CAHIER DU LAMSADE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
(Université Paris-Dauphine)
Unité de Recherche Associée au CNRS n° 825

KBS LIFE-CYCLE AND VALIDATION

CAHIER N° 133
janvier 1996

C. HAOUCHE [1]
C. ROSENTHAL-SABROUX [1]

reçu : novembre 1994.

[1] *LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16.*

CONTENTS

CYCLE DE VIE ET VALIDATION D'UN SBC

Résumé

Nous nous intéressons aux étapes de validation dans le cycle de vie d'un système à base de connaissances (SBC). Nous nous sommes focalisées sur l'utilisation d'un *modèle conceptuel*, élaboré pendant la phase d'acquisition des connaissances, pour valider un SBC. Cela implique de définir de nouvelles étapes dans le cycle de vie du SBC et de définir les processus de validation inhérents à ces étapes. Plus pratiquement, nous proposons une méthode de test utilisant en partie un modèle conceptuel à la KADS, plus précisément la structure d'inférence, pour tester le SBC. Nous nous efforçons de préciser, pour chaque étape de validation considérée, *quand* cette validation doit avoir lieu, *qui* sont les acteurs de la validation et, enfin, *comment* cette validation doit être menée.

KBS LIFE-CYCLE AND VALIDATION

Abstract

This research deals with validation steps within the KBS life-cycle. Our paper is centered on the use of the *conceptual model*, elaborated during the Knowledge Acquisition phase, for the KBS validation. We will, on the one hand, show the implications of such a use, that is, the need to specify new steps in the life-cycle as well as validating their outputs. On the other hand, we will propose a procedure to use one part of a KADS conceptual model - the inference structure - to test the KBS. We will, as far as possible, and for each considered step of the life-cycle, precise the following points: *When* the system will be validated, that is at which step of the life-cycle, *who* will be validating the system and *how* this validation will be achieved.

1 Introduction

"The primary functions of a software process model are to determine the order of the stages involved in software development " as Boehm said in (Boehm, 1988). For the KBSs, there are essentially two kinds of development methodologies (Dieng, 1990). The first one is the incremental development technique. This methodology is also called *rapid prototyping* (Hayes-Roth *et al.*, 1983). The major steps of this methodology are: identification, conceptualization formalization, implementation, and finally testing. This methodology does not integrate any conceptual modeling. On the contrary, the second kind of development methodologies are model-based and rely on a software development process. The steps involved in these methodologies are *(1)* acquiring requirements, *(2)* building the conceptual model (the analysis phase), *(3)* building the KBS, *(4)* validating the KBS, *(5)* installing and *(6)* maintaining the KBS.

However, the steps involved in a KBS life-cycle, and especially the analysis phase and the validation steps, are more complex than the ones involved in a "classical" software life-cycle. The existing approaches to validation generally do not use the conceptual model which is developed during the analysis phase. Our purpose is thus to integrate a validation step that uses this conceptual model to perform one aspect of the KBS validation. Our study is thus validation-centered and essentially based upon the KADS conceptual models proposed in the KADS methodology (Wielinga *et al.*, 1992).

Considering the whole life-cycle, we will focus on the following points: *When* the validation will intervene, that is at which steps of the life-cycle, *who* is expected to perform validation in the various life-cycle steps and *how* this validation will be achieved.

Concerning the KBS validation, we will take advantage of the *conceptual model* elaborated during the Knowledge Acquisition phase to test the KBS. We consider this conceptual model as a set of specifications for the KBS. This implies *(i)* to handle this conceptual model under an implemented form, *(ii)* to link it to the KBS and *(iii)* to build what we call *Valid Inference Paths (VIP)*. Furthermore, the results of these additional steps need to be validated.

In Section 2, we briefly describe the main steps of a classical KBS life-cycle. Since our study relies on a KADS conceptual model, we also give a short description of it.

In Section 3, we introduce additional steps in the previous KBS life-cycle. Two steps, namely, representing the conceptual model under an implemented form and building the VIP, are added between the phases *building the conceptual model* (2) and *building the KBS* (3). The third step, linking the KBS and the conceptual model, is added after the step (3).

Then we consider the validation steps that are relative to the following steps of the life-cycle: the validation of the conceptual model, the validation of the conceptual model representation, the validation of the inference paths, the validation of the links between the conceptual model and the KBS.

In section 4, we focus on the validation of the KBS. We propose a testing method of the KBS using the Valid Inference Paths and the links between the conceptual model representation and the KBS. This method is illustrated with an example described in 4.

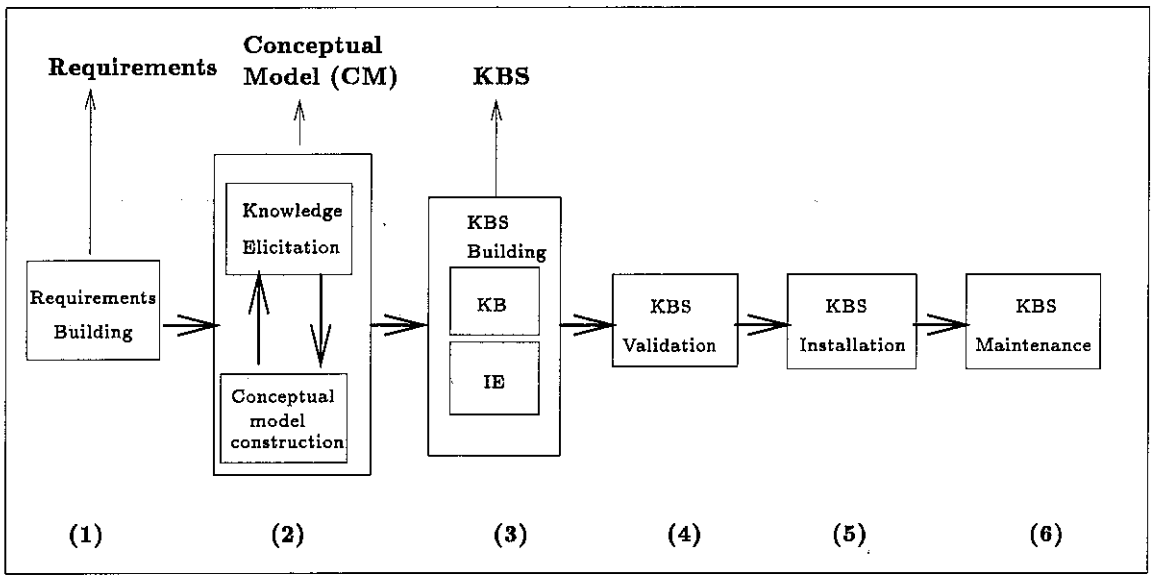

Figure 1: A Classical KBS Life-Cycle

2 Classical main steps in a KBS life-cycle

Some studies have already dealt with KBS life-cycle (Halker & Welz, 1989; Chabaud *et al.*, 1991; Wielinga *et al.*, 1992; Dieng, 1990). The main steps that are identified are as follows (see Figure 1):

2.1 Acquiring requirements

This step is an informal one. It is conducted by means of rough interviews to identify the user's needs. During this step, the expert, the knowledge engineer and the final user have to define these needs. Then, a decision is taken about the KBS appropriateness to solve the problem.

2.2 The analysis phase

The analysis phase aims at building a conceptual model (CM), that is a high level - implementation independent - description of the KBS. This phase is a cyclical one; it consists, on the one hand, of gathering the data that are relative to the problem. These data concern both the domain concepts and the different ways to use these concepts to solve the problem at hand. This gathering is conducted by the knowledge engineer who can use different elicitation techniques when interviewing the expert and the user (Dieng, 1990). We notice that these interviews are much more detailed than the previous ones and that they are relative to the domain knowledge. On the other hand, the knowledge engineer can use different kinds of methodologies

(KADS (Wielinga *et al.*, 1992), COMMET [1] (Canamero & Geldof, 1993), MACAO (Aussenac, 1989),etc.) in order to structure the knowledge that has previously been acquired and to build a conceptual model (Krivine & David, 1991; Wielinga *et al.*, 1992). Let us notice that the CM building can lead to acquire new data and vice-versa, and this, until the CM is valid.

The KADS modeling layers KADS[2] offers structures for building a conceptual model to guide the knowledge acquisition process. A KADS conceptual model is four-level organized[3], from the least abstract, which describes the domain concepts, to the most abstract, which represents the behavior of the system.

* The *domain layer* is a description of the domain concepts and of the relations between these concepts.
* The *inference layer*. In this layer, *roles* are attributed to the domain concepts. In addition, an *inference structure* describes all the links between the roles and the *inferences* that are involved in the problem solving method.
* The *task layer* specifies the goals assigned to the system and the way inferences are organized to achieve these goals.

2.3 Building the KBS

Despite the availability of a conceptual model, there is no systematic process to build a KBS from a conceptual model. However, some studies identify this phase as an *operationalization* one and propose tools, as the LISA language (Delouis, 1993) and the shell AIDE (Greboval & Kassel, 1992).

Two sub-phases are generally involved to build a KBS. One of them aims at choosing a representation formalism in order to build the knowledge base. The other one deals with the choice of the inference engine. Notice that these two sub-phases are strongly dependent.

2.4 Validating the KBS

Validating a KBS involves verifying the Knowledge Base and testing the KBS results and behavior.

Verification approaches

Various verifications tools have been developed. Most of them deal with detecting incoherencies (SACCO (Ayel, 1987), CHECK (Nguyen *et al.*, 1985)) or with proving the KB coherency (COVADIS (Rousset, 1988)). They also deal with detecting redundancies, potential circularities, subsumed rules (CHECK (Nguyen *et al.*, 1985)).

[1] KREST is the new name for COMMET.

[2] See (Hickman *et al.*, 1989; Wielinga *et al.*, 1992) for a detailed presentation of the KADS methodology.

[3] We used the KADS I methodology but we did not use the strategy layer. Notice that this layer is suppressed in KADS II

Testing approaches

Two main approaches have been used to test KBSs: the black box approach and the white box approach.

a - The black box approach

A test case is composed of a set of data and of the results that are expected from these data. These data are used to run the system. The results obtained are then compared to the expected ones. Notice that this approach deals only with the inputs of the system and with the outputs it provides. It does not consider the structure of the system.

b - The white box approach

The white box approach focuses on the structural aspects of the KBS. It handles a KB structure to study if all the parts of this KB are exercised (Preece *et al.*, 1993). For instance, if the KBS is rule-based, this approach allows to detect the rules which are not used on a given set of test cases, the ones that are often used, etc. Notice that some tools aim at generating test cases automatically; this is the case for SYCOJET (Vignollet, 1991).

2.5 Installation and maintenance

Two possibilities are to be considered. (i) The KBS is completely independent. In this case, it is used with no further effort. (ii) The KBS is to be integrated into a more complex information system. Then, some problems concerning interactions with other modules could occur. The maintenance phase may lead to review the conceptual model.

3 A proposal for validation steps in a KBS life-cycle

Our opinion is that the conceptual model which constitutes a set of specifications for the future KBS, must be used during the validation step. We also think that it is important to link these specifications to the KBS in order to build a KBS that actually corresponds to the initial requirements. Consequently, we propose three additional steps. The first one deals with representing the conceptual model under an implemented form (see (A) on Figure 2). The second one aims at building the inference paths (see (B)) on Figure 2). The third one aims at establishing links between the implemented from of the CM and the KBS (see (C) on Figure 2). The steps (A) and (B) are introduced between the steps *building the conceptual model* (2) and *building the KBS* (3), whereas the step (C) is introduced after the step (3). As we do not intervene on steps *installation* (5) and *maintenance* (6), they are not represented on Figure 2.

(A) Representing the conceptual model under an implemented form

The conceptual model constitutes a set of specifications and as it must be linked to the KBS, we need to handle it under an implemented representation. Thus, each layer of

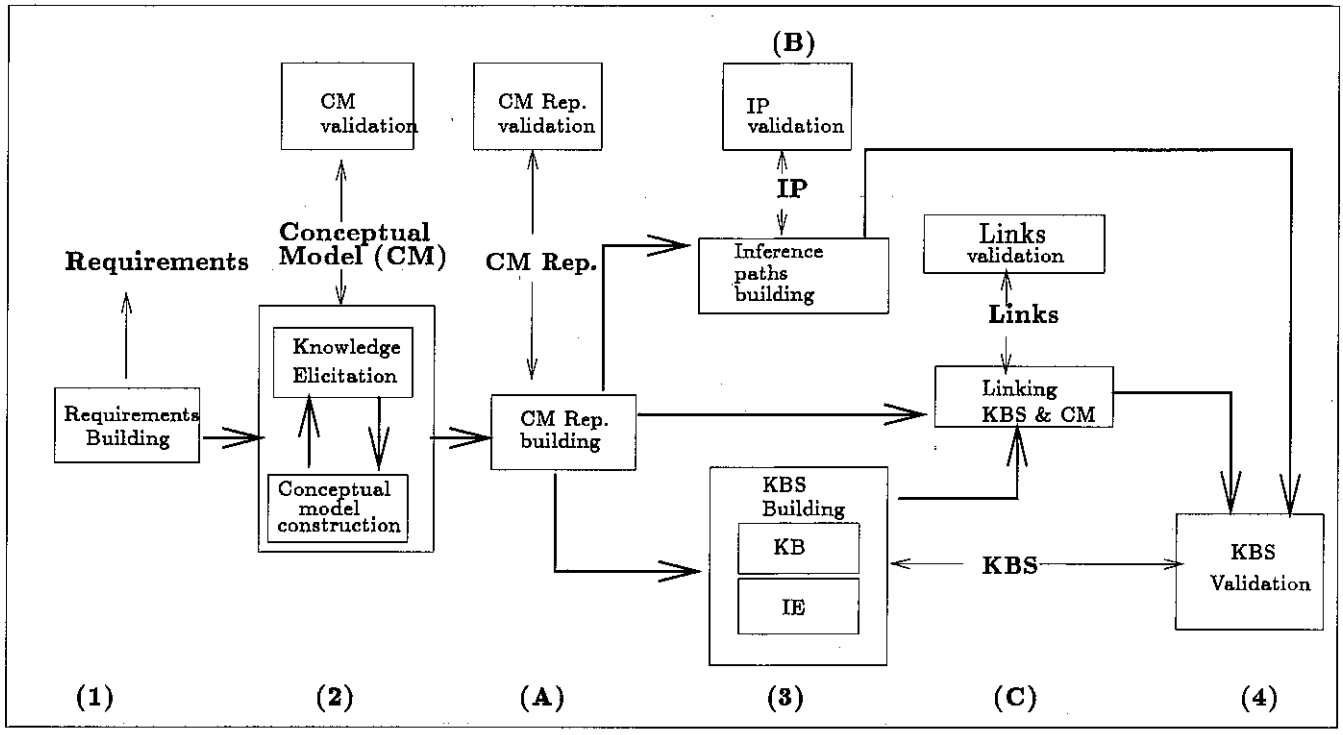

Figure 2: Additional Steps for a KBS Life-Cycle

the KADS conceptual model (the domain layer, the inference layer and the task layer) is represented. This step is performed by the knowledge engineer and/or by a programmer.

(B) Building the inference paths

We use the inference structure representation provided within the conceptual model representation, to derive all the inference paths. An inference path is a sequence of inferences between an initial inference and a final inference. An initial inference has only initial roles as inputs, and a final inference has a final role as output.

(C) Linking the implemented form of the CM and the KBS.

These links are used to build a high level trace during the testing phase. Up to now, there is no, to our knowledge, a method to establish such links. In our work, it is realised by the knowledge engineer who can use his knowledge of both the CM and the KBS.

Each step s of the KBS life-cycle is expected to produce outputs $o(s)$ from inputs $i(s)$. Some of these inputs can be seen as a kind of specifications. The outputs $o(s)$ are represented using a given representation. In our opinion, two kinds of validations are thus to be performed on $o(s)$. (i) A validation, which we call *internal validation*, which aims at verifying that the output $o(s)$ is represented in accordance with the representation formalism used at step s, and

(ii) an *external validation* which aims at assessing that the outputs *o(s)* verify the specifications expressed in *i(s)*.

We will focus on the validation of the conceptual model (2), the validation of the CM representation (A), the validation of the inference paths (B), the validation of the links (C). The KBS validation will be studied in the next section. We will specify the inputs and outputs for each of these steps as well as the kinds of validation which are necessary.

3.1 CM validation

For this phase, the inputs correspond to the CM and to the requirements which have been identified in phase (1). The outputs are a valid conceptual model. The representation formalism on which we focus is the one given by the KADS methodology.

The CM internal validation is performed by the knowledge engineer. As we said above, a KADS conceptual model is structured in 3 layers, namely: the domain layer, the inference layer and the task layer. Consequently, we identify two kinds of validation:

- An intra-layers validation that is a validation of each of the layers in the KADS conceptual model. For example, each inference used in the inference structure must have roles as inputs and one role as its output, according to the inference layer definition. For instance, on Figure 8, the inference *complete* has two input roles which are *domain model 0* and *initial specification 1* and has one output role which is *initial specification 2.*

- An inter-layers validation. It concerns the validation of the links between the different layers. For instance, each inference used in the task layer must be defined in the inference layer (Haouche, 1993) and every role of the inference layer must be linked to a concept defined in the domain layer. On Figure 8, the concepts that are linked to the roles are written in italics next to these roles. As an example, the concept *l-description* is linked to the role *complete specification.*

The CM external validation is performed by the expert, the knowledge engineer and the user. This validation is an informal one, and its goal is to state that the conceptual model satisfies the requirements. The acquired concepts must be suitable, the task hierarchy must be sufficiently detailed, ... For instance, considering the inference structure represented on figure 8, the expert is asked on the sequencing of the inferences. That is, does the inference *complete2* really follows the inference *complete*, etc.

3.2 The validation of the CM representation

The inputs for this step are the conceptual model and the CM representation. The output is a valid CM representation. Let us recall that our aim is to be able to handle the "paper" conceptual model under an implemented form according to the KADS methodology. Thus, we use a frame-based formalism in which every component of each layer is described by a frame. For instance, each inference is represented by the following attributes: input-roles, output-roles,

```
{!initial-specification
    !role:
    !input-for: !create !assemble1 !assemble2 !assemble3
    !output-for: !complete2
    !linked-to-domain: !lexeme-instance
    !linked-to-kbs: }
{ !complete
    !inference:
    !input-roles: !initial-specification1 !domain-model0
    !output-role: !initial-specification
    !linked-to-kbs: !regle-instanciation-composantes }
```

Figure 3:

linked-to-domain, linked-to-kbs). Each role is also described by a set of attributes (input-for, output-for, linked-to-domain) as we can see on Figure 3:

The CM representation internal validation is performed by the knowledge engineer and/or the programmer, and is a syntactic one regarding the representation language. For instance, each frame must be described by the appropriate attributes depending on its type (concept, inference, role, ...).

The CM representation external validation is performed by the knowledge engineer and/or the programmer and is an informal one. Its goal is to assess that the implemented form of the conceptual model corresponds to its paper form.

3.3 Inference paths validation

The inputs for this step are a subpart of the conceptual model representation, that is, the inference structure representation and the inference paths, which are represented by lists, as for example the inference path *(complete, complete2, create)*. The outputs are a set of Valid Inference Paths (VIP).

The inference paths internal validation. This phase aims at assessing that the outputs of the step *building the inference paths* (B) are effectively syntactically correct inference lists. This validation is performed by the programmer/knowledge engineer.

The inference paths external validation. This validation consists, on the one hand, of verifying that the order of the inferences in the lists corresponds to the one given in the inference structure. This is a syntactic validation performed by the programmer/knowledge engineer. On the other hand, an informal semantic validation consists (i) in suppressing semantically incorrect paths, and/or (ii) in adding some "parallel paths" (see the example in Section 4.1). We notice that this semantic validation is performed by the knowledge engineer and the expert.

3.4 Validating the links between the KBS and the CM

We focus now on the links[4] between the KBS and the inference structure. The inputs for this step are the implemented representation of the inference structure, the KBS, and the links between them. The outputs are the valid links, that is associations established between the two structures. These links are represented by the attribute "linked-to-KBS" on each inference and role, and by the attribute "linked-to-CM" on each rule as exemplified on Figure 4.

```
{ !complete
    !inference:
    !input-roles: !initial-specification1 !domain-model0
    !output-role: !initial-specification
    !linked-to-kbs: !regle-instanciation-composantes }
{!regle-instanciation-composantes
    !linked-to-cm: !completer
    !condition:
       (and [?instance !occurrence-de ?mot]
          [?lexeme !entree ?mot]
          (meta-frame ?lexeme ?m)
          (fork
             (then-do (copy-frame ?m (meta-frame ?instance)))
             (and [?lexeme !s-applique-sur ?arg]
                  (then-do [?instance !s-applique-sur ?arg]))
             (and [?lexeme !case-frame ?arg]
                (then-do [?instance !case-frame ?arg])))) }
```

Figure 4: The links **linked-to-kbs** and **linked-to-cm** between the inferences and the rules

The links internal validation consists of verifying that these links are symmetrical. It is performed by the knowledge engineer and/or the programmer.

The links external validation aims at verifying that each inference of the inference structure representation is linked to the KB and reciprocally. This is performed by the knowledge engineer and/or the programmer.

4 KBS validation

The inputs for this step are the valid inference paths, the KBS and the links between the CM and this KBS. The output is the validated KBS. There are many representation formalisms. The most common ones are the rule-based formalisms and the object-oriented formalisms.

[4]We recall that these links are established empirically by the knowledge engineer, relying on his knowledge and understanding of both the KBS and the CM.

The KBS internal validation consists of verifying whether the KB is represented in accordance with the formalism. Many tools have been developed to detect redundancies, subsumptions as well as incoherencies in such knowledge bases. This validation is mainly performed by the knowledge engineer and/or the programmer.

The KBS external validation

We focus here on the KBS test using the IP and we propose the following procedure:

1) The KBS is run on a data case and a code level trace is built.

2) The current inference path (CIP) is built from this trace using the links that are established between the CM and the KBS. This CIP constitutes a "high level trace".

3) The CIP is compared to the set of VIP. Two cases are to be considered:

   a) The CIP $\in$ VIP: the testing procedure is iterated on a new data case.

   b) The CIP $\notin$ VIP: The CIP is shown to the expert. If the latter identifies a valid inference path, this path is added to the set of VIP; else, the links between the CM and the KBS are used to localize the code that has to be revised.

An example

In this section, we present an illustration of the previous validation procedure. This procedure is currently being tested on a real world KBS, AMD (Cavazza *et al.*, 1992), for which we developed a CM (Haouche, 1993). This CM describes a system developed to understand medical summaries. Here, "understanding" means building a representation of the summary, called *a situation model.* In this situation model, all the implicit knowledge carried in the summary is made explicit by using common-sense and medical knowledge. A preliminary task consists of building *l-descriptions*, that is structures that gather the *semantic components* through *syntactic links* between *lexemes.* For instance, the l-description *l-descr25* (Figure 7) gathers the information which relates to the same concept *tumeur* and which is carried by the lexemes *nodule-13* and *gauche-15* (Figure 6).

To illustrate our testing procedure, we will focus on this subtask. The part of the inference structure concerning this subtask is thus presented (see Figure 8). On this inference structure, we can see the initial roles[5] *domain model 0* and *initial specification* and the final roles[6] *intermediary specification* and *complete specification.* Inferences which have initial roles as inputs are called *initial inferences* (*complete*) and the ones which have a final role as an output are called final inferences (*create, assemble1, assemble2, assemble3*).

The initial set of inference paths The following set is built using a procedure which yields all the paths that link an initial inference to a final inference. These paths specify all the intermediary inferences between an initial inference and a final inference.

[5] An initial role is an input to the inference structure.

[6] A final role is either an output of the inference structure or a role that is specified as final by the expert as for instance *intermediary specification.*

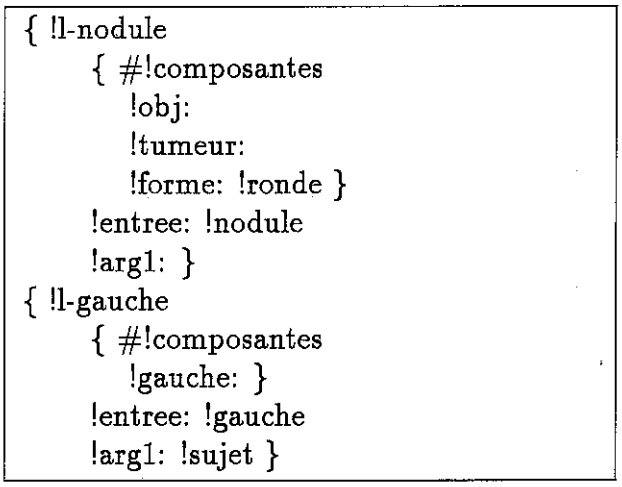

Figure 5: The lexical inputs *nodule* and *gauche*

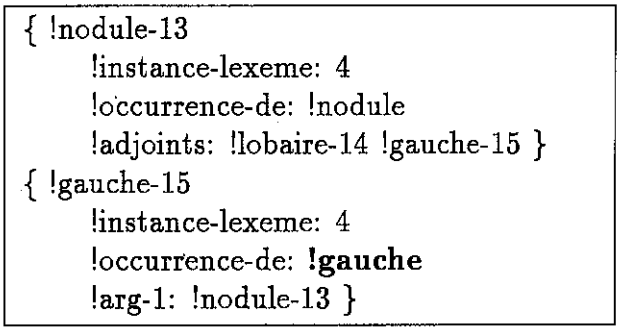

Figure 6: The lexeme instances *nodule-13* and *gauche-15*

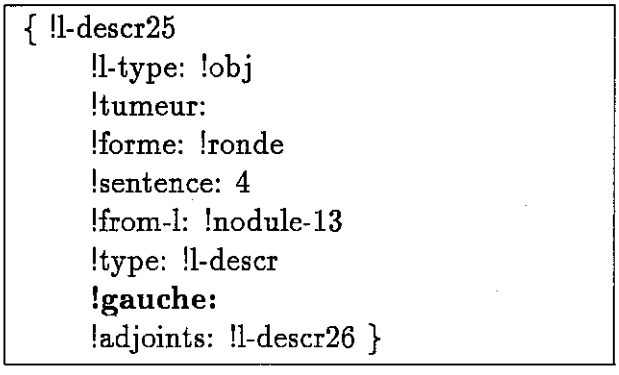

Figure 7: The l-description *l-descr25* created for the concept *tumeur*

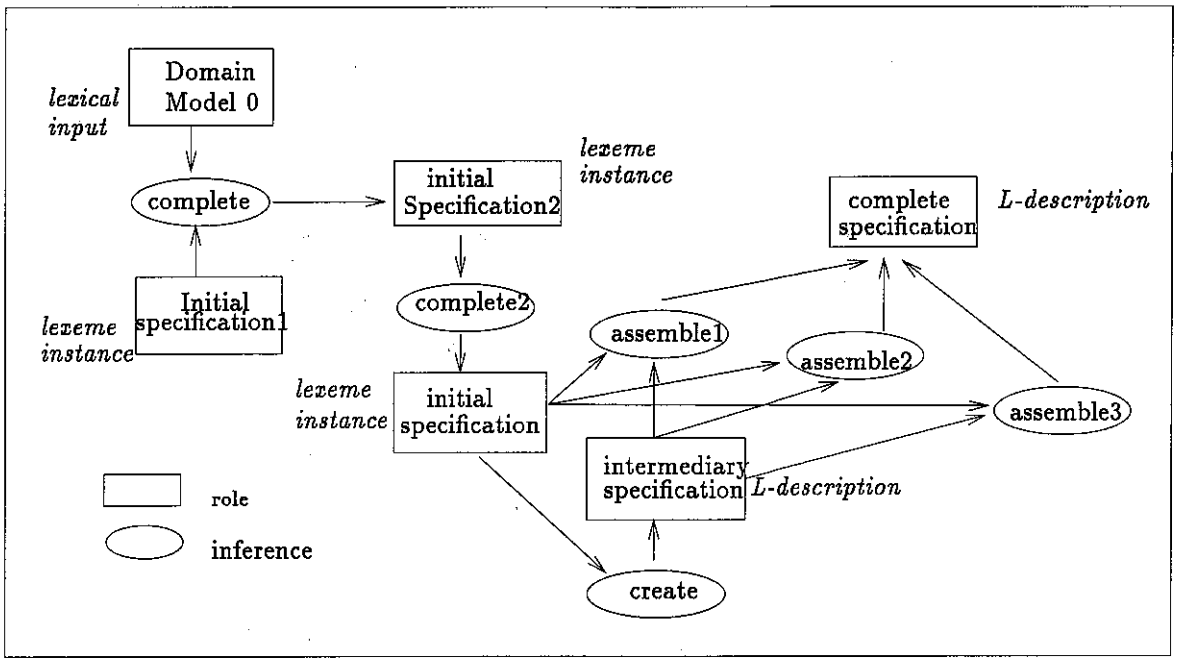

Figure 8: An inference structure for the sub-task *Building l-descriptions*

1 - (complete complete2 create)
2 - (complete complete2 create assemble1)
3 - (complete complete2 create assemble2)
4 - (complete complete2 create assemble2)
5 - (complete complete2 assemble1)
6 - (complete complete2 assemble2)
7 - (complete complete2 assemble3)

The set of Valid Inference Paths VIP The previous set of inferences is syntactically correct. However, some paths are not recognized as semantically correct by the expert. As an example, the expert tells that the path 5 is not semantically correct. In fact, among the lexeme instances, some will lead to the creation of l-descriptions, like *nodule* and some will not (*gauche*). On the other hand, the assembling process deals with transferring semantic components *on* l-descriptions. So the inference paths that do not create l-descriptions are not valid. For the same reason, the paths 6 and 7 are not valid.

Furthermore, all the semantically correct paths are not provided by the procedure, especially the ones that include "parallel inferences". For example, the three inferences *assemble1 assemble2 assemble3* may occur separately or simultaneously. Thus, the expert also provides the additional paths given below:

8 - (complete complete2 create assemble1 assemble2 assemble3)
9 - (complete complete2 create assemble1 assemble2)
10 - (complete complete2 create assemble1 assemble3)

11 - (complete complete2 create assemble2 assemble3)
Finally, the set of VIP is as follows: 1 2 3 4 8 9 10 11.

Using the VIP during the testing phase Let us consider the following example: *"The thyroïd palpation reveals a left lobar nodule".*
The KBS tracer yields the following set of fired rules:

!regle-instanciation-composantes (x 7 firings)
!create-L-description (x 5)
!transfer-comp-on-l-descr (x 1)
!specify-links-on-l-descr-*with-polarity* (x 5)

The rule "!regle-instanciation-composantes" has thus been fired seven times in order to complete the description of the initial lexemes. The rule "!create-L-description" has been fired 5 times to create five l-descriptions, etc.

Without giving the whole rules, we give below the links that have been established between these rules and the inferences.

The rule "!regle-instanciation-composantes" is linked to the inference !complete.
The rule "!create-L-description" is linked to the inference !create.
The rule "!transfer-comp-on-l-descr" is linked to the inference !assemble1
The rule "!specify-links-on-l-descr-*with-polarity*" is linked to the inference !assemble3

These links permit to build the *Current Inference Path* which is given below:

(complete create assemble1 assemble3).

This path is not a member of the VIP. Generally speaking, if a CIP is not a member of the VIP, the expert has to decide if this CIP is a valid path or not. In the first case (i.e. the CIP is valid), as the CIP does not match the inference structure, the latter has to be modified to take it into account. In the second case, the code level has to be corrected. In our case, the expert considers that the CIP is valid, thus leading to modify the inference structure (Figure 9). Notice that this may lead to add new paths which will have to be validated.

5 Conclusion

Many previous researches in Knowledge Acquisition have shown the importance of the modeling phase in a KBS life-cycle. Our study tends to show that an effective use of the conceptual model implies to introduce new steps in the KBS life-cycle, especially steps dealing with its explicit representation under an implemented form. On the other hand, we propose a procedure to test the KBS using this "implemented" conceptual model and, more precisely, its inference structure. Our approach clearly depends on the KADS methodology. Our future work will deal first with the extension of our experiment to the whole conceptual model and KBS, and second, we will investigate the use of the two other layers of the KADS conceptual model: the domain layer and the task layer. We will try for instance to build the inference paths using the task layer in order to limit the number of inference paths generated during the phase *building inference paths* (B) of our life-cycle.

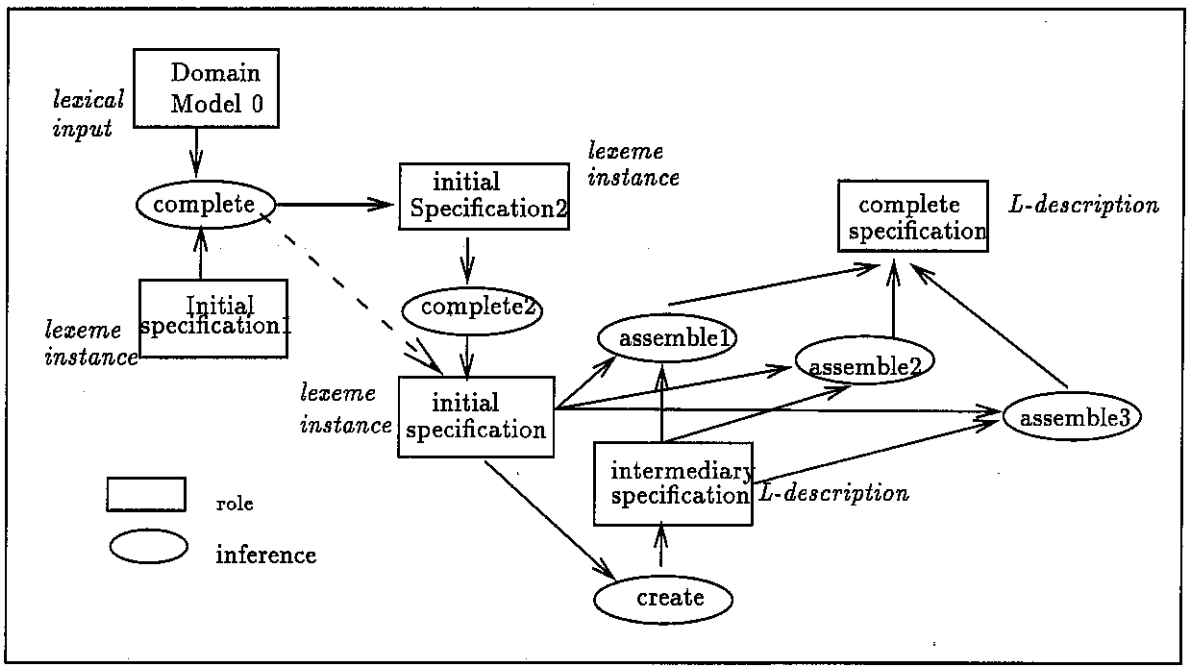

Figure 9: The new inference structure for the sub-task *Building l-descriptions*

Acknowledgments

We would like to thank Jean Charlet and André Molia for their helpful advices.

References

AUSSENAC, N. (1989). *Conception d'une méthodologie et d'un outil pour l'acquisition des connaissances expertes.* Thèse de doctorat, Université P. Sabatier, Toulouse.

AYEL, M. (1987). *Détection d'incohérences dans les bases de connaissances: SACCO.* Thèse d'état, Université de Savoie.

BOEHM, B. W. (1988). A Spiral Model of Software Development and Enhancement. *Computer*, pp. 61–72.

CANAMERO, D. & GELDOF, S. (1993). Coupling Modeling and Validation in COMMET. In *Proceedings of the European Workshop on Validation and Verification of KBSs*, Palma de Mallorca (Spain).

CAVAZZA, M., DORÉ, L. & ZWEIGENBAUM, P. (1992). Model-based natural language understanding in medicine. In K. C. LUN, P. DEGOULET, T. PIEMME & O. RIENHOFF, Eds., *Proc MEDINFO 92*, Amsterdam: North Holland.

CHABAUD, C., SOUBIÉ, J., BURATTO, F. & LOMPRE, N. (1991). *Cycle de vie des systèmes à bases de connaissances: objets et méthodes de validation ergonomiques.* Rapport Interne IRIT/91-48-r, IRIToulouse.

DELOUIS, I. (1993). *LISA, un langage réflexif pour la modélisation du contrôle dans les systèmes à bases de connaissances. Application à la planification des réseaux éléctriques.* Thèse de doctorat, Université de Paris-Sud.

DIENG, R. (1990). Méthodes et outils d'acquisition des connaissances. In *ERGO'IA90*, Biarritz, France.

GREBOVAL, C. & KASSEL, G. (1992). Une approche pour rendre opérationnels des modèles conceptuels : le générateur AIDE. *Revue d'intelligence artificielle*, **6**(1-2), 5–215.

HALKER, M. C. & WELZ, U. T. (1989). Software life-cycle for knowledge based systems. In *Actes des 9es Journées Internationales les Systèmes Experts & leurs Applications*, Avignon, France.

HAOUCHE, C. (1993). Using a Conceptual Model to Validate KBSs. In *Proceedings of the European Workshop on Validation and Verification of KBSs*, Palma de Mallorca (Spain).

HAYES-ROTH, F., LENAT, D. & WATERMAN, D. A. (1983). *Building Expert Systems.* Addison Wesley Publishing Company.

HICKMAN, F., KILLIN, J., LAND, L., MULHALL, T., PORTER, D. & TAYLOR, R. (1989). *Analysis for knowledge-based systems. A practical guide to the KADS methodology.* John Wiley and sons.

KRIVINE, J.-P. & DAVID, J.-M. (1991). L'acquisition des connnaissances vue comme un processus de modélisation : méthodes et outils. *Intellectica*, **2**(12), 101–137.

NGUYEN, T., PERKINS, W., LAFFREY, T. & PECORA, D. (1985). Checking an expert system knowledge base for consistency and completness. In *Proceedings of the 9 th International Joint Conference on Artificial Intelligence*, Los Angeles, CA.

PREECE, A., CHANDER, P., GROSSNER, C. & RADHAKRISHNAN, T. (1993). Modeling rule base structure for expert system quality assurance. In *IJCAI93 workshop on Verification and Validation of ES*, Montréal, Canada.

ROUSSET, M.-C. (1988). *Sur la cohérence et la validation des bases de connaissances:le système COVADIS.* Thèse d'état, Université d'Orsay.

VIGNOLLET, L. (1991). *Une approche pour la construction automatique de jeux de données de test pour des systèmes à base de connaissances.* Thèse de doctorat, Université de Chambéry.

WIELINGA, B., SCHREIBER, A. & BREUKER, J. (1992). KADS: a modeling approach to knowledge engineering. *Knowledge Acquisition*, 4(1), 5–53.

Contents