

A domain-restricted task-guided Natural Language Interface Generator

Gatius, Marta. & Rodriguez, Horacio
Universitat Politècnica de Catalunya
email: horacio@lsi.upc.es, gatius@lsi.upc.es

1 Motivation and introduction

Considerable attention has been paid in recent years to the subject of Natural Language Interfaces (NLI, see [4] for a survey on current trends in such technology). Starting with NL access to databases, NLI technology has evolved according to the needs of the different sectors involved. This evolution has followed several lines, improving both the quality and the capabilities of the interface functionality and the engineering features of the development process: the extension of the conceptual and linguistic coverage in order to deal with more complex domains and applications, the improvement of the friendliness of Person/Machine interaction, the integration of different modes of interaction leading to the multimodal/multimedia interfaces, the construction of transportable and integrated/modular systems, the reusing of software and linguistic components, etc.

Despite these efforts, the introduction of NLI technology as a component of a Software Application has been quite slow and has not reached the level predicted.

We think that there are three main reasons for this fact:

- From the end-user point of view, many experiences show that the main problem is that casual users lack knowledge about the limits of system capabilities (and we must take into account that casual users are the main potential market for NLI technology).
- From the developers point of view, the main problem is the cost of development, tailoring and maintenance of linguistic Knowledge Sources (KS). Although developing linguistic KS is expensive, it can be done by experts during the development phase. Tailoring and maintenance must be done at the exploitation or installation phase, usually by non-expert

people, and is therefore a supplementary problem.

- A final problem is related to the way interfaces are integrated into the whole computer application. Recent trends in software design methodology favour the strict separation in analysis, design and programming between communicative and functional modules. User Interface Management Systems (UIMS) and other CASE tools have been developed for dealing with the construction and maintenance of communicative modules. Unfortunately, such methodologies and tools do not include NLI development facilities and are not well suited dealing with knowledge-intensive software applications where extremely complex communicative relationships arise, producing a highly coupled communicative/functional modular structure.

In fact, the key point for these problems deals with the existence of different KSs involved in the interface performance. These sources differ in their content, both linguistic and domain and application-depending, in their size, in the way they are acquired and in their level of generality, i.e. in their different possibility to be reused with minor changes in different environments. Our claim is that, for the sake of effectiveness of the interface performance, dialogs must be constrained by the domain and the computer application.

Our approach consists of organising our knowledge in the following way: Information and Control are described declaratively and organised into separated data structures. Two main chunks of information, conceptual and linguistic, are considered. Conceptual information includes both domain depending and application oriented data. Linguistic information includes both grammatical (syntactic and semantic) and lexical data. Control includes the rules for performing the mapping between the different knowledge sources as well as the rules for generating the knowledge content of the generated interface. An overall view of our system architecture is shown in *Figure 1*.

A central issue of our approach is the specification in a declarative way not only of the domain but also of the system's functional units (tasks). We follow for this purpose the Task-Based Specification Methodology (**TBSM**) presented in [11]. In such methodology for system modelling, the conceptual information is composed by a **Model Specification**, including a **Domain Model** and a **State Model** and a **Process Specification** including functional and behavioural issues. All these aspects are covered by our Conceptual Model and are related

to their linguistic realization.

Building an interface implies an incremental acquisition process of the different knowledge sources needed for the interface performance. Some of these sources are quite general while others depend on the specific domain and task. These acquisition operations involved in the process of development of an interface can, thus, be grouped in three steps:

- First step. Building and maintenance of the **General Linguistic Knowledge** and a **General Conceptual Knowledge**. **General Linguistic Knowledge** includes a **Linguistic Ontology** that covers the syntactic and semantic information needed to generate the specific grammars and a **General Lexicon** that includes functional and domain and application independent lexical entries. **General Conceptual Knowledge** includes both the domain and application independent conceptual information and the meta-knowledge that will be needed in the following steps.
- Second step. Definition of the application in terms of the Conceptual Ontology (CO). When building a domain-constrained application-guided interface, both the **domain** description and the **task structure** description must be built and linked to the appropriate components of the CO.
- Third step. Definition of the Control structure. It includes the metarules for mapping objects in the Domain Ontology with those in the Task Ontology, the metarules for mapping the CO onto the Linguistic Ontology (LO) and those for allowing the generation of the specific interface knowledge sources, mainly the grammar and the lexicon.

In fact, as we will see in next section, a last modification of the linguistic sources is performed dynamically at run time for adapting themselves to the changing environment.

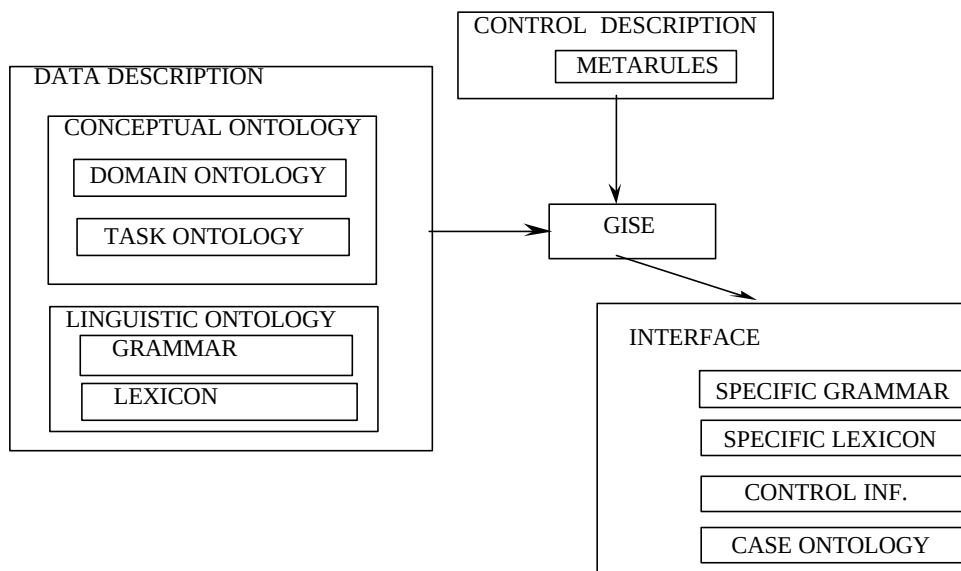


Figure 1: Overview of GISE

This methodological approach has been followed in our system GISE, that generates NL interfaces for heuristic classification Expert Systems.

Section 2 presents an overview of GISE (the system is described with more

detail in [5] and[6]). This paper focuses on GISE functionality rather than its architecture, specially on the control mechanisms that allow the generation of an specific interface taking into account the domain-constraints and the application-dependent tasks. Sections 3 and 4 present the main information structures while section 5 is devoted to the formalisation of control mechanisms. A detailed example of the performance of the system is presented in section 6. Section 7, finally, states the conclusions of our work.

2 Overview of GISE

GISE has been designed to provide NL access to a variety of Expert Systems (ES). ES represent a class of potential applications for NL interaction with specific problems, due to the characteristics of the dialogs involved (mixed initiative, explanation) and the lack of formalisation of the target system (in contrast, for instance, with the relational database model).

GISE was designed for communicating in Spanish and Catalan but can be transported to other languages with limited effort, modifying the Linguistic Ontology and some of the control metarules¹.

Efforts must be made at the knowledge level to build an explicit model of ES performance in order to interact with it. Our challenge in GISE is to define such a model without changing the internal operations of the ES.

The class of ES we have worked with belongs to a well known ES family: those performing heuristic classification, in which communication with users relies on an Ontology, i.e., a collection of objects owning properties and related to each other by means of different kinds of relationships. At least at a conceptual level, the Ontology represents the basis for modelling both the application domain and the ES performance.

If the CO establishes the model, a Case Ontology, i.e. a collection of instances of classes belonging to the Conceptual Ontology, will establish the framework for communication.

System interventions in dialog, consist of describing parts of such an ontology

¹ GISE was initially built for communicating in Spanish. Its transportation to Catalan was quite easy due to the closeness of the two languages.

or querying the user about unknown or not well established facts or properties of them. User interventions, on the other hand, are contributions to enriching the ontology or queries for a (partial) description of it.

The main task of GISE is the generation of NLI's able to support dialogues around the Ontology. Once generated, the specific interface will allow both the user and the system to create or to remove instances of classes of the Ontology, to fill their slots, to connect instances in different ways, to query about objects existence, their properties and relationships, and so on. A main constraint of GISE is the need to limit communications to changes in the CO.

The control, described by means of a set of metarules is defined declaratively and is separated from data. The domain-specific and application-dependent information, as well as the development process, are described in terms of the General Ontology, provided in step 1, that includes the General Conceptual Knowledge and the LO.

Figure 2 shows a possible architecture for a generated interface (other configurations can be built as well)². Three entities participate in the communication process: The User, the Interface and the Application. The interface owns its specific linguistic knowledge, provided by GISE and it shares the conceptual Knowledge with the computer application.

² Although we limit this paper to the generation of the linguistic components of the interface, other components, configuring the whole architecture of the interface are generated as well.

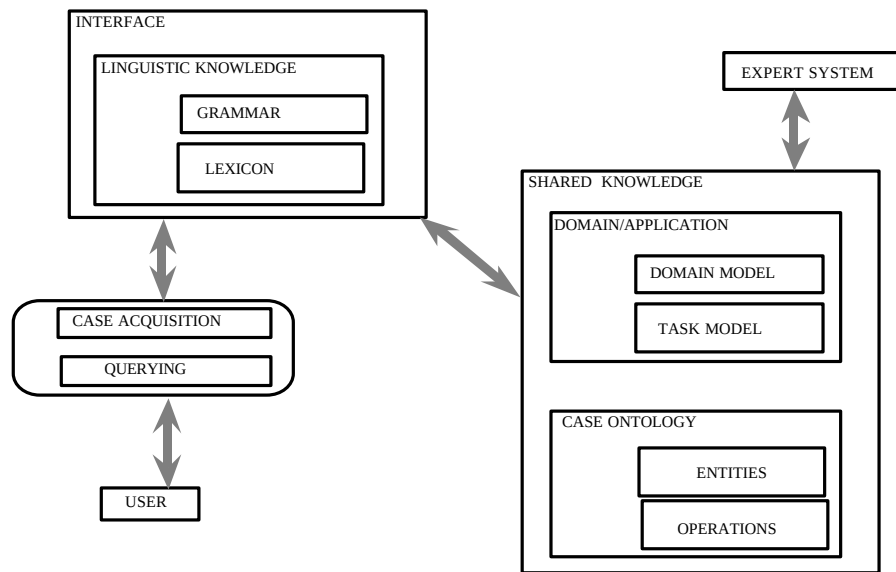


Figure 2 : Architecture of an Interface

The Case Ontology is built step by step while the communicative acts proceed. The initiative of such communicative acts is mixed. The user can provide the information spontaneously or as a result of a system request. The Case Ontology can also be modified by the computer application as a result of its functional behaviour. Aspects of the Case Ontology can also be described to the user as a result of a query or an order from the system.

In GISE we have focused on assuring linguistic coverage only to conceptual operations tied to the Ontology. For such a purpose, within the generated interface only the correct choices (according to the Ontology and the meta-grammar) are presented to the user. A left corner parser that returns the set of next acceptable categories at any state is used. Sentences can be directly typed or selected in a menu-based mode (similar to NL-Menu, see [10]).

The grammar includes both a syntactic and a semantic description of wellformedness. Non-terminal and terminal categories have a semantic signification tied to the CO, as well as syntactic patterns of realisation tied to the LO. The meta-grammar defines the conceptual coverage. The relationship between the two ontologies is used to define the linguistic coverage and to state

the syntactic-semantic properties of non-terminal and terminal categories of the grammar.

The grammar skeleton basically follows a DCG form, with syntactic and semantic features, where several extensions have been included:

- The possibility of attaching run-time preconditions to the firing of a specific rule.
- The existence of dynamic lexical categories, corresponding to lexical entries whose values are determined (selected) at running time by the system or the user (names of instances, attribute values).
- The incorporation of an interpretative semantic component based on a typified lambda calculus.

A prototype of the system has been built in Prolog for a IBM-PC and an interface to an existing ES for legal advising (SIREDOJ, [8]) has been generated.

3 Conceptual Ontology

In the CO all knowledge needed in the communication process including general conceptual, domain-specific, task-oriented and case-specific knowledge is described and structured.

The Ontology is implemented in a frame-like fashion, where basic elements are objects (identified set of properties or attributes) and basic relations are subsumption (**ISA**) and instantiation (**INSTANCE**).

Formally, we have defined the CO as a partial order finite set over the subsumption relation, where the top element is the least informative concept.

Information acquisition is done monotonically, thus not allowing new information to be inconsistent with the existing one. This limitation substantially reduces the semantic network complexity. The system allows orthogonal multiple inheritance.

The basic elements of the set are primitive concepts and attributes. Attributes can also be viewed as functions where the domain set is the set of concepts on

which functions are defined and the image set is the set of possible values of these functions.

A central issue in NL communicating systems is how to relate application concepts to linguistic information, i.e. how general domain independent relations between application concepts and its linguistic realisation can be established (see [2],[3],[7] and [9]). Penman Upper Model is a well known linguistically-motivated knowledge organisation structure that can mediate between the application domain and the kind of linguistic organisation more convenient for implementing the linguistic source. Upper Model has been built for providing general text generation facility. Our needs are more limited, because we are constrained to perform only frame-based operations. For these reason we have built a taxonomy of conceptual attributes according to its linguistic behaviour incorporating a subset of Upper Model classes and adapting it to our needs (in fact not only generalisations but also some specialisation of the Upper Model have been introduced in order to adapt it to our needs, especially due to the differences between Spanish and English syntax).

As we are mainly interested in linguistic coverage of frame-based operations special attention has been paid to the linguistic realisation of attributes.

Attribute classification allows different linguistic coverage for each attribute. For instance, the class OF_QUANTITY can be used for describing attributes referring to quantities. Attributes in this class are always associated with a unit and can be further subclassified depending on the differences in linguistic realisation (e.g. using a verb or an adjective). This classification is highly language-dependent, as can be shown by the following example.

Properties of attributes can also be defined with facets. Some of these facets are: transitivity, cardinality, default values, reflexivity, kind of inheritance, information percolation...

Consider the concept PERSON and JUAN as an instance of person. The representation in the CO of the concept PERSON and its attributes are shown in *Figure 3*. As can be seen, PERSON owns, among others, the attributes *sex*, *age*, *father* and *height*. All of these are described in the Ontology as subclass of ATTRIBUTE. Each of them owns several properties, some of which establish the linguistic realisation of the attribute.

The following are possible sentences to fill the attributes of the instance JUAN.

Juan tiene 30 a-os. / Juan is 30 years old
 La edad de Juan es 30 a-os.
 Juan mide 170 cm. / Juan is 170 cm high.
 La altura de Juan es 170 cm.
 El padre de Juan es Pedro. / Juan's father is Pedro
 Juan es hijo de Pedro / Juan is Pedro's son.
 ¿Cuántos a-os tiene Juan? / How old is Juan?
 ¿Quié n es el padre de Juan? / Who is Juan's father?

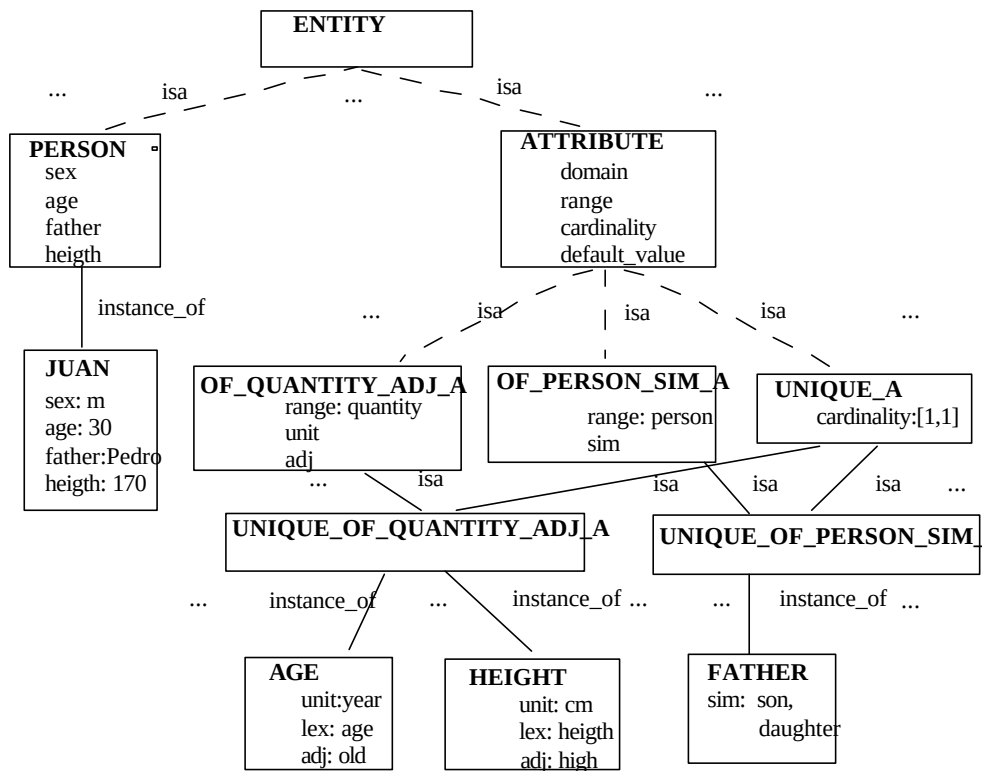


Figure 3: A fragment of the Conceptual Ontology

The concepts in the CO are organised in four levels: the meta-concept level, where all meta-information used in the definition and organisation of new concepts is represented; the general level, where basic concepts common to most applications are defined; the domain specific level, where concepts of the specific application are represented and instance or case level, where all concepts instances are represented. During the communication process at running time only the instance level can be modified .

For the task of interface generation, a detailed typification of the available operations has been established. We consider these operations both in isolation and combined. We distinguish between constructive and consulting operations. Simple constructive operations include creating and removing unconnected objects, filling attributes, adding or removing values and connecting objects. Complex operations involve several of these simple ones, e.g. creating a new object together with its attributes and classifying it in the Ontology.

An example of the CREATE_INSTANCE operation is shown in *Figure 4*.

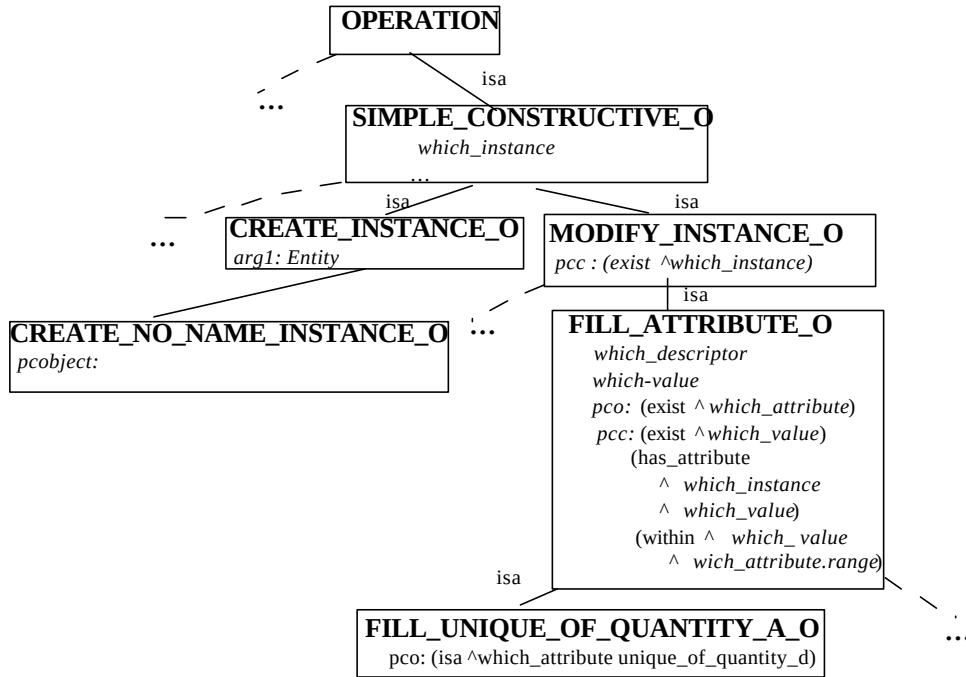


Figure 4: A fragment of Conceptual Ontology with operations classes

Just like attributes, operations are represented as objects in a taxonomic structure (formally they are considered as other objects). For each operation the number of its arguments and their classes are specified, as well as the preconditions that must be satisfied to perform the operation. We consider two kinds of preconditions: those executed in the generation process (Ontology Preconditions) and those executed in the communication process (Case Preconditions). Preconditions, like other concept attributes, can be inherited through taxonomic links. In order to consider only reasonable moves of interaction and offer them to the user by means of the generated grammar, our system allows the definition of macrooperations that implement some common ways the users have of describing their information. People usually communicate their knowledge following paths tied to the possible connections between the objects they are referring to. For instance, after mentioning a new object, users usually want either to describe it or to connect it to previously defined objects. Operations allowing partial, but not simple, description of a

recently created object or multiple connections to other objects (in focus) are also provided to the user.

A fragment of the CO where some operations appear is shown in *Figure 4*. In *this figure* we can see that each instance of `FILL_UNIQUE_OF_QUANTITY_A_O` owns a *which_instance* attribute, inherited from `SIMPLE_CONSTRUCTIVE_O`, that must be filled according to the *pcc* attribute (case precondition) inherited from `MODIFY_INSTANCE_O`. These instances own also a *which_attribute* and a *which_value* attributes constrained by the corresponding conditions.

Some of the preconditions can be stated at the generation level (those appearing as *pco*) and other will apply later, when running the interface (those appearing as *pcc*). In the operation of the example, at generation level only the existence of the class and the appropriateness of the attribute can be tested while the existence of the instance and the appropriateness of the value to be attached must be postponed.

4 Linguistic Ontology

The Linguistic Ontology is defined using the same form of representation used for CO.

LO covers the linguistic realisation of both objects and operations in the CO. We can, for instance, use nominal phrases to refer to conceptual entities denoting physical objects, and verbal phrases to refer to concepts denoting actions and operations on the CO (e.g. interrogative phrases for expressing consulting operations).

As has been mentioned in previous section, our approach for knowledge representation is based on the Penman Upper Model system for NL Generation. The linguistic core of Penman is Nigel, a large systemic-functional grammar of English based on Halliday's work. In the LO in our system, as in Nigel grammar in Penman system, most classes are assumed to be general while linguistic objects representing the specific aspects of the information to be expressed and the way of its expression are generated for each application. CO and LO are independent of each other although there must be appropriate mappings for expressing at linguistic levels the differences existing in the

conceptual level.

In linguistic objects we split constituency, order and functionality into different features. Syntactic and semantic features are also represented in linguistic objects (e.g. orthography, gender, number, semantic category, semantic interpretation, case and ontology preconditions).

The classes in the LO correspond to general linguistic structures. Values attached to attributes can refer to other classes (e.g. the value associated to the attribute *verb* of an ATTRIBUTIVE_SENTENCE must be an instance of the class STATE_VERB). Disjunctions and sequences can be used too (e.g. in a NOMINAL_PHRASE instance the value of the attribute *det* can be an instance of the class DEF or of the class INDEF and the value of the attribute *cset* is a sequence of instances attached to the attributes *det*, *modifiers* and *nominal_head*). Several superficial presentation are allowed for these constituents as expressed by means of the *pattern* attribute. As can be seen in Figure 5 , in linguistic objects not only constituent structure is described but constraints (as *agreement*) and information for further semantic interpretation (as numbers indicating the constituents interpretation order associated to each possible *pattern*).

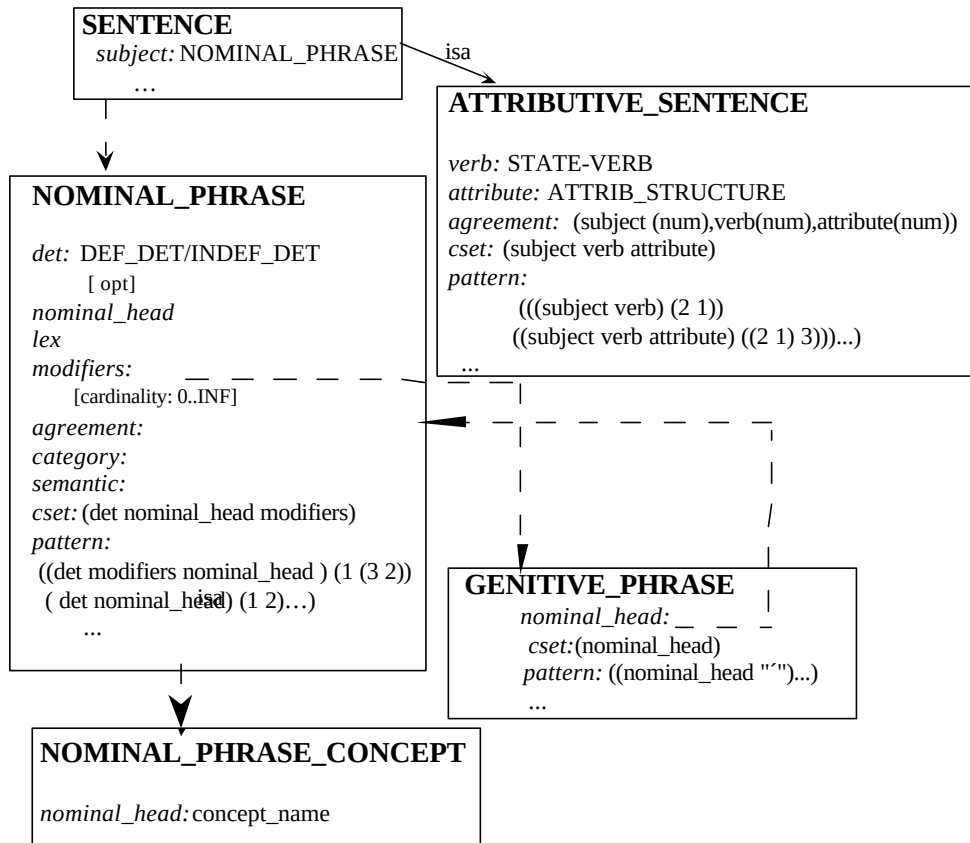


Figure 5: A fragment of the LO where some knowledge for attributive sentences is represented

5 Implementing Control

As has been mentioned in the introduction, some of the most important objectives of our system are friendliness and transportability. To reach these objectives GISE requires an easy tailoring of a new interface (a new domain, a new application) and of different kinds of users (casual or habitual, expert or

novice).

From the application specifications represented in the CO and the general linguistic knowledge represented in the LO, the Grammar and the Lexicon needed for the communication process can be generated automatically .

The generation process is done just once, before the communication process starts but a possible tuning of the grammar and the lexicon can be done dynamically during communication process according to the state of the case ontology. A set of metarules controls all the generation process. Metarules are applied over concepts of the CO to generate instances of the operational ontology, over these instances to generate instances of the LO and, finally, over these linguistic objects to generate the linguistic sources to be included into the interface, i.e. the grammar rules and the lexical entries.

Preconditions can be associated to metarules dealing with the state and kind of the communication process (updating/consulting, kinds of users). Conditions can also limit the operations that can be done at a time in a state, e.g. conditioning the application of a fill attribute concept operation instance to the previous existence of the concept instance or limiting the number of active concept instances at a time. Some of this conditions can also be associated with the grammar rules generated.

Preconditions established on the objects belonging to the CO, like those conditioning the existence of an object to the existence of another, can also be adapted to linguistic objects and they can be attached to lexical entries.

Metarules are implemented with a set of production rules by means of PRE, a production rules environment specially built for NL processing. PRE formalism is similar to the one used in others rule based languages (e.g. OPS5) although it incorporates some capabilities as the use of rulesets and a powerful and flexible control mechanism. In PRE, rules consist of a set of conditions establishing how and when rules must be applied, and a set of operations establishing the operations to be performed. Conditions on how rules must be applied consist of pattern matching conditions over the existing objects. Only four operations are allowed in the action part of the rule (creation and deletion of objects, activation of a rule set and assignment of values to variables) but the possibility of accessing the Ontology enrich considerably the power of the formalism.

Figure 6 describes an example of a metarule applied over CO. For each concept in the ontology, this metarule generates an instance of the operation **CREATE_NO_NAME_INSTANCE_O**, which creates a new concept instance from the concept name. **ins_op1** identifies the rule, **ontops** is the ruleset to which the rule belongs (in PRE rule application follows a ruleset firing mechanism), **forever** states the mode of rule application and **priority** controls the order of application of the rules within the ruleset. See [1] for a description of the PRE language. The following statement establishes the conditions to apply the rule, in this case the existence of an object in the CO. For each object satisfying the conditions, the action part is performed. In this case the **create** operator implies the creation of a new object in the working memory. The **:=** operator followed by the call to the **crea-frame** function implies the creation of a creation operation instance in the Operational Ontology; the **delete** operator removes the object from the working memory and the **apply-ruleset** operator fires the *sent_ci* ruleset, that contains rules in charge of the creation of the linguistic objects needed. In next section we will see with some detail the performance of the control over a simplified example.

```
(rule ins_op1
ruleset ontops
priority 1
control forever
(object ^nom ?nom ^pcc ?pcc)
->
(?x0 := (nom 'c_n_n_i_o ?nom))
(?x1 := (crea_frame
(list 'mk-frame ?x0 (list 'instancia 'c_n_n_i_o)(list 'arg1 ?nom) (list 'pcoobject ?pcc ))))
(create 'c_n_n_i_o ^nom ?nom ^arg1 ?nom ^pcoobject ?pcc)
(delete 1)
(apply-ruleset sent_ci))
```

Figure 6: An example of PRE rule

6 Following an example

We will now illustrate the performance of GISE with a simplified example taken from SIREDOJ. We will assume that there exists only three classes in the CO: CONTRACT, PERSON and ARCHITECT, being the last a subclass of PERSON³

If we want to allow the user to create instances of any of these classes we can start with the metarule in *Figure 6* for creating instances of the operations involved in the creation of concept instances in the Case Ontology, i.e. CREATE_NO_NAME_INSTANCE_O. Then other metarules will be in charge of mapping these operations to the corresponding instances in LO for further generating, by means of other metarules, the corresponding grammar rules and the lexical entries that will allow the user to express in NL the order of creating the desired instances when running the interface.

After performing the *ins_op1* metarule three instances of the class CREATE_NO_NAME_INSTANCE_O have been created (one corresponding to each conceptual class). *Figure 7* shows the corresponding instance generated for the concept *contract*.

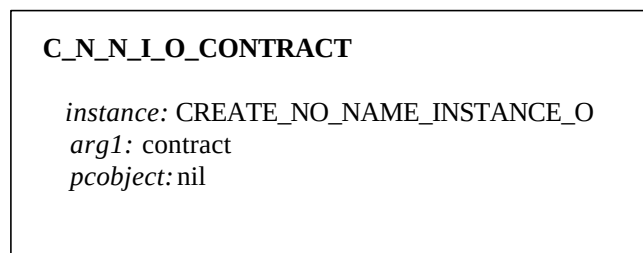


Figure 7: An instance of the creation operation

For each operation instance one or more linguistic objects instances are created. Attributes of these linguistic objects are also filled (semantic interpretation, category, semantic features associated to the categories, preconditions,...). In our example, for each operation instance, a NP (nominal phrase) is created and its attributes are filled (stating that a way of referring to the concept CONTRACT is by means of a NP headed with the word “contract”). The

³ Only the features directly involved in our comments are presented in the corresponding figures. The example Objects in CO and LO are also simplified.

metarule used to generated the corresponding NP is presented in *Figure 8*.

```
(rule sni1
ruleset sent_ci
priority 1
control forever
(c_n_n_i_o ^nom ?nom ^arg1 ?arg1 ^pobject ?pcc)
->
(?x0 := (nom 'n_p_c ?nom))
(?cat := (nom 'np 'concept_name))
(?lex:= (car(value ?arg1 'lex)))
(?x1 := (crea_frame
(list 'mk-frame ?x0 (list 'instance 'n_p_c)
(list 'category ?cat)(list 'semantic ?arg1)(list 'det 'indef)(list 'lex ?lex)(list 'pobject ?pcc ))))
(create np_c ^cat ?cat ^sem ?arg1 ^det indef ^lex ?lex ^pobject ?pcc)
(delete 1)
(apply-ruleset cob_ling))
```

Figure 8: A PRE rule for mapping arguments of operations into NP

One of the three instances generated when applying this rule, the instance N_P_CONTRACT, an instance of the class NOMINAL_PHRASE_CONCEPT for the concept contract is shown in *Figure 9*.

N_P_C_CONTRACT

instance: NOMINAL_PHRASE_CONCEPT
det: INDEF
lex:: contract
category: np_concept_name
semantic: contract
pobject: nil

Figure 9: An instance of NP for covering the concept CONTRACT

A grammatical rule for expressing the existence of an object instance is also implemented as an object in the LO and it is also generated by a metarule. The metarule used in the example to generate an instance of the class ATTRIBUTIVE_SENTENCE, used to create the grammar rule that will allow to utter this kind of operations presented in *Figure 10*.

The resulting instance is presented in *Figure 11*.

```
(rule sentci
ruleset sent_ci
priority 1
control one
(c_n_n_i_o)
->
(?x0::= (nom 'a_s_c_n_n_i))
(?x1::= (crea-frame (list 'mk-frame ?x0 (list 'instance 'attributive_sentence) (list 'subject 'np_concept_name) (list 'verb 'v_to_exist))
```

Figure 10: A PRE rule for mapping create_instance operation into an attributive phrase instance

For each NP_Concept instance in the LO a lexical entry is generated by the metarule showed in *Figure 12*. In this rule the number of the corresponding instance name and the appropriate determiner is obtained from the General Lexicon and a lexical entry with three attributes (string, category and semantic interpretation) is created.

A_S_C_N_N_I

instance: attributive_sentence
subject: np_concept_name
verb: v_to_exist
attribute: nil
cset: (np_concept v_to_exist)

Figure 11: An instance of the class attributive sentence

```

(rule sn_lex
  ruleset cob_ling
  priority 1
  control forever
  (np_c ^cat ?cat ^sem ?sem ^det ?det ^lex ?lexic )
  ->
  (?entrada := (cdr(car(get-immediate-value ?lexic 'information))))
  (?ent := (first ?entrada))
  (?num := (second ?entrada))
  (?string := (nom (get-determ ?det ?num ?ent) ?ent))
  (?x1 := (nom 'np ?arg1))
  (?list1 := ' (?cat ((num (?num) )))
  (?y4 := (crea_frame (list 'mk-frame ?x1 (list 'instance 'lex_ent)(list 'cat ?list1)(list 'str ?string) (list 'sem ?sem)
  (delete 1)

```

Figure 12: A PRE rule for generating lexical entries for Nominal_Phrases

Linguistic objects representing grammatical rules and lexical entries are further compiled together in order to have the data structures needed to generate a new interface.

In the example, the resulting grammar rule is:

Sentence_instance ->

NP_Concept_Name(number(N)) V_to_exist(number(N)) (2 1)

where the category *Verb_to_exist* corresponds to lexical entries of verbs indicating existence and *NP_Concept_Name* is the category for the lexical entry that represents a conceptual object. The syntactic feature Number is associated to both categories and it is used to control number agreement between them.

In the example above, the lexical entries generated are:

<a_contract>	NP_Concept_Name(number(s))	(Contract)
<a_person>	NP_Concept_Name(number(s))	(Person)
<an_architect>	NP_Concept_Name(number(s))	(Architect)

where each lexical entry consists of three fields: the string representing the entry, the semantic category and the semantic interpretation .

The other two lexical entries needed to build a NL sentence to create an instance with no name are:

<exist>	V_to_exist(number(s))	((lambda X, is (X))
<exists>	V_to_exist(number(p))	((lambda X, is (X))

These lexical entries belong to an initial lexicon where some general verbs and closed categories (e.g. prepositions) are represented.

Metarules used to generate the needed structures to recognise and to build NL sentences for other simple conceptual operations, as for the operation to fill a concept attribute, are quite similar to the metarules used in the example for the operation to create an instance. For each attribute class there is a specific linguistic realisation, i.e. different grammar rules are used for different attributes type.

More complex operations, like those referring to more than one instance can be built in NL with more complex grammar rules supporting coordination and/or subordination and reusing the objects created in the Operational and Linguistic Ontology for simple operations (to create an instance, to fill an attribute, to consult an attribute value, etc.).

7 Conclusions

We have presented here an approach to build NL interfaces to applications involving complex interaction. Our approach is based on the description of domain and operations for particular applications in a unified formalism, and on the building from these descriptions general ontologies, both conceptual and linguistic. A set of metarules implemented in a production-rules language, PRE will then proceed on these ontologies in order to obtain a dynamic grammar and a lexicon, tuned to the specific domain and application, that are adaptable to the evolution of the communicative process. The approach has been implemented in GISE, an interface generator for a class of Expert Systems.

References

- [1] Ageno A., Ribas F., Rigau G., Rodríguez H., Verdejo F. "TGE: Tlinks Generation Environment." Aquilex II WP Num. 8, UPC, Dept. LSI, 1993
- [2] Bateman J., Kaper R., Moore J., Whitney R. "A General Organisation of Knowledge for Natural Language Processing: the Penman Upper Model". Penman Development Note, USC/Information Sciences Institute, 1990
- [3] Dale, R. "Generating referring expressions". MIT Press, Cambridge (MA), 1992.
- [4] De Jon F.M.G., Nijholt A. (eds.) "Natural Language Interfaces" TWLT5. Fifth Twente Workshop on Language Technology. University of Twente. 1993
- [5] Gatus M., Rodríguez H. "Automatically generating linguistic knowledge sources from application specifications". Proceedings of CSNLP, Dublin, 1994
- [6] Gatus M., Rodríguez H. "Un Generador de Interfaces en Lengua Natural para Sistemas Expertos". Proceedings of SEPLN, Cordoba (Spain), 1994
- [7] Jacobs, P.S. "Knowledge Intensive Natural Language Generation". Artificial Intelligence Vol. 33, pp 325-378, 1987.
- [8] Martí M.A., Rodríguez H. "Interfaz en Lenguaje Natural para acceder a SIREDOJ" Proceedings II Int. Conference on A.I. and Law. Logroño, 1991.
- [9] Perkins, W.A.: "Generation of Natural Language from information in a frame structure" Data & Knowledge Engineering Vol 4, pp 101-114, 1989.
- [10] Thomson, G.: "Menu based N.L. interfaces to DB". Database Engineering 1985.
- [11] Yen J., Lee J. "A Task-Based Methodology for Specifying Expert Systems". IEEE EXPERT, Vol 8, pp 8-15, n°1 1993.