GOOAL AUTOMATIC DESIGN TOOL

A Role Posets based Tool to Produce Object Models from Problem Descriptions

Hector G. Perez-Gonzalez, Sandra Nava-Muñoz, Alberto Nuñez-Varela
Facultad de Ingenieria,Universidad Autonoma San Luis Potosi, Dr. Manuel Nava # 8, San Luis Potosi, Mexico

Jugal Kalita
University of colorado at Colorado Springs, 1420 Auston Bluffs Pkwy, Colorado Springs, CO, U.S.A.

Keywords: Object Oriented Analysis, Object Oriented Design, Software Engineering Education, Natural language Processing.

Abstract: A number of software analysts may produce different, perhaps all of them correct, solutions from one specific software requirement document. This is because natural language understanding is complex and because each analyst has distinct design experience. A methodology and approach that can be automated and that uses a proposed semi-natural language called 4WL used to accelerate the production of reliable accords between different stakeholders. The supporting software tool called GOOAL, Graphic Object Oriented Analysis Laboratory automatically produces simple object models (UML diagrams) from English or Spanish statements with minimal user participation. These statements, faithfully describe the original problem description sentences. The models are generated analyzing each sentence of the intermediate 4W language version of the original sentence set. With this methodology and supporting software tool, students of Object Oriented technology can visualize the design decisions being made by the system. This methodology and software tool has been used to support the learning process in object Oriented analysis and design courses. The original tool was developed to "understand" English and it was validated with design artefacts produced by several experts of the University of Colorado. The main results reported by the students, are related with the use of good design practices, a better understanding of UML language and a major interest in the pre programming process. Its technical contribution is the role posets technique.

1 INTRODUCTION

Although Natural Language (NL) processing is a very complex task (Allen 1995; McDonald 1992), it is possible to extract sufficient meaning from NL sentences to produce reliable models. This research proposes the use of a tool-supported methodology that helps in object oriented design (OOD) from English or Spanish problem sentences.

Our goal is to take a problem description given from the problem domain experts such as the one given below and produce the UML (Booch 1997) models for it. The following sentences describe a problem known as the dining philosophers and the second describe the towers of Hanoi problem both as shown in Rumbaugh's book (Rumbaugh et al. 1996).

Problem # 1: English Version

There are five philosophers and five forks around a circular table.
Each philosopher can take two forks on either side of him.
Each fork may be either on the table or used by one philosopher.
A philosopher must take two forks to eat.

Problem # 2: English Version

The towers of Hanoi is a game. There is a player that moves a stack of disks from one of three long pegs.

The player uses the third peg for manoeuvring.

Each disk has a different size.

Disk may be moved from the top of a stack on a peg to the top of the stack on other peg.

A disk is placed on another disk that is bigger than itself.

The problem description sentences in the examples have four and six English sentences respectively. We use the first as a running example in this paper. The verb forms used in the first example are *are*, *can take, may be, used, must take* and *eat*. The nouns are *philosopher(s), table, fork(s)* and *side*. The classes corresponded to the first example that could be accepted for modellers to construct their class diagram are: *Application* that works as main class, *Philosopher* and *Fork*. Each class abstracts the knowledge and behaviour of its corresponding real counterpart. Our challenge in the research reported in this paper is to obtain descriptive UML diagrams from NL general sentences using a systematic methodology and minimal user participation

2 RELATED WORK

An early paper by Abbot (1983) presented a systematic procedure to produce design models from NL sentences. This approach produces static or structural models obtained with high user participation. Saeki, Horai and Enemoto (Saeki et al. 1989) presented a process to derive incrementally a formal specification from natural language. They used simple “verb patterns” to identify linguistic subjects and objects. The user interactively decided which nouns became classes and which verbs became methods.

Cockburn (1992) presented a detailed analysis where he related relational nouns with objects, adverbs with polymorphism and the roles of objects. In a high level abstraction, Obsborne and MacNish (1996) eliminated ambiguity in NL requirements by the use of a Controlled language (CL) called Newspeak. Da Silva (1996) suggested an object oriented model and presented a formal diagrammatic notation (Object-Z) and a set of rules to transform semiformal sentences into formal. Burg and Van de Riet (1996) tried to minimize the participation of the user in the job of class and relationships extraction from the text. They used a very large lexicon to aid in semantic model validation and provide a new modelling language to illustrate results.

Hars and Marchewka (1997) presented an effort to produce dynamic analysis using a dictionary of about 23,000 single root words. Each word belonged to a concept category such as event, person, and location. The dictionary also contained word frequency information computed from a year’s volume of Time magazine. The process transformed the textual sentences into an internal representation, and constructed a tree structure. A separate algorithm resolved compound words such as chicken nuggets. This technique asked the user to resolve ambiguities not just at syntactic level but at semantic level also. Boyd (1999) discussed the linguistic metaphors in software design and analyzed prepositions, articles and interjections in addition to nouns and verbs. He used a process of syntactic normalization to produce a more precise sentence through two manual steps: syntax normalization and semantic exploration.

Borstler, Cordes and Carver (Borstler el al. 1992) presented a prototype that accepted well formed NL textual use cases and produces, with minimal user participation, UML static diagrams: classes, objects and simple relationships. A valuable feature of their work is the final traceability supported by hypertext technology. Overmyer, Lavoie and Rambow (Overmyer et al. 2001) presented a complete interactive methodology and a prototype tool that produced a subset of UML. However, the text analysis remained in good part a manual process.

We presented our seminal work at OOPSLA 2002 (Pérez-González & Kalita 2002) showing our initial results with just two simple examples and using just an English version tool. At OOPSLA 2005 (Pérez-González et al. 2005) we showed our Spanish version tool with an educational approach. Polajnar (Polajnar et al. 2006) presented CLIE (controlled language for information extraction) based in predicate logic. Zapata (Zapata et al. 2006) presented UN-Lencep a controlled language using the notion of pre-conceptual schema (Zapata et al. 2006b). They proposed a simple framework used by their automatic tool to produce classes, communication and state machine diagrams.

Our methodology accelerates the early software development process through the use of a software tool. The supporting software tool called Graphic Object Oriented Analysis Laboratory produces an intermediate representation of the original Natural Language sentences describing customers’ problem description sentences. It then analyzes them and proposes object oriented (OO) models (Booch 1997) thorough an iterative process. The value of our methodology consist of the use of a systematic process, and the value of our tool and its underlying techniques is the production of diagrams non just from a controlled language but from a free version of the problem sentences. That can be achieved because of the iterative participation of stakeholders even thinking that those interventions are not intensive. Particular difference of our techniques is the prediction of the probabilities every element has to become a class depending on the discourse.

3 METHODOLOGY

In order to build successful software products in a professional Software Engineering environment we have to define what success is. A software development project or product is considered successful if it meets the solicitor's requirements respecting the original schedule and budget.

The single biggest cause of software project failure is deficient requirements definition (Hofman & Lener 2001); specifically, inconsistencies and misunderstandings in the early stage of the process produce unpredictable consequences. The other important success factor is software architecture and design decisions. Good design decisions lead to a more maintainable product. Improving requirement definitions and software design practices will allow high-quality Software Engineering to become more to a reality. This section presents a methodology to maximize software reliability through automatic modelling techniques from natural language problem statements. This tool-supported methodology aims to facilitate stakeholders' communication by reducing early misunderstandings and promoting good design practices using the supporting tool for software design practice.

The analysis/design step generally takes a problem statement written in English, follows an object oriented methodology and produces a set of UML diagrams representing the proposed solution.

A number n of software (SW) analysts may produce n different (perhaps all of them correct) solutions from one specific SW requirement document. This happens because Natural Language (NL) understanding involves syntactic, semantic and pragmatic issues, (different backgrounds in persons produce different interpretations) and because of distinct design experiences.

We are proposing a methodology with a technique called role posets (Pérez-González & Kalita 2002) that will enable the early modelling of SW to be automated and a semi-natural Language called 4WL (Pérez-González & Kalita 2002) that are used as the main vehicles of the methodology to accelerate the production of reliable accords between different stakeholders. Figure 1 shows this process.

The supporting software tool (GOOAL: Graphic Object Oriented Analysis Laboratory) automatically produces object models from a natural language (English or Spanish) statements describing software requirements. Those models are generated by analyzing sentence-by-sentence an intermediate language (4W) version of the original sentence set.

3.1 4W Language

A 4WL sentence is composed of four general parts (Pérez-González & Kalita 2002): (W parts). They try to respond the questions: Who is the semantic subject?, What is happening?, Who is the receiver of the action? and Who is the indirect semantic object?. The subject is mandatory in every 4WL sentence. The last two parts are optional. Every 4WL sentence has an object that corresponds to the action referred to in the sentence. In linguistic terms, this is the noun of the subject. The presence of this part is mandatory in every English sentence we can process, and in a 4WL sentence also. From the OO point of view, this object cold be an instance of some class. When we use the term noun in this paper, in addition to a single noun, it could be a group of nouns (chicken nuggets) or a sequence of adjectives and a final noun (small blue birds).

EXAMPLE: DINNING PHILOSOPHERS

The five dinning philosophers' problem (Hofman & Lener 2001) is used here to illustrate the use of 4WL language. The English statements of the problem are given in the beginning of the paper and are repeated below. English sentences are automatically translated to 4WL. The English sentences in the specification are shown first in bold. The 4WL sentence(s) that correspond(s) to a single English sentence follow(s) in italics. Note that one English sentence can result in one or more 4WL sentences.

There are Five Philosophers and Five Forks around a Circular Table.

1.-Five forks are around a circular table.
2.-Five philosophers are around a circular table.

Each Philosopher can take Two Forks on either Side of Him.

3.-Each philosopher can take two forks on the side of each philosopher.
4.-A philosopher has side.

Each Fork may be either on the Table or used by One Philosopher.

5.-Each fork may be on the table.
6.-One philosopher may use each fork.

A Philosopher must take Two Forks to Eat.

7.-A philosopher must take two forks.
8.-<When preceding sentence> a philosopher eats.

The Software Tool called GOOAL developed to support this work automatically translates from

English to 4WL with little user participation The 4W grammar is not detailed here.

4 ROLE POSETS

We use the notion of partially ordered set of roles (Role Posets) (Pérez-González & Kalita 2002) to emulate the reasoning analysts perform when they model a problem. Software models are the product of a sequence of decisions taken. Different modellers can produce different models for a given problem depending on the decisions they make. The decisions can be influenced by different factors such as the design priorities and previous experience of the modeller, the nature of the problem itself and the priorities and expectations of the consumer. Role Posets are based on the mathematical concept of poset (Partially ordered set) and the thematic (theta) roles due to Chomsky (Chomsky 1965) and are used widely in many linguistics formalisms. We use Role Posets to construct the structures to model a problem. The decision that makes a noun becomes a class depends on the analysis of the complete problem, the potential future additions to the system and the perception, experience and motivation of the analyst. Some automatic methodologies (Borstler el al. 1992, see also Mich & Garigliano 1997) make class identification using the frequencies of all the nouns in the requirements text. We propose a heuristic algorithm to identify classes considering also the roles that the nouns play in every sentence:

The probability that a noun becomes a class varies proportionally to its importance or value in a text:

$$\text{probToClass(NounX)} = \text{value}_t\,\text{(NounX)} \qquad (1)$$

The importance of a noun in a text is computed as the sum of the importance of that noun in every sentence. We write it as:

$$\text{value}_t\,\text{(NounX)} = \sum\nolimits^{i=m}_{i=1} \text{value}_i\,\text{(NounX)} \qquad (2)$$

Where *m* is the number of sentences in the text and i is the index of the particular sentence.

The importance of a noun in a sentence S depends on the role it plays in this sentence. We write it as:

$$\text{value}_i\,\text{(NounX)} = \text{role}_i\,\text{(NounX)} \qquad (3)$$

Where i is the index of the particular sentence.

According to Chomsky (Chomsky 1965): “The role a noun plays in a sentence depends on the relative position it has in the sentence and on the semantics of the main verb of that sentence”. We express Chomsky’s statement as:

$$\text{role}_i\text{(NounX)}=\text{pos(NounX)}_i + \text{semantics (V)}_I \qquad (4)$$

Where i is the index of the particular sentence.

According to Haegeman (1991) “There is no agreement about how many such specific thematic roles (Θ Theory) there are and what their labels are. Some types are quite generally distinguished”. The selection of the roles is frequently a semantic intuition that is difficult to automate. Some types are quite generally distinguished”. According to Chomsky’s Government and Binding theory there are restrictions called the Theta Criterion on the Θ roles: “Each Argument is assigned one and only one theta role. Each theta role is assigned to one and only one argument”. Of course, we follow Chomsky’s ideas as just guidance in formulating our algorithms. Details of calculating the significance of roles for a specific piece of requirements text are not discussed here. Once the potential classes have been analyzed, their calculated probabilities are used to decide which ones deserve the class category. All of this are based on our proposed role machines and verb families (Pérez-González et al. 2005) explained below. We propose a universal partially ordered set of roles composed by: *Agent (Ag), User (Ur), Modified (Mr), Used (Ud), Whole (Wh), Part (Pt), General (Gl), Special (Sp), Theme, LSP (Location, situation or position)* and *attribute (At).*

Our prototype tool internally labels every noun in the text with the particular role it plays according to its associated verb and its relation with it.

At the end of this automated analysis, there is a list of nouns (and adjectivally qualified nouns), every one of them associated with a list of roles it plays.

4.1 Results and Evaluation

In our example we have the following nouns: **Na=Philosopher, Nb=Fork, Nc=Table, Nd=Side.** Table 1 shows results of the automatic generation of roles.

Table 1: Roles results for the five Philosophers problem.

Na = {User, User, Whole, Theme}
Nb = {Used, Used, Theme, Theme}
Nc = {LSP, LSP}
Nd = {Attribute, LSP}

From this, the universal role poset is reconstructed: *{Agent ( ), User (N1,N1), Modified ( ), Used (N2,N2), Whole (N1), Part (), General ( ), Special ( ), Theme (N1,N2,N2,), LSP (N3,N3,N4), Attribute (N4) }.*

The reason because N1, for example is associated with that particular list of roles, involves the use of a proposed state machine we called Role Machine.

We have designed a role machine for every one of a group of verb families.

We have identified 13 semantic families of verbs. Figure 1 shows the role machine corresponding to the *be verb* family with the slots being occupied by the words of our first 4W sentence: The machine shows the Noun *fork (Nb)* playing the role of Theme (Th) two times because the state the sentence follows denotes the state Th- meaning that the noun in the left side of the verb is a theme The verb *take* used in the second sentence belongs to the di-transitive modifier family of verbs.

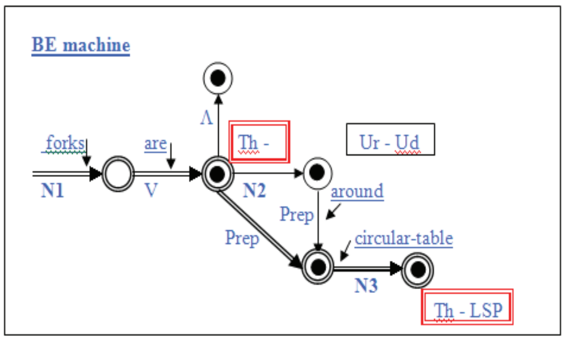


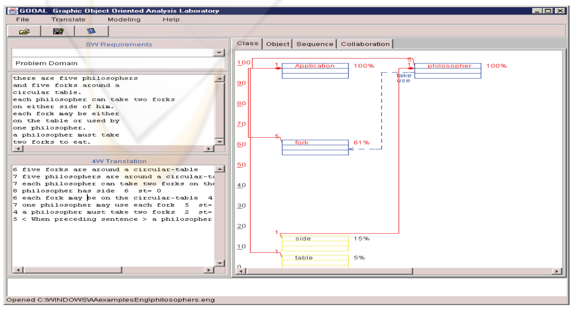
Figure 1: Role Machine for the *be verb* family using the sentence: *Five forks are around a circular table.*

We don't present more details of this technique due to lack space. After following the technique, we obtain the probabilities every noun (N) has to become a Class. Results of our example are shown in table 2.

Table 2: Result of analysis of nouns. Five Philosophers problem.

Noun	Noun´s name	Probability to be a class
Na	Philosopher	100%
Nb	Fork	61%
Nc	Table	15%
Nd	Side	5%

Figure 2 shows a simple class diagram obtained from our example. The vertical axis shows the probability a noun element has to become a class.



```
there are five philosophers
and five forks around a
circular table.
each philosopher can take two forks
on either side of him.
each fork may be either
on the table or used by
one philosopher.
a philosopher must take
two forks to eat.
```

```
6 five forks are around a circular-table
7 five philosophers are around a circular-ta
7 each philosopher can take two forks on the
8 philosopher has side  6  st= 0
6 each fork may be on the circular-table  4
7 one philosopher may use each fork  5  st=
4 a philosopher must take two forks  2  st=
5 < When preceding sentence > a philosopher
```

Opened C:\WINDOWS\AAexamplesEng\philosophers.eng

Figure 2: Final class diagram. Five Philosophers problem.

The tool can produce general static and dynamic UML diagrams. These diagrams and the 4W sentences may be used to detect and solve ambiguity.Class, These diagrams are based on the analysis of every one of the 4WL sentences.

5 EVALUATION

In order to prove the results obtained by the tool-supported methodology presented in this paper, we tested 82 undergraduate software engineering students. Each student was asked to evaluate both problems presented in this paper (philosophers and Towers of Hanoi). The students followed our methodology to identify the nouns in the text and assign them a probability that the noun or noun construction will become a class according to his level of knowledge and experience. Table 3 shows the test results for the Philosophers problem.

Table 3: Philosphers test results.

	Students	GOOAL
Philosopher	92,10%	100,00%
Fork	75,52%	64,00%
Side	23,42%	15,00%
Table	35,28%	2,00%

Table 4 shows the test results for the Hanoi problem.

Table 4: Towers of Hanoi test results.

	Students	GOOAL
Player	83,91%	100,00%
Disk	70,65%	99,00%
Stack	70,21%	67,00%
Top	28,52%	24,00%
Collection	25,00%	10,00%
Size	17,78%	10,00%

Note: the students' column is the students' average.

As we can see in the tables, all decisions taken by the systems are closely related with the ones taken by the students following the methodology The results are very promising and we can say that the tool will help in the use of good design practices and a better understanding of UML language.

6 CONCLUSIONS AND FURTHER WORK

The present paper has described a tool that supports a methodology that aims to accelerate the production of analysis and early design models. This methodology is based on semantic abstraction of linguistic elements and involves the use of the semantic role theory and a semi-natural language. The prototype tools have produced good results with problems described in no more than eight sentences and 100 words on an average. Observed advantages of the use of this tool are formalization, standard notation, validation, traceability, efficiency and early identification of misunderstood requirements. Individuals using it, will see how a group of sentences describing a problem are handled by GOOAL. The system takes decisions with minimal user participation, shows its interpretation in 4WL and produces model views of the problem. Unique features of this tool are the underlying methodology and the production of dynamic models. Although the tool, methodology and techniques expounded here are far from perfection, results using simple sentences are promising. Better results are expected if a more complete general dictionary is used as well as finer refinements in semantic classification of verbs into families. With the comparison of results between both GOOAL tool and the tests we are running with the students, it can be conclude that we can use this program as an educative tool.

REFERENCES

Abbott, R.,1983. Program Design by informal English Descriptions. In *Communications of the ACM*, 26(11).

Allen, J.,1995. *Natural Language Understanding*, Benjamin/Cummins Publishing Co.

Booch, G.,1997. *The unified Modeling Language User guide.* Addison-Wesley.

Borstler, Jurgen, Cordes, Carver. 1992. An Object-Based Requirements Modeling Method. *Journal of the American Society for Inf. Science* 43(1):62-71.

Boyd, N., 1999. Using Natural Language in Software Development. *Journal Of Object Oriented Progr.*

Burg, J., Van De Riet, R., 1996. Analyzing Informal Requirements Specifications: A first Step towards conceptual modeling. *Proceedings of the 2th International workshop on applications of natural language to information systems,* Amsterdam, The Netherlands, IOS Press.

Chomsky, N., 1965. *Aspect of the theory of syntax*. MIT Pr

Cockburn, A., 1992. Using Natural Language as a metaphorical Basis for Object Oriented Modeling and Programming. In *IBM Technical Report TR-36.0002.*

Da Silva, J.,1996. *Metamorphosis: An Integrated Object Oriented Requirements Analysis and Specification*. Lancaster University.

Haegeman, L., 1991. *Introduction to government and binding theory*. Wiley.

Hars, A., Marchewka, J.,1997. The Application of Natural Language Proc. requirements Analysis. *Journal of Management Information Systems*.

Hofmann, H., Lener, F.,2001. Requirements Engineering as a Success Factor in Software Projects. *IEEE Software*, (pp 58)

McDonald, D.,1992. Robust Partial-Parsing Through Incremental Multi- Algorithm Processing. In Lawrence, E., *Text Based intelligent systems*. (pp 83-100). Associates Publishers.

Mich, L., Garigliano, R., 1997. NL-OOPS A Tool for Object Oriented Requirements Analysis. In *The LOLITA Project: The First Ten Years*, Vol.2 Applications, Springer-Verlag.

Osborne, M., MacNish, K., 1996. Processing Natural Language Software Requirement Specifications *IEEE Computer Society DL.* (pp 229-237).

Overmyer, S., Lavoie, V., Rambow, O., 2001. Conceptual Modeling through Linguistics Analysis Using LIDA. *23rd International Conference on Software engineering*.

Pérez-González, H., Kalita, J., 2002. Automatically Generating Object Models from Natural Language Analysis, *Companion OOPSLA 2002.*

Pérez-González, H., Kalita, J., Nunez-Varela A. Wiener, R., 2005. GOOAL: An educational Object Oriented analysis Laboratory, *Companion OOPSLA 2005.*

Polajnar, T., Cunningham, H., Tablan, V., Bontcheva, K.,2006. *Controlled language IE Components Version 1.* EU-IST Integrated Project (IP) IST-2003-506826 SEKT, D2.2.1 Report Sheffield.

Rumbaugh, J., Blaha, M., Lorensen, W., Eddy, F., Premerlani, W. ,1996. *Object Oriented Modeling and Design*. Prentice Hall.

Saeki, Horai, M., Enemoto, H.,1989. Software Development Process from Natural Language Specification. In *Proceedings of the 11th International conference on SW Engineering IEEE,* Computer Society Press.

Zapata, C., Gelbukh, A., Arango, F.,2006. UN-Lencep Obtencion Automatica de Diagramas UML a partir de un leguaje controlado. *Avances en ciencias de la computación VII Encuentro Internacional de Computacion ENC 2006*, ISBN 968-5733-06-6.

Zapata, C., Gelbukh, A., Arango, F., 2006. Pre-conceptual schema: a UML isomorphism for automatically Obtaining UML Conceptual Schemas. *Research in computing Science: Advances in Computer Science and Engineering.* Vol 19, (pp 3-13).