Integrating Natural Language Oriented Requirements Models into MDA

María Carmen Leonardi María Virginia Mauco
INTIA - Departamento de Computación y Sistemas
Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Pcia. de Buenos Aires
Argentina
{cleonard, vmauco}@exa.unicen.edu.ar

Abstract. MDA is a software development framework where the core is a set of automatic transformation of models. One of these models, the CIM, is used to define the business process model. Though a complete automatic construction of the CIM is not possible, we think we could use some requirements models and strategies adapting them to be used in the MDA framework. We present an OCL based transformation to obtain a structural object-oriented CIM from natural language oriented models.

Keywords: Natural language oriented requirements models, MDA Computer Independent Model, UML object diagram, OCL specifications

1. Introduction

The Model Driven Architecture [1], known as MDA, is a framework for software development defined by the OMG [2]. Key to MDA is the importance of models and transformations between them in the software development process. MDA defines how models defined in one language can be transformed into models in other languages. An MDA development process generally begins with a Computer Independent Model (CIM) which describes the business system independently of the software system to be implemented. There is not too much work on this model, and although it is not possible to construct it automatically [3], there are some works in this direction [4].
We have been working with natural language oriented models which describe the Universe of Discourse [5]. In particular, we have defined manual derivation strategies to obtain object conceptual models [6, 7] and formal specifications [8] from them. We think that some of these manual strategies may be formalized in order to define a semiautomatic transformation from natural language oriented models to a CIM. Through this transformation it would be possible to integrate these models in the MDA framework.
In this paper we present an OCL [9] based transformation process to define a CIM from natural language oriented models, concretely the Language Extended Lexicon (LEL) and the Scenario Model [5]. We also discuss how Requirements Engineering models can fit into the MDA framework, and the possibilities, difficulties and benefits of defining automatic transformations in the first stage of development.

The paper is organized as follows. Section 2 introduces the MDA Framework. Section 3 briefly describes the natural language oriented models used. Section 4 presents our transformation process, exemplifying and discussing each rule. In Section 5 we discuss the automatic transformation process. Finally, Section 6 presents some conclusions and future work.

2. The MDA Framework

MDA is an approach to the full lifecycle integration of enterprise systems comprised of software, hardware, humans, and business practices. MDA is based on modeling different aspects and levels of abstraction of a system, and exploiting interrelationships between these models [10]. In MDA, all artifacts such as requirements specification, architecture descriptions, design descriptions, and code are regarded as models.

One of the key features of this framework is the notion of automatic transformations that are used to modify one model in order to obtain another one. MDA defines how models expressed in one language can be transformed into models in other languages. The Model-Driven development is divided into the following main steps [1]:

- Construct a model describing the business system that is called Computer Independent Model (CIM).
- Construct a model with a high level of abstraction that is called Platform Independent Model (PIM).
- Transform the PIM into one or more Platform Specific Models (PSMs).
- Transform the PSMs to code.

A transformation describes how a model in a source language (source model) can be transformed into a model in a target language (target model). The success of MDA depends on the definition of transformation languages and tools that make a significant impact on full forward engineering processes. MDA is still evolving and many products claim to be complaint with it.

3. Natural Language Oriented Requirements Models

The models presented in this section are well known, used and accepted by the Requirements Engineering community. A complete description of them can be found in [5]. The models are:

Language Extended Lexicon (LEL): It is a structure that allows the representation of significant symbols of the Universe of Discourse. It is composed by a set of symbols which have a name (and a set of synonyms), notions, and behavioral responses. LEL symbols define objects, subjects, verbal phrase and states. When describing LEL symbols two rules must be followed simultaneously: the "circularity principle" and the “minimum vocabulary principle ".

Scenario Model: A scenario describes situations in the Universe of Discourse. A scenario is connected to the LEL and it is composed by: a title to identify it, a goal describing its purpose, a context to define geographical and temporal locations and preconditions, actors which are entities actively involved in the scenario generally persons or organizations, a set of resources that identify passive entities with which actors work, and a set of episodes where each episode represents an action performed by actors using resources. An episode may be explained as a scenario; this enables a scenario to be split into sub-scenarios.

4. The OCL based transformation process to define a CIM

In this section we present a process to obtain an object diagram representing the structural aspects of a CIM. The process consists of a set of steps that apply OCL based tranformation rules to natural language oriented models to define an object oriented diagram. These rules come from the formalization of some of the heuristics proposed in [6, 8], where a complete description can be found.
The process takes as the source model a LEL and a Scenario model from a concrete case study, and follows the steps described below to organize the application of the transformation rules :
- *Identification of classes*: taking as input LEL symbols classified as subjects and objects, transformation rules named TRC1 and TRC2 propose the definition of one class for each symbol. TRC2 also defines the methods for the classes coming from object LEL symbols.
- *Identification of methods*: considering behavioural responses of subject LEL symbols, transformation rule named TRM1 defines the methods for the classes coming from subject LEL symbols (obtained after applying TRC1). Then, transformation rule TRM2 completes the corresponding parameters.
- *Identification of relationships*: the object diagram is completed with the definition of inheritance, aggregation and association relationships through transformation rule TRR by analysing notions of LEL symbols defined as classes.
It is important to remark that this strategy must be complemented with the participation of software engineers who will adjust the results obtained after the application of the transformation rules.
All the tranformation rules mentioned above are completely described in Section 4.2.

4.1 Source and Target Models

Our target model are the Core Package Relationships and the Core Package Backbone from UML V1.5 metamodel [11] that show the structural aspects of a class diagram.
To describe the tranformation rules between the source and target models in a consistent way, we must describe LEL and Scenario Model using an UML object diagram. In this way, we can manage the transformation between them in OCL. Figure 1 shows this UML object diagram which was defined considering the structure and the construction process of LEL and Scenario Model proposed in [5].

4.2 The Transformation Rules

This section describes the transformation rules that allow the mapping between the models. The transformation language we use is based on the transformation language proposed in [1, 9], which is an OCL extension. Each transformation rule specification contains a name, the signature, a brief natural language description, and the OCL

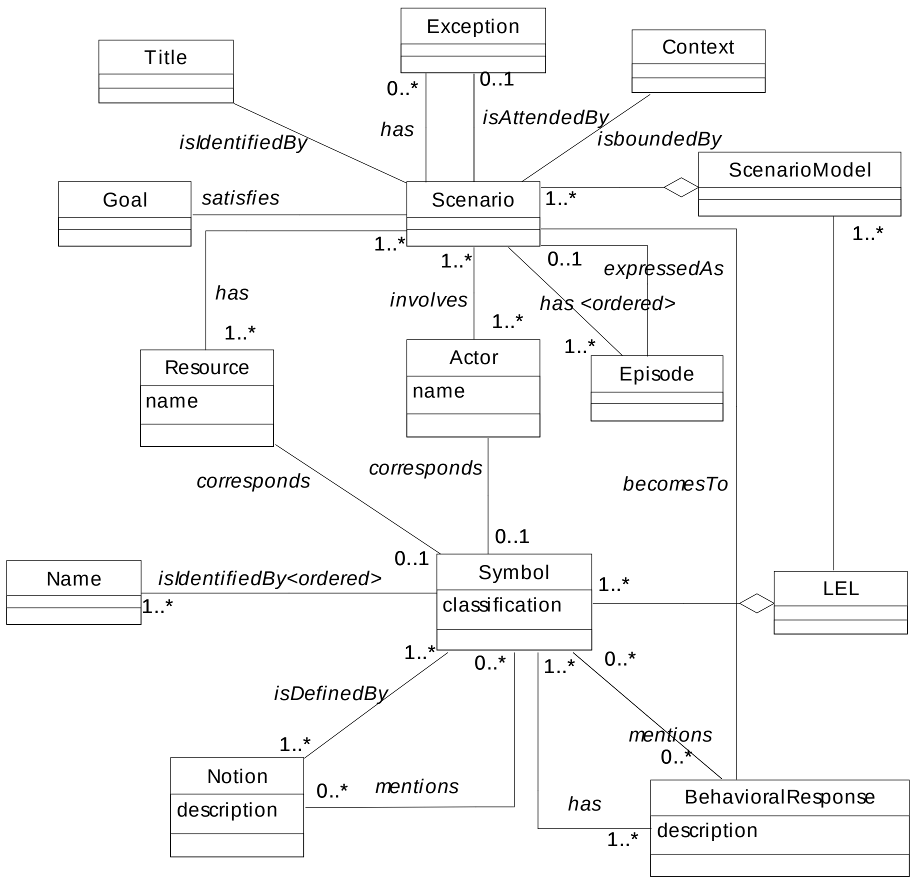


Figure 1. Object-oriented diagram of LEL and Scenario Model

specification. Parameters may be any of the components of the requirements model shown in Figure 1 or any of the components of the target model, referenced in each transformation rule as RM and UML respectively. Besides, another parameter may be included to represent the transformation process model, identified as TP, which contains all the classes with the dictionaries of the language used in the construction of the requirements models (an English dictionary in this case).
We illustrate the application of each rule with examples taken from a Milk Production System [8]. In some rules we also mention how the result would have been if we had used the manual strategies proposed in [6, 7, 8].

TRC1: Transformation SubjectToClass (RM, UML, TP)
-- *Description: Each subject LEL symbol becomes a UML class. The attributes are defined as follows: for each notion that does not contain a LEL symbol, the transformation identifies nouns and defines them as attributes.*
SOURCE: S1: RM:: Symbol
D: TP:: Dictionary

```
TARGET: C1: UML :: Class
SOURCE CONDITION
          S1.classification :: subject
TARGET CONDITION
          C1.name = S1.isIdentifiedBy → first()
          let
              plainNotions :Set =
                         S1.isDefinedBy →excludes (n/ n.mentions -> notempty())
              nounofNotions: Set =
                     plainNotions → collect(n/ D.returnNouns(n.description)) asSet
                at: OrderedSet=
                         C1.features → collect (f/ f.oclIsTypeOf (Attribute))
           in
                     at → forAll (a/ nounofNotions → one (n: String / n = a.name))
```

DAIRY FARMER
NOTION
Person in charge of all the activities in a dairy farm.
He has a name.
He has a salary.
He may have one or more employees.
BEHAVIOURAL RESPONSE
He assigns to a group each cow of the dairy farm.
He saves birth.
He computes individual production of a group.
He computes birth date for each dairy cow or heifer.

Figure 2. Dairy Farmer LEL Symbol

Figure 2 shows a LEL symbol defining a Dairy Farmer. By applying the transformation rule TRC1, the class shown in Figure 3 is defined:

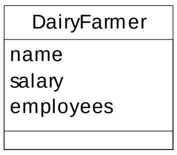


Figure 3. Dairy Farmer class

One of the main problems of this transformation is that it misses noun groups. As the method returnNouns, belonging to the Dictionary class, only detects separate nouns, every noun is a potential attribute, thus generating more and sometimes inappropriate attributes. However, noun groups detection may be included following linguistic approaches [12, 13].

TRC2: Transformation ObjectToClass (RM, UML, TP)
-- *Description: Each object LEL symbol becomes a UML class. The attributes are defined as follows: for each notion that does not contain a LEL symbol, the transformation identifies nouns and defines them as attributes. Methods are defined adding SET and GET prefixes for each attribute.*

```
SOURCE: S1: RM:: Symbol
             D: TP :: Dictionary
 TARGET: C1: UML :: Class
 SOURCE CONDITION
                              S1.classification :: object
 TARGET CONDITION
                              C1.name = S1.isIdentifiedBy → first()
                              let
                                   plainNotions: Set =
                                S1.isIdentifiedBy →excludes (n/ n.mentions → notempty())
                               nounOfNotions: Set =
                                           plainNotions → collect(n/
                                           D.returnNouns(n.desription)) asSet
                               at: OrderedSet=
                                C1.features → collect (f/ f.oclIsTypeOf (Attribute))
                              oper: OrderedSet =
                                     C1.features → select (f/ f.oclIsTypeOf(operation))
                              in
                                 at → forAll (a/nounsOfNotions →
                                        one (n: String /n= a.name))
                                 oper → forAll(o/ at → one(a / o.name = "set" concat
                                (a.name) or o.name = "get" concat (a.name)))
```

The application of the transformation rule to the object LEL symbol Plot, whose notion is described in Figure 4, gives as result the class and attributes shown in Figure 5. By applying the manual heuristics from [6, 8], we would have obtained the following attributes: identification, location (discarded by TRC2 because the notion contains a LEL symbol), size, starting date (TRC2 only considers the noun date), period of duration (TRC2 takes each of them separately). In both last cases, the problem is that the dictionary does not recognize noun groups, as we have mentioned before. Besides, the attribute days obtained applying TRC2 would not be an attribute following the manual approach because human judgement would have realized they are the way in which periods are measured.

PLOT
NOTION
It is a part of a field.
It has an identification.
It has a location inside the field.
It has a size.
It has a starting date.
It has an approximated period of duration in days.
In any time it is occupied by one group.

Figure 4. Plot LEL Symbol

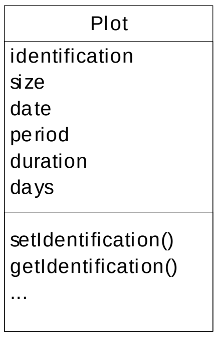


Figure 5. Plot class

TRM1: Transformation SubjectBehavioralResponsesToMethods(RM, UML,TP)
-- *Description: Each behavioral response of a subject LEL symbol modeled as a class by TRC1 becomes a method.*
SOURCE: S1: RM:: Symbol

```
          D: TP :: Dictionary
-- D. ProcessString deletes spaces between strings, and deletes articles, prepositions
and conjunctions, returning nouns and verbs concatenated by _
TARGET: C1: UML :: Class
SOURCE CONDITION
            S1.classification:: subject
            C1.name = S1.isIdentifiedBy → first ()
TARGET CONDITION
            let
                behavioralNames : Sequence =
                     S1.has → (collect (br/ D.processString (br))) → AsSequence
                methods : Sequence =
                        C1.features → collect (f/ f.oclIsTypeOf(Operation))
            in
                methods → forAll ( m/ behavioralNames → one (n: String /
                n= m.name))
```

Applying the transformation rule TRM1 to the LEL symbol shown in Figure 2, the methods described in Figure 6 are obtained.

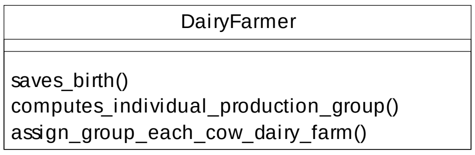
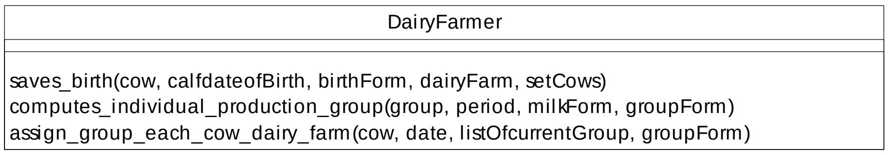

DairyFarmer
saves_birth()
computes_individual_production_group()
assign_group_each_cow_dairy_farm()

Figure 6. Methods of Dairy Farmer class

TRM2:Transformation SubjectInformationToMethodParameter (RM, UML)
-- *Description: Each behavioral response of a subject LEL symbol originates a scenario [5]. This is modeled with the relationship becomesTo (Figure 1). The rule models actors and resources of each scenario as parameters of the method obtained by TRM1 from the behavioral response that originated the scenario. The actor referring to the subject LEL symbol in consideration is excluded.*

```
SOURCE: S1: RM:: Symbol
TARGET: C1: UML :: Class
SOURCE CONDITION
                       S1.classification :: subject
            C1.name= S1. isIdentifiedBy → first ()
TARGET CONDITION
                  let
                     opers: OrderedSet =
                         C1.features → select (f/ f.oclIsTypeOf(Operation)
                  in
                       opers → forAll( o/ o.parameter =
                       (S1.has → select (description=o.name).becomesTo.has →
                       collect (name)) union
                       (S1.has → select (description=o.name).becomesTo.involves
                       → excludes (S1) → collect (name)))
```

For example, for each method previously defined by TRM1 (Figure 6), parameters are identified considering the scenarios involved: Assign a group to a cow, Manage birth, Compute group individual production [8]. As parameters come from resources and actors, they are modeled as classes when the corresponding resource and actor is a subject or object LEL symbol (TRC1 and TRC2); for example, parameters cow and groupForm in the method assign_group_each_cow_dairy_farm (Figure 7). When the resource or the actor does not belong to the LEL, two things may happen. It may be a word that does not need a LEL entry because it belongs to the minimum vocabulary [5], or it may represent a set. In the first case, it is modeled with a primitive class or type (parameter date, Figure 7), and in the second one no new classes are needed because the parameter is a set of a class already defined (parameter listOfCurrentGroup, Figure 7, corresponds to a set of group).



```
DairyFarmer

saves_birth(cow, calfdateofBirth, birthForm, dairyFarm, setCows)
computes_individual_production_group(group, period, milkForm, groupForm)
assign_group_each_cow_dairy_farm(cow, date, listOfcurrentGroup, groupForm)
```

Figure 7. Defining parameters to methods of Dairy Farmer class

TRR: Transformation LELRelationshipsToClassRelationships (RM, UML, TP)
-- *Description: This transformation applies to subject as well as object LEL symbols. Notions of a LEL symbol, called L1, modeled as a class are analyzed in order to detect other LEL symbols also defined as classes. For each LEL symbol detected, named L2, the definition of an association relationship between the corresponding classes is considered, taking into account the following issues:*
INHERITANCE RELATIONSHIPS: L1 and L2 have the same classification (object or subject). Besides, L1 appears in one of the notions of L2. The involved notions of L1 and L2 contain, in a complementary way, two kind of verbs [13]: bottom-up verbs (is a, is a type of, is a class of) or or top-down verbs (is, may be, may be classified as, classifies as).
AGGREGATION RELATIONSHIPS: in the notions of the LEL symbol considered as container, verbs of the type "component_composition_verb" must appear [13]: "to consist / to contain / to include / to form, to compose, to divide" (these three last in passive voice[1]). In the notions of the “component” symbol, verbs of the type content_composition_verb must appear [13]: "it is part, it belongs, it is a component, it is included", among others. As it is not possible to automatically distinguish between an aggregation or a composition relationship, the transformation rule defines the relationship as an aggregation.
ASSOCIATION RELATIONSHIPS: any relationship between LEL symbols that does not represent a relationship of the previous types, represents an association. The verb that appears in the notion (classified as general verb in [13]) is taken as the name of the association.
A complete justification for TRR may be found in [6].

[1] We decided to eliminate the verbs to have and to posses as indicators of aggregation relationships since, from our experience, they are commonly used by stakeholders to describe properties of concepts.

```
SOURCE: S1: RM:: Symbol
             D1: TP:: Dictionary
TARGET: C: UML :: Class
SOURCE CONDITION
                 C.name = S1.isIdentifiedBy(first)
TARGET CONDITION
let
cadidateInheritanceNotions: Set=
        S1.isDefinedBy → select (D1.BottonUpVerbsIncludes(n.description))
CandidateAggregationNotions: Set=
        S1.isDefinedBy → select(D1.Component_Composition.Includes(n.description))
CandidateAssociationNotions: Set=
        S1.isDefinedBy → excludes(cadidateInheritanceNotions union
        candidateAggregationNotions)
in
cadidateInheritanceNotions →forAll (n.mentions → exists (s: Symbol / s.classification =
S1.classification and Class.allInstances→ exists (c1 / c1.name = s.name) and
s.isDefinedBy → exists(n1/ n1.mentions-> includes(S1) and
D1.TopdownVerbsIncludes (n1.description)))
         implies  G.oclIsTypeOf(Generalization) and G.child = c1 and G.parent = C
         and c1.generalization = G and C.especialization = G)[2]
candidateAggregationNotions → forAll (and n.mentions → exists (s: Symbol /
s.classification = S1.classification and (Class.allInstances → exists (c1 / c1.name =
s.name)) and s.isDefinedBy → exists (n:notion / D1.
Content_Composition_VerbIncludes (n.description) and n.mentions → includes (S1)))
         implies A.oclIsTypeOf (Association) and A.connection → at(1).participant = C
         and A.connection → at (1).aggregation = aggregate and  A.connection →
           at(2). participant= c1 and A.connection → at (2).aggregation=none)
candidateAssociationNotions →forAll( n.mentions → exists(s: Symbol/
class.AllInstances → exists(c/c.name=s.name))
          implies A.oclIsTypeOf(Association) and A.connection → at(1).participant= C
          and A.connection → at (2).participant= c1)
```

By applying the transformation rule TRR to the LEL symbols of Figure 8 we obtain a hierarchy with Cow as the superclass and Dairy Cow, Heifer and Calf as subclasses.
Analyzing the LEL symbol Field shown in Figure 9, a notion with a LEL symbol modeled as a class (Plot, Figure 5) containing the “component-composition” verb is “divided into” is found. Besides, a “content-component” verb is found in the notion of the LEL symbol Plot (“it is a part of …”, Figure 4). Therefore, the transformation rule TRR defines an aggregation relationship between Field and Plot classes.
Considering the LEL symbol Dairy Farmer (Figure 2), the transformation rule TRR takes the notion “Person in charge of all the activities in a dairy farm.” because it mentions another LEL symbol, Dairy Farm (Figure 10), modeled as a class. Inheritance and aggregation relationships are rejected because the verb involved is a general one. Then, a general association is defined between both classes.

[2] To simplify the OCL expression we have omitted the expression to define c1 in the right side of each implies expression.

COW
NOTION
…
It may be a calf, a heifer, or a dairy cow.
…

DAIRY COW
NOTION
It is a female cow which has had at least one calf.
…

CALF
NOTION
It is a cow of less than 12 months age.
…

HEIFER
NOTION
It is a female cow of 12 months age or more which has not had a calf.
...

Figure 8. Some LEL symbols

FIELD
NOTION
Land where cows eat pasture.
It has an identification.
It has a precise location in the dairy farm.
It has a size.
It has a pasture.
It has an hectare loading.
It is divided into a set of plot.
It has a list of previous plot.
...

Figure 9. Field LEL Symbol

DAIRY FARM
NOTION
…
It is managed by a dairy farmer.
…

Figure 10. Dairy Farm LEL Symbol

5. Discussing the Transformation Process

The application of the transformation rules allows a systematic definition of a tentative object-oriented CIM. Though a manual derivation produces a better and more accurate model definition, transformation rules are a starting point to deal with the great amount of requirements information. They provide a systematic and consistent way of defining CIM´s in MDA framework. The CIM should be later refined by a human, who will correct and complete it.
Considering our experience with manual derivation strategies and the semiautomatic transformation we propose in this paper, we want to discuss the following issues:
- Our proposal is mainly based in the metamodel of LEL. The transformation rules were defined considering the way in which the concepts of the Universe of Discourse are described, explicitly defining structural and behavioral aspects of them. For example, definition of classes is based on the classification of LEL symbols, automatically modeling one class per each subject or object LEL symbol. The strategy to find methods and parameters is also based on the structure of the models. However, to identify attributes we have to analyse the text of notions. In this first approach, we follow a basic linguistic strategy to find nouns in notions, causing some of the

problems presented in Section 4.2. In order to address this problem, and enhance and refine the strategy, a linguistic analysis must be done [12, 13, 14, 15].
- We think the free style to express the content of notions and behavioral responses of LEL symbols makes difficult the automatic processing of the information they describe. Manual heuristics could use human intelligence to take the final decision. In some cases, it would be possible to define a standard form of writing without restricting the power of expression of natural language.
- Though LEL and scenarios have a precise structure, the use of natural language allows the same semantics to be usually expressed with many different natural language sentences. For example, in some cases the same concept may be described with a noun or a verbal phrase since each essential concept has a root expression as a noun, a verb or even as an adjective [15]. The manual strategies already mentioned use human judgement to decide if a verbal phrase should be modeled as a class or as a method. An automatic transformation takes always the same decision loosing, in some cases, the real meaning of the essential concept. In our proposal LEL verbal phrases remain as methods of classes modeling subject LEL symbols. We take this decision to avoid the definition of classes with only one method, as advised in [16]. Later, this may be modified by the software engineer.
Natural language oriented models are widely used in requirements modeling due to their well-known advantages [5]. This kind of requirements models have to be reinterpreted by software engineers into a more formal design on the way to a complete implementation. Therefore, a semiautomatic transformation to map their knowledge into conceptual object models would be really useful. Our proposal is a first step into this direction, aligning with the MDA framework.

6. Conclusions and Future Work

In this paper we have sketched a first proposal to define, in a semiautomatic way, an early objet oriented CIM starting from natural language oriented requirements models. The transformation process we propose fits into MDA process as it can be automated, and as a consequence it may be implemented by a tool, enhancing in this way the construction of the first MDA model, currently obtained in a manual way [1, 3]. In addition, we also take advantage of all the time and effort the definition of requirements and business models consumes, thus reducing the gap between requirements and other development models.
Transformation rules are a concrete automatization of some of the manual heuristics proposed in [6, 8], and then they involve fix decisions about certain modelization issues. As a consequence, this strategy unavoidably needs software engineer´s participation in order to adjust the results obtained after the application of the transformation rules.
In order to complete the transformation process, we must define the transformation rules of the business rule model [17], based on the manual heuristics proposed in [6]. We must also define transformation rules to include the dynamic aspects of the models; in this case, we want to define rules for the definition of interaction diagrams from scenarios. To do this we may study approaches like [12, 14]. We also want to study the possibility of formalizing other object oriented model derivation strategies, for example the proposal presented in [18] that defines an object model from i*. As

another step to improve the complete strategy and making consistent source and target models specification, we will propose an UML profile to define the requirements models used in the transformation strategy.
As we have discussed in Section 5, it would be necessary to incorporate linguistic approaches to achieve a better processing of the information. In addition, we have to test the strategy in more case studies.
Traceability plays a crucial role. The transformation process we have proposed allows the trace between the source and the target. However, we want to enhance this mechanism by defining another complementary and independent model to capture and represent the relationships created by the application of the transformation rules, as the one proposed in [6].

References

[1] Kleppe, A., Warmer, J., Bast, W., *MDA Explained: The Model Driven Architecture™: Practice and Promise*, Addison Wesley, 2003.
[2] OMG: Object Management Group. http://www.omg.org/
[3] Eriksson, H., Penker, M., *Business Modeling with UML: Business Patterns at Work*, John Wiley & Sons, 2000.
[4] "Business Processes and the OMG: an overview", http://www.omg.org/bp-corner/bp-files/The_OMG_BP_Corner_INTRO_Paper_3-2-04.pdf.
[5] Leite, J.C.S, Hadad, G., Doorn, J., Kaplan, G., "A Scenario Construction Process", *Requirements Engineering Journal*, 5(1), Springer Verlag, 2000, pp. 38-61.
[6] Leonardi, M.C., "Una Estrategia de Modelado Conceptual de Objetos basada en Modelos de Requisitos en Lenguaje Natural", Master Thesis, Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina, November 2001.
[7] Leonardi, M.C., "Enhancing RUP Business Model with Client-Oriented Requirements Models", *UML and the Unified Process*, ed. Favre, L., IRM Press, Chapter 6, 2003, pp. 80-115.
[8] Mauco, M.V., "A Technique for an Initial Specification in RSL", Master thesis, Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina, July 2004.
[9] Warmer, J., Kleppe, A., *The Object Constraint Language: Getting Your Models Ready for MDA*, Second Edition, Addison Wesley, 2003.
[10]D´Souza,D., "Model Driven Architecture", www.omg.org/mda/presentation.html.
[11] Unified Modeling Language Specification. V.1.5. March2003. http://www.omg.org/ocl
[12] Díaz, I., Pastor, O., Moreno, L., Matteo, A., "Una Aproximación Lingüística de Ingeniería de Requisitos para OO-Method", *Proc. IDEAS'2004: IIV Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software*, Perú , May 2004, pp.270-281.
[13] Juristo, N., Moreno, A., López, M., "How to Use Linguistic Instruments for Object-Oriented Analysis", *IEEE Software*, 17(3), May/June 2000, pp. 80-89.
[14] Li, L., "Translating Use Cases to Sequence Diagrams", *Proc. of the Fifteenthe IEEE International Conference on Automated Software Engineering,* 2000, pp 293-296.
[15] Boyd, N. "Using Natural Language in Software Development", *Journal of Object Oriented Programming-JOOP*, 11(9), February 1999, pp 45-55.
[16] Meyer, B., *Object-oriented Software Construction*, Prentice Hall, 1997.
[17] Leite, J.C.S.P, Leonardi, M.C., "Business Rules as Organizational Policies", *Ninth International Workshop on Software Specification and Design*, IEEE Computer Society Press, Japan, April 1998, pp. 68-76.
[18] Castro, J., Mylopoulos, J., Alencar, F. M. R. , Cysneiros Filho G., "Integrating Organizational Requirements and Object Oriented Modeling", *Proc. of the Fifth International Syposium on Requirements Engineering,* Canada, 27-31 August, 2001. pp. 146-153.