

# A Novel CASE Tool based on Pre-Conceptual Schemas for Automatically Obtaining UML Diagrams

## Una Novedosa Herramienta CASE basada en Esquemas Preconceptuales para la Obtención Automática de Diagramas UML

Carlos Zapata, PhD.<sup>1</sup>, Alexander Gelbukh, PhD.<sup>2</sup>, Fernando Arango, PhD.<sup>1</sup>

<sup>1</sup>Grupo de Investigación en Ingeniería de Software  
Escuela de Ingeniería de Sistemas, Facultad de Minas  
Universidad Nacional de Colombia Sede Medellín

<sup>2</sup>Computing Research Center, National Polytechnic Institute, Mexico  
cmzapata@unal.edu.co, farango@unal.edu.co, gelbukh@gelbukh.com

Recibido para revisión 26 de Marzo de 2007, aceptado 15 de Junio de 2007, versión final 31 de julio de 2007

**Resumen**—Las herramientas CASE han asistido a los analistas en el trazado de diagramas UML y otros tipos de diagramas para el desarrollo de software. Sin embargo, la tarea previa al trazado de diagramas, que es la comprensión del discurso del interesado, no es soportada por las herramientas CASE tradicionales. Para este fin, el Procesamiento del Lenguaje Natural propuso un nuevo tipo de herramientas CASE, que incluye tanto interpretación del lenguaje natural como generación de diagramas UML. En este artículo se introduce UNC-Diagramador, una novedosa herramienta CASE para la representación gráfica del discurso del interesado mediante esquemas preconceptuales. UNC-Diagramador también es capaz de transformar automáticamente los esquemas preconceptuales en tres diagramas de UML 2.0. Finalmente, el uso del UNC-Diagramador se demuestra con un ejemplo

**Palabras Clave**—Herramientas CASE, Esquemas Preconceptuales, UML 2.0, Proceso de Desarrollo de Software.

**Abstract**—Assistance is provided, in software development process, to Analysts in drawing UML diagrams and others by means of CASE tools. However, the task of the Stakeholder discourse understanding, a previous process in diagram drawing, is not supported by traditional CASE tools. In order to complete this task, Natural Language Processing has proposed a new kind of CASE tools, including both natural language interpretation and UML diagrams generation. We introduce, in this paper, UNC-Diagrammer, a novel CASE tool for graphically representing the Stakeholder discourse by means of Pre-conceptual Schemas. We also show that UNC-Diagrammer is capable of automatically transforming Pre-conceptual Schemas

into three UML 2.0 diagrams. We finally demonstrate the use of UNC-Diagrammer through an example.

**Keywords**—CASE Tools; Pre-Conceptual Schemas, UML 2.0, Software Development Process.

### I. INTRODUCTION

WHEN a Stakeholder needs a solution to his/her information problems, an Analyst can begin a software development process. In the beginning, this process often comes up with a series of Analyst-Stakeholder interviews, which result in a preliminary interpretation of the problem [1]. Then, Analyst must convert his own interpretation into a set of diagrams (commonly UML diagrams) to continue with the process. Computer-Aided Software Engineering (CASE) has been created to provide assistance to Analysts in drawing these diagrams and assisting with other tasks, like code generation, scheduling development tasks, controlling software versions, and so on [2].

There are many CASE tools like Rational Rose [3], ArgoUML [4], Poseidon [5], and Fujaba [6], which provide assistance for UML diagram drawing, consistency checking, and code generation. Unfortunately, these tools lack mechanisms for either discourse interpretation or representation; the Analyst must create his own diagrams and then, aided by the CASE tool, he/she can draw them. In other words, there is no assistance in the conceptualization of the Stakeholder discourse, and this unguided process generates

too many interpretation problems, because Stakeholder cannot understand UML diagrams, and, consequently, he/she cannot validate such diagrams.

As a result, Natural Language Processing has a new trend to generate automatically UML diagrams from natural language. Projects like LIDA [7], RADD [8], NL-OOPS [9], CM-BUILDER [10], and NIBA [11], are testing the future of this trend. They are part of a new generation of CASE tools, closer to the Stakeholders. On the downside, every CASE tool uses a different representation of the Stakeholder discourse, and, even worse, uses a different representation for every target diagram.

We introduce, in this paper UNC–Diagrammer, a CASE tool for drawing a graphical representation of the Stakeholder discourse, the so-called Pre-conceptual Schemas [12], which contains a set of elements translatable to UML diagrams. UNC–Diagrammer also makes these transformations from Pre-conceptual Schemas into three kinds of UML 2.0 diagrams: Class, Communication, and State Machine Diagrams.

This paper is organized as follows: in Section II we discuss the state-of-the-art in CASE tools, including a new trend in CASE tools for automatic obtaining of UML diagrams. In Section III we introduce UNC–Diagrammer, a CASE tool for assisting analysts in drawing Pre-conceptual Schemas and translating them into UML diagrams. In Section IV we present an example of the use of UNC-Diagrammer. Finally, in sections V and VI we present conclusions and future work.

## II. CASE TOOLS: STATE-OF-THE-ART

### A. Conventional CASE tools

In the early-Eighties, Manley coined the term Computer-Aided Software Engineering (CASE) to identify a set of

processes, techniques, and tools emerged with the recently created Software Engineering [2]. CASE Technology has been growing up ever since this moment, but only until middle-Nineties it could reach the top of modeling technologies, with the development of UML-based CASE tools like Rational Rose [3].

Nowadays, there are many UML-based CASE tools: ArgoUML [4], Poseidon [5], Together [13], WithClass [14], Fujaba [6], and so on. Figure 1 shows a snapshot of a screen in ArgoUML [4], a typical UML-based CASE tool. There are seven available diagrams to draw in ArgoUML (Class, Use Case, State Machine, Activity, Collaboration—currently named Communications in the UML 2.0 standard—, Deployment, and Sequence Diagrams). For diagram development in ArgoUML, the Analyst must know the diagram syntax and must have a self-interpretation of the Stakeholder discourse domain, so he/she can represent it in the seven available diagrams. Furthermore, if a Stakeholder wants to take part of this process, he/she must know the syntax of every UML diagram to know how exactly his/her discourse matches every UML diagram. This task is often difficult to complete for Stakeholders, because they are people with lesser or none of the required expertise in UML modeling.

Additionally, some of the described UML-based CASE tools (certainly a little subset of them) have a mechanism for consistency checking between diagrams. ArgoUML is one of such tools, and the button “By Priority” in the bottom-left corner in Figure 1 represents this mechanism. In conventional CASE tools every diagram must be independently created, and there is no unified view about either the problem or the Stakeholder discourse. This is why consistency checking is needed in this kind of tools. Consistency checking has been a topic of research in Software Engineering for many years, and even there is no satisfactory agreement about this topic.

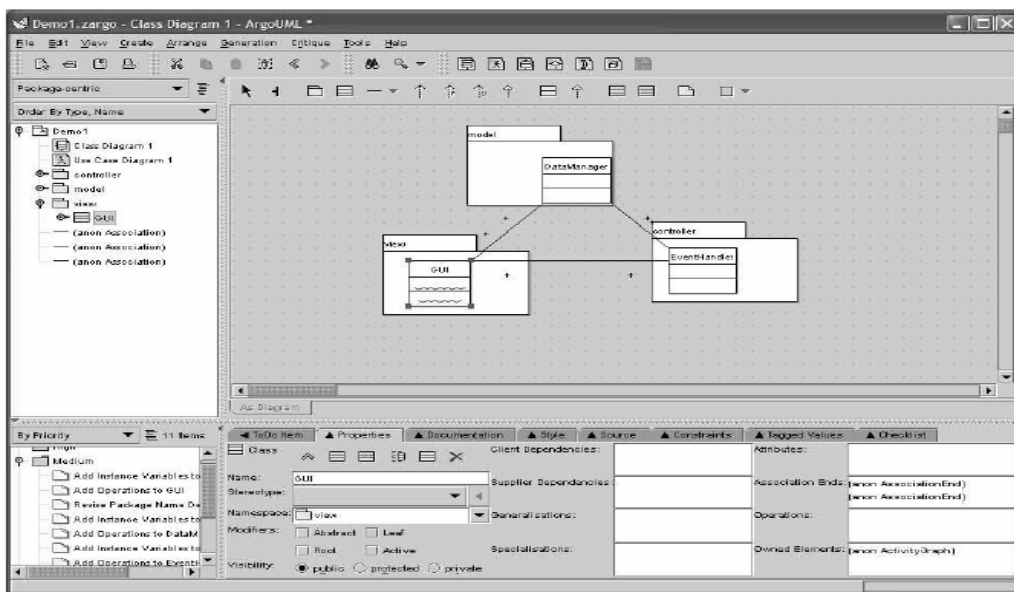


Figure 1. Snapshot of the ArgoUML®, one of the traditional CASE tools.

### B. Automatic generation of UML diagrams: a new kind of CASE tools

By the end of the Nineties, there was a number of projects which was using natural language processing for automatic or semi-automatic generation of UML diagrams. Some of these projects developed CASE tools including this technology. Their results are discussed in this Section.

Linguistic Assistant for Domain Analysis (LIDA Project) is a semi-automated approach for building class diagrams from a discourse in natural language. In the CASE tool generated by the LIDA project [7], some words in the natural language discourse are identified as nouns, verbs, and adjectives, and the tool calculates frequencies of word appearance in the text. Then, Analyst must decide if a specific word will be mapped up to a class, an attribute, an operation or a relationship in the class diagram. Mapping process is left intentionally to Analyst and there are no rules or suggestions to ease this task. Finally, LIDA CASE tool shows the generated diagram and makes it available to be edited.

Rapid Application and Database Development (RADD Project) is a project for automated obtaining of Entity-Relationship Models (ERM) from natural language [8]. In RADD, once an ERM is obtained, a “moderated” dialogue starts to enhance completeness of the diagram.

Class Model Builder (CM-Builder Project) is a CASE tool which performs linguistic analysis of documents in order to obtain three results: a list of candidate classes, a list of candidate relationships, and a conceptual model in an interchange format [9]. In other words, CM-Builder was developed for automated obtaining of UML class diagram.

Natural Language Object-Oriented Product System (NL-OOPS Project) is a CASE tool for semi-automated obtaining of UML class diagram [10]. The starting point is a natural language discourse and NL-OOPS needs Analyst cooperation in the identification process. NL-OOPS identifies elements which can simultaneously belong to several categories. Consequently, mapping process requires active participation of the Analyst, who must decide about the final category for every element.

Natural Language Requirements Analysis (NIBA Project) is a CASE tool developed for obtaining various UML diagrams [11]. NIBA has a strong linguistic analysis that is based in the so-called NTMS (Naturalness Theoretical Morpho-syntax) and the KCPM (Klagenfurt Conceptual Pre-design Model). NIBA can generate various kinds of UML diagrams like class and activity diagrams. For every diagram, KCPM has a different intermediate formalism, varying from tables to graphs. For this reason, consistency problems may arise in the transformation process.

### C. Problems to be solved

Despite the efforts of researchers in CASE tools, there are still problems to be solved:

- Many of these CASE tools are focused on only one

diagram. In a typical software development process there are many diagrams related to the same model.

- The only tool that does construct many diagrams is NIBA, and every diagram needs a different intermediate representation in this tool. For this reason, consistency problems between diagrams can be generated.

### III. UNC-DIAGRAMMER: A CASE TOOL FOR AUTOMATIC TRANSFORMATION OF PRE-CONCEPTUAL SCHEMAS INTO UML DIAGRAMS

UNC-Diagrammer (Universidad Nacional de Colombia—UML Diagram maker) tries to solve the problems of the previous section, based on two hypotheses:

- Stakeholder discourse contains the needed information for automatic drawing of several types of UML diagrams.
- A unified representation of the Stakeholder discourse helps to avoid some of the common consistency problems between UML diagrams.

UNC-Diagrammer uses Pre-conceptual Schemas [12] for representing Stakeholder discourse.

#### A. Pre-conceptual Schemas

A Pre-conceptual Schema (PS) [12] is an intermediate stage of the mapping process between natural language and conceptual UML diagrams. Consequently, a PS contains a graphical representation of a set of UML elements originated from different UML diagrams, in such a way that stakeholders can “read” the diagram to understand its meaning. Figure 2 shows the basic symbols of Pre-conceptual Schemas.

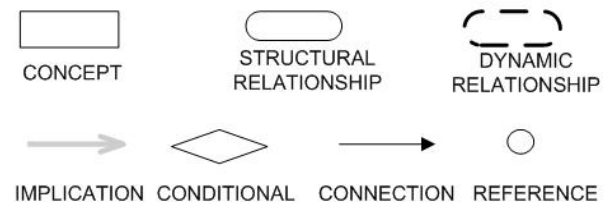


Figure 2. Basic symbols of Pre-conceptual Schemas.

The contents of every symbol are the following:

- A concept can be either a noun or a noun phrase.
- A structural relationship can be a verb that refers to a permanent relationship between two concepts. Verbs like “to be” and “to have” are intended to be structural relationships.
- A dynamic relationship can be a verb that refers to a temporal relationship between two concepts. Verbs like “to register” and “to dispatch” are intended to be dynamic relationships.
- An implication must link two dynamic relationships. The first one is the antecedent and the second one is the consequent of an “if... then” relationship.
- A conditional is a clause formed by concepts and operations between concepts. The value of a conditional can be one of the following values: true or false.
- A connection is an oriented arrow to join:

- A concept with a—structural or dynamic—relationship.
- A—structural or dynamic—relationship with a concept.
- A conditional with a dynamic relationship.
- A reference is a circle with a number within. A reference helps to identify connections between several distant elements.

#### B. An Overview of UNC-Diagrammer

UNC-Diagrammer is a CASE tool based on Pre-conceptual Schemas with the capabilities of generating three UML 2.0 diagrams: Class, Communication, and State Machine Diagrams. Figure 3 depicts the architecture of UNC-Diagrammer.

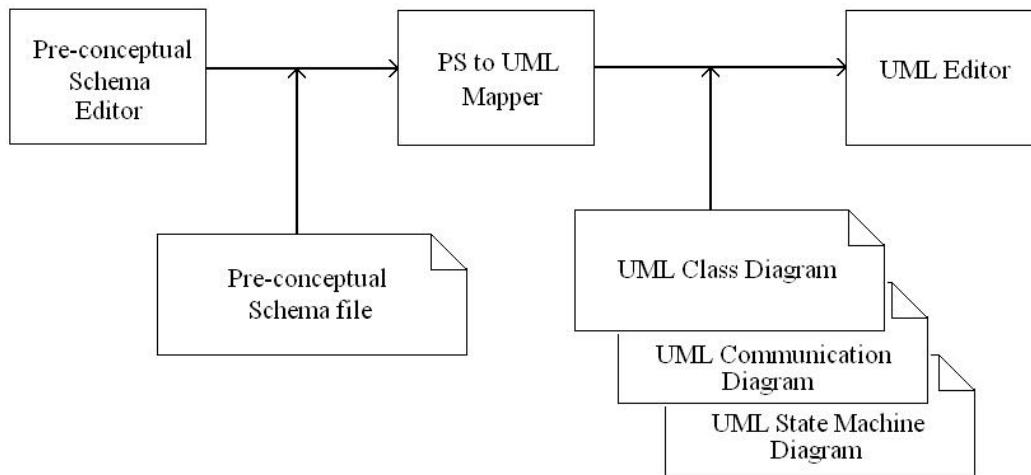


Figure 3. Architecture of UNC-Diagrammer

The following is a description of the UNC-Diagrammer functions and components:

- Pre-conceptual Schema Editor: Allows visually designing and editing Pre-conceptual Schemas. PS Editor is based on a subset of Microsoft Visio application. Figure 4 shows a snapshot of the PS Editor.
- Pre-conceptual Schema File: PS Editor saves information in a XML file which is used as an input for the next component. The XML file mixes both graphical features (for example, the position in the screen, the shape of the elements, and the features of the shape) and logical features (for example, the connections between two elements) of the Pre-conceptual Schema. An example of the XML file belonging to a Pre-conceptual Schema is provided in Section 4.
- PS to UML Mapper: Automatically translates Pre-conceptual Schemas to three UML Diagrams: Class, Communication, and State Machine Diagrams. PS to UML Mapper is based on the C# language.
- UML Files: PS to UML Mapper saves information of the three obtained diagrams in one XML file. Again, the XML file mixes both graphical features and logical features of every target diagram into one single file. An example of the XML file that describes the three target diagrams is provided in Section 4.
- UML Editor: Allows visually editing the obtained UML

diagrams. UML Editor is also based on a subset of Microsoft Visio application.

As in well-known CASE tools, Analyst must interpret Stakeholder discourse for translating it into Pre-conceptual Schemas. However, Pre-conceptual Schemas are closer to Stakeholder discourse and Stakeholder can validate them. In traditional CASE tools, Stakeholder validation is very difficult, because Stakeholder commonly lacks technical knowledge about UML.

For using UNC-Diagrammer, the Analyst (referred to Figure 4) must execute the following actions:

- (1) Create and edit Pre-conceptual Schemas in Edition Space, using PS symbols.
- (2) Press the “Save” button.
- (3) Press the “Convert” button.
- (4) Optionally, edit obtained diagrams.

#### IV. AN EXAMPLE OF THE USE OF UNC-DIAGRAMMER

Figure 5 presents a Pre-conceptual Schema made in UNC-Diagrammer and corresponding to a discourse about a farm dairy. Pre-conceptual Schema has intended to be understandable by stakeholder, as we can see in Figure 5; the demonstration of this assertion is outside the scope of this paper.

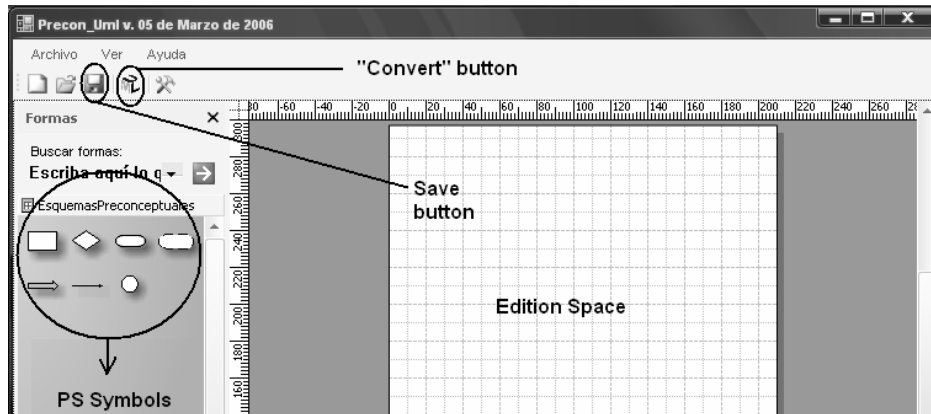


Figure 4. Snapshot of UNC-Diagrammer

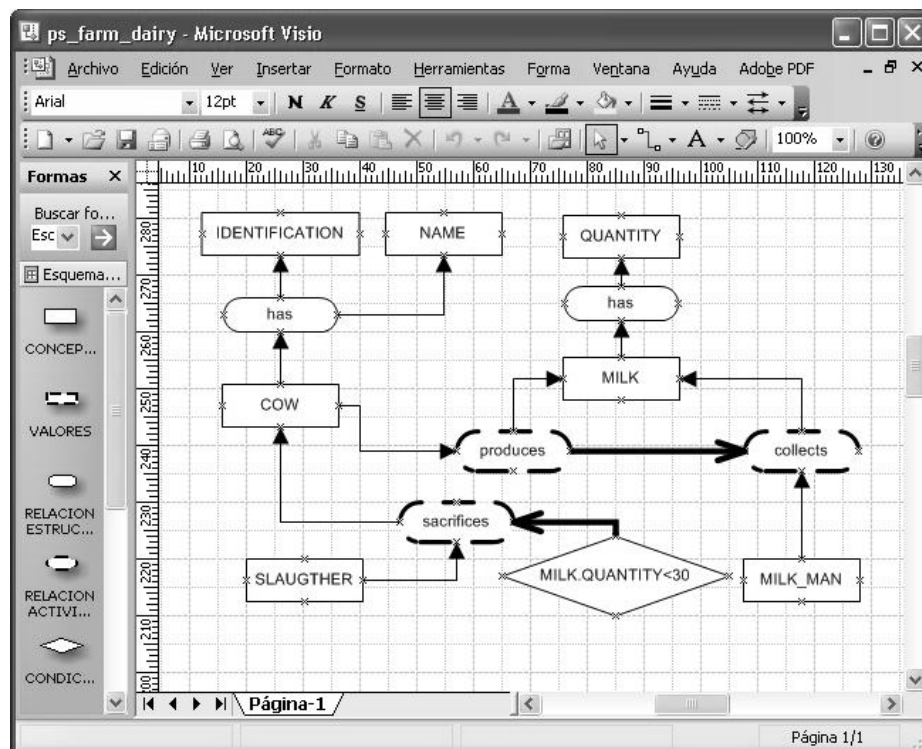


Figure 5. Pre-conceptual Schema of a farm dairy

This Pre-conceptual Schema can be expressed in the following sentences (written in a kind of controlled language):

- Cow has identification.*
- Cow has name.*
- Milk has quantity.*
- When cow produces milk, milk\_man collects milk.*
- When milk.quantity<30, slaughter sacrifices cow.*

These sentences are part of a broader discourse. Pre-conceptual Schemas are the product of the Analyst interpretation of Stakeholder discourse. Furthermore, Pre-conceptual schemas are easily understandable by stakeholders, who are capable of suggesting improvements to this schema.

As we stated before, UNC-Diagrammer stores a first XML

file for the Pre-conceptual Schema. Figure 6 shows a fragment of this file. This Figure contains a set of tags for describing the implication between the “milk.quantity<30” conditional and the “sacrifices” dynamic relationship. Under the <shapes> tag, the file describes the elements belonging to the Pre-conceptual Schema. There is a set of tags (<XForm>, <Event>, <LayerMem>, <Misc>, <Char IX=”0”>, and <Geom IX=”0”>) for expressing graphical features of the elements. Also, for every shape the file provides identification, a name, a type, and a master (a special code for identifying the type of element). The <Text> tag defines the message displayed inside the shape. In this way, we can describe all the elements of the Pre-conceptual Schema in order to complete a transformation process into UML diagrams.

```

- <Shapes>
- <Shape ID="26" Type="Shape" Master="4">
+ <XForm>
+ <Event>
+ <LayerMem>
+ <Misc>
+ <Connection IX="0">
+ <Connection IX="1">
+ <Connection IX="2">
+ <Connection IX="3">
+ <Char IX="0">
- <Text>
  <cp IX="0" />
  sacrifices
</Text>
</Shape>
- <Shape ID="30" NameU="CONDICIONAL" Type="Shape" Master="5">
+ <XForm>
+ <Event>
+ <LayerMem>
+ <Misc>
+ <TextXForm>
+ <Char IX="0">
+ <Connection IX="0">
+ <Connection IX="1">
+ <Connection IX="2">
+ <Connection IX="3">
+ <Geom IX="0">
- <Text>
  <cp IX="0" />
  MILK.QUANTITY<30
</Text>
</Shape>

```

Figure 6. Fragment of the XML file for the Pre-conceptual Schema depicted in Figure 5

Figures 7–9 show the results of pressing “convert” button, after Pre-conceptual Schema has been created. Internally,

UNC-Diagrammer applies a set of conversion rules encoded in C#, for automatic obtaining of Class diagram, Communication diagram, and State Machine diagram. Analyst can edit the resulting diagrams, using the Visio-based templates for this task.

UML diagrams generated by UNC-Diagrammer are stored also in a XML file, as depicted in Figure 10. In this Figure, the “Milk” class is represented in conjunction with its attributes and operations. Again, the XML file has a set of tags for expressing graphical features of the diagram. Furthermore, the image of the “Milk” class is composed by three images: one for the name of the class, one for the attributes, and one for the operations. In a similar way, the XML file stores all of the obtained UML diagrams.

### V. CONCLUSIONS

There is a new trend in CASE tools focused on assisting Analysts in tasks of automated conceptual design of UML diagrams, instead of only drawing them. UNC-Diagrammer follows this trend, using the so-called Pre-conceptual Schemas.

The use of UNC-Diagrammer can assist Analyst with UML diagram generation, and provides syntax for the use of Pre-conceptual Schemas as a way to represent stakeholder discourse. The resulting diagrams are consistent to each other, because they are generated from the same Pre-conceptual Schema.

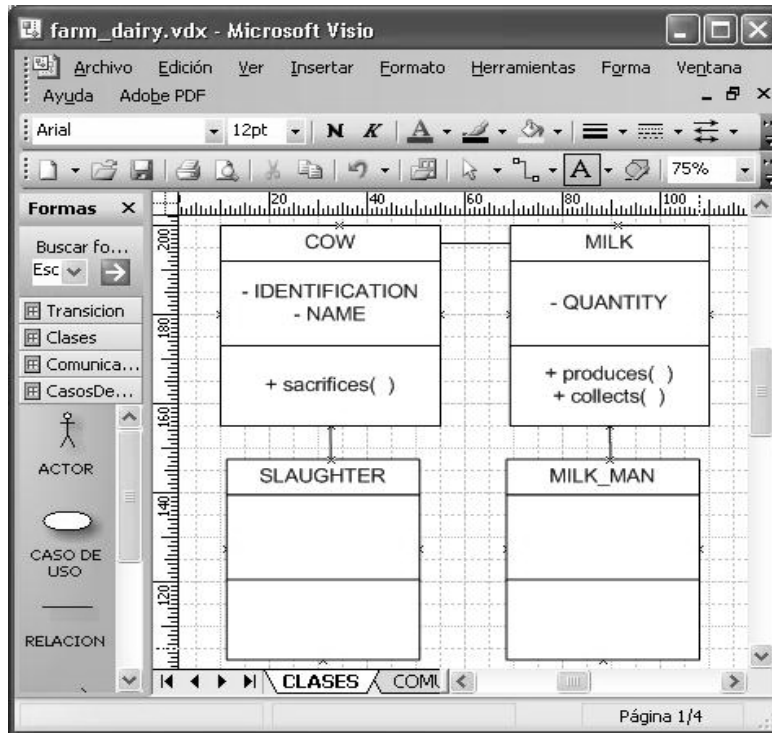


Figure 7. Snapshot of the resulting Class Diagram

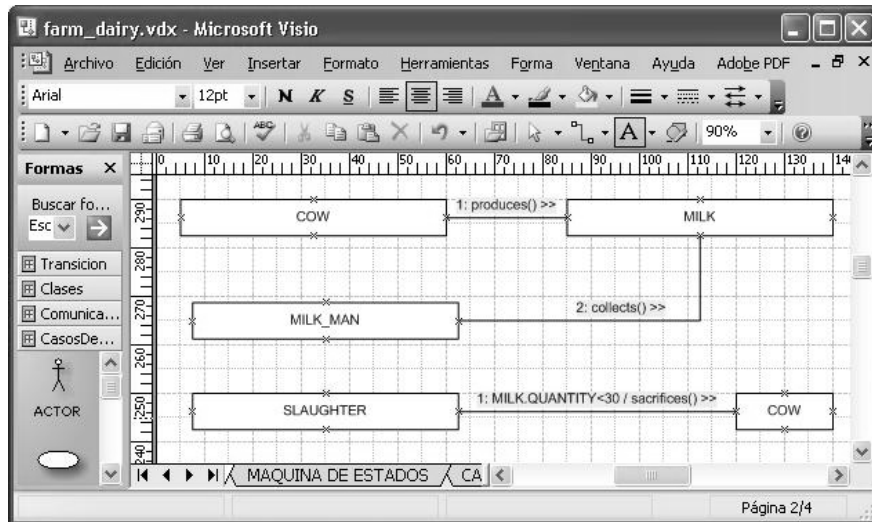


Figure 8. Snapshot of the resulting communication diagram

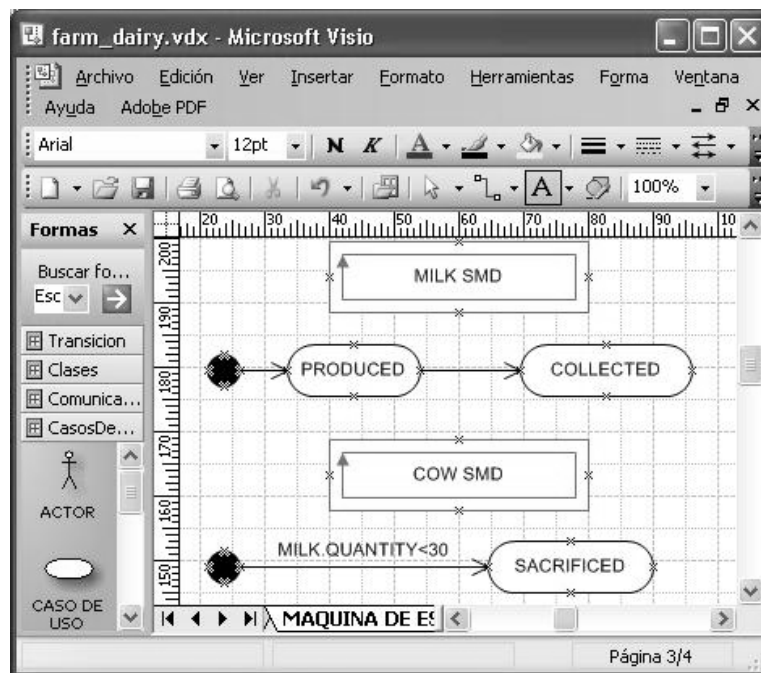


Figure 9. Snapshot of the resulting state machine diagram

UNC-Diagrammer is based on .NET and Microsoft Visio technologies. The diagram editors use the Microsoft Visio functionality and the transformation process is encoded in C# under .NET.

## VI. FUTURE WORK

UNC-Diagrammer has many tasks to accomplish for future development:

- Nowadays UNC-Diagrammer is only capable to generate Class diagrams, Communication Diagrams and State Machine Diagrams, and the set of UML available diagrams—including sequence diagrams, use case diagrams

and so forth—must be completed.

- Interoperability with another CASE tool must be accomplished, to complete the process of translating UML diagrams to source code.
- A major task of a broader project is the automatic obtaining of Pre-conceptual Schemas from natural language (or maybe from a controlled natural language in the beginning), trying to make UNC-Diagrammer closer to the stakeholder, in his/her own language.

## ACKNOWLEDGMENT

This work is supported in the following projects: “Construcción Automatica de Esquemas Conceptuales a partir

de Lenguaje Natural”, financed by DIME and “Definición de un Esquema Preconceptual para la Obtención Automática de Esquemas Conceptuales de UML”, financed by DINAIN and managed by DIME.

```

- <Shapes>
- <Shape ID="2053" Type="Shape" MasterShape="5">
+ <XForm>
+ <Event>
+ <Misc>
+ <TextBlock>
+ <Connection IX="0">
+ <Connection IX="1">
+ <Connection IX="2">
+ <Connection IX="3">
+ <Char IX="0">
+ <Para IX="0">
  <Tabs IX="0" />
+ <Geom IX="0">
- <Text>
  <cp IX="0" />
  <pp IX="0" />
  <tp IX="0" />
  MILK
</Text>
</Shape>
- <Shape ID="2054" Type="Shape" MasterShape="6">
+ <XForm>
+ <TextBlock>
+ <Char IX="0">
+ <Char IX="1">
+ <Para IX="0">
  <Tabs IX="0" />
+ <Geom IX="0">
- <Text>
  <cp IX="0" />
  <pp IX="0" />
  <tp IX="0" />
  - QUANTITY
</Text>
</Shape>
- <Shape ID="2055" Type="Shape" MasterShape="7">
+ <XForm>
+ <TextBlock>
+ <Char IX="0">
+ <Char IX="1">
+ <Char IX="2">
+ <Char IX="3">
+ <Char IX="4">
+ <Para IX="0">
  <Tabs IX="0" />
+ <Geom IX="0">
- <Text>
  <cp IX="0" />
  <pp IX="0" />
  <tp IX="0" />
  + produces ( ) + collects ( )
</Text>
</Shape>
</Shapes>

```

Figure 10. A fragment of the XML file that stores UML generated diagrams

## REFERENCES

- [1] R. Pressman. *Software Engineering: A Practitioners' Approach 5th edn.* New York: McGraw-Hill, Inc., 2001.
- [2] D. Burkhard, and P. Jenster, "Applications of Computer-Aided Software Engineering Tools: Survey of Current and Prospective Users", *Data Base 20*, no. 3, pp. 28-37, 1989.
- [3] Rational Rose. [Online] Available: <http://www-306.ibm.com/software/rational/>.
- [4] ArgoUML. [Online] Available: <http://argouml.tigris.org/>.
- [5] Poseidon. [Online] Available: <http://gentleware.com/index.php>.
- [6] Fujaba. [Online] Available: <http://wwwcs.uni-paderborn.de/cs/fujaba/>.
- [7] S. P. Overmyer, B. Lavoie, and O. Rambow, "Conceptual modeling through linguistic analysis using LIDA", in *Proceedings of ICSE 2001*, Toronto, 2001.
- [8] E. Buchholz and A. Düsterhöft, "Using Natural Language for Database Design", in *Proceedings Deutsche Jahrestagung für Künstliche Intelligenz*, Saarbrücken, 1994.
- [9] H. Hamain and R. Gaizauskas, "CM-Builder: An Automated NL-based CASE Tool", in *Proceedings of the fifteenth IEEE International Conference on Automated Software Engineering (ASE'00)*, Grenoble, 2000.
- [10] L. Mich, "NL-OOPS: From Natural Natural Language to Object Oriented Requirements using the Natural Language Processing System LOLITA", *Journal of Natural Language Engineering* 2, no. 2, pp. 161-187, 1996.
- [11] NIBA Project, "Linguistically Based Requirements Engineering - The NIBA Project", in *Proceedings 4th Int. Conference NLDB'99 Applications of Natural Language to Information Systems*, Klagenfurt, 1999, pp. 177-182.
- [12] C. M. Zapata, A. Gelbukh, and F. Arango, "Pre-conceptual Schema: a UML Isomorphism for Automatically Obtaining UML Conceptual Schemas", *Research in Computing Science: Advances in Computer Science and Engineering*, no. 19, pp. 3-13, 2006.
- [13] Together. [Online] Available: <http://www.borland.com/us/products/together/index.html>.
- [14] WithClass. [Online] Available: <http://www.microgold.com/>

**Carlos Mario Zapata J.** Actualmente se desempeña como Profesor Asistente en la Escuela de Ingeniería de Sistemas de la Facultad de Minas de la Universidad Nacional de Colombia Sede Medellín. Es Ingeniero Civil, Especialista en Gerencia de Sistemas Informáticos, Magíster en Ingeniería de Sistemas y PhD. en Ingeniería-Sistemas; todos los títulos son de la Universidad Nacional de Colombia. Sus áreas de trabajo son: Ingeniería de Software, Ingeniería de Requisitos, Procesamiento de Lenguaje Natural, Lingüística Computacional y Estrategias Pedagógicas para la Enseñanza de la Ingeniería.

**Alexander Gelbukh** es Profesor-Investigador titular «C» del Laboratorio de Lenguaje Natural y Procesamiento de Texto del Centro de Investigación en Computo del Instituto Politécnico Nacional (México). Es Maestro en Ciencias (matemáticas; con distinción) de la Universidad Estatal Lomonósov de Moscú, Facultad de Mecánica y Matemáticas, Departamento de Matemáticas, y Doctor en Ciencias de Computación (Ciencias Tecnológicas) del Instituto de la Información Científica y Técnica de toda Rusia. Sus áreas de trabajo son: Inteligencia Artificial, Procesamiento del Lenguaje Natural y Lingüística Computacional.

**Fernando Arango I.** Trabaja como Profesor Asociado en la Escuela de Ingeniería de Sistemas de la Facultad de Minas de la Universidad Nacional de Colombia, sede Medellín. Es Ingeniero Civil de la Universidad Nacional de Colombia, Magíster en Planeación y Gestión de Recursos Hídricos de la Universidad de Colorado State (Estados Unidos de América) y Doctor en Informática de la Universidad Politécnica de Valencia (España). Sus áreas de trabajo son: Ingeniería de Software, Ingeniería de Requisitos, Métodos Formales y Lenguajes Declarativos.