Artificial intelligence tools for software engineering: Processing natural language requirements

M. Bras & Y. Toussaint

ARAMIIHS,[1] 31 Rue des Cosmonautes, 31077 Toulouse Cedex, France

ABSTRACT

In an industrial and technological domain such as space, natural language messages and documents constitute the main medium of information exchange between different kinds of human operators.Through the presentation of the Linguistic Engineering for Software Development project, we focus on specification requirements documents, which are governed by standards and quality criteria and where quality problems may entail different kinds of errors such as inconsistency between documents or even loss of functionality between the initial requirements and the coding phase. To improve quality control of requirements documents,we aim at mastering the linguistic material in order to enable computational extraction and use of the informational content of requirements documents. We propose Linguistic Engineering tools to build semantic representations of a set of software requirements, then Artificial Intelligence Tools to handle these representations in order to assist software engineers in the task of defining traceability links between requirements.

1. INTRODUCTION

In an industrial and technological domain such as space, natural language messages and documents constitute the main medium of information exchange between different kinds of human operators. The major problems arising in producing and handling documents such as proposals, requirements documents, user's manuals, maintenance manuals, integration and operation procedures are related to the size of the documents to be handled (for example in the development of a commercial satellite an average of 50000 documents are produced and referenced) and to the use of a domain specific sublanguage. Concerning the specification phase, for example, quality problems in specification requirements documents may entail different kinds of errors such as inconsistency between documents or even loss of functionality between the initial requirements and the coding phase. Mastering the use of the sublanguage would save time and money, increase the writers productivity, produce documents of better quality, and help to master the overall work process. A way to cope with these problems is to proceed to a linguistic analysis of the technical sublanguage in order to be able to master it.

Through the Linguistic Engineering for Software Development (LESD) Project, and the more ambitious program of Research and Development it is part of, we aim at providing Linguistic Engineering tools for Software Engineering, particularly for the specification phase, since a significant amount of documents written in natural language (NL) are produced along this phase.

Specification documents are composed of sets of requirements defining the

---

[1]ARAMIIHS, joined research laboratory MATRA MARCONI SPACE France - CNRS (French National Center of Scientific Research)

functional aspects and characteristics of a system. We work on a corpus of 50 requirements taken from the functional requirements document of the In Orbit Infrastructure Ground Segment (IOI-GS) of the System Architect Support Contract project. The part of the document we are interested in defines the IOI-GS space vehicle monitoring and control capabilities such as the control of the automatic systems of the flight configuration, the monitoring of the health of the flight configuration, of the on-board resources and of the on-board configuration, the detection of anomalies, etc...

As far as the quality of the requirements is concerned, standards have been defined by IEEE, ESA or NASA to guide the writing of requirements. On the one hand, these standards specify linguistic constraints in order to limit irregularities that can be involved by the use of NL such as ambiguity, polysemy or vagueness. They occur as rather informal constraints stating for example that a requirement should be a simple sentence and should not convey neither ambiguity nor vagueness. They act as syntactic constraints (restricting sentence and phrase construction) and as lexical constraints (restricting the use of words and terms to be used). On the other hand, these standards specify the properties —software engineering constraints— that a requirements document has to verify such as consistency, completude, traceability, verifiability and modifiability. The quality of a requirement document is function of the degree to which it fulfills these constraints.

The tools we intend to design will help the engineers to control the quality of the requirements they write, both from linguistic and software engineering points of view. The tools we are developing at present within the framework of the LESD project aim at assisting the analysis of requirements in order to extract their informational content and to represent it. In a second step, they aim at assisting the user in various information retrievial tasks leading to the construction of a structured set of requirements. On the basis of the requirement set structure, the system will help the engineer to build traceability links between requirements.

Traceability in a software project is ensured by a set of links between requirements that traces the evolution and choices in the development of a software. As traceability is involved in other quality criteria such as consistency or completude, we focused on tools which aim at mastering traceability in NL requirements documents.

In traditional approaches (Fig. 1), software engineers have to deal with NL specification requirements documents.in order to generate a formal specification, invoking their specific knowledge of the domain.

With the LESD project, we aim at providing the engineer with a semantic representation of each requirement. The informational content of requirements documents has to be extracted in order to build a formal representation of each requirement. This is performed by a set of Natural Language Processing (NLP) tools. In a second step, we want to provide reasoning tools for the engineer to set links on these formal structures so that he can be assisted in his work of producing a further document. Let us mention that LESD does not aim at an automatic translation from NL requirements to a formal specification (Fig.2)

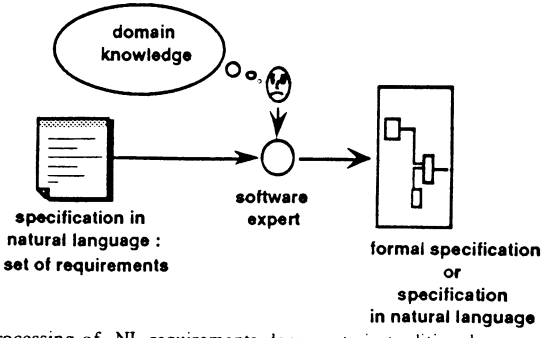


Fig 1 : Processing of NL requirements documents in traditional approaches

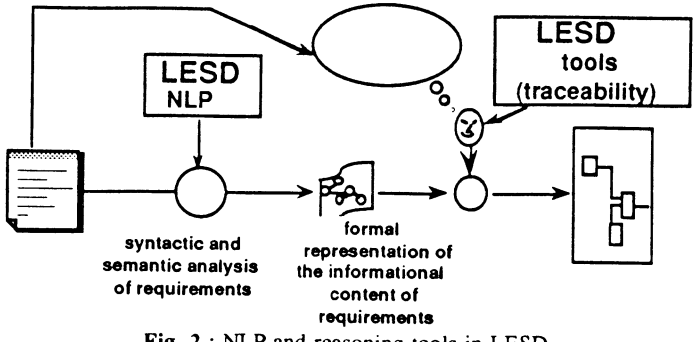


Fig. 2 : NLP and reasoning tools in LESD

2. THE LESD SYSTEM

Representing the meaning of a requirement or extracting the informational content conveyed by a sentence requires several kinds of knowledge and consequently several levels of analysis :

- **Morphological knowledge** : first the words of a sentence, together with their canonical form, have to be identified.

- **Syntactic knowledge** : then the way the words are associated to form phrases (group of words) and sentences has to be investigated in order to point out the syntactic structure of the sentence. The general relations expressing formal links between word categories is described in the grammar of the language.

- **Semantic knowledge** : each word of the sentence conveys a meaning and so does the sentence as a whole. From a referential point of view, semantics can be roughly described as a relation between the elements of the language (words, phrases and sentences) and the elements of the speaker's representation of the real world (objects and events). This relation can be established via an internal (formal) representation of the sentence. From a mere linguistic point of view, semantics concerns the relations between words within the linguistic system.

- **Pragmatic and background knowledge** : context and general world knowledge are necessary to choose the right interpretation of a sentence. These knowledge refer to the implicit aspect of meaning which is conveyed by a sentence.

Thus, understanding a sentence requires **linguistic knowledge** (morphological, syntactic and semantic) and **extra-linguistic knowledge** (pragmatic and world knowledge).

In the LESD system, the linguistic knowledge is represented in the so called **Language Model** including :
- the dictionary which contains for each entry morphological, syntactical and semantic data,
- a set of rules combining these basic data :
  - the grammar including morphological and syntactic rules
  - a set of semantic rules associated to the syntactic ones.

The world knowledge, which is here equivalent to the description of the technical domain refered to by the requirements, is described in a knowledge base called the **Domain Model**. It contains a semantic network representing the concepts of the domain. They are structured by relations such as taxonomic relations (classification from the most specific to the most general concept) and meronomic relations (decomposition of a concept into its components).

The general architecture of the LESD system is given in figure 3.

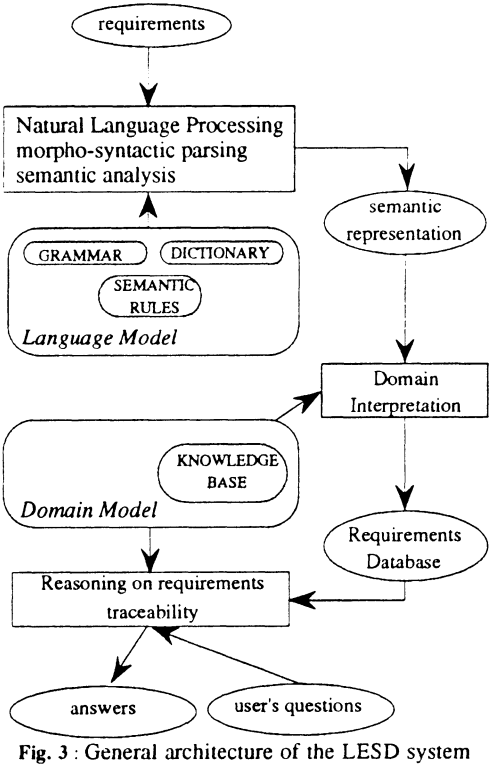


Fig. 3 : General architecture of the LESD system

• **The NLP Component** implements morpho-syntactic and semantic analyses based on the Language Model. The morpho-syntactic parser of the LESD system was developed on the theoretical basis of GPSG (General Phrase Structure Grammar, cf Gazdar et al. [6]), in the framework of the Alvey Natural Language Tools project. It is called the Grammar Development Environment (GDE, cf. Grover et al. [7]). We have adapted it to our corpus of requirements. The NLP component is presented in section 3.

• **Each requirement is parsed by the NLP component and a semantic representation**

Transactions on Information and Communications Technologies vol 2, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517

is built. Then, it is necessary to interpret this representation in its context. This is performed by the **Domain Interpretation Component.** The interpreted representation is then integrated in the Requirements Database including a complex semantic network representing the content of the requirements. The interpretation process is described in section 4.

• With the **Reasoning Component**, we aim at helping the engineer to build traceability links between requirements. Traceability testing does not mean a simple mapping between all the concepts involved in two requirements. It involves inference techniques on a strongly structured knowledge representation of the universe of the discourse. To exploit the Domain Model and the Requirements Database a question language and an answering mechanisms have been defined. They are detailed in section 5.

3. THE NATURAL LANGAGE PROCESSING COMPONENT

In this section we describe the way the NL requirements are processed in order to extract their informational content. We explain what kind of linguistic data is associated to the words in the lexicon and the way these data are combined to get the so-called semantic representation of the requirement. Our aim here is not to give an exhaustive description of the NLP Component but to illustrate the complexity of the linguistic information required at this level.

For the sake of illustration, we will take simple requirements from our corpus such as :

req0 : The IOI-GS shall monitor the status of the space vehicle
req1 : The IOI-GS shall control the automatic systems of the space vehicle.

3.1. Morpho-syntactic analysis

As a result of this analysis the words of the sentence have to be identified and organised along the syntactic structure of the sentence, as illustrated in figure 2.

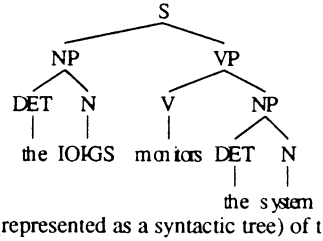


Fig.4 : Syntactic structure (represented as a syntactic tree) of the simplified requirement : *The IOI-GS monitors the system.*

The rules which are used to derive this representation can be seen as a set of rewriting rules. Here are for example the rules associated to the 3 upper nodes of the tree of figure. 4, S (for sentence), NP (for Noun Phrase), VP (for Verb Phrase) :

S --> NP, VP
NP --> DET, N
VP --> V, NP

The rules are actually not that simple as they need to convey much more precise information in order to control the association of morphological and then of syntactic constituants.

We are now going to describe the morphological and syntactic information associated to the words in the lexicon. The general format of a lexical entry is the following :

word : written form
: phonological form
: morphological and syntactic information
: semantic information

The linguistic data associated to the words are conveyed by **features**, as it is shown in the following example giving morphological and syntactic features* of two lexical entries associated to the noun *system* :

N [LAT +, COMPOUND NOT, FIX NOT, INFL +,
POSS -, PLU -, PER 3,
NUM -, PN -, PRO -, COUNT +,
SUBCAT NULL]

N [LAT +, COMPOUND NOT, FIX NOT, INFL +,
POSS -, PLU -, PER 3,
NUM -, PN -, PRO -, COUNT +,
SUBCAT PP, PFORM OF]

The first set of features gives morphological information : *system* has a latine origine (LAT +), which gives rises to a certain kind of derivation, it is not a compound word (COMPOUND NOT), as opposed to *space vehicle* for example, it is neither a suffix nor a prefix (FIX NOT), and a prefix or a suffix can be added to it (INFL +), e.g. to derive the word *subsystems*.

The second set of features combines morphological and syntactic information : *system* is not a possessive case (POSS -), as opposed to *system's*, it is a third person (PER 3) singular form (PLU -), as opposed to the plurals *data* or *systems*. Thus this feature will be modified by the morphological rule deriving *systems* from *system* and the suffix *s*.

Then the syntactic features NUM -, PN -, PRO -, COUNT + indicate that *system* is not a numeric, not a proper noun, not a pronoun, and is a countable word.

The last set of features gives information on the potential complements of the noun *system* : in the second entry SUBCAT PP, PFORM OF indicates that *system* subcategorizes a Preposition Phrase beginning by *of*, this entry corresponds to the use of the word *system* in *the automatic systems of space vehicle.*

The first entry is associated to the use of *system* without any complement (SUBCAT NULL). Those features depend strongly on the underlying syntactic theory.

The same kind of morpho-syntactic description is given for verbs and auxiliaries Here are for example two entries associated to the verbs *control* and *monitor* :

V [LAT +, COMPOUND NOT, FIX NOT, INFL -,
FIN +, REG +, VFORM NOT , PRD -, PSVE -,
NEG -, AUX -, PAST -,
AGR N2[CASE NOM, PLU -, PER 1],
SUBCAT NP, ARITY 2]

* For the sake of illustration, we give the lexical entries after compilation, that is after adding a set of default features, but for the sake of simplicity, we do not give all the features normally associated to an entry.

V [LAT +, COMPOUND NOT, FIX NOT, INFL +,
FIN -, REG +, VFORM NOT , PRD -, PSVE -,
NEG -, AUX -, PAST *,
AGR N2[CASE *, PLU *, PER *],
SUBCAT NP, ARITY 2]

From a morphological point of view, the first entry describes the forms of the verb that cannot be derived, i.e. that will not bear any suffix (INFL -) and appear consequently under a finite form (FIN +). On the opposite, the second entry describes the forms of the verb that can bear a suffix (INFL +), for example the past suffix *ed*, since this verb is regular (REG +), thus it appears at the morphological level under an infinitive form (FIN -). Both entries describe the active form of the verb (not passive PSVE -). The combination of the features VFORM and PRD is used to distinguish different verb forms : present participle, gerund, past participle, passive, infinitive without *to*, and default value, as it is the case in our examples.

From a morpho-syntactic point of view, the described forms are not negative (NEG -) and they are not auxiliaries (AUX -). The first entry does not describe a past form (PAST -), whereas the second can describe a past (PAST)* in *control* + *ed* or *control* + *ing*.

From a syntactic point view, the AGR feature is used for the agreement of Noun Phrases and Verbs. The agreement specified in the first entry is the one required for example for *I control* (first person of the singular). Other entries with different agreement values will be used to parse *the systems control* (third person of plural), *you control* (second person) or *we control* (first person of plural)*. The derived form of the verb (second entry) have a variable agreement *control* is a transitive verb and requires a NP as an object complement (SUBCAT NP). Thus it requires two arguments (subject and object) (ARITY 2).

All the morpho-syntactic features associated to the lexical entries are used, first by the morphological rules, then by the syntactic rules that yield the tree structure (cf Fig.2). The N (noun) and V (verb) symbols actually point out a set of features rather than a grammatical category. The rules can be viewed as set of constraints on these features : they constraint the word associations through feature combinations. A simple example of these constraints on word association is the agreement principle. The following figure illustrates the satisfaction of the agreement constraint during the parsing process.

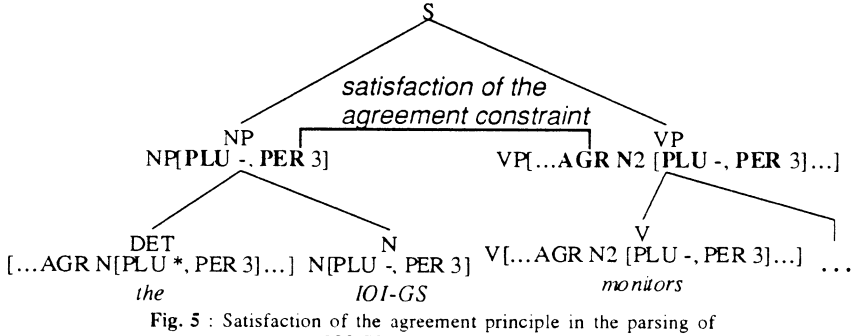


Fig. 5 : Satisfaction of the agreement principle in the parsing of *the IOI-GS monitors (the system)*

* Morphological default rules are actually associated to morphological features such as AGR and are used to add all the entries corresponding to the verbs derivation

3.2. Semantic analysis

The role of the NLP component is to build a semantic representation, that conveys the informational content of the sentence. This representation will be exploited in relation with the informations on concepts refered to by the sentence, which are described in the Domain Model.

The semantic representations produced by the NLP component look like expressions in first order logic. We give for example the representation of the requirement req0 (Fig. 6).

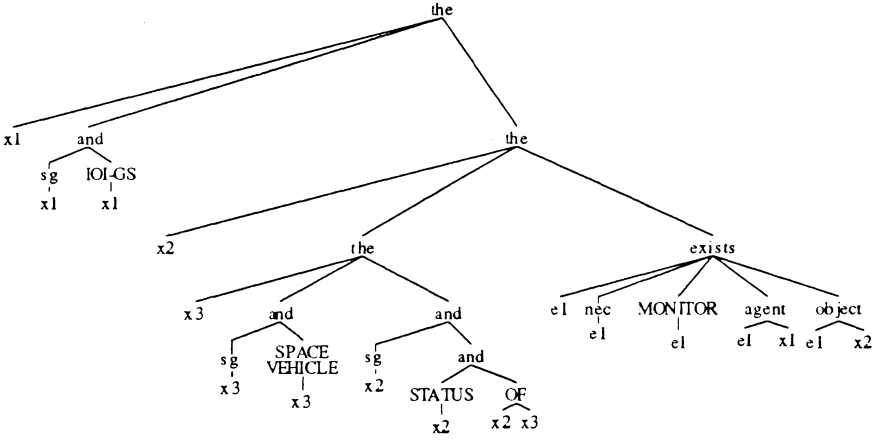


Fig. 6: Semantic representation of the requirement :
req0 : The IOI-GS shall monitor the status of the space vehicle.

This representation, here on a schematic form, expresses that there exists an event e1, that is a monitoring event, that has a necessary modality, the agent of which is x1, representing the IOI-GS, the object of which is x2, representing the status of the space vehicle. The classical predicative representation à la Montague (cf. Dowty et al. [4]) is enriched with information on the semantic role of the event participants, following Fillmore case structures [5].

The semantic information associated to an entry in the dictionary conveys a part of his meaning. For categories such as nouns and verbs, it gives:

• the concept to which the word refer. It is usually noted as the word in capital letters. For example, the concept SPACE VEHICLE is associated to each entry of the word *space vehicle*. This concept will then be used to access the knowledge base of the Domain Model.

• semantic features to characterize the semantic properties of the word. These features are used to classify the word according to a semantic typology.

For example, to describe the nouns of our corpus, we use the features :

H+ for human or machine entities which have capacities to compute, control, etc...,

CTRL+ for the entities that have the capacity to control a process,

SYST+ for the system nouns such as *space vehicle, subsystems, systems.*

DATA+ for the data nouns such as *data, configuration data, trajectory, telemetries.*

PROC+ for the processes and activities expressed by nouns such as *use, progress.*

STAT+ for state nouns such as *status, health, safety.*

These features are transmitted to noun phrases and can help to characterize the

agent and object of a predicate. For example here is the semantic description of the verb *monitor* :

[MONITOR, ACT + , CTRL +,
OBL_ARG [AGT: and (CTRL+ , H+), OBJ: or(PROC+, SYST+, STAT+, DATA+)],
.....]

The concept associated to *monitor* is MONITOR. It is a controlled (CTRL+) activity (ACT) that requires as obligatory arguments an agent (described as CTRL + **and** H+) and an object (described as a process **or** a system **or** a status **or** a data).

The syntactic structure of the sentence and its semantic structure are strongly connected. For example, it is obvious here that the object refered to by the (syntactic) subject of the verb *monitor* is the (semantic) agent of the monitoring event e1. As a matter of fact, the construction of the semantic representation is parallel to the parsing of the sentence. A semantic rule is associated to each syntactic rule : partial semantic representations, which are expressions of the lambda calculus, are constructed for each partial syntactic analysis which the parser yields, by taking the partial semantic representation of each local tree and building larger expressions according to the formulas associated with the rules which define the local trees (composition). This construction is based on beta-reductions. The final expression is totally reduced.

The final expression, given above in Fig.6 is actually constructed in two steps : a first "unscoped" expression is built before the final one. This two step analysis is based on Alshawi et al [1].

As we showed it for nouns and verbs, elements of the semantic representation are present at the lexical level. They are combined later on in the parsing process with other elements brought by the semantic rules. Most of the time, the information for the expression composition are conveyed by the semantic rules, they appear as partial lambda expressions. However, for categories such as determiners and auxiliaries, the semantic information at the lexical level conveys a part of the resulting semantic representation.We give for example the partial expression associated to the determiner *the* acting as a quantifier on the variable x (Fig. 7).

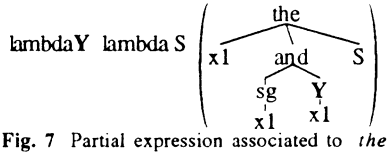


Fig. 7 Partial expression associated to *the*

The semantic rules which associate *the* and *system* operate a composition and the partial expression described in Fig. 8 is obtained.

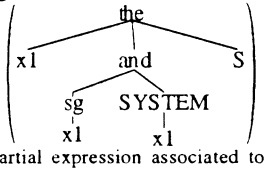


Fig. 8 : Partial expression associated to the NP *the system*

In this section, we wanted to point out the complexity of the linguistic information that has to be associated to word entries in the dictionary to enable the automatic processing of the requirements. The morphological and syntactic description are strongly related to the theory which is underlying to the parsing system. It is also important to mention that the semantic description in terms of semantic features requires a good knowledge of the specific domain described by the requirements, and therefore an important investment.

4. THE CONCEPTUAL INTERPRETATION

The NLP component takes linguistic knowledge into account in order to produce a syntactic and semantic analysis of the requirement. The conceptual interpretation component is based on semantic networks whose functions are :

- to map linguistic entities (nouns, verbs...) to concepts of the technical domain and link them with conceptual relations,

- to integrate the conceptual interpretation of a requirement in the requirement base which contains the conceptual interpretation of all the requirements already treated. The mode of construction we propose for the requirement database is strongly dependant on the nature of the linguistic material we deal with : they are sequences of independant requirements which are different from usual texts in which stences are semantically related.

A requirement is represented by a set of concept linked together by conceptual relations. The requirement req1 (*The IOI-GS shall control the automatic systems of the space vehicle*) is represented by the following concepts, where `req-1` identifies the requirement and defines the modality (*shall* translated by `necessary`) and the main activity of the requirement (*control* translated by `control-1`):

```
req-1
        IS-A            requirement
        MODALITY        necessary
        ACTIVITY        control-1
IOI-GS
        IS-A            system
space-vehicle
        IS-A            flight-element
control-1
        IS-A            control
        AGENT           IOI-GS
        OBJECT          auto-syst-of-spv
auto-syst-od-spv
        IS-A            system
        IS-A            automatic entity
        LOC             space-vehicle
```

The next sections present some aspects of the identification and construction of the conceptual interpretation of the requirements. Our goal is to represent in the requirement database the functionalities a system should implement. This set of functionalities will be incremented when processing new requirements. An important characteristic of this database is that functionalities are static and are not executed : the consequences of the activity of `controling the automatic systems of the space vehicle by the IOI-GS` are not taken into account.

4.1. Mapping words into concepts

The conceptual interpretation is based on a minimal knowledge base. This knowledge base is structured into a lattice with the IS-A relation. A concept is described by the following structure, close to conceptual language notation such as KL-ONE (cf. Kobsa [8]):

```
concept
        IS-A            concept_a
        IS-A            concept_b
        RELATION_1      concept_1
        RELATION_2      concept_2
```

`concept` inherits properties from $concept_a$ and $concept_b$ (cf. Brachman [3]). The properties which are specific to `concept` are given by the set of couples ($relation_i$, $concept_i$).

The five main classes of concepts we have defined and their associated relations are presented in Fig. 9.

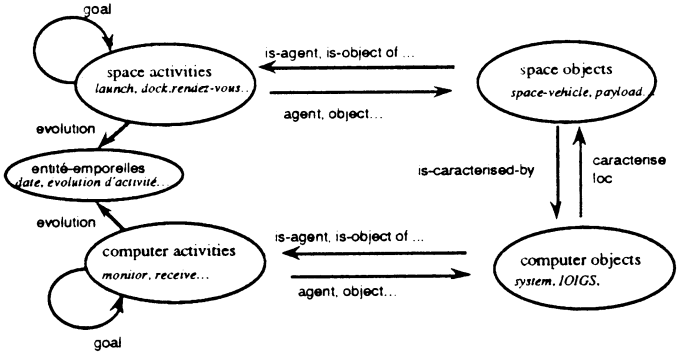


Fig. 9 : general structure of the initial knowledge base

We adopted a representation of quantification based on the notion of classes and instances in conceptual languages. We mentioned above (section 1) some constraints imposed by standards on NL requirements to avoid vagueness and ambiguity. As a consequence of these linguistic constraints, only three form of quantification occur in requirements :

(i) definite and singular quantification (the + singular noun),
(ii) definite and plural quantification (the + plural noun),
(iii) indefinite plural quantification (plural noun),
(iv) indefinite singular quantification (any + singular noun).

(i) is used to refer to a unique entity of the domain. The noun will be mapped to an instance of concept. In our domain `space-vehicle` is an instance of `flight-element`.

(ii) to (iv) correspond to the notion of class. Properties associated to a class or events involving a class are inherited by any sub-class or instance of this class.

4.2. Adding new concepts to the knowledge base to create the requirement database

The minimal knowledge base will be incremented when processing new requirements from the requirements document. For each word, the lexicon provides the basic concept associated to the word. In each requirement, the relation between nouns and their complements or between the verb and its arguments (agent, object...) are determined by the case structure analysis (section 3.2). For example, the word *system* corespond to the concept `system`. At the linguistic level, the semantic analysis of the Noun Phrase *the systems of the space-vehicle* is given by the logical form :

the(x2, space-vehicle(x2) ∧ sg(x2),the (x1, system(x1) ∧ sg(x1) ∧ loc(x1,x2)))

The concept which represents this expression is a x2, a sub-class of SYSTEM :

x2

IS-A	system
LOC	space-vehicle

where system and space-vehicle are concepts from the minimal knowledge base. If a concept with the same definition than x2 does not exist in the knowledge base, it will be added following our *classification algorithm.*

The classification algorithm has to ensure consistency of the knowledge base from the *type* point of view. This notion of type is well known in logic programming or in conceptual language. When a concept from a requirement is identified (x2, for example), the algorithm try to locate the concept in the knowledge base following the IS-A information. For each subclass of the concept system, the algorithm compare the properties of the class with the properties of the new concept x2 : (LOC space-vehicle). If there exists a concept identical to x2, then x2 is replaced by this concept. If it does not exist, then x2 will be added to the knowledge base. For the sake of illustration, let us assume that the current knowledge base is represented by Fig. 10, then an informal illustration of the assertion of x2 in the current knowledge base is given in Fig. 10b. The algorithm locates systems of the space vehicle as a son of the concept system and as a father of the concept automatic systems of the space vehicle. The link between system and automatic systems of the space vehicle is cut and the new concept systems of the space vehicle is inserted.

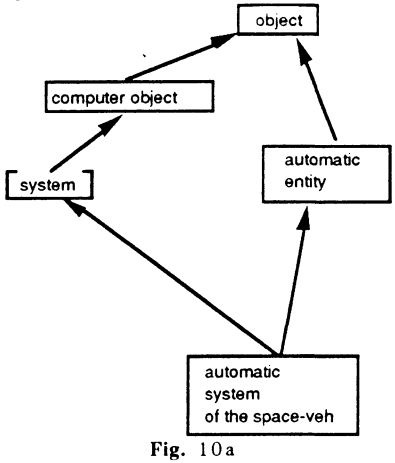


Fig. 10a

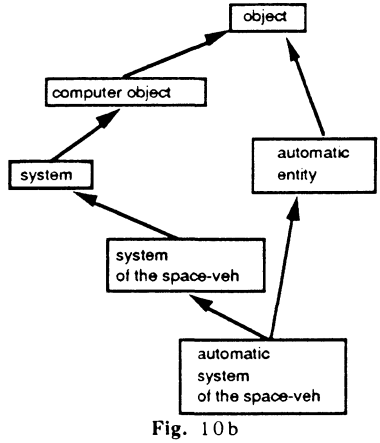


Fig. 10b

Fig. 10 Insertion of the concept associated to the expression "the systems of the space-vehicle" (arrows represent IS-A relations)

The same algorithm is used to classify the activity of any requirement. The concept associated to the activity of the requirement *the systems shall monitor the space-vehicle* is described by the following structure :

the systems monitors the space-vehicle

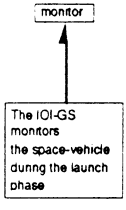
IS-A	monitor
AGENT	system
OBJECT	space-vehicle

where monitor is a concept of the knowledge base.

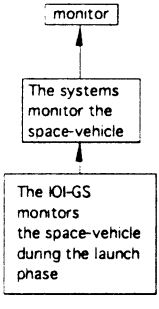
Now, let us interprete the following sequence of five requirements :

```
req2- The IOI-GS shall monitor the space-vehicle during the
launch-phase.
req3- The systems shall monitor the space-vehicle.
req4- The systems shall monitor the flight-elements during the
launch-    phase.
req5- The IOI-GS shall monitor the space-vehicle.
req6- The systems shall monitor the flight-elements.
```

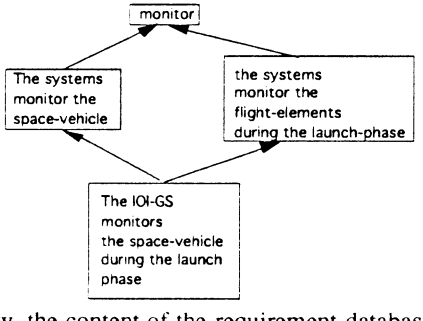
If `req2` is the first requirement to be analysed, its activity is located in the requirement database as a son of the `monitor` activity :



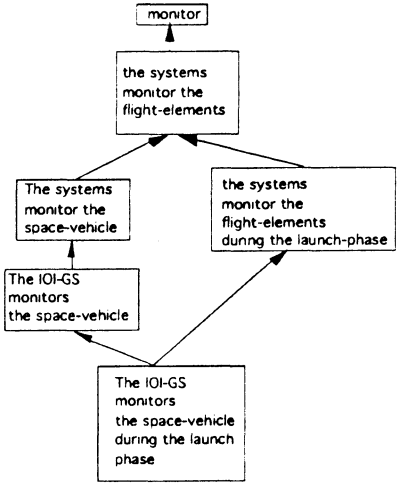
In the initial knowledge base, `IOI-GS` is defined as a specific `system`. The classification algorithm will locate the `req3` activity as more specific than `monitor` (in general) but more general than the `req2` activity. Then, the current requirement base is :



In the knowledge base, the `space-vehicle` is one specific `flight element`. `Req4` activity is more specific than `monitor` in general. It cannot be compared to `req3` activity because `space-vehicle` is more specific than `flight-element` but `during the launch-phase` is more specific than "no information about location in time". However, `req4` activity is more general than `req2` activity. The current requirement database is now :



Adding the last activity, the content of the requirement database is now :



The classification algorithm ensures that all the concepts from the knowledge base are classified according to their properties and inheritance in the overall base is preserved. We will study in the next section how this structure can be used for reasoning on the requirement database. The knowledge base is built under the Knowledge Craft environnement and the classification algorithm is written in Common LISP in this environment.

5. REASONING ON REQUIREMENTS : THE TRACEABILITY IN A SPECIFICATION

The syntactic, semantic and conceptual analysis of requirements has been designed in order to perform reasoning operations the conceptual structures. Our goal in this article is to propose an interpretation of one specific criterion of specification quality : the traceability.

Working from the customer's request, the engineer writes the specifications of the system and its subsystems in a top-down way. This initial definition phase is followed by different phases which should result in the development of several modules to be integrated in a bottom-up way. At each level of the upward phase, tests are perfomed to compare the module's functionality with the client's request. Traceability expresses the links between requirements at different levels. In the downward phase, knowing the origin of a requirement is very useful. In the upward phase, it facilitates control tests and allows identifying any of the initial requirements which have not been satisfied.

5.1 Traceability as a questioning language

The definition of traceability in standards is very intuitive. No objective criterion can be identified in order to "calculate" traceability links and an important amount of knowledge from the domain should be tacken into account in order to propose links between requirements from two separate specification. (cf. Toussaint [9]) shows that such a general objective as building traceability links cannot be reached directly without defining more precise criteria : the following requirements correspond to different semantic networks but define the same functionality :

The system shall monitor the space-vehicle.
The system shall have the capability to monitor the space-

vehicle.
The system shall be able to monitor the space-vehicle.
The system shall provide space-vehicle monitoring.

As a first step in mastering traceability, we propose tools to help the engineer to establish an internal traceability in order to structure requirements within one specification. The "adaptative" way we conceived this tool makes different kind of research and structuration possible depending on the user (engineer) interest : the engineer asks a question about an entity or an activity using a language based on concepts and conceptual relations. The LESD traceability tool will search for requirements involving these entities. The answer to the question is a list of requirements. It is structured following the notion of type.

A question is based on concepts and relations. `Q1` is a question which searches all the requirements which involve the `space-vehicle` concept :

```
Q1 :  ?- [space-vehicle]
```

Question `Q2` search for any requirement whose functionality involves the monitoring of the space vehicle by the IOI-GS :

```
Q2 :  ?- [IOI-GS AGENT monitor OBJECT space-vehicle]
```

The question can be viewed as being a path in the semantic network and the answer as the list of requirements which contain this path or any path involving concept derived from the concept in the question using the $(\text{IS-A}^{-1})^*$ relation. If figure 11a represents the semantic network, figure 11b identifies paths in the network ; the answer will be any requirement containing one of these paths.

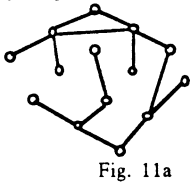


Fig. 11a

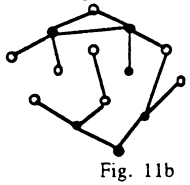


Fig. 11b

It is also possible to use variables in the questioning language. The following question `Q3` defines the variable `?x` as an `entity` (a very general object) and searches any requirement in which an ?x is agent of the monitoring activity which object is of type system. The answer will be sorted following the notion of type of the entity `system`.

```
Q3 ?- {?x IS-A entity} [?x AGENT monitor OBJET system]    (system)
```

Fig. 12 gives an example of answer to `Q3`: `req1` and `req3` involves the concept `system` which is the object of a monitoring activity which agent is an entity ; `Req-4` involves the `on-board-system concept` ; `req7` and `req10` involves the `on-board automatic system` concept.

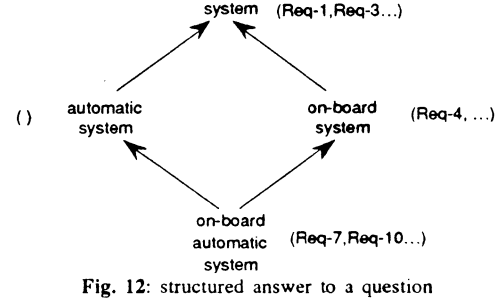


Fig. 12: structured answer to a question

The questioning language helps the engineer in finding requirements following the semantic criteria defined in the question. The structure of the answer takes into account the notion of specificity of the concepts involved in the requirements which follow the criteria expressed in the question.

6. CONCLUSION

The problem of quality in software engineering is a major problem for software development. Although formal methods and languages offer ways to increase quality, natural language is necessary in the preliminary phases of specification, as it is easier to use and more adequate to the expression needs. Consequently, mastering the linguistic features of a technical sublanguage is necessary to increase the quality in software production. Moreover, it should be pointed out that constraints imposed by standards on NL requirements make the domain very propicious to the development of NLP and Linguistic Engineering tools.

The reasoning tool we defined for traceability is the first step towards the identication of linguistic characteristic of quality criteria such as consistency, completude, modifiability. Experimentation of the questioning language will give further elements for the refinement of the intuitive notion of traceability and then for the construction of new Artificial Intelligence tools for Software Engineering.

REFERENCES

[1] Alshawi H., Carter J., Van Eijck J., Moore R., Moran D., Pereira F., Smith A., Pulman S.: *Intern Report on the SRI Core Language Engine*, SRI Cambridge Computer Science Research Centre, Cambridge UK, 1988.

[2] Borillo M, Borillo A, Castell N, Latour D., Toussaint Y, Verdejo MF, Applying linguistic Engineering to Software Engineering : The Traceability problem, *Proceedings of ECAI'92* , Vienne, Austria, Aug 1992.

[3] Brachman R.J., "What IS_A Is and Isn't : An Analysis of Taxinomic links in semantic networks", *Computer*, pp 30-36, October 1989

[4] Dowty D., Wall R., Peters S.: *Introduction to Montague Semantics*, Reidel, Dordrecht, 1981.

[5] Fillmore C.J: *The case for case. in Universals in Linguistics Theory*, edited by Buch and Harms; New York: Holt, Reinhart, Winston, 1968.

[6] Gazdar G., Klein E., Pullum G., Sag I: *Generalized Phrase Structure Grammar*; Harvard University Press, Cambridge, 1985.

[7] Grover C., Briscoe E., Carroll J, Boguraev B: The Alvey Natural Language Tools Grammar (second release); University of Cambridge, Computer Laboratory, TR-162, 1989.

[8] Kobsa A., "Utilizing knowledge : The components of the SB-ONE Knowledge representation workbench" in *Principles of semantic networks : exploration in the Representation of Knowledge*, Sowa J., Ed Morgan Kaufman, 1990.

[9] Toussaint Y., Méthodes Informatiques et linguistiques pour l'aide à la spécification de logiciel, *PhD thesis*, Université Paul Sabatier, Toulouse, France, Oct 1992.