Knowledge Acquisition (1994) **6**, 343–368

Acquisition and validation of software requirements

BRIGITTE BIÉBOW AND SYLVIE SZULMAN

Laboratoire d'Informatique de Paris-Nord, Université Paris-Nord and C.N.R.S., 93470 Villetaneuse, France

(Received 13 July 1993 and accepted in revised form 11 April 1994)

This paper presents a knowledge-based software engineering tool, DASERT (Detection of Anomalies in Software Engineering Requirements Texts), to acquire and validate functional requirements in natural language. The user describes the functional specifications through informal methods, using graphics with comments in natural language. During this elaboration step the system validates the document by processing the comments semantically to detect ambiguities or inconsistencies. To do so it uses natural language processing and knowledge base engineering.

DASERT's kernel is a KL-ONE-like semantic network, which helps the semantic parsing of the comments and their semantic representation. This knowledge base is first initialized by the acquisition of the lexical domain knowledge, then progressively enriched with the domain terminology given by the user and with the requirements knowledge extracted from the user's graphics and texts.

During initialization and enrichment, the network manager validates the knowledge structurally. This ensures the logical consistency of the base which is then checked for inconsistencies and ambiguities specific to the domain of software requirements.

From a software engineering point of view, the originality of DASERT is that it provides a semantic checking of an informal specification by interpreting the natural language comments. From a knowledge acquisition point of view, DASERT allows acquisition from texts to build the kernel of a knowledge base which is then used to guide the semantic parsing of texts during the acquisition of the specification itself. Moreover, the representation formalism provides a unified view of acquisition and validation.

1. Introduction

The interactions between Software Engineering and Knowledge Engineering are becoming stronger and stronger. Ramamoorthy *et al.* (1991) review research in AI technology for SE and include a detailed bibliography. They point out that *many hard problems in software engineering still remain to be solved* and that *software engineering is a knowledge intensive activity.* Bolton *et al.* (1992) give a comprehensive survey of the research on knowledge engineering and requirements engineering; it would seem that to realize an expert system for software engineering requirements is out of reach for the moment. The on-going projects aim to provide the specifier with an intelligent assistance and the research presented here follows these lines. The point addressed is the elaboration of informal requirements and their semantic validation by detection of anomalies.

Since Balzer *et al.* (1978) and Borgida *et al.* (1985), the informal approach to

Based on a paper presented at the 7th European Knowledge Acquisition Workshop, Toulouse, France, September 1993.

1042-8143/94/040343 + 25$08.00/0

requirements is known to be more fruitful and realistic than the formal one. As most development teams have adopted informal methods using graphics and natural language (NL) comments, a lot of software engineering tools have been developed for them. Few of the existing tools include semantic verification, and none is based on the semantic processing of NL comments. Even without a formal frame, though, the detection of inconsistencies becomes possible by using knowledge-based techniques. The goal of DASERT is to detect anomalies in requirements documents by semantically processing the NL comments. Through the description of a system, this paper presents an approach for knowledge acquisition from natural language in the domain of the validation of informal functional specifications in software engineering.

At the beginning, the knowledge base contains knowledge about software engineering, functional specifications and the real world that would be found in any application. From this hand-made kernel the user, aided by DASERT, initializes the knowledge base: this step is that of domain† lexical‡ knowledge acquisition. Within the informal specification method proposed by the system, the user may describe the terminology of the application domain: this is domain terminology acquisition. During the requirements description, DASERT enriches the base with the natural language comments: this is requirements acquisition. In the KADS (Breuker & Wielinga, 1985) terminology, all this acquired knowledge corresponds to domain knowledge. New knowledge is inserted in the base only if it does not introduce any inconsistencies: this is formal validation. The base is then checked to detect anomalies commonly found in requirements documents: this is informal validation, based on some expert knowledge in requirements. The whole process relies on NL processing, so the system includes linguistic knowledge; static knowledge in syntactic and semantic lexicons and in the kernel of the base; dynamic knowledge in the NL processing part of the system.

DASERT provides semantic processing of NL and knowledge base management: its cornerstone is a KL-ONE-like semantic network (Brachman & Schmolze, 1985). The network helps to represent the knowledge expressed by the analysed text and to validate it by verifying its consistency with the knowledge in the base. If consistency is verified, the network is enriched with this new knowledge.

The system, which has been developed for semantic processing of NL and knowledge representation, is presented here from a knowledge acquisition angle. It is shown how semantic processing of NL may help the acquisition of specifications in requirements engineering, and how a semantic network, which is often used in NL processing as a representation tool, may model the static part of the domain knowledge (in the KADS sense) in a structure which provides its logical consistency.

This paper introduces the system and its goal through an extract of a real specification and a typology of the anomalies commonly found in requirements documents. It goes on to describe the knowledge base and NL processing. All the

† Domain means domain of the software engineering application which is described in the requirements with DASERT's help.

‡ *Lexical means "which have an entry in the syntactic lexicon and which have a conceptual* representation", the word "temperature" for example. It must be distinguished from "terms of the domain" which are not necessarily simple syntactic entries (for instance, syntagms such as "temperature level").

acquisition steps are then detailed: aided modeling for the initialization of the base, automatic modeling for domain terminology acquisition, final enrichment with the specification itself and the detection of anomalies. The validation phases which are detailed through the acquisition examples are summarized briefly at the end.

2. DASERT

DASERT allows the building of functional specifications using graphical methods with natural language comments. The originality of this software engineering tool is that it takes the semantics of comments into account. The system is based on the idea that it is possible to represent structured texts of specifications and to detect inconsistencies or ambiguities. It involves textual and graphical interfaces, using a specification method.

2.1. AN EXAMPLE OF A FUNCTIONAL SPECIFICATION TO BE ACQUIRED

In the prototype, a top-down method with natural language comments describing functions and data called SADT (Ross & Schoman, 1977) is used. The SADT diagrams include boxes to describe functions and arrows to describe data; each graphical object is labeled. The functions are broken down into sub-functions; the lowest level functions are commented in NL through a structured textual interface in order to define what the functions do and what data to use. Terms of the application domain may be defined in a glossary. Figure 1 presents an example† of the user interface for SADT including a SADT diagram, the comments describing a function and an extract of the glossary for a clocking application.

As shown in Figure 1, natural language appears in the diagrams for the labels of the graphical objects, in the glossary for terms and their definitions, and in the comments to define functions and data which appear in the diagrams. The definitions are very concise and are just phrases, not real sentences. The language used includes only infinitive sentences and noun phrases. But the phrases can be quite complex, with subordinate clauses.

2.2. THE TYPOLOGY OF ANOMALIES IN REQUIREMENTS TEXTS

Having studied a number of industrial requirements documents it was possible to propose a typology of some of the most frequent ambiguities and inconsistencies in the comment part of the functional specification. They may be classified according to one of the following criteria:

The domain of software requirements
This involves expert knowledge on errors in requirements. For instance, several definitions of the same object must be checked for consistency. If the user gives the following definitions

(1) *badge input*: *badge index*
(2) *badge input*: *time of badge input*

† DASERT is used for French; the text of this figure is a direct translation to exemplify the language used. In what follows French will also be given when the NL processing is involved. The semantic representations will be in English.

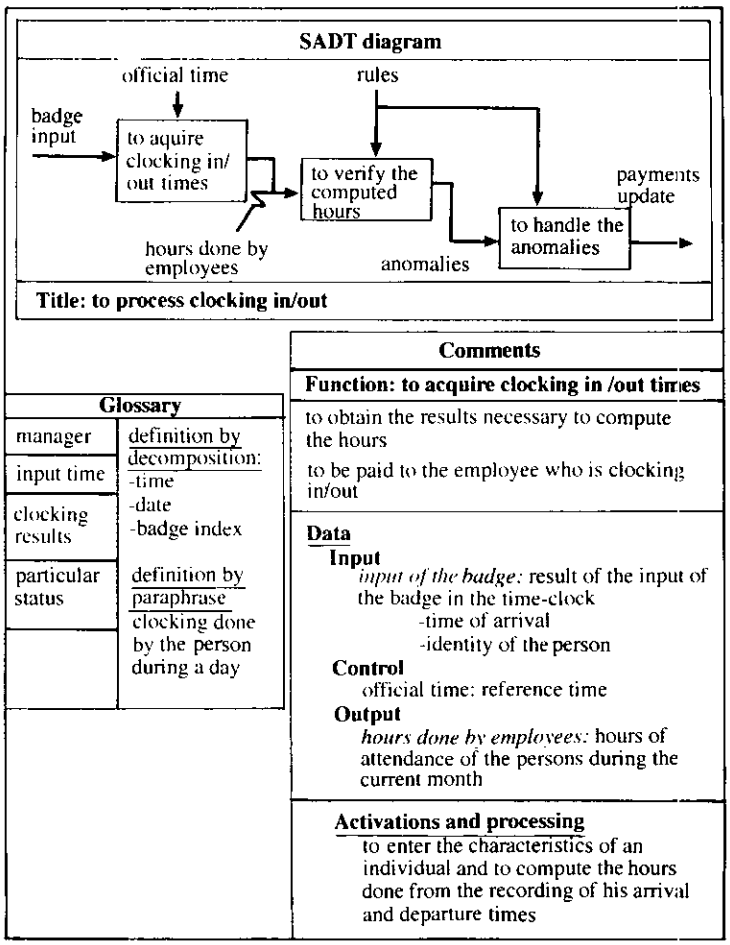

FIGURE 1. An example of a functional specification.

the user has to be warned. The two definitions do not seem consistent because an index is not a temporal object whereas the time is.

However, the two definitions

(3) *badge input*: *time the person entered*
(4) *badge input*: *time the badge was entered*

are compatible. This is because there is a metonomy rule (which allows the use of one word for another) on identifier-identified, and a badge identifies a person.

To allow redundancy increases the expressiveness allowed to the user. It is thus possible to cross-match the multiple definitions to gain a more precise meaning of the defined object; this will show up any inconsistent definitions.

Another example which shows the interest of such a system is to detect when two data with the same definition are given different names, as in

(3) *badge input*: *time the person entered*
(5) *time of arrival*: *time the person entered*

In this case, the user must be warned.

The domain of the application
This involves application domain knowledge, especially terminological knowledge. For instance, *the recording of the sensor was good* would mean "the sensor been recorded correctly", if we are talking about stock control, or "the sensor has correctly recorded", if we are talking about measurement acquisition.

The method
This means either that the syntax of the graphics and their comments must respect that of the method, or that the semantics of the method must be respected. In fact the semantics of the informal graphical methods consisered here is not formally or even precisely defined; if this were the case, formal tools could validate the requirements better than a system based on NL processing. The examples given in this paper are written in SADT, but they could easily be translated into any graphical top-down method with NL comments.

One of the simplest rules in this kind of method is that the label of a data (resp. function) must be interpreted as a computational data (resp. action). Consequently, a data labeled *input badge* would be rejected because it would be interpreted as a physical object and not as a computational data, whereas *badge input* would be accepted because a metonymy rule allowing the use of "action" for "result of action" would replace it by *result of the input of the badge* which is a data.

To give a more complex example, the SADT method explicitly states that any data defined at level A_i must appear at level A_{i+1}, either as such or broken down into subdata; this rule entails verifying that the graphical objects exist, and that their labels are identical, paraphrastic, or in a breaking down relation. The problem is that linguistic processing becomes difficult.

Inherently linguistic criteria
They correspond to commonsense knowledge and to classical problems of semantic processing of NL. For instance, the word *absence* means "non-presence" or "non-existence" and the system must choose. The polysemy (several meanings for one word) is strongly limited by the context. The restriction of the language to the domain of the requirements and of the considered application facilitates the processing. So, the metonymy "content-container" (as in *the whole house applauded*) is not covered: taking a file for its record is not accepted.

2.3. DASERT ARCHITECTURE

The system is made up of three components (Figure 2):

- the *user interface,* which allows the specification to be described through a graphical editor and a text editor, following the chosen method
- the *natural language system,* which classically performs a morphological analysis, a lexical analysis, a syntactic analysis, and finally a semantic analysis using the knowledge base
- the *knowledge base system,* which enhances and validates the base.

A specification session begins by acquiring from the user the graphical objects and

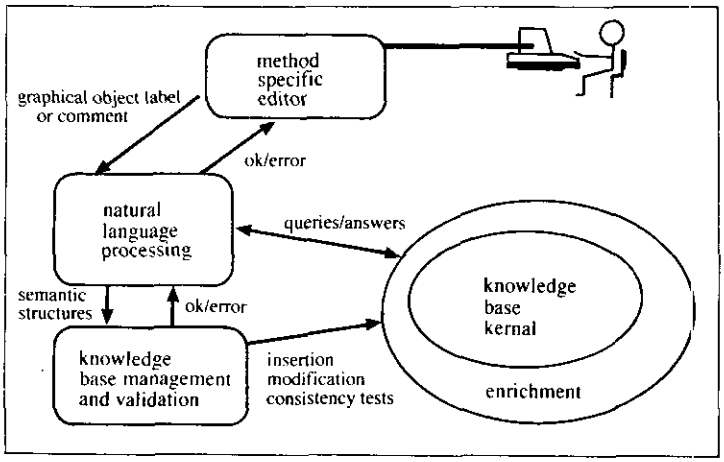

FIGURE 2. DASERT architecture.

comments which describe the functional specifications of the application. On user request the graphical object labels and the comments are sent to the natural language processing system. The comments define functions or data; the natural language system transforms labels and comments into knowledge base objects, with user help in cases of unresolved ambiguity. These semantic objects enrich the base under the control of the base manager. The validation process ends the session.

3. The knowledge base

The knowledge base is a terminological component of a KL-ONE-like semantic network, which generalizes semantic networks and frames and allows classification (Brachman & Schmolze, 1985; Woods & Schmolze, 1992). It includes nodes which represent concepts, and links which describe relations between concepts. A concept is defined by its relations with other concepts. The concepts are organized into a taxonomy of subsumption along which the properties are inherited. Semantic networks are often used in natural language processing and knowledge representation; KL-ONE-like networks are now called terminological systems to underline their sound theoretical basis in terminological logics and to break away from older generation semantic networks. As logic-based inheritance networks, they allow inferences and have a precise semantic interpretation (Nebel, 1990). As terminological languages, they are designed to describe terms (or concepts) by their necessary and sufficient conditions and to classify generic or individual terms. This corresponds exactly to representing descriptive models and classifying or identifying descriptive knowledge in the way Manago *et al.* (1992) pose the problem but in a logic-based frame.

Taxonomies are often used in knowledge acquisition. In KOD (Vogel, 1988), they describe objects of the physical world and actinomies describe the behavior of the expert when reasoning. In DASERT, the taxonomy is used not only for the knowledge representation of physical or abstract objects but also for the conceptual representation of a textual specification. It includes an actinomy describing the

functions but this is nothing like the actinomy of KOD. In (Kietz, 1988) a taxonomy is created from expert-given facts and their relations, expressed in a formal language which is very restricted with respect to KL-ONE.

The knowledge graph project (James, 1991, 1992) relies on the same basic ideas as KL-ONE, as other semantic networks (Sowa, 1991) and as conceptual graphs (Sowa, 1984), which have been used for a long time in the fields of knowledge representation and natural language. The knowledge graphs have been developed especially for knowledge representation in expert systems, and they provide a restricted number of relevant relations. In the latest version, individual and generic concepts are distinguished; the defined relations are either structural or conceptual. The problems of defining the structural primitives [the epistemological level in the sense of Brachman (1979)] and the conceptual primitives (concepts or relations) have already been addressed by Woods (1975), Wilks (1977) and Brachman (1979). KL-ONE-like network offer a few structural primitives with a logic-based semantics allowing the definition of sound and complete deduction algorithms, even if the tractability of these algorithms relies on a trade-off with the expressiveness of the language. The choice of the conceptual primitives depends on the application domain and addresses the conceptual level. The problem is that there is no general method to govern this choice which, moreover, depends on the aim of the developed system.

The KREME knowledge editing environment (Abrett & Burstein, 1987) allows the building and storage of knowledge bases. It implements its knowledge data base in a KL-ONE-like network; it allows frames, rules, procedures and behaviors to be described in the same environment; it includes a classifier for frames. It offers tools for validation and consistency checking. This use of a semantic network as a representation formalism and a knowledge base manager, is the same as that of the DASERT's network. The expressiveness of DASERT's network is much less rich and it could advantageously use KREME as knowledge base editor.

The XTRA project (Allgayer *et al.*, 1989) seems perhaps the closest to DASERT, since it involves NL processing and KL-ONE-like knowledge bases. But in fact it is merely a sophisticated interface used to query an expert system, using NL and touch screens. There are two KL-ONE-like networks, one for the linguistic knowledge and the other for general and domain-specific world knowledge, the system's goals and the user's beliefs and goals. XTRA analyses and generates NL and may modify its model of the user during the querying. The only things it has in common with DASERT are NL and KL-ONE features, its goal being totally different. WASTL (Jansen-Winkeln, 1988) adds to XTRA a knowledge acquisition module using KADS, which is closer to the lexical and terminological aided acquisition that will be described in section 6.

3.1. THE SEMANTIC NETWORK

Let us begin by briefly reviewing the main features of a KL-ONE-like network. Concepts are either *generic* to represent a class of individuals, or *individual* to represent an individual of the domain. Generic concepts are either *primitive* or *defined.* A primitive concept is not wholly defined, the set of its properties expresses necessary but not sufficient conditions for an individual to belong to the class. On the contrary, a defined concept is wholly described by its relations with other

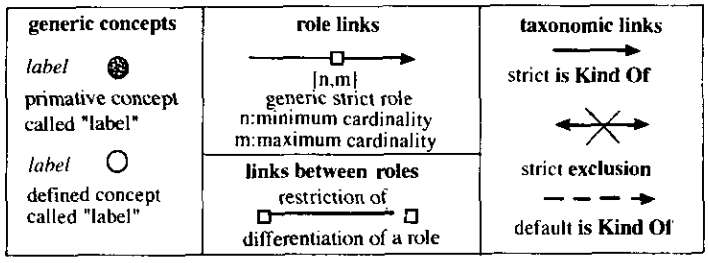

FIGURE 3. Graphical syntax of the main network objects.

concepts. For instance, **human** is usually a primitive concept, but **parent** may be defined as a human who has children: to be a human and to have at least one child is necessary and sufficient to be a parent.

The *role* links describe relations between concepts with value constraints and/or number constraints. When a concept is inserted into the network, the system verifies that it complies with all the constraints expressed by the roles. The *isKindOf* links describe the taxonomic hierarchy along which the properties are inherited. In DASERT's semantic network *exception* and *exclusion* links exist. All links are either *default* or *strict*. An exception link holds on a default link and prevents the inheritance of the default property. The strict links do not support exception links. This extension to KL-ONE has been reported by Coupey (1989). Default and exception abilities have been explained in Biébow *et al.* (1989), and will not really be used in the following.

Figure 3 presents the graphical syntax used in the examples. A classification mechanism allows a concept to be inserted in the best place. Knowing that a concept is characterized by its set of links to the other concepts, the classifier determines the optimum hierarchical relations between concepts from their properties.

Coupey (1989) studied the epistemological level of this network [following the terminology of Brachman (1979)] by defining the knowledge structuration primitives as inheritance, and the logical level by giving a network interpretation in default logic (Reiter, 1980). Figure 4 gives an example with its logical interpretation where, to make things simple, only strict links are defined.

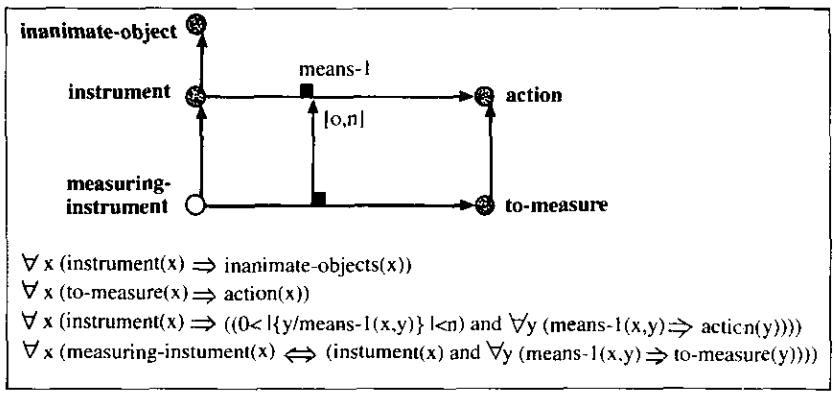

FIGURE 4. Logical interpretation of a simple network example.

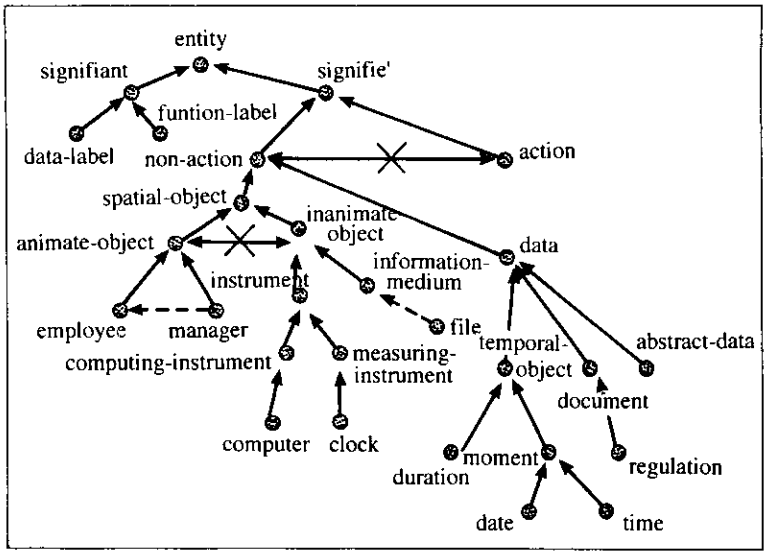

FIGURE 5. Extract of the initial kernel of the semantic network.

3.2. THE INITIAL KERNEL OF THE KNOWLEDGE BASE

Figure 5 is an extract of the concepts of the taxonomy of the kernel; this kernel describes some conceptual primitives of our representation system but is shown without the primitive relations represented by "role" links so as to make the diagram readable. The most general level of the knowledge is common to all application domains.

Of course, the taxonomy is specific to the domain of requirements described by graphics and natural language. For instance, the **significant** concept allows a graphical object label to be represented; it is distinguished from the **signifié** concept which represents what the label models, i.e. its meaning. The **action** and **non-action** concepts reflect the dichotomy of the functional specifications, which describe the data and action on these data. This initial kernel is encoded manually; fortunately it can be reused.

3.3. THE ENRICHMENT OF THE KNOWLEDGE BASE

This kernel is enriched by the acquisition of domain and specification knowledge, using the classification process. The knowledge base manager receives from the NL processing a set of objects which represents the lexical entry or sentence in the network language. Since these network objects may already exist, the base manager begins by looking for the existence of each object in the network. If it does not exist, it is created. The point of insertion of a new concept is determined by the properties of the concept, its roles and their value concepts. A new concept has a father concept in the base kernel called its *connected concept.* The classifier searches the right place for a new concept below the connected concept, by comparing all the properties of the new one with all the properties of the sons of the connected concept. This algorithm is based on the work of Coupey and Fouqueré (1993).

3.4. SOME GENERAL DETAILS ABOUT THE BULDING OF THE KNOWLEDGE BASE

In terminological logics, the semantics of a concept does not rely on its name. Two concepts which differ only in their names are not semantically distinct, unless they are primitive, i.e. not wholly defined. The relations between a primitive concept A and others are necessary conditions for a concept B to belong to the class of A, but not sufficient. As it is difficult to define the most general concepts of a taxonomy by necessary and sufficient conditions, these are often primitive. This implies that the semantics of the concept network description does not exactly match the intuitive meaning of this concept which is expressed by its name. The name of a primitive concept allows a human user to intuitively complement the formal description, and it is sometimes necessary to look at the network to understand the exact meaning of a concept. Most of the concepts which define the vocabulary of the application of a specification are primitive; those which represent the texts of the specification are defined, i.e. wholly defined by their links with others, and their name no longer has any intuitive utility. These concepts are prefixed by a "$" in the following (see for instance Figure 11). This is the name of their connected concept, the first of their fathers without a prefix, which gives a more immediate understanding.

Using primitive concepts limits the possibilities of the classifier. Since it works only with sufficient and necessary conditions, the classifier can put a concept B under a primitive concept A only if the description of B says so (or if it can be deduced from this description). So, the classification is wholly automatic only for the representation of the texts of a specification, but the classifier may suggest a classification to the user when creating the domain knowledge base, as shown in section 5.

Some natural language processing systems using terminological logics represent text by individual concepts, mixing text syntax and semantics together in the network structure. In our knowledge base, the few individual concepts correspond to individuals in the described domain, for instance "the personnel management file". Specification texts describe classes of individuals rather than single individuals: employees, badges, actions, etc. To represent "the badge of the employee" with an individual concept would lead to representing "the badge of the employee with a specific schedule" by another individual concept, thus losing the specialization relationship between them. As almost every syntagm may be specialized by adding a noun complement or a prepositionnal group, the syntagms will be represented by generic concepts and the relation between them will be expressed by an isKindOf link between the generic concepts. The choice in DASERT is not to represent texts, but the domain these texts describe, even if it is sometimes close.

4. Natural language processing

Studying functional specifications showed us that the syntax of the language used includes essentially infinitive sentences and noun phrases, and that the texts are strongly structured into labels of graphical objects and definitions of functions, terms or data; it appears useful to get a set of structures called "templates" which correspond to these structures. Each template specifies a semantic interpretation of

its contents: graphical object label, term definition, function definition, data definition, definition by paraphrase, definition by examples, definition by decomposition, etc. Inside a template, the language is hardly constrained. The content of a template and its structure are sent from the editor to the NL processor, which is classically made up of three steps:

a) morphological and lexical analysis
b) syntactic analysis
c) semantic analysis.

The morphological and lexical analyser has to bind each word in the sentence with an entry of the syntactic lexicon described below.

Because several analyses may be given for one definition, the syntactic analyser creates a graph of syntactic dependence where each node is a syntactic description of a word, and each vertice is a possible dependency relation between two nodes.

The last step is semantic and is the most important one in DASERT. It provides the representation of the text that will enrich the knowledge base. The system works under the compositionally hypothesis: it represents a phrase from the representation of its words and their relations. This representation is structured as a chain of concepts and semantic relations, beginning with the concept representing the "head" word of the sentence (noun if noun phrase, and usually verb if verb phrase).

What the semantic process does is:

- link each lexical word in the sentence with its *connected concept* of the network kernel
- transform the grammatical functions given by the syntactic parser into semantic cases using a case grammar
- define a set of objects which represents the interpretation of the analysed sentence, in the semantic network language. These objects are descriptions of subconcepts of the connected concepts whose role links are usually the expression of the semantic cases. This semantic description is sent to the base manager which has to insert it in the knowledge base.

Linguistic knowledge is thus found in the syntactic and semantic lexicons, the kernel of the knowledge base and the procedural rules of the semantic analyser.

- In the syntactic lexicon, each word has as many descriptions as syntactic uses. An entry may be the root of a French word such as "fin", for example, which could be either the beginning of the verb "finir" (*to complete*) or the noun "fin" (*the end*). It may be a compound word such as "plage horaire" (*time slot*). For each syntactic use the entry in the semantic lexicon is given.
- The semantic lexicon binds an entry either to a kernel which represents it directly, or to a rule to disambiguate multiple interpretations of a word when the semantic cases in the network are not sufficient (the context-sensitive interpretation of the word "*absence*" may signify "*non-presence*" or "*non-existence*"), or to a rule to explain the representation of the entry by a more complex semantic network structure than a single concept (such as the representation of the adjective "*individual*" by an **owns-1** role with value concept **person**).

- The kernel of the base contains generic concepts which often correspond to the semantic representation of lexical knowledge.
- The semantic analyser uses two kinds of procedural rules: general rules of semantic interpretation as for nominalizaton or for support† verbs, and specific rules bound to one entry of the semantic lexicon.

Apart from its object implementation and its semantic representation, the NL processing in DASERT offers no more particular features than do other systems of lexical or terminological extraction from texts or dictionaries, as for instance Slator (1989). The whole process and especially the semantic rules will be illustrated throughout sections 5 and 6. For more details on the NLP part, see (Biébow & Szulman, 1991).

5. Domain knowledge acquisition

The system helps the knowledge base administrator to define the domain vocablary. From existing documents, he takes out the relevant terms with some of their domain specific uses. This process can been done automatically, using for example LEXTER (Bourigault, 1992) (but this tool does not structure the knowledge at all).

The terms to be acquired may be either words or phrases. In DASERT, the concepts corresponding to a lexical word are acquired via a specific lexical acquisition tool which uses NL processing but offers a particular interface. It is the job of the base administrator to initialize the base with the domain vocabulary. There is a a terminological acquisition tool, the glossary, to define domain phrases, because this seems to be a specification task. The specifier usually gives a complex definition of the term, not only its lexical expression. But of course, the administrator may use he same tool to define compound nouns or set expressions of the domain. So lexical acquisition is distinguished in the following from terminological acquisition, although NL processing and the enrichment of the base are the same.

A term is a candidate for inclusion as a new kernel concept to describe the semantic interpretation of lexical entries in the application domain. For example, in a clocking application, the French verb "pointer" (in current use to *point out*) has to be defined as *to clock in/out.* A concept is described by its name (a word) and by its roles (its semantic relations with other concepts). A new concept may be inserted into the network only if it can be distinguished from all the previous ones. If, however, the administrator does not want to or cannot distinguish two concepts other than by their names, they can be defined as primitive concepts.

The system performs syntactic analysis on the syntagms selected by the administrator and asks him to define the unknown words in the syntactic lexicon, through a query/answer dialog. From the syntactic relations found between words it proposes a semantic interpretation of these relations. Finally the administrator fixes the semantic relations between existing concepts and the candidates (new concepts).

† A support verb is used to make a verbal periphrase, and functions as a semi-auxiliary: "faire un voyage" (*to make a trip*), "effectuer" (*to make*), "faire" (*to do*), "réaliser" (*to perform*) are support verbs.

The system asks the administrator to specify the value or the cardinality of those roles that it has not been able to determine from the analysis of the syntagms.

If the concept does not exist or if it is defined as a primitive to be distinguished from another concept, the system classifies it in the hierarchy from its semantic relations. This classification is proposed to the administrator who may accept, refuse to modify it.

5.1. LEXICAL KNOWLEDGE ACQUISITION

Let us suppose the administrator wants to insert initial concepts for the actions "vérifier" (*to verify*)† and "valider" (*to validate*) which even those working in the knowledge acquisition domain find difficult to define. The following sentences are found in the requirements:

"vérifier chaque année les badges individuels" (*to verify each year the individual badges*)

"vérifier chaque semaine l'horlage de la pointeuse" (*to verify each week the time mechanism of the time-clock*)

"vérifier l'enregistrement du pointage" (*to verify the recording of the clocking in/out*)

"la validation du pointage est faite par le pointeau" (*the validation of the clocking in/out is done by the supervisor*)

"le chef de service valide l'anomalie, retard du car" (*the manager validates the anomaly, late arrival of bus*)

"la pointeuse ne peut pas valider les résultats du pointage" (*the time-clock can't validate the clocking in/out results*).

Let us suppose the base involves the **action** concept as in Figure 6 and the semantic representation of all the words used.

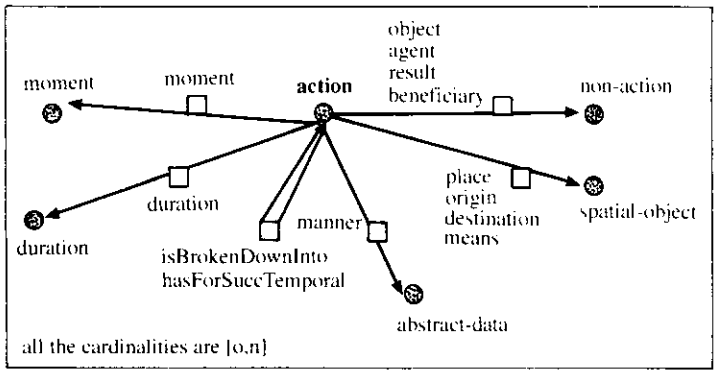

FIGURE 6. The **action** concept.

† Throughout this paper the following conventions have been adopted: the text that is analyzed appears between quotes, either in normal type (French) or italics (literal English translation); concept and role names appear in bold type.

action	agent	object	moment
vérifier (*toVerify*)	?	badge (*badge*) horloge (*clock*) donnée-abstraite (*abstract-data*)	année (*year*) semaine (*week*)
valider (*toValidate*)	pointeau (*supervisor*) chef de service (*manager*) pointeuse (*time-clock*)	donnée-abstraite (*abstract-data*)? anomalie (*anomaly*)	

FIGURE 7. Accepted values of the semantic cases of **toValidate** and **toVerify.**

The system analyses the sentences above first syntactically, then semantically. The following semantic relations are found, using the same principles as Condamines (1992) for the syntagmatic relations but only for the verbal phrases. Figure 7 gives all the values of the semantic cases encountered in the parsed sentences for each concept to be defined. ? indicates that there was no occurrence of this semantic case. The name of the new concept is usually the word itself, unless it is specified by the manager.

The system generalizes the value of the semantic relations found and proposes a concept which subsumes all the concept values found. For example it gives the **non-action** concept for the **object** role value of **vérifier** (*toVerify*), because the **non-action** concept subsumes **badge** (*badge*), **horloge** (*clock*) and **donnée-abstraite** (*abstract-data*).

The system describes the new concepts in the semantic network language. In the examples, the connected concept of "valider" (*to validate*) or "vérifier" (*to verify*) is **action.** Their properties are the roles described above. The NL processor sends the network objects to the base manager which inserts the new concepts in the base, one after the other. Then it presents the set of inherited roles and their value concepts to the administrator for confirmation of the role values that were either found by the NL system or inherited. The administrator may specify the cardinality or the value concept of a role and describe them as strict or default, as excepted or not.

Insertion of toVerify

Since the **action** and **toVerify** concepts differ only in name, the system asks the administrator if it can consider them as identical. The administrator may accept, refuse or modify the analysis. Let us suppose that the administrator refines the

action	agent	object	moment
vérifier (*toVerify*)	?	non-action (*non-action*)	moment (*moment*)
valider (*toValidate*)	objet-spatial (*spatial-object*)	donnée-abstraite (*abstract-data*)	?

FIGURE 8. Generalization of the values of the semantic cases of **toValidate** and **toVerify.**

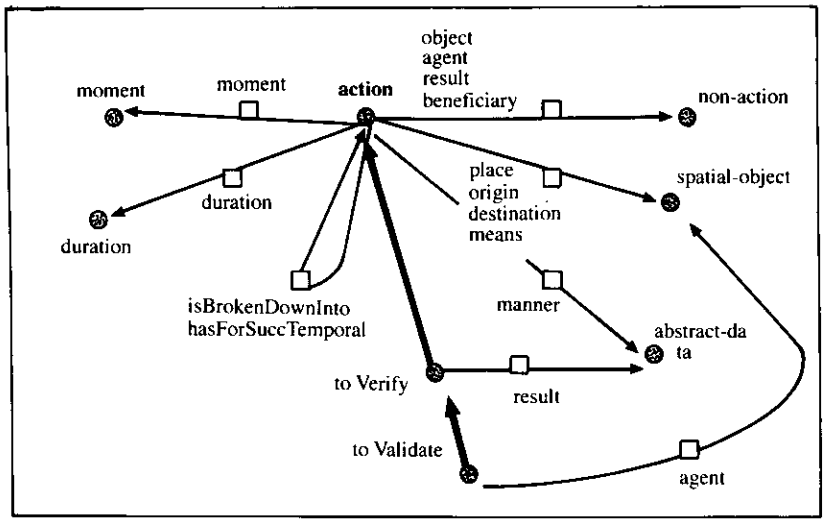

FIGURE 9. The kernel base below **action** after the insertion of **toVerify** and **toValidate.**

result role value of **toVerify** and gives it the restricted value **abstract-data** because, in the domain in question, the result of a verification is an item of information.

Insertion of toValidate

The base manager now suggests classifying **toValidate** as a brother of **toVerify** because its **agent** role value is restricted to the **spatial-object** concept. If the administrator decides to also restrict the **result** role value of **toValidate** to **abstract-data,** the base manager will classify **toValidate** under to **Verify** as shown in Figure 9.

5.2. TERMINOLOGICAL KNOWLEDGE ACQUISITION

Once the kernel of the knowledge base has been defined for an application, DASERT is ready for specification tasks. The user may build a glossary of the application terms, which are composed of lexical items already represented by basic concepts of the kernel at the lexical knowledge acquisition step. The enrichment of the base with the representation of these application terms is called terminological acquisition.

Let us describe the insertion of the term "heure de pointage" (*clocking in/out time*). The NL system analyses this noun phrase. The syntactic analyser gives the following outputs: a noun "heure" (*time*) and a nominalization "pointage" (*clocking in/out*) bound by a complement noun relation. The head of the phrase is "heure" (*time*). The semantic analyser represents the term by a **$time1** concept which has a **moment-1** role with the **toClock** concept value as shown in Figure 10.

6. Specification acquisition and validation: a detailed example

The goal of this section is to show how the system helps the specifier during the acquisition of the specification, through an example of multiple definitions of a data. We have chosen an example from the requirements of a clocking system written in

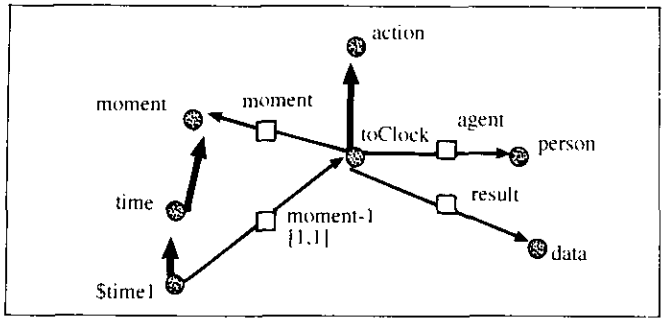

FIGURE 10. Insertion of the term **clocking in/out time** represented by the **$time1** concept.

SADT by an industrial partner. The goal of the clocking application is to compute the hours of attendance of the company's employees. When an employee goes in or out of the work place he puts a personal badge in a time-clock. The system records the clocking in/out times and computes the hours of attendance.

We will focus on a class of data, labeled "résultats du pointage" (*clocking in/out results*), which is defined in several different functions. The problem is that the requirements used to specify the application formally or to implement it are totally confusing because of the data:

- in some functions, the term "résultats du pointage" (*clocking in/out results*) designates the same object, but in more or less detail.
- in others, which are not at the same level of refinement, this name designates different objects.

Even if the first point is acceptable, the second one is in contradiction with good specification or programming practices. These ambiguities and inconsistencies have to be removed, for instance by giving different names to different objects and by making their links, if any, explicit. This work must be done at the requirements step, but not at the programming one, when each programmer decides how to implement this shared data. We show in the following how the system detects these inconsistencies.

6.1. A FEW DETAILS

The requirements define 15 functions in detail with one page of comments for each. The data "*clocking in/out results*" is defined eight times, including six different definitions. We give just four, of which the others are simply paraphrases, in a totally arbitrary order. They are given in the original French form with a literal translation.

résultats du pointage
(*clocking in/out results*)
(1) heure, date, indice, code erreur (*time, date, index, error code*)
(2) heure, date, indice du badge, code erreur (*time, date, index of the badge, error code*)

(3) pointages effectués par un individu au cours d'une journée (*clockings in/out done by an individual during a day*)
(4) pointage du jour et résultats du traitement de la veille (*clocking in/out of the day and results of the processing of the previous day*)

(2) is more precise than (1) concerning the index; implicitly, these are the results of one particular clocking action, corresponding to the introduction of the badge. In (3), all the clocking results of a given day are considered. In (4), it is the results of two consecutive days. "*Clocking*" is used with the meanings "*action*" in the label of the data, "*result of action*" in (3), "*processing of result of action*" in (4).

When comparing (1) and (2), the system will detect the extra information on the index. It will ask the user if the previous data was underdefined; if so, it will replace the previous definition by the new one; if not, it will refuse to represent two definitions for one data and will ask the user to choose.

When comparing (1) or (2) with (3), the system will represent on the one hand only one clocking action, and on the other hand a set of clocking actions. When comparing (3) and (4), an inconsistency is also detected on the temporal domain of the set of actions. In these last two comparisons the system will also refuse the multiple definition of the data and will ask the user of choose.

Note that the order of the comparisons does not matter: to compare (1) or (2) with (4) will lead to the same inconsistency as with (3).

6.2. THE NETWORK AFTER ACQUIRING THE USER GRAPHICS

When the user defines a new SADT object through the graphical editor, the identity of the object is sent to the network with its label. The network builds a new concept under the function or data label concept. After parsing the label, its representation is bound into the network to the previous concept by a role which is a restriction of the **signifié** role of the **significant** concept. If the label has already been parsed, a corresponding concept exists in the network and there is no enrichment.

In Figure 11, the graphical object label is represented by the **$label1** concept and its meaning is represented by a concept named **$data1**. This last one is a kind of data which is the value concept of the result role of the action **toClock**; as playing this role is a necessary part of its definition, an inverse role **result-1** is built. The fact that

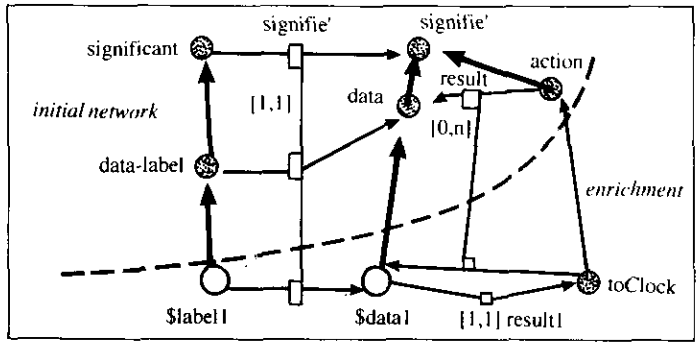

FIGURE 11. Enrichment by the label "*clocking in/out results.*"

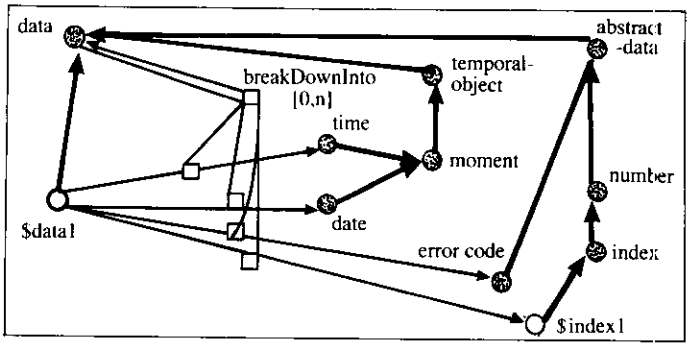

FIGURE 12. Representation of the definition of **$data1** "*clocking in/out results*" by decomposition.

"*results*" is in the plural is interpreted here by default as one of the expressions of a class description: "*managers are employees*" would also be interpreted by generic concepts. The interpretation of plurals and definite and indefinite articles raises a lot of difficulties in NL processing.

6.3. EXAMPLE OF ENRICHMENT WITH A COMMENT

Now that the label has been inserted, the user comments on the graphics. In the function F-1, the data "*clocking in/out results*" is described in a decomposition template as in definition (1) above. The default interpretation of the plural in "*clocking in/out results*" is made explicit at the level of this definition by means of the decomposition template. It should be noted that, although a plural may usually be interpreted by a generic expression, as in Figure 11, it may sometimes be interpreted by an explicit set concept as in "*the number of employees is increasing*", or by an explicit set of roles as in "*clocking in/out results include time, date, identity of the person*". The plural of "*clocking in/out results*" which appears here in a decomposition template, describes the data as a set of several objects and is represented by a set of **breakDownInto** roles, as shown in Figure 12.

In another function F-2, the user gives the same definition of "*clocking in/out results,*" but adds a precision to the index, as in definition (2) of 6.1. The network representation of "*index of the badge*" will be as in Figure 13.

In order to represent this precision, the enrichment module has to replace the old value of the inherited **breakDownINto** role of **$data1** "*clocking in/out results*" by

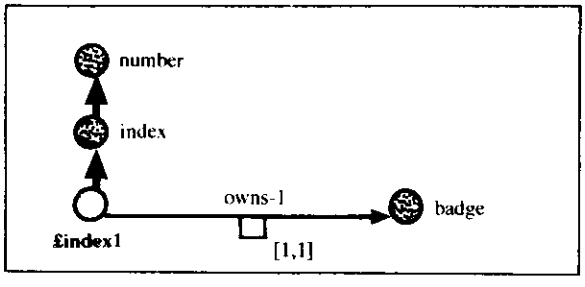

FIGURE 13. Representation of "*index of the badge.*"

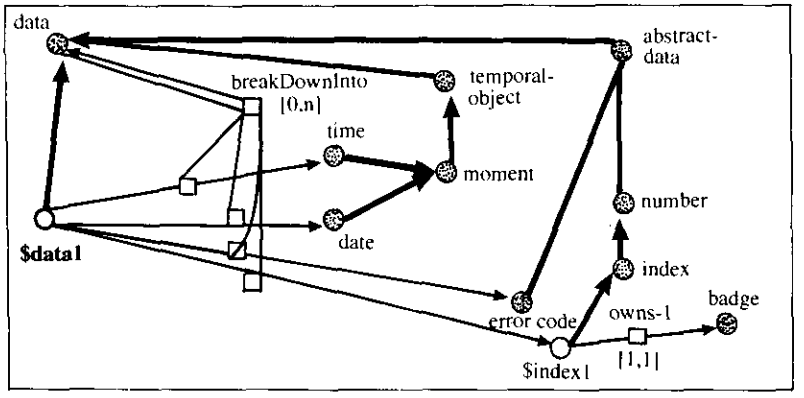

FIGURE 14. Representation of the two definitions of **$data1** by decomposition.

the new representation. Before any modification, the module asks the user if it is correct. If it is, the final result is that given in Figure 14.

6.4. AN EXAMPLE OF INCONSISTENCY DETECTION

In a third function F-3, the user defines the data "*clocking in/out results*" in a data definition template by:

pointages effectués par un individu au cours d'une journée (*clockings in/out done by an individual during a day*)

What follows is the parsing which leads to the interpretation "*results of clockings in/out of a person during a day*", and an informal presentation of the principal rules used for the analysis.

- "*day*" is the complement of "*done*" and not of "*clockings in/out*", and it plays by default a "*duration*" role because of the preposition "*during*".
- "*individual*" is translated into the concept **person** and plays an **agent** role for "*done*" because of the past principle and of the preposition "*by.*"
- "*to do*" is a support verb of the action "*toclock in/out*", so it is not represented and its complements become complements of "*to clock in/out*".
- The action "*to clock in/out*" is defined as instantaneous, so its time complement is not a duration but a moment.
- "*clockings in/out*" is a nominalization in the plural which means either "action" or "result (of the action)" or both. As it is used in a definition template of a data, it must be a data; though an action cannot be a data its result can (see Figure 5). So, "*clocking*" is interpreted by "*clocking in/out result*". But "*clocking*" must also be interpreted as an action, because of its complements. This leads to representing "*clockings*" with both meanings, so by two concepts **action** and **result (of the action).**
- When a nominalization in the plural gives rise to two concepts **action** and **result (of the action)** the plural is on both **action** and **result (of the action).**

Each point raises difficulties in NL processing and needs shrewd rules in order to be processed. For instance, the rule on the support verbs leads to the disappearance of

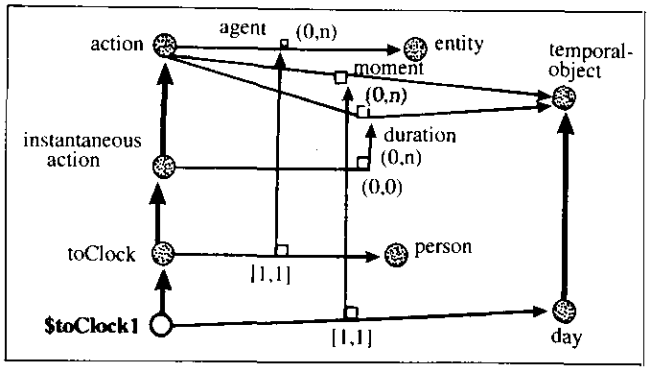

FIGURE 15. Representation of "*clocking in/out by a person during a day.*"

the verb and the transfer of its complements to the nominalization it supports. The fact that an action may be either instantaneous or lasting allows the right interpretation of its time complement. The possible metonymy between **action** and **result of the action** is very common in requirements, but to be obliged to interpret one word by two concepts is not an obvious process. As we have seen in section 4, all the necessary linguistic rules are contained in the semantic lexicon and the semantic analyser, which uses linguistic descriptions of some kinds of words (such as nominalization) and also the semantic network.

To make it easier to read, Figure 15 shows the representation of the partial noun phrase "*clocking in/out by a person during a day*" by the **$toClock1** concept. The **toClock** action is represented as being instantaneous with only one **person**-type **agent**; these properties are inherited by the new concept, which specializes **toClock** by adding a precision to the **moment** role. Indeed, **toClock** cannot accept a **duration** role that was given by default but accept a **moment** role that is allowed for interpreting "*during.*"

In order to represent the whole noun phrase, the plural must be processed. In the right part of a definition template, the plural on the noun phrase is explicitly represented by a **set** concept. A rule about the expression "*set of results of actions*" gives its common interpretation "*set of results of each action*". This finally leads to the interpretation "*set of results of each clocking in/out of a person during a day*".

The analyser knows that the relation "*set of...*" corresponds to an **element** role. The concept **set** must be specialized by a new **$set1** concept with an **element** role restricted to **$data11** which represents the meaning of "*results of clocking in/out of a person during a day*". Finally, the definition template is translated into an **isDefinedBy** link between the data "*clocking in/out results*" and the set specialization, respectively **$data1** and **$set1** of Figure 16.

Detection of inconsistencies

The enrichment expressed in Figure 16 implies that the data "*clocking in/out results*" would be a result of an action **toClock**, defined by a set of results of an action **toClock**. The validation module detects an anomaly because a set seems to be confused with one of its elements, and the semantic network is not enriched. The system warns the user that the final interpretation "*set of results of each clocking*

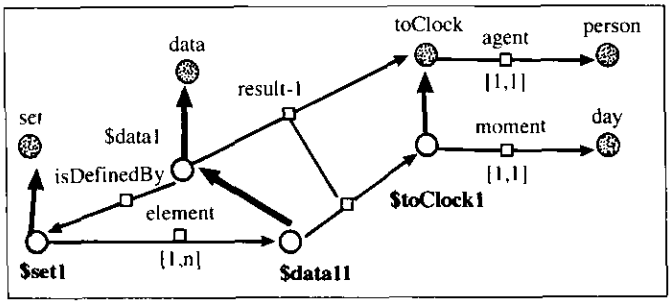

FIGURE 16. Representation of "*clocking in/out results*" by "*set of results of clocking in/out of a person during a day.*"

in/out of an individual during a day" leads to this anomaly. The user has to decide if it is an error or not; it is possible in this case either to rewrite a more explicit text or to modify the label of the data, which would be better.

Analysing definition (4) of 6.1 would lead to the detection of an inconsistency on the **moment** role of the clocking in/out action referred to in the label (a precise day) and in the definition (this day and the day before).

7. Validation

We agree with Bolton *et al.* (1992) that a formal representation of the requirements makes validation considerably easier, by avoiding ambiguities and inconsistencies, and allowing inferences and deductions. As they have said, a representation such as ours can easily be understood by the user or be transformed; translated into natural language for instance. But from a knowledge-based point of view, the problem of requirements validation cannot be solved entirely by the choice of a requirements knowledge representation.

We use the term requirements validation with the general meaning that Laurent (1992) gives to the term validation for knowledge-based systems. Validation covers a formalizable process called "objectie validation" or "verification" and also a non-formalizable process called "subjective validation" or "evaluation." In DASERT, verification is performed by the network manager and evaluation by a validation module which detects inconsistencies or ambiguities specific to the software specification domain.

7.1. VALIDATION AT THE LOGICAL LEVEL BY THE SEMANTIC NETWORK

From a logical point of view, the knowledge base is assumed to be unambiguous and consistent. Coupey (1989) shows that under some constraints (no cycle on default links) there exists an extension of the network in default logic (Reiter, 1980) (that provides the consistency) and that there is only one extension (so there is no ambiguity).

Each time an object is added during the acquisition process, the network manager verifies the structural constraints which guarantee the existence of one and only one

extension. It verifies syntactic contraints including the respect of the syntax of the network language, and the fact that it does not create a concept which already exists. It applies deduction rules to:

- detect taxonomic contradictions, such as deducing that a concept B is, and is not, a kind of A.
- verify the value and cardinality of roles, for example that the **signifié** role value of a **data label** must be **data.**
- detect forbidden cycles which would lead to the existence of several extensions in Reiter's sense.

The choice of this KL-ONE-like network makes the objective validation of the knowledge base and the verification of its structural correctness the same. The fact that the network managers guarantees the consistency of the base also underlies the work of Hors and Rousset (1993). This is so when the network is a terminological component; other KL-ONE-like networks (Nebel, 1990) include an assertional component which is not structured and in which the predicates are terms of the terminological component. These networks, which are hybrid, provide a greater expressiveness, with a more expensive deduction mechanism. In addition no structural constraint can help in the validation of the assertional rules, and the usual validation techniques (Ayel & Rousset, 1990) have to be applied.

7.2. VALIDATION AT THE CONCEPTUAL LEVEL

The knowledge base is validated from a logical point of view by the choice of the representation formalism. But this does not solve the problem of the "conceptual" correctness of the base according to the classification of Brachman (1979), or of the "knowledge level" according to Newell (1982), since the choice of conceptual primitives does not depend on a particular formalism. No formalizable process can guarantee that the choice of conceptual or knowledge primitives is relevant, that the domain structure is acceptable or that the knowledge base is correct from the point of view of the specifier. All these notions are qualitative and fuzzy; we shall call the process which validates them and which is part of the subjective validation or evaluation of Laurent (1992), the conceptual validation.

DASERT evaluates the network from a conceptual point of view to prevent anomalies in the requirements. This involves some expert knowledge on the requirements domain. There are two distinct ways, both involving the checking of the network to detect anomalies depending on the domain interpretation of the name of the network objects.

The first depends on the expressiveness of the network itself. For instance, the transitivity of the taxonomic relation allows inferences in the network; the transitivity of the meronimic relation part-of does not and is considered as depending on the interpretation of the name of the **isPartOf** role. This is not the case in all semantic networks. So it is the concern of the validation module to verify some simple constraints: circularity on some roles is forbidden, such as **isPartOf** or such as **hasForTemporalSuccessor** on **action**; a constraint such as "a set cannot be confused with one of its elements", etc.

The second involves more complex mechanisms which use the previous kinds of validation to detect the kind of anomalies listed in 2.2 in the way illustrated in 6.4.

8. Conclusion

The aim of our knowledge-based system is to prevent inconsistencies and ambiguities in functional requirements by taking into account the semantics of the NL part. These requirements are designed using graphical methods with comments in NL. The system provides NL processing and incremental knowledge representation in a semantic network.

Taking into account the semantics of the informal comments allows the specification to be more sound. The validity of each new definition is checked for its insertion in the network, which confirms the consistency of the whole base from a logical point of view. The final knowledge base represents the NL part of the functional requirements; it may then be validated by checking possible inconsistencies from a requirements expert's point of view.

As we have seen, DASERT is original in that it provides a semantic checking of an informal specification by interpreting the natural language comments. It also allows acquisition from texts to build the kernel of a knowledge base which is then used to guide the semantic parsing of texts during the acquisition of the specification itself. In addition, the representation formalism provides a unified view of acquisition and validation.

An implementation of the whole system exists using Smalltalk80 and has been demonstrated (Biébow & Szulman, 1991). Of course, the natural language processing part is known to be difficult and is limited in this prototype. But it allows the relevance of the proposed approach to be verified.

It seems obvious that the part of the system that is used for the aided acquisition of the lexical and terminological knowledge from text would be useful in acquisition tools. For instance, it is easy to imagine that in a tool such as Open-KADS (Tang *et al.*, 1993) it would help a knowledge engineer to build the knowledge model and to define the term dictionary.

The anomaly detection principles may be applied to knowledge acquisition in several ways. For instance, the problem of consistency of multiple definitions is common when more than one expert is involved, and detection would be useful. One could generalize to detect anomalies during the building of knowledge-based systems, so long as experts would be able to characterize these anomalies.

We would like to thank Rose Dieng for her valuable suggestions and continued encouragement.

References

ABRETT, G. & BURSTEIN, M. H. (1987). The KREME knowledge editing environment. *International Journal on Man–Machine Studies*, **27**, 103–126.

ALLGAYER, J., HARBUSCH, K., KOBSA, A., REDDIG, C., REITHINGER, N. & SCHMAUKS, D. (1989). XTRA: a natural-language access system to expert systems. *International Journal of Man–Machine Studies*, **31**, 161–195.

AYEL, M. & ROUSSET, M.-C. (1990). *La cohérence dans les bases de connaissances*. Cepadues-Editions, France.

BALZER, R., GOLDMANN, N. & WILE, D. (1978). Informality in program specifications. *IEEE Transactions on Software Engineering*, **4**, 94–103.

BIÉBOW, B., COUPEY, P. & SZULMAN, S. (1989). Using exceptions in a semantic network for a natural language application. In *Proceedings of the 2nd European Conference on Speech Communication and Technology* (*EUROSPEECH-89*), Paris, France, pp. 34–37.

BIÉBOW, B. & SZULMAN, S. (1991). Validation de cahiers des charges rédigés en langage naturel. In *Proceedings of the 11th International Conference on Expert Systems and their Applications*, Avignon, France, pp. 255–268.

BOLTON, D., JONES, S., TILL, D., FURBER, D. & GREEN, S. (1992). Knowledge-based support for requirements engineering. *International Journal of Software Engineering and Knowledge Engineering*, **2**, 293–319.

BORGIDA, A., GREENSPAN, S. & MYLOPOULOS, J. (1985). Knowledge representation as the basis for requirements specifications. *IEEE Computer*, **18**, 82–91.

BOURIGAULT, D. (1992). Surface grammatical analysis for the extraction of terminological noun phrases. In *Proceedings of the 15th International Conference on Computational Linguistics* (*COLING-92*), Nantes, France, pp. 977–981.

BRACHMANN, R. J. (1979). On the epistemiological status of semantic networks. In N. V. Findler, Ed., *Associative networks: representation and use of knowledge by computers*, Academic Press, pp. 3–50.

BRACHMAN, R. J., SCHMOLZE, J. G. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, **9**, 171–216.

BREUKER, J. & WIELINGA, B. (1985). KADS: structured knowledge acquisition for expert systems. In *Proceedings of the 5th International Conference on Expert Systems and their Applications*, Avignon, France, pp. 887–900.

CONDAMINES, A. (1992). Aide à l'acquisition de connaissances par la spécification de la terminologie d'un domaine de spécialité. In *Proceedings of the 3rd Journées d'Acquisition des Connaissances* (*JAVA-92*), Dourdan, France, pp. 89–100.

COUPEY, P. (1989). Etude d'un réseau sémantique avec gestion des exceptions. *PhD thesis*, University of Paris-Nord, France.

COUPEY, P. & FOUQUERÉ, C. (1993). Default subsumption in terminological languages. *Internal Report*, University of Paris-Nord, France.

HORS, P. & ROUSSET, M.-C. (1993). A formal framework for structured and deductive knowledge consistency checking. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence Workshop on Validation, Verification and Test of Knowledge-based Systems* (*IJCAI-93*), Chambéry, France, pp. 59–66.

JAMES, P. (1991). Structuring knowledge using knowledge graphs. In *Proceedings of the 5th European Knowledge Acquisition Workshop* (*EKAW-91*), Crieff, Great Britain, pp. 128–139.

JAMES, P. (1992). Knowledge graphs. In R. P. VAN DE RIET and R. A. MEERSMAN, Ed., *Linguistic Instruments in Knowledge Engineering*, Elsevier Science, pp. 97–117.

JANSEN-WINKELN, R. M. (1988). WASTL: an approach to knowledge acquisition in the natural language domain. In *Proceedings of the 2nd European Knowledge Acquisition Workshop* (*EKAW-88*), Bonn, Germany, pp. 22.1–22.15.

KIETZ, J.-U. (1988). Incremental and reversible acquisition of taxonomies. In *Proceedings of the 2nd European Knowledge Acquisition Workshop* (*EKAW-88*), Bonn, Germany, pp. 24.1–24.11.

LAURENT, J.-P. (1992). Proposals for a valid terminology in KBS validation. In *Proceedings of the 10th European Conference on Artificial Intelligence* (*ECAI-92*), Vienna, Austria, pp. 829–834.

MANAGO, M., CONRUYT, N. & LE RENARD, J. (1992). Acquiring descriptive knowledge for classification and identification. In *Proceedings of the 6th European Knowledge Acquisition Workshop* (*EKAW-92*), Heidelberg, Germany, pp. 392–405.

NEBEL, B. (1990). *Reasoning and Revision in Hybrid Representation Systems*. Lecture Notes in Artificial Intelligence, J. SIEKMAN, Ed., Springer-Verlag.

NEWELL, A. (1982). The knowledge level. *Artificial Intelligence*, **18**, 87–127.

RAMOORTHY, C. V., LUIS, M. & YOUNG-CHUL, S. (1991). On issues in software engineering and artificial intelligence. *International Journal of Software Engineering and Knowledge Engineering*, **1**, 9–20.

REITER, R. (1980). A logic for default reasoning. *Artificial Intelligence*, **13**, 81–132.

ROSS, D. T. & SCHOMAN, K. E. (1977). Structured analysis for requirements definition. *IEEE Transactions on Software Engineering* **SE-3,** 6–15.

SLATOR, B. M. (1989). Extracting lexical knowledge from dictionary text. *Knowledge Acquisition,* **1,** 89–112.

SOWA, J. (1984). *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley.

SOWA, J. (ed.) (1991). *Towards the expressive power of natural language. Principles of semantic networks: exploration in the representation of knowledge.* Morgan Kaufman.

TANG, X., RABAUX, E. & MAESANO, L. (1993). Knowledge modeling with Open KADS. *Atelier d'Ingénierie des Connaissances et des Données,* Strasbourg, France, pp. 55–68.

VOGEL, C. (1988). *Génie cognitif.* Masson, France.

WILKS, Y. (1977). Good and bad arguments about semantic primitives. *RR no 42,* AI Department, University of Edinburgh.

WOODS, W. A. (1975). What's in a link: foundations for semantic networks. In D. G. BOBROW and A. COLLINS, Ed., *Representation and Understanding: studies in Cognitive Science,* Academic Press, New York, pp. 35–82.

WOODS, W. A., SCHMOLZE, J. G. (1992). The KL-ONE family. *Computers Mathematical Applications,* **23,** 133–177.