Pre-conceptual Schema: a UML Isomorphism for Automatically Obtaining UML Conceptual Schemas

Carlos Mario Zapata Jaramillo,[1] Fernando Arango Isaza,[1] Alexander Gelbukh [2]

Universidad Nacional de Colombia,
Carrera 80 No. 65-223 Oficina M8-112 Medellin, Colombia
{cmzapata, farango}@unalmed.edu.co

Computing Research Center (CIC), National Polytechnic Institute,
Col. Zacatenco, 07738, DF, Mexico
www.Gelbukh.com

Abstract. Software development methodologies improve model quality. Conceptual schemas are representations of the universe of discourse for development purposes. UML had become a de-facto standard in software modeling. Obtaining UML diagrams from natural language descriptions is a very attractive goal. In this paper we present a proposal for improving some drawbacks from the previous work on this area. We call the proposed representation a preconceptual schema. It is an intermediate stage between natural language and UML conceptual Schemas. Finally, we show a case study for rules applying.

1 Introduction

Quality has been increasingly important in software development; from the point of view of Software Engineering, quality in software is the result of the application of a disciplined and methodological approach, covering all aspects of software life cycle [1].

Through a methodological approach, models have capital importance, because they permit specification understanding, communication among members of development team, future maintenance of the system, and reuse of code and specifications. In this methodological approach, UML had become a de-facto standard for software development [2].

Two trends in Software Engineering have become to gain importance for analysts:

- Firstly, CASE tools have improved capabilities of analysts to create UML diagrams. The main goal of CASE tools is support for drawing and editing diagrams, but responsibility for domain knowledge and obtaining of UML diagrams from natural language is left to analysts [3].
- Secondly, there is an increasing wave for the automated obtaining of UML diagrams. In this trend, responsibility for domain knowledge and obtaining of

C. M. Zapata Jaramillo, F. Arango Isaza, A. Gelbukh. Pre-conceptual Schema: a UML Isomorphism for Automatically Obtaining UML Conceptual Schemas. In: Advances in Computer Science and Engineering. J. *Research in Computing Science*, **19**:3–14, 2006.

UML diagrams from natural language is left to computers, and theoretical studies have been made for rules definition to achieve this goal.

There are several works on the second trend [4–9], but certain problems are still open. Generally speaking, many proposals in this trend tends to work on only a kind of diagram (Entity-Relationship Model, Class diagram, and so on), but in the presence of two or more diagrams, consistency problems are common.

In this paper, adopting the second trend, we propose an approach for automated UML diagrams obtaining through the use of a new graph named "Preconceptual Schema" and a set of translation rules.

This paper has been organized as follows: in section 2 we survey some works on automated UML obtaining; then, in section 3 we describe Preconceptual Schema as a new approach to this issue. Section 4 is devoted to the definition of rules for automatic transformation from Preconceptual Schemas and three kinds of UML diagrams. A case study in Spanish language is developed in section 5, and in sections 6 and 7 we discuss some conclusions and future works, respectively.

2 Automated UML obtaining: a survey

Many researchers in the world are trying to obtain UML conceptual schemas in an automated way from natural language. This trend has been a dream for modelers since the early attempts of Peter Chen, the father of Entity – Relationship Model (ERM) [10], who defined a set of rules to obtain ERM from an English discourse [11]. They were simple rules, but it was possible to find many counter-examples for each of them; for this reason, Chen named them "suggestions" more than "rules". In the same way of thinking, Coad and Yourdon defined another set of rules for Class Diagram, in the early years of object-orientation [12]. Neither Chen nor Coad and Yourdon had the intention to automate their rules, but they generated the basis for a new research in modeling.

Progress in Software Engineering has increased development and use of many new tools, named Computer-Aided Software Engineering (CASE) Tools [3]; for modelers, these tools have become electronic assistants for model drawing. However, CASE technology has been founded under the assumption that modelers have to interpret the domain of discourse and they may convert natural language specifications into the required diagrams, and then they can use CASE tools for drawing these diagrams. Automated assistance begins in this moment, but there's no help in previous stages of the process.

A semi-automated approach has been developed by LInguistic assistant for Domain Analysis (LIDA) Project [4]. In LIDA, a classification for kinds of words is made along a discourse in natural language; LIDA identifies nouns, verbs, and adjectives, and it calculates frequencies of word's appearance in the text. With this information

at hand, modeler must decide if the word will be mapped to a class, an attribute, an operation or a relationship in the class diagram. Mapping process is, therefore, owned by the modeler, with little assistance of the LIDA tool.

Rapid Application and Database Development (RADD) Project [5] was designed to obtain ERM from natural language in an automated way, and initiating a "moderated" dialogue to enhance completeness of the diagram. However, RADD was designed for ERM, and its creators didn't define mapping for another kinds of conceptual schemas, e.g. UML diagrams.

A different approach was defined by Cyre in Automatic Specification Interpreter (ASPIN) Project [6]. In ASPIN, modeler can create multiple diagrams (e.g. timing, State-Transition, Blocks, etc.) for describing a control system specification, and then ASPIN can create a consolidated representation of the system, based on those diagrams. Although ASPIN doesn't use natural language (it only accepts a restricted form of language, specific to the domain of control systems), its approach is useful for demonstrating the possibility of joining diagrams together in a single representation (it uses Conceptual Graphs for this purpose). However, ASPIN only works with control systems domain, and lacks generality for working with another paradigm (such as UML, for example).

CM-BUILDER project [8] was developed for automated UML class diagram acquisition. The process begins from natural language specifications, but it requires a previous knowledge about domain of the problem. This knowledge must be represented through semantic nets, with almost every category of the class diagram in them. Semantic nets, like these, are very complex to acquire, and they don't guarantee mapping process if a word doesn't match a category in them.

NL-OOPS (Natural Language Object – Oriented Product System) Project [7] uses a semantic net for mapping process too. However, rules used by the process lack of generality, and identified elements must belong to several categories simultaneously. Therefore, mapping process requires active participation of the analyst, who must decide the final category for every element.

Both CM-BUILDER and NL-OOPS were developed for UML class diagram acquisition and they don't identify elements for another UML diagrams. On the opposite, NIBA Project [9] was developed for multiple UML diagrams acquisition (mainly class and activity diagrams). It uses the so called KCPM (Klagenfurt Conceptual Predesign Model), a model with various kinds of elements to achieve mapping process, from tables to certain dynamic diagrams. KCPM is not unified, and its use depends on the kind of target diagram; as a consequence, it's difficult to guarantee consistency between diagrams, because every element for every diagram comes from different forms of KCPM.

Work has been done in this area, but problems still remain; particularly, problems are unsolved for consistency reasons, standardization (UML is a standard, but many works try to obtain some other formalism), and connectivity between formalisms. In

the next section, we define a new proposal for an intermediate stage between natural language and UML conceptual Schema: Preconceptual Schema.

3 A new approach: Preconceptual Schema

3.1 Justification

Some of the works listed in the previous section uses an intermediate formalism, in order to facilitate mapping process. Semantic nets, tables, dynamic graphs and conceptual graphs are some of the mechanisms used by these projects for representing natural language discourse.

An intermediate formalism acts as a facilitator of the mapping process between natural languages and UML diagrams. It's needed because natural language representation lacks of certain elements and relationships required for mapping process.

3.2 Definition

In this paper we use a new kind of intermediate formalism, and we call it "Preconceptual Schema". The term "preconceptual" was coined by Heidegger [13] and it refers to previous information, acquired in some way, of a concept. In the knowledge stages, Piaget [14] identifies a stage, later to linguistic knowledge, but previous to conceptual knowledge, and he called it "preconceptual stage".

In some way, we are trying to gather some useful information for mapping process from natural language to UML diagrams. To achieve this goal, we need an intermediate stage, founded in the linguistic information, but with certain knowledge of the later phase of the process: the conceptual one. As a consequence, our approach needs a new schema, an intermediate schema for facilitating mapping process between natural language specifications and UML diagrams. We called it "Preconceptual Schema" PS because of the above reasons. Our main goal is the definition of PS syntax and to prove UML diagrams can be contained in PSs.

PS must accomplish three asserts:

- PS must be obtainable from natural language (demonstration is out-of-scope of this paper).
- PS must be isomorphic with UML diagrams (through the set of rules defined in section 4).
- PS must be rewritable in a disambiguated form of language, a simple discourse with little or no ambiguities.

Acting as an isomorphism, PS must represent generalities of UML diagrams, as an integrated view of the same model. For this reason, consistency problems must disappear, because starting point for every diagram drawing is the same.

3.3 Notation

In ASPIN [6], they used Conceptual Graphs CG as an intermediate formalism for representation of many diagrams simultaneously. The reasons for this election are representativeness and versatility. But, thinking about our goals, Conceptual Graphs poses some drawbacks for us:

- Representation of a concept, which is included in various phrases simultaneously, is complicated. For this goal, CGs use a dotted line called "coreference"; however, this mechanism is more intricate as far as the same concept appears over and over again.
- CGs are preeminently structural graphs. For the sake of variety, we need a schema capable of representing both structural and dynamic properties, and CGs only can represent structural features.
- Usage of a symbol for many kinds of representations produces ambiguity. In CGs concepts are used for both nouns, verbs and adjectives, and relationships are used for both semantic cases and other semantic relations.

Preconceptual Schemas are founded on CG, but we have made some changes for supporting main features of the mapping process. The main symbols in Preconceptual Schemas are Concepts (rectangles) and relationships (ovals), but differing from CGs, in concepts we can only put nouns, and in relationships we can only put verbs. If a concept is repeated too many times in a discourse, in PSs will appear only once; furthermore, all the relationships to this word in the discourse will be represented as relationships with this only one concept. In this way, PSs will be integrated graphs and they will not be like many CGs with correferents.

For dynamic purposes, in PSs there are two new elements:

- Thick arrows express implication. These elements only can be connected from one relationship to another, and it means the target verb it's only performed if the source verb is performed (as a precondition in if-then phrases).
- Rhombs express conditionals. In the inner space of the rhomb, we can put expressions with concepts and operators; these expressions must be true or false.

Similarly to CGs, in PSs thin arrows represents directed connections. In PSs, They must be:

- Concept-relationship connection: it means one concept achieves an activity expressed by the relationship.
- Relationship-concept connection: it means one concept receives an activity expressed by the relationship. In this case, we can include a preposition in the arrow (if it's needed)
- Conditional-relationship connection: it means one activity (expressed by the relationship) will be executed if the answer for the conditional matches the word included in the arrow (we must include that word).

In directed connections, we can include a number for distinguishing actions before and after an implication occurs.

In Figure 1 we can see the main symbols of PSs.

Fig. 1. Main symbols of Preconceptual Schema

3.4 Additional Considerations

In order to define and use properly PSs, we must warn you about two special considerations:

- We have only defined the main symbols used in PSs. Users of PSs must decide the best way to express their needs in terms of them.
- Concepts admit the use of compound nouns. Again, user must decide usage.

In the next section, we define a set of rules for automated transformation between PSs and three kinds of UML 2.0 diagrams: class, communication (in previous versions of UML, it was called "collaboration" diagram) and state machine diagrams.

4 Rules for automated transformation from Preconceptual schema to UML 2.0 diagrams

In the previous section, we defined main features of PSs as an intermediate formalism to perform transformation between natural language specification and UML diagrams. Rules for obtaining PSs from natural language are out-of-scope of this paper. Instead of, we must present rules for transformation from PSs to three UML diagrams, for Spanish language (some rules can be different in English language).

4.1 Rules for Class Diagram

4.1.1. A source concept from a “tiene” relationship is a candidate class.
4.1.2. A target concept from a “tiene” relationship is a candidate attribute.
4.1.3. Both the source and target concepts from an “es” relationship are candidate classes (an exception is made if one or both concepts are adjectives or proper nouns). The relationship itself is a candidate inheritance with source concept as a candidate daughter class, and target concept as candidate parent class.
4.1.4. A concept defined as a candidate class by one or more rules, and as a candidate attribute by another set of rules, is a class.
4.1.5. Relationships corresponding to either activity verbs or realization verbs are candidate operations of target concepts (if these concepts are defined as candidate attributes by one or more rules, the operations are assigned to their owner candidate classes).
4.1.6. A candidate operation between two classes generates a candidate association between these classes.
4.1.7. A “tiene” relationship between two concepts, which are identified as candidate classes by one or more rules, generates a candidate aggregation relationship, with source concept as the whole and target concept as the part.
4.1.8. Concepts identified as object classes in communication diagram are candidate classes in class diagram.
4.1.9. Relationships identified as messages between objects in communication diagram are candidate operations of target object class in class diagram.

4.2 Rules for Communication Diagram

4.2.1. The source set of concepts and relationships from an implication connection is a candidate guard conditions.
4.2.2. Expressions included in conditionals are candidate guard conditions.
4.2.3. Target relationships after either an implication or a conditional are messages. The source concept will be source object class and target concept will be target object class. There must be series of messages jointed by objects.

4.3 Rules for State Machine Diagram

4.3.1. Past participle messages identified in communication diagram are candidate states for target object class.
4.3.2. Sequence between states in State Machine Diagrams depends on identified and numbered sequences in communication diagrams.

5 A case Study in Spanish language for rules applying

The rules described in section 4 are for Spanish language, because our Research Group is trying to obtain conceptual diagrams from specifications in this language.

The case study relates to a pizzeria and its production and delivery processes. In Figure 2 we can see Preconceptual Schema from this domain, and in Table 1 we summarize the application of some rules. The application of the rules for this case study was hand-made for academic purposes; as a future work, we are planning to build a CASE tool with the automation of this method.

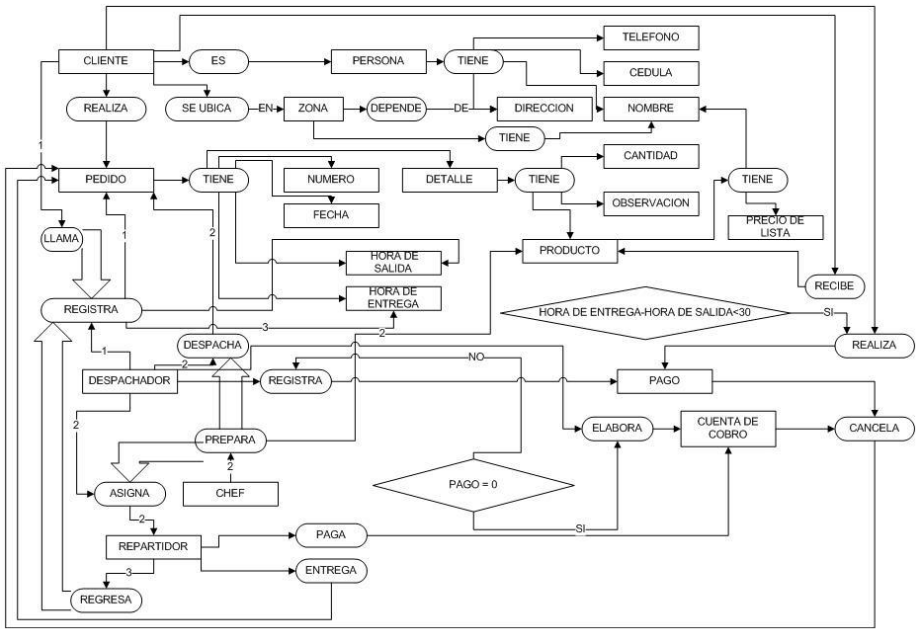

Fig. 2. Preconceptual Schema of production and delivery process of a pizzeria.

Some pieces of this diagram must be rewritten as follows:

- Clients are persons.
- Details have a quantity, an observation and a product.
- If the difference between delivery hour and exit hour is lower than 30 minutes, then dispatcher registers payment, else dispatcher makes a devolution report.
- Whenever client calls, dispatcher registers order.

Note how this way of textual representation of PSs must express, in a single way, the Universe of Discourse associated with a model of the world.

Table 1. Rules application for some elements of PS.

PS Element	UML Diagram	UML Element	UML associated element	Applied Rule
Client	Class	Candidate Class	Person	4.1.1., 4.1.3.
Order	Class	Candidate Class		4.1.1.

Product	Class	Candidate Class		4.1.2., 4.1.4.
Address	Class	Candidate Attribute	Person	4.1.2.
Register	Class	Candidate Operation	Order	4.1.5., 4.1.9.
Chef prepares product	Communication	Guard Condition	Assign	4.2.1.
Deliver	Communication	Candidate Message	Deliverer, Order	4.2.3.
Delivered	State Machine	Candidate State	Order	4.3.1.

From Table 1, we can see how product is initially defined as a candidate attribute by 4.1.2. rule and then redefined as a candidate class by 4.1.4. rule. Furthermore, in UML associated element appears some elements needed for the element definition; e. g. "Client" needs "Person" for its inheritance, and "deliver" needs source object class ("deliverer") and target object class ("order").

With the hand-made application of rules described in section 4, we must obtain diagrams showed in Figures 3, 4 and 5.

6 Conclusions

In this paper we have presented Preconceptual Schemes, an intermediate stage between natural language specifications and UML diagrams. Furthermore, we have developed a set of rules for automated obtaining of three kinds of UML diagrams: class, state machine, and communication diagrams. We have assumed Preconceptual Scheme as being obtainable from natural language (demonstration is out-of-scope of this paper), and we have concentrated in the next transformation stage.

With the case study, we showed it's possible to obtain those UML diagrams (class, state machine, and communication diagrams) with some limitations, but with the main features of the diagrams. Preconceptual Schemas are even very simple, and they represent restricted natural language specifications; however, they lack of mechanisms for representing words like adverbs and adjectives. Even with limitations, they can describe a domain of discourse in a complete and understandable way.

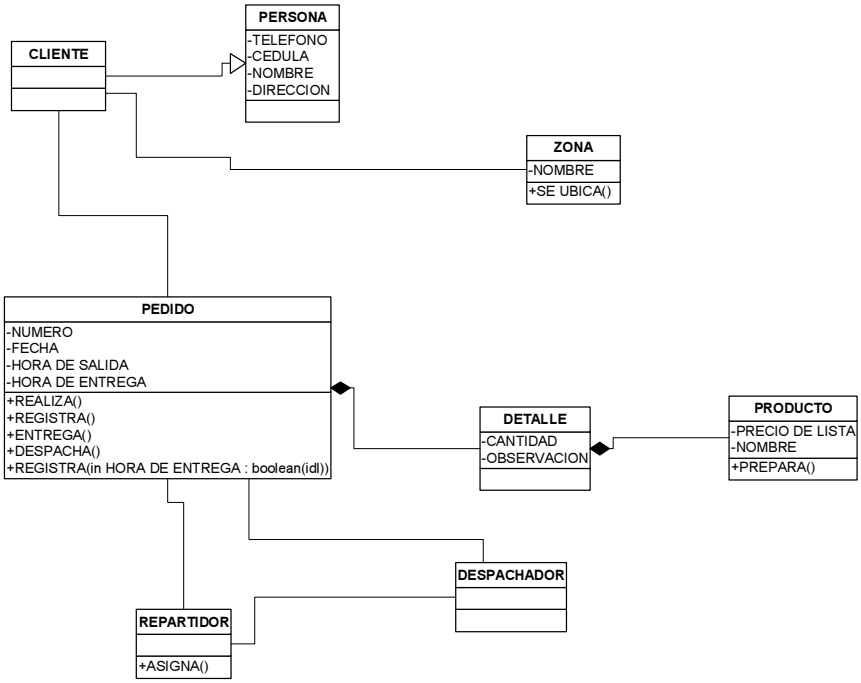

Fig. 3. Resultant Class Diagram from PS in the Figure 2.

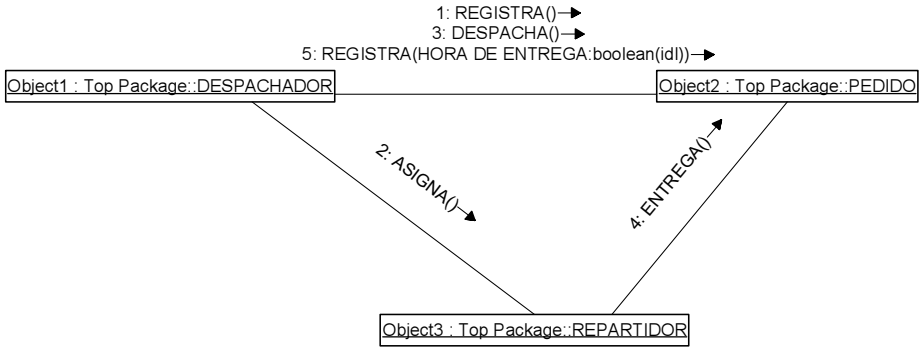

Fig. 4. Communication Diagram from PS in the Figure 2.

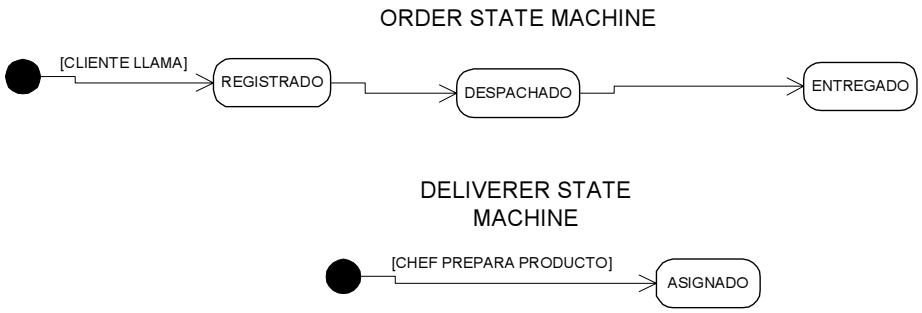

Fig. 5. Resultant State Machine Diagrams from PS in the Figure 2.

7 Future Work

- As a continuation of this work, it will be needed the definition of additional sets of rules for automated obtaining of UML diagrams (e.g. timing or activity diagrams), and additional rules for the diagrams here described (e.g. rules for multiplicity of associations in Class diagram, rules for "on exit" actions in state machine diagram, and so on). Furthermore, we must develop a prototype to prove application of this work in the second trend described by section 1 (introduction).

- We must generate rules for Preconceptual Scheme obtaining from natural language.

- Furthermore, we must generate new mechanisms for representation in Preconceptual Schemas (e.g. adverbs and adjectives) for extending their functionality and scope.

References

1. Pressman, R.: Software Engineering: A Practitioners' Approach. 5th edn, McGraw-Hill, Inc, New York (2001)
2. OMG: UML Specification. Available: http://www.omg.org/uml
3. Burkhard, D. and Jenster, P.: Applications of Computer-Aided Software Engineering Tools: Survey of Current and Prospective Users. Data Base Vol. 20, No. 3 (1989) 28-37
4. Overmyer, S.P., Lavoie, B., y Rambow, O.: Conceptual modeling through linguistic analysis using LIDA. In: Proceedings of ICSE 2001, Toronto, Canada. (2001)
5. Buchholz, E. y Düsterhöft, A.: Using Natural Language for Database Design. In: Proceedings Deutsche Jahrestagung für Künstliche Intelligenz. (1994)
6. Cyre, W.: A requirements sublanguage for automated analysis. International Journal of Intelligent Systems, Vol. 10, No. 7. (1995) 665-689.

7. Mich L.: NL-OOPS: From Natural Natural Language to Object Oriented Requirements using the Natural Language Processing System LOLITA. Journal of Natural Language Engineering, Cambridge University Press, Vol. 2, No. 2. (1996) 161-187.
8. Harmain, H. y Gaizauskas, R. : CM-Builder: An Automated NL-based CASE Tool. In: Proceedings of the fifteenth IEEE International Conference on Automated Software Engineering (ASE'00), Grenoble. (2000)
9. 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) 177 – 182.
10. Chen, P. P.: The Entity–Relationship Model: Toward a Unified View of Data. ACM Transactions on DataBase Systems, Vol. 1, No. 1. (1976)
11. Chen, P. P.: English Sentence Structure and Entity–Relationship Diagrams. Information Science, No. 29, Vol. 2. (1983) 127-149.
12. Coad, P. y Yourdon, E.: Object – Oriented Analysis. New Jersey: Yourdon Press. (1990)
13. Heidegger. M.: Protokoll zu einem Seminar über den Vortrag "Zeit und Sein". En: Zur Sache des Denkens, Tübingen (1976) 34.
14. Piaget, J.: The origins of intelligence in children (2nd ed.). New York: International Universities Press (1952)