Computer Aided Requirements Engineering

G. Grosz
Laboratoire MASI - Université Paris 6
4, place Jussieu 75005 Paris France
email : grosz@masi.ibp.fr

C. Rolland
Université Paris 1 - Sorbonne
17, rue de la sorbonne
75231 Paris Cedex 05 France

Abstract

Our aim is to do an advanced CASE tool for supporting the Requirements Engineering (RE) process. The RE process is based on : (1) acquisition of domain dependent knowledge using natural language statements, (2) representation of the semantic contents of natural language statements through a real-world description with graphical notations, easy to understand and to manipulate, and (3) a guided and partly automated mapping of the real-world description to an object-oriented conceptual schema.
Our CASE tool has an expert system architecture. These different possibilities are thus achieved by means of a rule base and a user-friendly graphical interface. It is called CARE (Computer Aided Requirements Engineering).
In this paper, we introduce the different concepts used in the real-world description, namely: actor, event, entity and their associated relationships. We briefly present the object-oriented conceptual model and some of the mapping rules. We propose different criteria for deducing simple behaviour.

1 Introduction

In this paper, we want to tackle two different but interrelated problems : (1) the recognition and description of the problem space and the definition of the solution space when designing Information Systems (IS) , and (2) the interfaces provided by a Computer Aided Software Engineering tool (CASE). The purpose of Requirements Engineering (RE) is to abstract and to conceptualize the relevant parts of an application domain [Dubois 89], in order to obtain a conceptual specification of the IS : the conceptual schema. In our CASE tool, the conceptual schema is based on an object-oriented model, but this is only of minor importance. In fact, as far as possible, we want to hide the conceptual model from the designer. It can be hidden only if the mapping between the real-world description and the conceptual schema is sufficently understood and formalized. We believe that it is the tool duty to direct the designer for building the final solution. This consideration has, at least, two advantages :
- it allows designers to forget the conceptual model and to concentrate on the description of the problem space using the real-world concepts.

- it opens analysis activity to a broader audience; as real-world concepts are easy to understand and manipulate, even occasional "designers" are able participate in the definition of the description.

In order to efficiently support the RE process for information systems, CASE tools must evolve from simple managers of IS specifications to being more intelligent tools, see for instance [Bouzeghoub 85, Cauvet 88]. They must be able to guide the designer to reach effective solutions, able to react to his decision, and proposing the most appropriate action. The design of these new tools must include a new formulation of the RE process itself [Dubois 86]. This must take into account all modelling tasks and the linkage of these tasks. A formalization of this process, entirely based on Artificial Intelligence (AI) techniques, gives a new perspective to designing of these tools and belongs to the Software Engineering approach [Arango 87].

CASE tools must offer simple and easy to use interfaces, understandable by all the potential intervening parties, i.e. all the people who can participate in the construction of the conceptual schema : designers, managers and end-users. Most of the time, CASE tools provide graphical interfaces [Kersten 87] and sometime natural language interfaces [Rolland 91, Bouzeghoub 85, Black 87]. Depending on their personal ability to understand conceptual design and the possibilities offered by the CASE tool, they will choose the most appropriate interface.

Our experience with CASE tools [Cauvet 88, Proix 89] has shown that designers spend much time assimilating sophisticated conceptual models. They prefer to use simple and unconstrained models to make a first "draft" of their conceptual schema. The semantic of this draft is fuzzy; it may be incomplete. Then step by step, the designer constrains and refines his description in order to build the conceptual schema. This schema is a formal representation of the solution space while still respecting the constraints of the conceptual model. Then, during the maintenance phase, as the real-world evolve, the IS has to evolve too. The designer may have to modify the conceptual schema. To make the right modifications implies having a precise knowledge of the conceptual schema, and more importantly, a real expertise about the real world. For the sake of efficiency, we propose a solution to this problem.

First, the designer describes the problem space as completely as possible. Then, for instance using a CASE tool, the designer will transform the first draft by answering questions (automaticaly generated by the tool according to pre-defined pattern), taking decisions, until the conceptual schema is realized. If the first draft is complete enough and the CASE tool smart enough, the designer has the opportunity to forget the conceptual model. Then, during the maintenance phase, if

significant modifications have to be performed on the conceptual schema, the designer will go back to his first draft. At this level, it will be easier for him to understand the changes without having to worry about the consequences on the conceptual schema. His interlocutors may also understand the description and help him more efficiently to find out what to modify. In this paper, we are not concerned about the mapping of this schema in a given technical environment. We limit our work to the construction of the conceptual schema.

Our work may be compared to the transformation of conceptual to logical schema [Tan 84]. This task has been partly automated by a better comprehension of the underlying process. The purpose is automatically to map a computer independent description onto a logical schema for a given computer. We try to do the same but at a higher level. We want to automate, as far as possible, the transformation of a real-world description to conceptual schema.

The paper is structured as follows: the next section presents the general architecture and the global functions of CARE. Then, in section 3, we describe the different concepts used in the real-world description. We propose different criteria useful for defining either the structural and the behavioural properties of an IS. Section 4 gives an overview of the rule base with the presentation of a detailed example. In order to complete the real world description, we define dynamic patterns. The underlying meta-rules are discussed in section 5. The paper will be illustrated by the description of a stock control system.

2 General architecture

The goal of CARE is to provide an active and intelligent support to IS designers. Intelligent support is developed to provide instruments to make the design activity more effective. Instruments are :

- guidance of the RE process
- automatization of part of the design tasks.

The designers can interact with the tool using two kinds of interface:

- natural language [Rolland 91],
- graphical ; using the graphical interface, one can use the different proposed models.

As shown in figure 1, the fact base contains all the information concerning the current description of the IS. The rule base contains all the required knowledge for transforming the description to the conceptual schema.

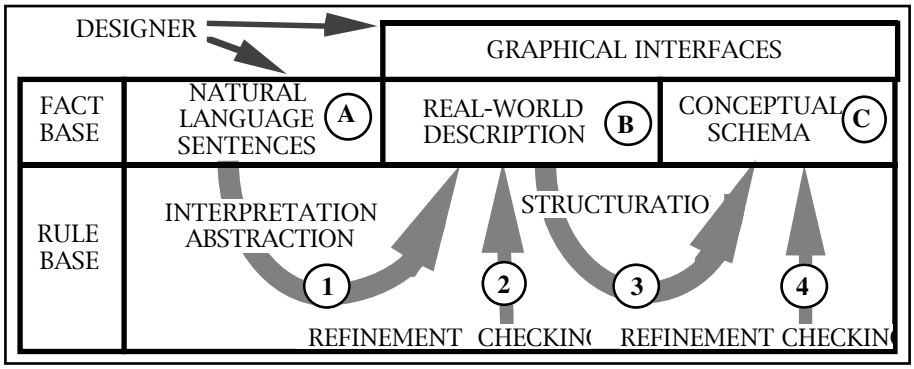

Figure 1: CARE's architecture and functionalities

The fact base includes three levels of facts:
(A) Sentences expressed in a subset of the French language and represented by means of syntactic trees.
(B) Classes of facts are described in a semantic network called the "real-world description". It is based on three concepts : actors, entities, events and their relationships (see section 3 for definition and graphical notations).
(C) Elements of the conceptual schema are also expressed in a semantic network. The conceptual model is object-oriented. For a precise description, see [Brunet 91]. Notice that this conceptual model allows one to identify objects and describe their structure (in terms of association and references) and their behaviour (in terms of operations and events). This model is derived from the REMORA methodology [Rolland 87]. This model is supported by graphical notations, different then the ones used for the real-world model description.

The rule base, activated by an inference engine, transforms, validates and refines the fact base. According to the fact's level, it performs different kinds of actions:
(1) from natural language statements to a real-world description, following the Fillmore's "Case for Case" theory [Fillmore 68], it performs an abstraction which transforms the perception of concrete facts into a description in terms of classes. The main concept of the Fillmore theory is the notion of case**.** Cases are types of relationships that groups of words have with the verb in any clause of a sentence. We have adapted this notion to the purpose of establishing problem-statements allowing the construction of the IS real-world description. We extended it in two directions : (1) cases are applicable to clauses and (2) the classification of cases has been revised.
(2) the real-world description is :
- refined in order to make precise the relationships between the different nodes of the semantic network, which may lead to the discovery of relevant entities or actors;
- checked. Simple rules are activated, which complete the description with forgotten elements. For instance, an actor must trigger at least one event.

(3) from a real-world description to the conceptual schema, by taking decisions, a structure is given following the constraints inherent to the conceptual model. We call this action "structuration".
(4) if the designer chooses to directly use the conceptual model, then his schema is checked and may be refined.

Our tool simplifies the conceptual design process for three different kinds of designers :
- the totally novice ones ; they can describe their problem space using natural language sentences and then use the two transformation steps to build the final conceptual schema
- those who have already understood the real-model concepts ; they use the graphical interface and are driven by the tool to build the final conceptual schema
- the skilled designers ; they can directily describe the solution space using the conceptual model, the tool helps them to check and to refine their description.

A prototype of this CASE tool is now operational. The interface is implemented using the object-oriented programming language ObjectiveC; the knowledge base is implemented in Prolog. The tool runs on SUN stations. An industrial version, named ALECSI, will be commercialized by the French company CRIL by the end of 1991.

3. The real-world description concepts and the conceptual model

The real-world description is a representation, using simple concepts, of the information system in the broad sense, as defined by the FRISCO group in the IFIP Working group 8.1 [IFIP 90]. It is a broad description in the sense that it deals with all the actors and entities including those who will not be represented in the final IS. It is based on three concepts : actor, event, entity. In this section, we present the concepts and their associated relationships. We then introduce the different criteria useful for characterizing these relationships. We end with a short discussion.

3.1. The concepts

Each concept has **properties**, these properties are close to those defined in the ER model [Chen 76]. All the relationships between concepts are binary.

An **actor** has the ability to activate events which process, change entities or actors by means of events. An actor is someone or something (for instance a person, a cashier, a computer,etc) who can interact on the IS. We distinguish two kinds of actors : the active and the passive (see figure 2 for the graphical notations).

An **active actor**'s behavior is not fully constrained. Thus, he can trigger events without justification. In other words, an active actor may intend to trigger events,

and we do not want to know where the intention comes from. On the other hand, a **passive actor**'s behavior is known from the start. He is an intermediary, he transmits information provided by an active actor. For instance, the man who receives the order in a stock control system is considered a passive agent. For the sake of completeness, we think it is useful to describe passive actors because they are part of the information flow inside any organization.

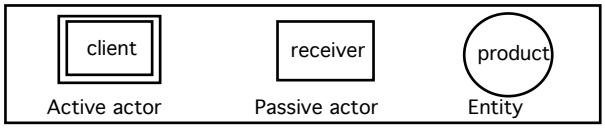

Figure 2 : Graphical notations for actor and entity

An actor can be either represented inside the IS as an object (for instance the client) or not (the manager). The important rule is : an actor will be considered as an object in the conceptual schema if another actor triggers events which accesses him or implies a modification of him.

An **entity** represents a relevant phenomenon of the real-world. This concept is close to the entities as defined in the ER model [Chen 76]. The main difference with the ER model is the distinction between entities and actors. For instance in the stock control system, a product is an entity (see figure 2 for the graphical notation). Entities are related in a binary relationship. But this means no more than that entities share a static relationship which remains to be precised. An entity is modified by events. Entities may or may not be represented in the final IS as objects, depending on the goal of the application. Entities can be split up or collapsed to form objects.

An **event** causes changes inside the organization. Our notion of event is close to the one defined by Naveen Prakash [Prakash 90] and [Sernadas 88]. It includes both the recognition of a state change and the induced actions. However, we distinguish two kinds of events : the external ones and the internal ones (see figure 3 for the graphical notations). The orientation of the graphical notation indicates where the event comes from and what it is acting on. In figure 3, it comes from clients and it is acting on products.

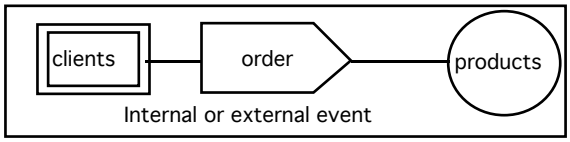

Figure 3 : the graphical notation for events

An **external event** describes an intention coming from an actor : it can be a loan, the purchase of a new book. It represents the actor's possibility to interact with the organization. An **internal event** reflects a rule for the management of the organization. For instance, the rule, which determines what to do when the

organization for a product is under a given limit, is considered as an event. When mapped into the conceptual model, an event can be represented as a combination of object, action and event.

3.2. Characterizing relationships

The real-world model concepts and relationships are not precise enough to build the corresponding conceptual schema. For the refinement and structuration phase, more information is needed. In the following, we present some meaningful criteria. All the relationships presented in the previous paragraph are characterized by cardinalities. These cardinalities are the following:

- four of them are the well-known cardinalities defined by Chen in the ER model [Chen 76], namely : partial or total (resp. 0 or 1 for the minimum) and simple or multiple (resp. 1 or n for the maximum).
- we add a pair of new ones. In order to caracterize the behavior during the relationship, we make a distinction between permanent and variable relationships.

A **permanent** relationship between two occurrences of entities A and B (it may be also actor or event) can be formally defined as :

$$\forall\, a_t \in \mathrm{Ext}(A),\ \forall\, b_t \in \mathrm{Ext}(B),\ \text{if } a_t\ \Re\ b_t \text{ then}$$
$$\forall\, t' > t \ \text{ if } a_{t'} \in \mathrm{Ext}(A) \text{ then } a_{t'}\ \Re\ b_{t'}$$

A **variable** relationship between two instances of occurrences A and B (it may be also actor or event) can be formally define as :

$$\exists\, a_t \in \mathrm{Ext}(A),\ \exists\, b_t \in \mathrm{Ext}(B),\ a_t\ \Re\ b_t,$$
$$\forall\, t' > t \text{ if } a_{t'} \in \mathrm{Ext}(A) \text{ then } a_{t'}\ \neg\Re\ b_{t'} \text{ and } a_{t'}\ \Re\ b'_{t'}$$
$$\text{and } b'_{t'} \in \mathrm{Ext}(B) \text{ and } b'_{t'} \neq b_{t'}$$

Notation: Ext(A) (resp Ext(B)) refers to all the instances of A (resp B), t' and t are points in the time. $a_t \Re b_t$ means that $a \in$ Ext(A), $b \in$ Ext(B) at time t and a is in relationship with b at the same time t.

More generally, when reasoning about this cardinality, one may consider the life cycle of a given instance, not the life cycle of the relationship. In other words, a relationship is **permanent** when, considering an instance of A related to an instance of B, that B will always be the same during A's life cycle. For instance, the relationship between a given order and the client who asks for it is permanent : a given order is related to the same client for its entire life cycle. A relationship is **variable** when one instance of A is related to different instances of B during his life cycle. For example, the relationship between a given client and his orders is variable : a given client is related to different orders during its life cycle - even if he cannot have more than one at the same time. Note that (1) the variable cardinality

does not overlap with the multiple one and (2) when a relationship is variable in one way it can be permanent in the other.

3.3. Discussion

We think that this model is powerful enough to describe most real-world phenomena. It is flexible and simple enough to be understood by novice designers. The different cardinalities are easy to understand, they may be asked through simple questions and allow complex reasonning. They define the type of structural relationship between entities. They allow one to deduce constraints on the creation and deletion of instances. Our definition of permanence/variability subsumes the notion of life cycle inclusion. In conjunction with the partial/total cardinality, it allows one automatically to deduce some of the constructors defined in the conceptual model. For example, assuming that a relationship is "total, simple, permanent" between *order* and *client* and "partial, multiple, variable" between *client* and *order*, the tool can automatically deduce a referential link from *order* to *client*.

It may be argued that the user should be aware of these concepts before being able to do the conceptual modelling. However, if they are not, they just have to use the natural language interface. Since these concepts are close to the one defined in the Fillmore approach, the mapping between natural language sentences and these concepts is made easier.

4. The rule base

4.1. Refinement triplets

We propose a new formulation of the RE process in AI terms and its implementation within an expert design tool. This section presents the knowledge base of the tool which is homogeneously composed of knowledge triplets. They are structured as : <situation, decision, action>. [Grosz 90a] describes this formalism, its usefulness and how we take advantage of it. In this paper, a situation is a piece of specification, a decision is a choice that has to be stated by the designer and an action is a transformation or an addition of the specifications. Making decisions explicit helps in presenting the entire RE process in an homogeneous way. A triplet can be seen as a rule, but an important dichotomy is made in the premise between the situation and the decision. This distinction is useful because it shows precisely what the application relates to and what a design decision is. Triplets reflect the decision-oriented view of the RE process. In order to define the various triplets of the knowledge base, we have experimented with an automatic learning technique [Grosz 90b]. We chose a learning technique based on examples. We focused on the part of the rule base which transforms the static part of the real-world description

to the conceptual one, following the constraints inherent to the conceptual model. We defined all possible examples of <situation, decision, action>. i.e., for each possible combination of cardinalities, we enunciated all the possible conceptual structures, the differences between them were expressed with design decisions. Then, we activated the induction process in order to obtain a set of coherent triplets subsuming the examples.

We do not think it is useful to enumerate the entire list of triplets we have. Here, our purpose is not to make a complete description of the system, we only want to give examples of the triplet base. For a given situation, we will not present all the triplets it is possible to trigger. Only the relevant ones, i.e. the ones we fire, are shown.

A minimal description for a stock control system can be "Clients order products". Using CARE's natural language interface, the tool will automatically generate the following figure 4 :

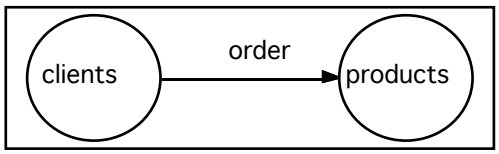

Figure 4 : An interpretation of : "Clients order products"

Then, the designer is asked to designate the actor and to provide the cardinalities between clients and products. He might say:
*Clients are actors. From **clients to products**, the relationship is (1) **partial**, a client may not have order any product, (2) **multiple**, a client may have ordered more than one product, (3) **variable**, a client may order different products during his life cycle. From **products to clients**, the relationship is (1) **partial**, a product may not be ordered, (2) **multiple**, a product may have been ordered by more than one client, (3) **variable**, a product may be ordered by different clients during its life cycle.*

All refinement triplets are structured as follows :
- a situation : a part of the real-world schema;
- some decisions which are informations the designer has to state;
- some actions which transform the situation as an expert designer would do. The following triplet can be triggered (where A is *client* and B is *products*). For the sake of clarity, we present it graphically in figure 5 :

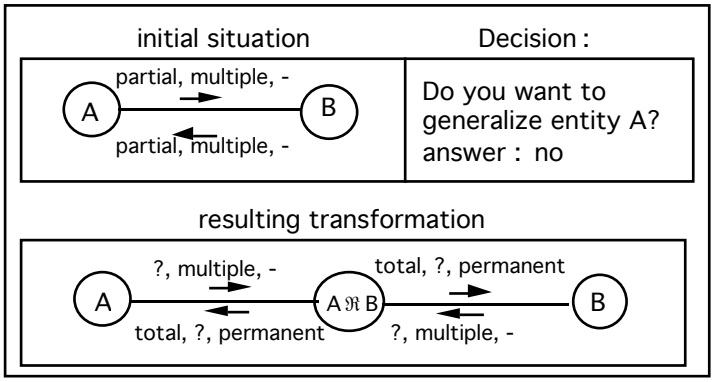

Figure 5 : A refinement triplet

Notations: the"-" means that the value has no importance, the "?" means that this value has to be stated by the designer. "A ℜ B" is an entity which represent the intersection of A's life cycle and B's life cycle, i.e. "A ℜ B" characterizes the relationship of instances of A related to instances of B. The answers given by the designer are in bold face. If the answer is "**yes**" the triplet is not triggerred.

This triplet expresses the following rule :
IF there is a situation where the cardinalities are "partial, multiple" in one way and "partial multiple" in the other way (here, we do not care about permanence/variability)
AND if the designer do not want to generalize the node A "*client*" (the designer is asked to decide if it is relevant or not to differentiate *clients* who have ordered *products* and those who, currently, have not)
THEN the designer must identify and label the relationship. So, in this example, "A ℜ B" is labelled "*order*". This triplet leads to the emergence of the event "*order"*.
After applying the previous triplet to the initial situation and considering that *client* is an actor, we obtain the following schema (figure 6):

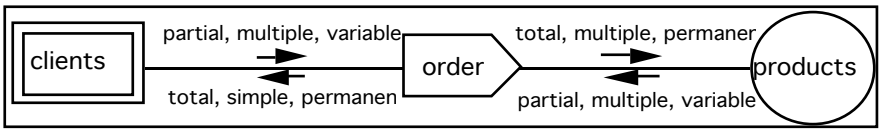

Figure 6 : intermediary schema

Then, different triplets can be applied to the different parts of the resulting situation. For the sake of clarity, we only detail the ones we use. The relationship between *order* and *products* (resp. A and B in the triplet) can be transformed using the following triplet (figure 7):

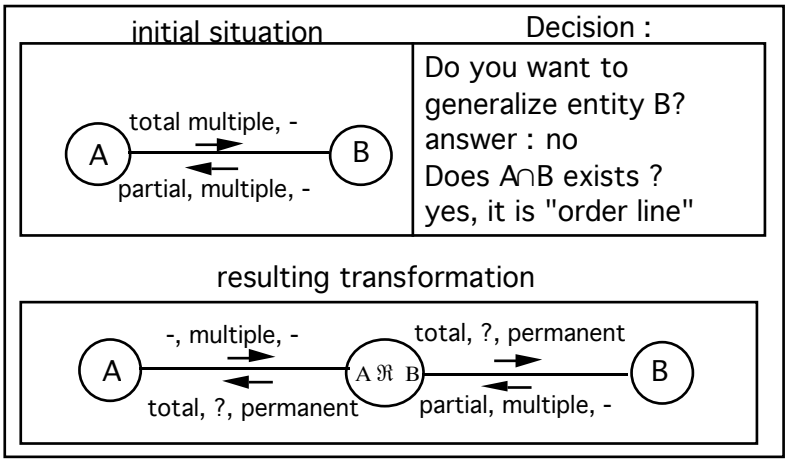

Figure 7 : A refinement triplet

We obtain figure 8:

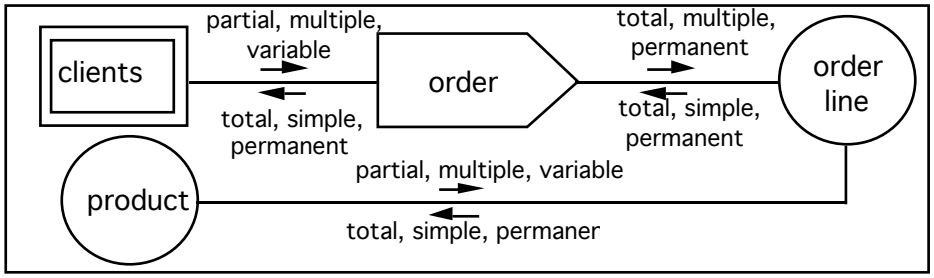

Figure 8 : intermediary schema

4.2 Dynamic patterns

There is a big gap in the literature about the recognition of dynamic patterns. We think that there are some sub schemas, i.e. dynamic patterns, which may be characterized by some given criteria. These patterns are deduced from the study of different conceptual schemas and their associated real-world description. Some mechanisms are reproduced in specific situations. We want to re-use conceptual specifications independently of a given domain. These application-independent patterns must be characterised by criteria. The designer might be able to recognize these criteria. Then, the associated pattern concerning the dynamic of the system can be deduced by the tool and included in the current description of the system in order to refine it.

An important pattern is the **critical resource**. A critical resource is something one cannot share. It concerns entities and corresponds to a specific dynamic pattern. When it is detected by the designer, he is asked to make his current description correspond to the indicated pattern. This may lead him to discover, or at least to put forward, new entities, actors and more importantly, events. For instance, in a stock control system, a product is a critical resource. It is ordered by a client and when there is not enough product, orders are put on a waiting list. Then, when the product becomes available again, the orders are filled, for instance in the FIFO

manner. Of course, there are many ways of managing critical resources, whether it be a single printer connected to a network or a book in a library. But a general frame does exist, see in figure 9, and when using this frame, adaptations have to be made. For instance, in the stock control system, suppliers and consumers are different actors and only suppliers add new resources. On the other hand, in a library, suppliers are either book-sellers or borrowers. The "add" event is triggerred when a new book arrives and when borrowers bring books back.

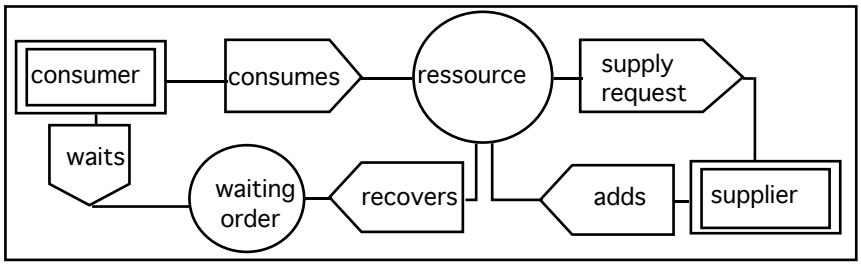

Figure 9 : dynamic pattern for critical resource

Let's go back to our example and illustrate the use of dynamic patterns. The recognition of these patterns can be made at any stage of the refinement of the real-world description. If dynamic patterns are highlighted at an early stage, relevant actors, events and entities will be put forward before. Now, but this could have been done at previous stage, the tool asks the designer to recognise critical resources. We think it is clear, in view of the definition given in the previous paragraph, that "*product*" is a critical resource. The designer has to identify what is already present and what is missing in his schema according to the critical resource pattern. The correspondences are :

actor "consumer" <=> *actor "client"*
event "consumes" <=> *event "order"*
entity "resource" <=> *entity "product"*

The refinements are:

actor "supplier" <=> *actor "product supplier"*
entity "waiting order" <=> *entity "waiting order"*
event "recovers" <=> *event "recovers"*
event "adds" <=> *event "delivers"*
event "supply request" <=> *event "supplier order"*
event "waits" <=> *event waits for products"*

This leads to the following schema (figure 10):

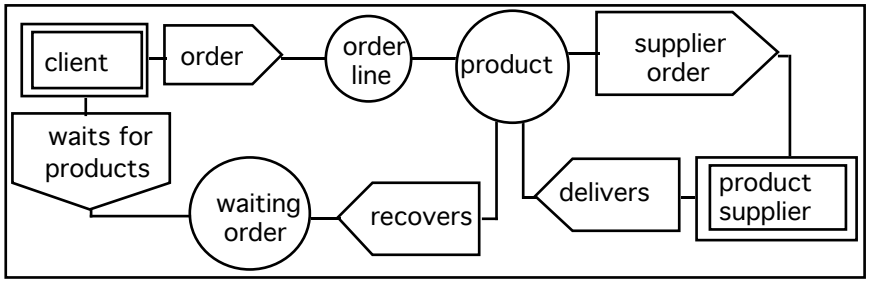

Fig 10 :Using dynamic pattern for the resource "product"

At this stage, no more refinement's triplets can be applied. The tool proposes to map this schema on to the conceptual model by means of other triplets called "structuration triplets". However, we do not present these triplets here, see [Grosz 90b] for details. But, in the same way, by taking decisions, the designer will progressively obtain the static conceptual structure illustrated by figure 11 :

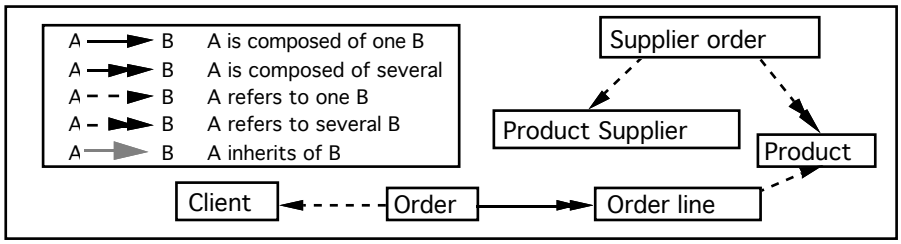

Figure 11 : The static part of the conceptual schema

The dynamic conceptual schema is based on objects (represented with rectangles), actions (represented with rounded boxes) and events (represented with triangles). For the stock control system, the current description leads to the following schema (figure 12):

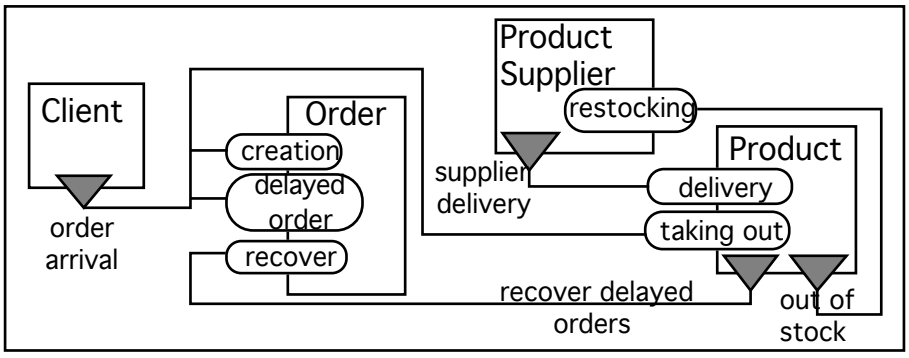

Figure 12 : The dynamic part of the conceptual schema

5 The underlying meta-rules

The RE process is an iterative process. However, the rule base is made of different groups of triplets. For instance, a group of triplets is dedicated to finding the static structure ; another one for the refinement of the real world description. All the different groups which compose the rule base are autonomous. They are linked together by means of different meta-rules. The meta-rule underlying the previous section, is the following. After providing the tool with a simple description of the

problem space, refinement's triplets are exhaustively tried, they lead to a situation where none of them can be applied any more. Then, the triplets which give the structure of objects are triggered.

In fact, we consider the hierarchy between the different ways provided for describing the problem space, from the fuzziest to more constrained one, from the natural language to the conceptual model. The general philosophy underlying the meta-rules is the following: finishing the transformation (and the possible completion) of the fuzziest description before beginning the transformation to more precise level. However, the designer can decide to use the natural language interface even after making a first representation using the real-world description concepts. The tool will immediately integrate these new requirements and transform them onto the real-world description. Finally, the tool will take into account conceptual specification when the real-world description is complete enough.

6 Conclusion

We presented a CASE tool, called CARE (Computer Aided Requirements Engineering), with an expert system architecture. The aim of this tool is to help the designer in the construction of an object-oriented conceptual schema. Different interfaces are provided, a natural language interface and a graphical one.The later allows one to describe (1) the problem space in a simple model called a "real-world" model, and (2) the solution space with the object-oriented model.

The real-world model is based on the notions of actor, event and entity. In order to transform this model into the conceptual one, we looked for the minimal information allowing the tool to deduce the largest number of conclusions. We propose different cardinalities integrating temporal features and a criterion allowing it to deduce dynamic patterns.

Future work will be developed in three major directions:
- the study of visibility between actors and between actors and entities; our aim is to allow the designer to describe precisely who needs what information ; this will add a new dimension to the description of the real-world : the formalisation of the communication inside an IS;
- the search for other dynamic patterns;
- the improvement, through concrete development, of the real-world description.

References

[Arango 87] G. Arango, I. Baxter, P. Freeman : "*A Framework for Incremental Progress, the Application of AI to Software Engineering*", Research Report, Department of Information and Computer Science, Univ. of California, Irvine, USA, May 1987.

[Black 87] W. J. Black : "*Aquisition of Conceptual Data Models from Natural Language*", 3rd European chapter of ACM, Copenhague, 1987.

[Bouzeghoub 85] M. Bouzeghoub,G. Gardarin,E. Metais: "*Database Design Tool:an Expert System Approach"*,Proc. of the 11th VLDB Conference,Stockolm, August 1985.

[Brunet 91] J. Brunet : "*Modeling the World with Semantic Objects*", Proc. of the IFIP Working Conf. on "*The Object Oriented Appraoch in Inofrmation System*", Quebec, Canada, October 1991.

[Cauvet 88] C. Cauvet, C. Rolland, C. Proix : "*Information Systems Design: an Expert System Approach*" in Proc. of the Int. Conf. on Extending Database Technology, Venice, March 1988.

[Chen 76] P.P.S. Chen : "*The entity relationship model : towards a unified view*" ACM Trans. on data base systems, Vol 1, Nb1, 1976.

[Dubois 86] E. Dubois, J. Hagelstein, E. Lahou, A. Rifaut, F. Williams : "*A data Model for Requirements Engineering*", 2nd Int. Conf. on Data Engineering, Los Angeles, 1986

[Dubois 89] E. Dubois, J. Hagelstein, A. Rifaut : "*Formal Requirements Engineering with ERAE",* Philips journal of research, Vol 43, N°3/4, 1989.

[Fillmore 68] C.J. Fillmore : "*The Case for Case*", in Universals in linguistics theory; Holt, Rinehart and Winston, Inc., E. Bach/R.T. Harms (eds) 1968.

[Grosz 90a] G. Grosz, C. Rolland : "*Knowledge Support for Information System Design*", Jerusalem Conference on Information Technology 90, Jerusalem, Israel, october 1990.

[Grosz 90b] G. Grosz, C. Rolland : "*Using Artificial Intelligence Techniques to Formalize the Information System Design Process*", Data-base and EXpert system Application 90 Conference, Vienna, Austria, august 1990.

[IFIP 90] IFIP WG 8.1 Task Group : "*A Framework of Information Systems Concepts*", Interim report N°3, Paul Lindgreen (ed), 1990.

[Kersten 87] M.L. Kersten,H. Weygand,F. Dignum,J. Boom : "*A Conceptual Modelling Expert System*", 5th Entity Relationship Conference, Dijon, France, 1987.

[Prakash 90] N. Prakash : "*On Event Modelling*", Internal Repport, Dpt of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India, December 1990.

[Proix 89] C. Proix, C. Rolland : "*Alecsi: Atelier Logiciel Expert en Conception de Systèmes d'Information*", 2nd Journée Internationale sur le Génie Logiciel et ses Applications, Toulouse, France, December 1989.

[Rolland 87] C. Rolland, O. Foucault, G. Benci, "*Conception des systèmes d'information : la méthode REMORA*", Eyrolles, 1987.

[Rolland 91] C. Rolland, C. Proix : "*Natural Language Approach to Conceptual Modelling*", in "Conceptual modelling, Databases and CASE, an integrated view of information systems development", Ellis Horwood (pub), 1991.

[Sernadas 88] C. Sernadas, J. Fiafeiro and A. Sernadas : "*Object-Oriented Conceptual Modelling from Law*", The Role of Artificial Intelligence in Databases and Information Systems, C-H Kung and R. Meersman (eds), Proc of the IFIP 2.6.8.1 Working Conference, Guangzhou 1988, North Holland.

[Tan 84] Tan Tahn Joo, Tan Kah Poh and Goh Ah Moi : "*Datadict : A Data Analysis and Logical Database design tool*", Proc. of the 10th Very Large Data-Base Conference, Singapore, August 1984.