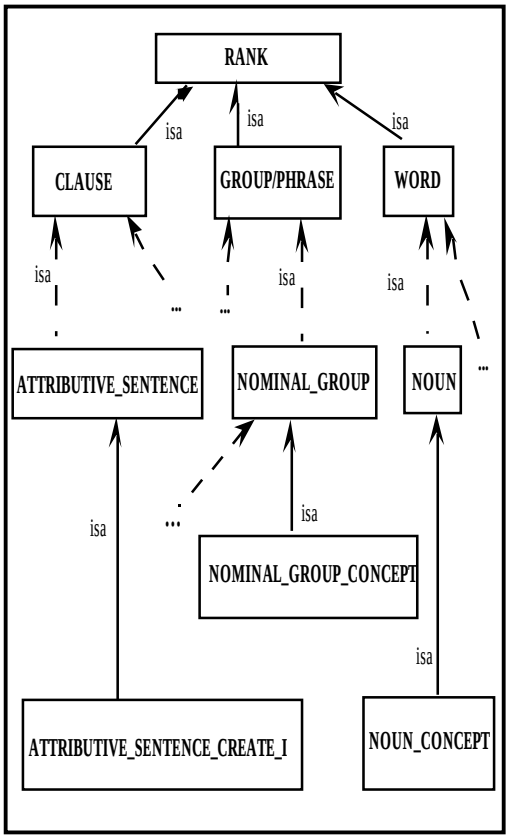

Figure 3: A fragment of the linguistic ontology

LO classes and instances are described by a set of attributes. These attributes represent the syntactic and semantic information associated with each class. In Figure 4, the description of the class **attributive_sentence** is shown. Each linguistic class has an associated linguistic category, represented in the attribute **cat**. The sequence of the syntactic constituents of each class is represented in the attribute **cset** (constituent set). For reasons of efficiency, the information on the superficial presentations of the class is represented by means of a different attribute, named **pattern**. The value of the attribute **pattern** is the set of the presentations allowed for the constituents of a class. Information for further semantic interpretation is represented by numbers indicating the order of interpretation of the constituents associated with each possible pattern. Each constituent is represented by an attribute whose value belongs to an existing linguistic class.

The syntactic and semantic agreement between the constituents of a linguistic class is represented by the attribute **agreement**. The value of this attribute is a list in which constituents are associated with its syntactic and semantic features. The semantic information is based on a typified lambda calculus. In terminal classes, the semantic interpretation consists of a lambda function and is represented by an attribute. Functional information associated with the class is represented in the attributes **simple**, **polarity, mood, voice** and **theme.** The range of these attributes is a closed set of values.

As can be seen in the Figure 4, the value of the attribute **cset** of the class **attributive_sentence** is the sequence of its constituents: **subject**, **neg**, **verb** and **attribute.**

There is an attribute representing each of these constituents, its value being a linguistic class belonging to the LO. The linguistic classes **nominal_like_group** and **nominal_group** belong to the class **group**. The linguistic classes **copulative_verb** and **no** are terminal closed classes, representing always the same lexical entries. The attribute **pattern** represents all the possible superficial representations of an attributive sentence, <**subject neg verb attribute**> and <**subject verb attribute**> are being two of them. The numbers associated with each pattern give the order of interpretation of the constituents. The attribute **agreement** indicates that the number of the constituent **subject** and the number of the constituent **verb** must be the same. The value of the attribute **simple** is **yes/no**, indicating that the attributive sentences can be simple or complex. The attribute **polarity** indicates that the elements belonging to the class can be expressed both in a positive and in a negative form. The value of the attribute **mood** is **indicative/interrogative**, indicating that the modality of an attributive sentence can be indicative and interrogative. Finally, the attribute **voice** indicates that only the value **active** is allowed for attributive sentences.

ATTRIBUTIVE_SENTENCE

cat: attributive_sentence
cset: subject neg verb attribute
pattern:
(((subject verb neg attribute)(3 ((2 1) 4)))
((subject verb attribute)((2 1) 3))...)
subject: **nominal_group**
verb: **copulative_verb**
attribute: **nominal_like_group**
neg: **no**
agreement: subject (num) verb (num)
simple: yes / no
polarity: positive / negative
mood: declarative / interrogative
voice: active

*Figure 4: The class **attributive_sentence***

In terminal instances, the lexical semantic interpretation, the lexical realization and the category are represented in attributes. For each application, instances of the linguistic classes necessary to represent CO concepts and operations are created. Instances of the class **clause** are created to represent the NL expression of CO operations. Application concepts and their attributes are represented in instances belonging to the class **group**. For each application, the non-terminal instances are represented as the interface grammar rules and the terminal instances as the interface lexical entries.

The grammar generated is represented as a Definite Clause Grammar (DCG) augmented with syntactic and semantic features. Each grammar rule incorporates semantic information indicating the interpretative order of its components. This information is based on a typified lambda calculus. Dynamic mechanisms were incorporated to adapt these resources dynamically to the application performance during the communication process. These mechanisms improve the functionality of the interface by reducing the space and time for processing user interventions. The needs of these mechanisms depend on the application complexity and size. The dynamic mechanisms are implemented as preconditions attached to the grammar rules that are evaluated at run-time and categories whose value is set during the communication process.

Grammar preconditions are obtained from the case preconditions represented in the operation classes. For example, the operations to modify an instance have an associated precondition governing the existence of the instance. When performing the linguistic realization of these operations, this precondition is attached to the corresponding grammar rules. The grammar preconditions are evaluated dynamically, restricting the grammar rules to be considered to the ones representing the possible operations at each step of the communication process.

Dynamic categories were incorporated into the generated grammar to obtain the proper values associated with these categories at run-time. Two different types of dynamic categories are distinguished: dynamic categories representing instances of concepts generated at run-time and those representing a proper noun and a number that will be introduced by the user at run-time.

3 Relating the conceptual ontology to the linguistic ontology in order to obtain application-restricted interfaces

One of the most important goals in this work is the study of how general relations between two ontologies representing different types of knowledge can be established. This study have been focused on the relations between the application specifications represented in the CO, described in previous section, and the general linguistic knowledge represented in the LO. A general process relating the CO to the LO in order to obtain application-restricted NL interfaces is described in this section.

The general scheme of the process of generating the interface linguistic sources is described in Figure 5.

```
for each CONCEPT in APPLICATION_ONTOLOGY do
 generate_CO_instance_operations_modifying_concept_instance (CONCEPT)
 generate_CO_instance_operations_consulting_concept_instance (CONCEPT)
endfor
for each OPERATION_INSTANCE in CASE_ONTOLOGY do
 generate_LO_non_terminal_instances (OPERATION_INSTANCE)
 for each ARGUMENT in OPERATION_INSTANCE do
  generate_LO_terminal_instances (OPERATION_INSTANCE,ARGUMENT)
 endfor
endfor
obtaining_grammar_rules_from_LO_non_terminal_instances
obtaining_lexical_entries_from_LO_terminal_instances
```

Figure 5: The general scheme of the generation procedure

Basically, this process consists of relating the possible operations for an application represented in the CO to the LO. The process is performed in three steps. The first step consists of generating instances of the CO operations for the application concepts represented in the ontology. Instances of operations creating, modifying and consulting instances are generated for each ontology concept. Different operations are generated considering the classes of the concepts and the attributes.

The second step of the generating process consists of creating the LO instances supporting the expression of the operations generated in the first step. For each operation, one or more instances of the class **clause** and instances of the class **group** are created. Finally, the third step consists of representing the LO instances created in the second step as DCG rules and lexical entries. The linguistic instances expressing the conceptual operations are represented as interface grammar rules. The linguistic instances representing the operation arguments (these are the concepts and their attributes) are represented as the interface lexical entries.

The first two steps of the process are performed by a set of control rules. Rules are of the form: **conditions --> actions**. Conditions basically consist of descriptions of objects. Rules are applied over objects in the CO and the LO satisfying required descriptions. The actions performed by the rules are operations consulting and modifying the objects in the CO and LO[3] . Rules are grouped in rulesets. Each

[3] *For the sake of effectiveness, ontologies objects are represented in an active data storage device, the working memory, where these objects can be consulted, created and deleted.*

ruleset performs a different action and each rule in the ruleset considers a different type of object. A mechanism controlling the activation of the rulesets and the rules in the rulesets has been incorporated.

Control rules are implemented in the Production Rules Environment (PRE), a rule-oriented environment specially built for NL, described in [1]. It incorporates the capabilities necessary for the control rules performance: the use of rulesets, a powerful and flexible control mechanism and a working memory where objects can be created, consulted and modified. In PRE the rulesets are organized in a multilevel hierarchy allowing inheritance. Each ruleset ensures the activation of the rules that belong to it. The top of the hierarchy is the TOP ruleset. This ruleset must be present in any PRE application, whilst other rulesets are optional. All rules in PRE consist of two sets: the condition set and the action set. The condition set consists of all the statements describing the conditions governing the application of the rule[4] . The action set consists of all the actions to be performed when applying the rule. Each object in the working memory (WM) is described by a word describing the type of the object and a variable number of attribute-value pairs. Only four operations are used in the action part of the rule: creation and deletion of objects, activation of a rule set and assignment of values to variables. The assignment operation allows the use of LISP functions. In the general set of control rules designed and implemented to generate NL interfaces for different applications, only six different LISP functions are used.

An example of a control rule, implemented in PRE, is shown in Figure 6. This rule ensures the generation of an instance of the operation **create-instance-with-no-name** (**ocinn**) for each concept in the CO. Previously to the application of this rule, the CO concepts had been represented in the working memory. Each CO concept is described by the word **object**, its name (represented by the attribute **^name** and the variable **?name**, with the concept name) and the preconditions associated with the concept (**^pcobject ?pcc**). This rule is applied over all WM objects satisfying this description. The action part of the rule consists of five statements. In the first statement, the result of the function **create-name,** concatenating the name of the operation **ocnni** to the concept name (represented in the variable **?name**), is assigned the to the variable **?op**. The second statement creates an instance of the CO operation **ocnni** by calling to the predicate **create-object.** The third statement calls to the function **add-slots** to fill the instance attributes **concept** and **pcc**. A WM object representing the generated instance is created in the fourth statement. Finally, the last statement uses the operator **delete** to remove the WM object representing the CO concept that has been previously matched.

The user can define different control rules considering different levels of linguistic coverage. Current implementation makes possible, through the performance of the appropriate control rules, the generation of grammars covering ellipsis, subordinate clauses, a variety of anaphoric references and other linguistic phenomenon. However, to define a new rule it is necessary to know the rules formalism as well as the details about the CO and LO implementation. In order to make the implementation of the system transparent to the user and facilitate its use, a general basic set of control rules for generating application-restricted NLIs was defined and implemented. This basic set of rules establish general relations between the CO and the LO in order to obtain the linguistic resources necessary for application-restricted interfaces. The same basic set of control rules can be used for different types of applications.

Three different types of interfaces have been taken into account when defining the basic set. These types are interfaces guiding the users to describe particular cases of the application general knowledge, interfaces allowing the users to consult the application knowledge, and interfaces supporting both, consults and descriptions.

The rules in the basic set have been revised to generate the minimum number of grammar rules to express, in a natural way, all possible operations over the CO. Several linguistic realizations for the same operation are considered. Current implementation ensures the generation of grammars covering ellipsis, coordination, direct and indirect interrogation and other linguistic phenomenon. The basic set of control rules has been applied to an expert system in law for generating an application-restricted NL interface. This set is described bellow.

```
(rule instance_operation
 ruleset creating_instance
 priority 1
 control  forever
 (object ^name ?name ^pcobject ?pcc)
 ->
 (?op := (create-name 'ocinn ?name))
 (?COop := (create-object ?op 'ocinn))
 (?opsnarg := (add-slots ?op '((con ?name) (pcc ?pcc))))
 (create ocisn ^name ?op ^concept ?name ^pcobject ?pcc)
 (delete 1) )
```

*Figure 6: The rule i**nstance_operation***

[4] *In addition, other features are provided in the condition set describing the rule. These are: rule, identifying the rule; ruleset, identifying the ruleset to which the rule belongs; control, stating the rule application mode; priority, controlling the application rule order within the rule set and the statements establishing the conditions governing the working memory objects to apply the rule.*

In the basic control rules proposed, rules are grouped in 8 rulesets. The first ruleset is the TOP ruleset, responsible for the initialization process. This ruleset checks the initial conditions, describing the type of the interface to be generated, and activates the appropriate ruleset for each case. Four different rulesets are in charge of the first step of the process, when instances of CO operations are created. These rulesets are: the ruleset **creating_instance**, the ruleset **modifying_instance**,the ruleset **consulting_instance** and the ruleset **consulting_attribute**. The set **creating_instance** is responsible for generating instances of the operations creating case ontology objects: the operation **create-instance-with-no-name**, for creating anonymous instances and the operation **create-instance-with-name**, for creating instances with a proper noun. This set is applied over all WM objects representing the CO concepts. For each concept, instances of the two operations are created in the case ontology as well as in the WM. A simplification of the rule **instance_operation**, belonging to this ruleset is described in the Figure 6. The ruleset **modifying_instance** is responsible for generating instances of the operations modifying case ontology objects.

Three different rulesets are responsible for the second step of the process, when instances of LO objects supporting the NL expression of the application operations are created. These rulesets are the ruleset **grammar**, the ruleset **argument**, and the ruleset **lexical_entries**. The first ruleset performs the creation of the LO instances representing the different operations generated for an application. The ruleset **argument** ensures the different types of operation parameters. The ruleset **lexical_entries** performs the creation of the LO terminal instances representing the operation parameters. These instances correspond to the interface lexical entries.

The linguistic instances created in this step of the process incorporate conceptual information related to the possible operations for the application. For each operation class, several linguistic instances are created. The operation arguments are represented as the constituents of the LO instances generated. The operation preconditions are represented as the dynamic conditions associated with these linguistic instances. The syntactic information is inherited from the linguistic instances classes. This information consists of the pattern (all possible presentations of the constituents) and the syntactic and semantic agreement between the constituents. These linguistic instances are represented as DCG rules.

Terminal linguistic instances representing each operation argument are generated. In this process, the different types of arguments are considered. The types of arguments distinguished are concepts (classes and instances), attributes (represented by simples names or compound names) and values (close values, open values, menu values and values representing concept instances).

The four attributes representing each interface lexical entry, the string, the category, the semantic interpretation and the general linguistic class the category belongs, are described in terminal instances. The semantic information, semantic restrictions associated with the category and the semantic interpretation, is obtained from the operation arguments. The syntactic information, the category, the syntactic restrictions (number and gender) and the string, is obtained from the general lexicon.

Control rules considering different types of user (casual or habitual, expert or novice) can also be incorporated to the basic set. New rules can also be added to the basic set when new information in relating conceptual knowledge to linguistic knowledge has to be considered. Different control rules can be created to encode the conceptual information representing the application functionality in the grammar rules and lexical categories, following different criteria.

4 Applying the proposed architecture to an expert system on law

The proposed organization has been applied to an expert system on law specializing in building agreements, SIREDOJ (Intelligent System for Legal Information Retrieval). The basic function of the system is to classify the legal cases introduced by the user according to the knowledge base of the system in order to obtain the legal conclusions. In its first review, SIREDOJ was originally built as a monolithic system with communicative and functional tasks fully integrated. By incorporating the described knowledge organization and thus controlling the user-application communication, major improvements have been achieved in efficiency and friendliness.

In order to obtain the appropriate NL-interface for SIREDOJ, the expert system has been represented on the basis of the CO general level. 33 concepts (15 belonging to the general level and 18 to the application level) and 31 different attributes are required to represent the expert system. All these attributes have been classified according to the basic attribute taxonomy. The 31 attributes used in the application representation belong to 7 different classes. 10 different menus representing values of concept attributes are also defined in the CO. New lexical entries referring to all application concepts and attributes are incorporated into the general lexicon. 183 classes are distinguished in the implemented general LO. 23 of these classes are closed classes (articles, prepositions, conjunctions and auxiliary verbs) and 160 are open classes. The control rules described in previous section are applied over the CO and LO in order to generate the interface linguistic resources. Using the current implementation of the basic set of control rules, only

the 23 closed classes and 17 open classes of the LO are used for generating the linguistic resources needed in the interface to SIREDOJ. The generated grammar contains 28 rules and the generated lexicon 96 lexical entries. The linguistic coverage considered corresponds to all information the users were asked for by the system in previous interface. The generated interface guides the user to introduce NL sentences containing any required information by the system in any state. Information requiring a long menu chain in the original ES interface can be expressed in one simple sentence in the new interface.

5 Conclusions

As was pointed out in the introduction, the natural language mode is an appropriate mode for person-machine communication because NL is a friendly mode for expressing complex and diverse knowledge. There are, however, major problems to be solved in NL communication. This work is devoted to providing solutions to these problems and thus improving efficiency in the development and performance of NLIs. For this reason, a mechanism based on the performance of a set of control rules relating general linguistic knowledge to the applications specifications in order to obtain automatically the NL best suited for each application, is proposed. The representation of the relevant knowledge in the process of generating an interface in separate data structures (conceptual ontology, linguistic ontology, general lexicon and control rules) gives a great flexibility in adapting linguistic resources to different applications and users. Reusing the general knowledge bases and control rules reduces the high cost of the process of developing NL interfaces. Encoding application domain and functionality in the generated grammar and lexicon plus guiding of the user to introduce the required information, substantially improves the functionality of the interface. Dynamic mechanisms have been incorporated to use information available at run-time in order to reduce dynamically the number of interface grammar rules and lexical entries that must be considered. As a result, the space and time for processing user interventions is not large, even when the meanings expressed are complex.

Applying the designed architecture to an expert system in law has proved how interfaces to KBSs can improve their functionality by incorporating NL. Once the main shortcomings inherent in the use of NL have been solved, the semantic complexity and friendliness of the NL mode can improve communication. Following the proposed generation process, the cost of creating an application-restricted grammar is reduced to the cost of adapting the applications specifications to the general CO plus introducing the application vocabulary in the general lexicon. The process of representing the application knowledge in the CO could be simplified by the incorporation of various aids. In particular, guidance in adapting the representation of the domain and functionality of the application to the CO would facilitate this task.

References

[1] Ageno A., Ribas F., Rigau G., Rodríguez H., Verdejo F. “TGE: Tlinks Generation Environment.” Acquilex II .WP Num. 8, UPC, Dept. LSI, 1993.

[2] Bateman J. et alt. “A General Organization of Knowledge for Natural Language Processing: the Penman Upper Model”. Penman Development Note, USC/Information Sciences Institute, 1990.

[3] Dowing, J. and Hirschman, L. (1988) “A Dynamic Translator for Rule Pruning in Restriction Grammar”. V. Dahl and P. Saint-Dizier (eds.) “Natural Language Understanding and Logic Programming, II. Elsevier Science Publishers B.V. (North-Holland), 1988

[4] Gatius M., Rodríguez H. “Automatically generating linguistic knowledge sources from application specifications”. Proceedings of The Cognitive Science of Natural Language Processing (CSNLP). Dublin, 1994.

[5] Gatius M., Rodríguez H. “A domain-restricted task-guided Natural Language Interface Generator.” Proceedings of the second edition of the workshop Flexible Query Answering Systems 96 (FQAS’96). Roskilde University (Danmark), 1996.

[6] Gómez-Pérez, A. et alt. “Towards a Method to Conceptualize Domain Ontologies”. Proceedings of the ECAI workshop on Ontological Engineering. Budapest, 1996.

[7] Henshel, R. and Bateman J. “Application-driven automatic subgrammar extraction”. ACL workshop on Computational Environments for Grammar Development and Linguistic Engineering. Madrid, 1997.

[8] Perkins, W.A.: “Generation of Natural Language from information in a frame structure” Data & Knowledge Engineering Vol. 4, pp 101-114, 1989.

[9] Thomson, G. “ Menu based N.L. interfaces to DB”. *Database Engineering*, 1985.

[10] Yen J., Lee J. “A Task-Based Methodology for Specifying Expert System”. IEEE EXPERT, Vol. 8, pp 8-15, n_1 1993