A Rational Agent for the Construction of a Semantic Model*

PAUTRET Vincent
Université de Rennes I, ENSSAT
6 rue de Kérampont, BP 447
Lannion, France, 22300
Vincent.Pautret@enssat.fr

Abstract

This paper presents a methodology that aims at building knowledge models from a natural language description of a domain. Our methodology is based on the establishment of a dialogue with the knowledge engineer of an application. This dialogue is motivated by the Semantic Differentiation Process, which solves problems related to acquisition and modelling.

Moreover, the dialogue can be naturally formalised within a theory of communicating rational agents. We can thus consider a more complete automation of the process of modelling and show how to integrate our methodology into this type of theory.

Introduction

Knowledge Based Systems separate the semantic model - which handles the system knowledge - from the reasoning process - which uses this knowledge. The main advantage of this approach is that only the semantic model has to be changed to handle a different application domain. However, the creation of a semantic model for a given application is a manual process, which is difficult to automate (Paris and Vander Linden (1996)).

Tools (Heijst et al. (1997)) or workbenches ((Mikheev and Finch (1995), (Delisle (1996)) already exist that aim at building semantic representations at the domain level (using the vocabulary of KADS (Wielinga et al. (1992)). With these tools and workbenches, conceptual knowledge models (like ontologies) independent of the application domain are built. However, the knowledge engineer task remains fastidious. One of the difficulties in completely automating the acquisition and modelling process comes from a lack of interaction with the knowledge engineer.

In order to improve these interactions (and thus to facilitate modelling), we propose a methodology based on a natural language dialogue with the knowledge engineer. This methodology can be implemented into a rational agent. In this way, this agent is given capabilities of modelling by means of conceptual diagrams defined in our methodology. We show how to make this integration within the formal theory of communicating rational agents of Sadek (Sadek (1991), (Sadek et al. (1997))).

Section 1 introduces the bases of the methodology. Section 2 explains how to integrate it into a theory of rational agents for its effective implementation. The last section presents the guidelines to implement our methodology into a rational agent.

1 Bases of the methodology

The methodology aims at building a semantic model of a domain from a natural language description. It is based on three successive stages: the acquisition stage, the modelling stage, and the transfer stage.

The acquisition stage consists of the analysis of each domain description utterance. A morpho-syntactic analysis is followed by a semantic analysis in order to build a semantic representation of each utterance. In an iterative way, these representations are integrated into a general model: the Construction Model (CM).

The modelling stage consists of an interactive reorganisation of the CM once the description process is completed.

The transfer stage extracts the relevant information from the CM and builds the semantic domain model.

* This work was realised within the framework of a PhD in France Télécom R&D.

1.1 Construction Model

On the one hand, Construction Model must have a sufficient expressiveness, which makes it possible to represent domain knowledge and knowledge related to its own structure at the same time. On the other hand, it must have a flexible enough structure, which can be handled simply and efficiently.
We use a formal language based on KL-ONE-like description logic. The central part of the model is a semantic network whose nodes are concepts and whose arcs are semantic domain or modelling relations (for example *subconcept*, *composition*, *property*, etc.) The representation language also offers the possibility to express *abstract* concepts (as a composition of concepts and relations of the network), as well as constraints and negative knowledge related to the concepts and relations of the network.

1.2 Basic tools for knowledge acquisition

During the modelling process, which is based on dialogue, the knowledge engineer utterances are analysed and the relevant information has to be extracted from them. For this purpose, we use two tools to acquire knowledge from texts.
The first tool is a robust morpho-syntactic analyser, which produces a syntagmatic graph (Giguet (1998)) where each node is a syntagm and each relation is a syntactic relation. A syntagmatic graph is produced for each utterance of the description.
The semantic tool makes use of the results of the morpho-syntactic tool to produce a semantic representation of each utterance. Thanks to four basic operations, it integrates this representation into CM. The first operation identifies the concepts that are already known. The second one is related to generalisation and organises the concepts into hierarchies. The third one calculates the common characteristics to the concepts. Finally, the last one places the semantic relations resulting from the semantic analysis into CM.

1.3 Semantic Differentiation Process

The Semantic Differentiation Process is based on a set of generic conceptual diagrams, whose role is to modify the CM structure. We follow an empirical process to exhibit modelling problems and to define a conceptual diagram as a solution to each one.

An initial situation and several final situations define a conceptual diagram. Situations are expressed in terms of the language of CM representation, namely as sets of first order logic formulae. A situation corresponds to a particular structuring of generic concepts and generic relationships between these concepts. A condition is associated with each final situation. The conceptual diagrams are represented in the following form (we will use more readily a chart of the initial and final situations of a diagram as on the example of figure 1).

Name_of_the_diagram
<Initial_situation>
<condition_1> <Final_situation_1>
…
<condition_n> <Final_situation_n>

where *Initial_situation* and *Final_situation_k* ($k \in \{1,\ldots,n\}$) refer to the initial situation and the *n* final situations associated with the diagram, and *condition_k* ($k \in \{1,\ldots,n\}$) refer to the condition associated with *Final_situation_k*.

Diagrams are divided into three main families. The first family (two diagrams) is dedicated to the integration problems. The second one (seven diagrams) allows model simplifications while the third one (eleven diagrams) allows modifications of the model structure. The diagrams constituting the first family are applied during the acquisition stage while those of the two other families are applied during the modelling stage. The diagrams are ordered according to the importance of the modifications they produce on the model. For example, the simplification diagrams are applied before the modelling ones.
A diagram can only be used once the model under development has validated the initial situation. When the initial situation has been instantiated, a dialogue begins with the knowledge engineer until one of the conditions associated with each final situation is validated. CM is then restructured to resemble the final situation, which corresponds to the condition. The role of this dialogue is to determine the best transformation of the model by the considered conceptual diagram for the problem under consideration. An algorithm of processing of graphs carries out the passage from the initial situation to the selected final

situation, which directly removes assertions from the model or adds some to it.

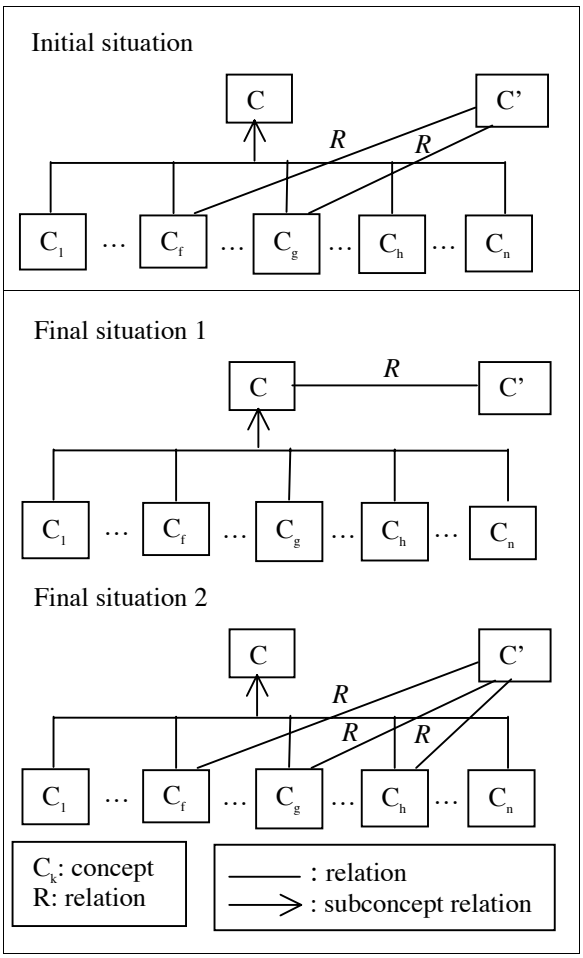

Figure 1: Factorisation diagram

Figure 1 shows the *Factorisation diagram*, which belongs to the second family of diagrams. The role of this diagram is to factorise a relation from the subconcepts to their supconcept (final situation 1) or to add relations, which would have been forgotten by the knowledge engineer (final situation 2). The first case makes it possible to reduce the number of relations and thus the complexity of the model. With the second one, supplements can be added to the model after missing information has been detected.

The initial situation shows several subconcepts C_f, …, C_g of concept *C*, which have the same relation *R* on the same concept *C'*.

In final situation 1, relation *R* is placed on concept *C*, whereas in final situation 2, relation *R* is extended between *C'* and some of the subconcepts of *C* (concepts C_f, …, C_g, …, C_h).

When the initial situation is detected in the model, the evolution of the model is determined thanks to the following dialogue with the knowledge engineer:

Q1 – Subconcepts C_f, …, C_g of C have the same relation R with C'. Have all subconcepts of C this relation with C'?

With a positive answer, the model is transformed like final situation 1 (by adding relation *R* on *C* and by removing *R* between subconcepts of *C* and *C'*). With a negative answer the dialogue proceeds as follows:

Q2 – What are the different concepts that have this relation R with concept C'?

The model then evolves to situation 2.

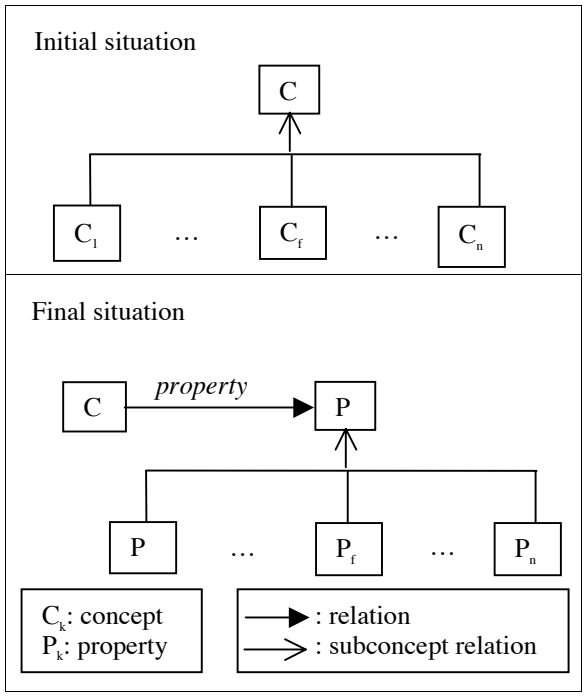

Figure 2: Property Extraction diagram

Figure 2 shows the *Property Extraction diagram*, which belongs to the third family of diagrams (modelling stage). It is intended to extract a particular property of a concept from its structure. From the structural viewpoint, a definition of a concept could be compared to the complex terms extracted by LEXTER (Bourigault (1994)). A concept comprises a word in head position (the headword) and a list of qualifications. Each qualification is recursively composed of a word and a list of qualifications. For example, *a specialised server for weather forecast* has the word *server* as headword and *weather forecast* and *specialised* as qualifications. In the context of

the diagram of figure 2, *weather forecast* is seen as a property of the concept.
Hence, if the system detects different qualifications of the concepts that have the same headword, the system asks the knowledge engineer whether it is possible to gather the qualifications under a same property. The dialogue associated to this diagram is very simple. The system only needs to ask the knowledge engineer if s/he wants to apply the diagram. If s/he answers positively, the system asks the knowledge engineer for the name of the property. Otherwise, the system does not modify the model.

1.4 A short example

Here is an example of dialogue in the AGS application, which is a vocal query, over the telephone, of the shared-revenue information voice services directory (applied to the areas of "employment" and "weather forecast") hosted by France Télécom. *K* stands for the knowledge engineer, *S* for the system, and N for note on the current modelling state.

K: *There exist servers for weather forecast, servers for employment and mail order servers*.
N: The system creates the concept "server" and places the three types of servers as subconcepts of this abstraction (by using the second basic operation, cf. section 1.2).

K: *The servers for weather forecast and the servers for employment have telephone numbers*.
K: *That is all*.
S: *O.K.*

S: *The servers for weather forecast and the servers for employment own a telephone number. Do mail-order servers own a telephone number as well?*
K: *Yes*.
S: *O.K.*
N: The system factorizes the relation on the concept "server" by applying the Factorization diagram.

S: *I detect some properties of the servers (weather forecast, employment, and mail order). Is it possible to group together these properties under the same concept?*
N: The system applies here the Property Extraction diagram.

K: *Yes*.
S: *Could you give the name of the concept?*
K: *Theme*.
S: *O.K.*
N: Then the system builds the final model.

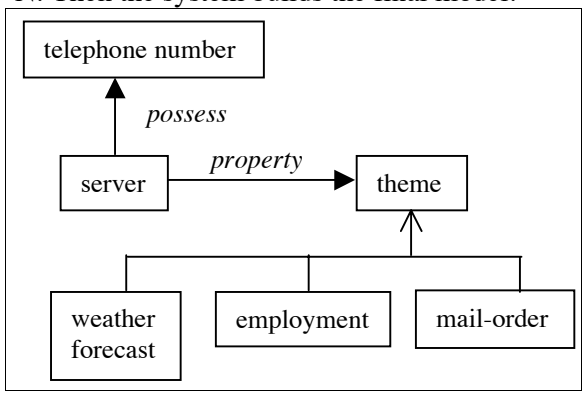

Figure 3: Example of a semantic model

2 Integration of the methodology into a rational agent

The core of our methodology is the dialogue with the knowledge engineer. The communicative rational agents provide a theoretical framework that is particularly adapted to the formalisation of this dialogue. In this way, we extend the theory of rational agents proposed by Sadek (Sadek (1991)), thus giving the agents the ability to build a semantic model of an application while following the interactive principles of our methodology.

2.1 The theory of rational agency

The whole theory of rational agency is expressed in a homogeneous multimodal logic of mental attitudes and actions (or events). Modal operator of belief B_i satisfies a *KD45*-model. The resulting agents are fully introspective and have consistent beliefs. Formula B_ip is read "property p is a logical consequence of the beliefs of agent *i*". The mental attitude of intention is defined as a complex combination of primitive mental attitudes like belief and choice[1], as a relatively similar way as Cohen and Levesque (1990). Formula I_ip is read "agent *i* intends to bring about proposition p".
In order to reason about action, two modal operators are introduced, *a* being an action expression and ϕ a formula: *Feasible*(a,ϕ) means that *a* can take place and if it does, ϕ

[1] For sake of simplicity, we only focus on belief and intention. For more details, see Sadek (1991).

will be true after that, and $Done(a,\phi)$ means that a has just taken place and ϕ was true before that.
The formal theory provides a set of axioms that specify rational agent behaviour in multi-agent environment and flexible behaviours according to the type of desired agent. For example, the first ones allow infer chains of actions corresponding to the agent intentions and the second ones generate cooperative reactions.

2.2 Primitive action of modelling

We propose to formalise conceptual diagrams with primitive actions of the theory of rational agency. Conceptual diagrams are then directly usable and can be planned by the agents like the other actions.

2.2.1 Action model

For each action to be planned, the classical components preconditions and effects are defined with the following meaning. Preconditions refer to the statements that must be true for the action to be performed. Effects refer to the statements that are intended to hold in the world following the performance of the action. Actions are represented in actions schemata:

<*actor*, Action(*parameters*)>
P: ϕ_1
E: ϕ_2

where *actor* refers to the agent of the action, *parameters* refers to eventual parameters of the action, ϕ_1 refers to the preconditions, and ϕ_2 refers to the effects.
For example, the communicative act of an agent *i informing* an agent *j* that a proposition ϕ holds is defined by the actions schema:

<*i*, Inform(*j*, ϕ)>
P: $B_i\phi \wedge \neg B_i(B_j\phi \vee B_j\neg\phi)$
E: $B_j\phi$

Thus an agent who achieves the act to inform of ϕ aims that *j* believes that ϕ is true. It cannot make it if it thinks itself ϕ true and does not think that *j* already has a belief on ϕ.

Among the axioms, which define the characteristics of an action in the theory, we exhibit the one, which refers to the preconditions[2]:

$$B_i(Feasible(a) \Leftrightarrow \phi_1) \quad (1)$$

where ϕ_1 refers to preconditions of a.

2.2.2 Actions and conceptual diagrams

In order to express conceptual diagrams of our methodology into the theory, we propose to associate with them the primitive actions, which make the transformation from their initial situations to each one of their final situations. In particular, this is made possible thanks to the fact that the situations can be described in a logical language. In this way, each one of these primitive actions corresponds to a modification of the mental state of the agent. We define actions associated with conceptual diagrams in the following way:

<*i*, Action_1>
P: $B_i(initial\_situation \wedge condition\_1)$
E: $B_i(final\_situation\_1)$

<*i*, Action_2>
P: $B_i(initial\_situation \wedge condition\_2)$
E: $B_i(final\_situation\_2)$

...

In this schemata, *i* refers to the agent (the modeller agent), which applies the methodology, *initial_situation*, *final_situation_1*, … refer to logical formulae associated with situations of a conceptual diagram, and *condition_1*, *condition_2*, … refer to logical formulae, which describe conditions for each final situation of the conceptual diagram.

2.2.3 Axioms for conceptual diagrams

Unlike actions defined in the theory of Sadek, the primitive actions that we associate with each conceptual diagram are not planned according to their effects. The conceptual diagrams are applied as long as possible. Therefore, their planning by the modeller agent does not depend on the goals it seeks, but rather on the current situation of its mental state, i.e. on its knowledge.
The following axioms schema corresponds to this strategy. As soon as an action can be

[2] *Feasible(a)* is the syntactic abbreviation of *Feasible(a,True)*.

applied, it must be done. The calculation of feasibility results from the axiom (1) of the theory.

$$B_i(Feasible(a)) \Rightarrow I_i Done(a) \qquad (2)$$

where *a* refers to a primitive action associated with a conceptual diagram.

Conditions associated with each alternative of a conceptual diagram are the source of the dialogue to apply the diagram (see section 1). We have to supplement the model of communicative behaviour of modeller agent in order to convey to it the capacity to initiate the dialogue with the knowledge engineer. This dialogue increases its knowledge until it can determine which alternative of the diagram (i.e. which primitive action) to apply. For each primitive action, we introduce an axioms schema of the following form:

$$B_i(initial\_situation) \wedge \phi \Rightarrow I_i \psi \qquad (3)$$

where *initial_situation* is associated with the corresponding conceptual diagram. ϕ (condition of release of the primitive action) and ψ (goal starting the corresponding dialogue) are expressed according to *condition_k* (condition associated with the primitive action).
For example, ϕ can be defined by $\neg Bif_i(condition\_k)$ and ψ by $Bif_i(condition\_k)$. $Bif_i\phi$ is a syntactic abbreviation defined in the theory and means that agent *i* knows if ϕ is true or not.
When several actions (resulting from the intentions derived by the axioms) are applicable in the same state, logic does not make it possible to choose the order in which they are applied. In an implementation, we need to be careful to follow the order defined by the methodology. This is achieved by a process of control of the inferences.

2.3 Example

We show the reasoning process related to the application of the factorisation diagram presented in section 1. The primitive actions associated to the diagram are:

<*i*, Facto_1>
P: $B_i(initial\_situation \wedge relation(R,C,C'))$
E: $B_i(final\_situation\_1)$

<*i*,Facto_2>
P: $B_i(initial\_situation \wedge \neg relation(R,C,C')) \wedge \bigwedge_{k\in\{1,\ldots,n\}} Bif_i(relation(R,C_k,C'))$
E: $B_i(final\_situation\_2)$

where R, C, C_1, …, C_n, and C' are the relations and the concepts identified in the conceptual diagram (cf. figure 1).

The two axioms schemata associated to the primitive actions are respectively[3]:

$$B_i(initial\_situation) \wedge \neg Bif_i(relation(R,C,C')) \Rightarrow I_i Bif_i(relation(R,C,C')) \qquad (4)$$

$$B_i(initial\_situation) \wedge B_i(\neg relation(R,C,C')) \wedge \neg[\bigwedge_{k\in\{1,\ldots,n\}} Bif_i(relation(R,C_k,C'))] \Rightarrow I_i Bref_i(\iota x\ relation(R,x,C') \wedge subconcept(x,C)) \qquad (5)$$

where R, C, and C' are the relations and concepts identified in the conceptual diagram (cf. figure 1).

Let us suppose that the modeller agent is configured in such a way that the conceptual diagram is applicable, i.e. $B_i(initial\_situation)$ can be derived from its mental state. Moreover, let us suppose that it can infer no knowledge in connection with $relation(R, C, C')$, i.e. it can only conclude $(Bif_i(relation(R, C, C'))$.
The instantiation of the axiom (4) thus generates the intention of the agent to know if the concepts C and C' are or not linked by R. Then the traditional mechanisms of planning of the theory take over (cf. Sadek (1991)) and produce an act of dialogue aiming at requiring missing information to the knowledge engineer (cf. dialogue *Q1* of section 1.3).
If the answer to this question is positive, the axioms of rational behaviour of the theory involve $B_i(relation(R, C, C'))$. All the preconditions of the primitive action Facto_1 are then checked and the axioms (1) and (2) induce the execution of the first alternative of the conceptual diagram (Facto_1). The resulting mental state of the agent conforms to the final situation 1.
If, on the contrary, the answer is negative, the agent acquires the knowledge: $B_i(\neg relation(R, C, C'))$. The axiom (5) then applies as long as the agent does not have a knowledge

[3] $Bref_i(\iota x\ \phi(x))$ means that agent *i* knows the objects, which check the property ϕ.

supplements on *relation*(R, C_k, C') for all $k \in \{1,..., n\}$. These produces the intention at the origin of the act of dialogue aiming at requiring of the knowledge engineer the whole of the subconcepts of C in relation R with C' (cf. dialogue *Q2* of section 1.3). The primitive action Facto_2 can then be carried out and leads to a mental state that conforms to the final situation 2.

3 Integration into an operational system

3.1 The Artimis technology

The Artimis technology of France Télécom R&D provides a generic framework to instantiate intelligent dialoguing agents. Such agents can interact cooperatively in natural language with human users.

Artimis software is composed of four main modules: a rational unit (which is the kernel of the system), a natural language interpretation unit, a natural language generation unit, and a domain knowledge management unit (Sadek et al. (1997), Sadek (1999)).
The rational unit conveys the agent the ability to dialogue and to reason about knowledge and action.
The natural language interpretation unit uses island-driven parsing and semantic completion (Sadek et al. (1997)). Island-driven parsing means that small syntactic structures in the text are spotted, with as few range dependencies as possible. The semantic completion builds a well-formed logical formula with the result of the parse.
The natural language generation unit verbalises dialogue acts produced by the rational unit.
Finally, the domain knowledge management unit contains a representation of the domain knowledge. It provides several functions (like concepts identification) to have access to the knowledge.

Artimis software works in lab versions on several real applications like the AGS one. It is written in Quintus Prolog.

3.2 Guidelines for the integration into an Artimis dialogue agent

We present, in this section, the needed modifications in order to integrate our methodology into an Artimis dialogue agent.

3.2.1 Modification of the two natural language components

We have to modify the natural language interpretation unit at two levels.
Firstly, the system must take into account all the words of the utterance in order to detect the new concepts. We make the assumption that the sentences are syntactically and semantically correct.
Secondly, a robust syntactic analysis, based on an approach such as (Giguet (1998)) must be implemented to get as much information as possible on the relations between the concepts.
In order to solve syntaxico-semantic ambiguities we introduce two particular relations: *unknown* and *context*. The *unknown* relation means that the analyser detects a relation but can not determine its exact nature. The relation *context* means that two concepts are present in the same utterance without any other information.
Example:
Input sentences:
There are servers and telephone numbers.
Results:
concept([server]) concept([telephone, number])
relation(context, [server], [telephone, number])

The natural language generation unit recovers the vocabulary necessary to the generation of sentences related to the domain thanks to the interpreter, which keeps the link between the concepts of CM and the vocabulary of the description.

3.2.2 Modification of the rational unit

In order to increase the reasoning capabilities of the rational unit so that it can direct the construction of the semantic model as well as the dialogue with the knowledge engineer, we add the logical axioms and the primitive actions that we defined in section 2. The rational unit should then not be rebuilt but rather updated.

3.2.3 Modification of the knowledge management unit

The main modifications concern this module.
We extend the language of representation of the model with the primitives of the CM.
In order to be able to insert a new knowledge in the model, we add the four basic operations.

The original identification function is modified to take into account the modification of the knowledge representation language.
The second function builds the hierarchies by using the structure of the concepts. For example, " server for employment " and "server for weather forecast" belong to the same hierarchy since they are two "servers". They are generalised by the concept "server".
The third function places relations between two concepts and their qualifying common part. For example, "server for employment" and "employment theme" have the common part "employment".
The last one is a transfer function between the result of interpretation and CM.

The organisation algorithm of the model tries to instantiate the initial situations of the diagrams in a definite order. This order depends of the priority associated to each diagram. The priority is given according to the transformations carried out by the diagram: the more significant the transformations, the weaker the priority. When a situation is validated, the corresponding formulae are injected into the rational unit. This one then takes over to calculate a question. Following the answer of the knowledge engineer, a new knowledge is asserted and the process starts again from the beginning. When no diagram is applicable, the algorithm stops and the knowledge engineer is provided with the model.

4 Conclusion

We define a methodology of semantic modelling of a domain. It is based on conceptual diagrams that formalise the incremental evolutions of the structure of the semantic model during its construction. A dialogue with the knowledge engineer directs the application of these diagrams.
We also formalise the use of our methodology within a theory of communicating rational agents. This specification provides the rational agent with new reasoning capabilities, which aim at building a semantic model by questioning the knowledge engineer and by applying the conceptual diagrams according to the principles of our methodology.
We thus open prospects for automation since effective agents implementing this type of theory are already operational like, for example, those resulting from Artimis technology (Sadek et al. (1997), Sadek (1999)).

References

Bourigault D. (1994) *Lexter, un logiciel d'Extraction de Terminologie. Application à l'acquisition à partir de textes*. École des Hautes Études en Sciences Sociales, Paris.

Cohen P.R. and Levesque H.J. (1990) Intention is choice with commitment, *Artificial Intelligence* 42(2-3):213-262.

Delisle S. (1996) Le Traitement Automatique du Langage Naturel au Service de l'Ingénieur de la Connaissance : le Système READER. *International Conference on Natural Language Processing and Industrial Applications*, Moncton (New-Brunswick, Canada).

Giguet E. (1998) *Méthode pour l'analyse automatique de structures formelles sur documents multilingues*. Thèse de Doctorat Informatique, Université de Caen, France.

Mikheev A. and Finch S. (1995) Towards a Workbench for Acquisition of Domain Knowledge from Natural Language. *Proceedings of EACL'95*, Dublin, Ireland.

Paris C. and Vander Linden K. (1996) Building Knowledge Bases for the Generation of Software Documentation, *COLING'96*, University of Copenhagen, Denmark.

Sadek M.D. (1991) *Attitudes mentales et interaction rationnelle : vers une théorie formelle de la communication*, Thèse de Doctorat Informatique, Université de Rennes I, France.

Sadek M.D., Bretier P., and Panaget F. (1997) ARTIMIS: Natural Dialogue Meets Rational Agency, *IJCAI-97*, Japan.

Sadek D. (1999) Design Considerations on Dialogue Systems: From Theory to Technology – The Case of Artimis- *Proceedings of the ESCA TR Workshop on Interactive Dialogue for Multimodal Systems (IDS'99)*, Klöster Irsee, Germany.

Van Heijst G., Schreiber A.TH., and Wielinga B.J (1997) *Using Explicit Ontologies in KBS Development*, International Journal of Human-Computer Studies, 42(2/3) : 183-292.

Wielinga B., Schreiber A., and Breuker J. (1992) *KADS: A Modelling Approach to Knowledge Engineering*. Knowledge Acquisition 4(1):5-53.