Automatically Generating Object Models from Natural Language Analysis

Hector G. Perez-Gonzalez
Department of Computer Science
Univ. of Colorado at Colo. Springs
Colorado Springs. CO, USA
(719) 262-3325
hectorgerardo@acm.org

Jugal K. Kalita
Department of Computer Science
Univ. of Colorado at Colo. Springs
Colorado Springs. CO, USA
(719) 262-3432
kalita@pikespeak.uccs.edu

ABSTRACT

Our goal is to enable rapid production of static and dynamic object models from natural language description of problems. Rapid modeling is achieved through automation of analysis tasks. This automation captures the cognitive schemes analysts use to build their models of the world through the use of a precise methodology. The methodology is based on the use of proposed technique called role posets. and a semi-natural language (called 4W). Original problem statements are automatically translated to 4W language. The produced sentences then, are analyzed with role posets to produce static model views. Finally the 4W sentences are used to generate dynamic views of the problem. This set of methods maximizes analysis process agility, promotes reusability and constitutes a valuable tool in the learning process of object thinking.

Keywords

Object oriented modeling, computational linguistics, discourse analysis, set theory, object oriented education.

1. PROBLEM DESCRIPTION

The successful development of any software system depends on the communication between customers and software developers. Customers communicate in common, widely comprehensible, but at the same time vague and potentially contradictory natural language (NL). Computer specialists communicate using precise, (but at the same time, not widely understandable) formal languages. Automatic generation of object models through and intermediate from informal NL requirement documents may accelerate accords between stakeholders. Past similar efforts[2][4] produce static object views and the associated automatic tools are based in methodologies considering noun frequencies. Our techniques look for simplicity and effectiveness considering also semantic issues and discourse analysis and simplification.

2. METHODOLOGY

The purpose of this methodology is to promote rapid software development, reusability, and support memorable experiences in "object oriented thinking" and it is mainly supported by the use of automated tools. The general steps are as follows:

OOPSLA'02, November 4-8,2002 Seattle, Washington, USA

ACM 1-58113-626-9/02/0011.

Step 1. Analysts obtain a set of describing requirement documents coming from stakeholders of different kinds. (several documents from final users, several from different clients, etc.)

Step 2. The name of the problem, problem domain's (PD) name and possible sub-domains are declared.

Here is an example of one such simple requirements document:

Problem Domain: *"Operating systems"*

Sub-domain: *"Concurrent programming"*

Problem name: *"Dinning philosophers simulation"*

Problem Description: *There are 5 philosophers and 5 forks around a circular table. Each philosopher can take 2 forks on either side of him. Each fork may be either on the table or used by one philosopher. A philosopher must take 2 forks to eat.*

Step 3. Each one of the documents is processed using an automatic tool considering associations between documents with the same kind of authors (users, clients, managers, etc). The tool produces design view diagrams (class, objects, sequence and activity diagrams) that are validated by the user.

Step 4. Any participant, stakeholder or mainly any student of object oriented (OO) design may modify, delete or add new sentences to any requirement text to identify the consequences in real time and produce a memorable learning experience.

Step 5. Produced information is stored associated with the declared problem domain in order to promote reusability.

The proposed techniques produce OO static and dynamic model views of the problem in Unified Modeling language (UML) [1].

3. THE 4W LANGUAGE

In order to have enough simple sentences to be analyzed with the role posets technique, we propose a subset of English called the 4W language (4WL) to which original sentences are translated. In the 4WL certain expressions show a syntactic subject who performing an action, an optional syntactic object receiving the action (if the verb is transitive) and an optional prepositional phrase giving information about adverbial or adjectival relation to some other word in the sentence. In addition, we can have a link between sentences, a relationship that gives us information about time sequences or conditional rules between them. In summary, a 4W sentence tries to answer the following four questions related to a particular object: **What** does the object do?, **Who** receives the action?, **Which** other participates? and **When** does it happen? 4W sentences are displayed in a 4W table with each slot filled with a word.

Automatic 4W translation of our requirements example is:

1. five philosophers are around a circular table

2. five forks are around a circular table

3. Each philosopher has side.

4. Each philosopher can take two forks on side.

5. Each fork may be on the table.

6. Each fork may be used by one philosopher.

7. A philosopher take two forks.

8. A philosopher eats (When 7).

This semi-formal language (4WL) is expressive enough to translate most declarative sentences, clear enough to be understood and validated for problem domain users and unambiguous enough to produce didactic design diagrams.

4. ROLE POSETS

Partially ordered sets of roles (Role Posets) are a conceptual framework to take analysis and design decisions to produce OO static model views (Class, Objects and use cases diagrams). It is based on the linguistic concept of theta roles [3] and the mathematical concept of partially ordered sets [5].

Attributes, behavior, and relationships are the components that an entity (represented by a noun) must have in order to be promoted to a class. These three components can be discovered through the analysis of the role every noun plays with the complete requirements texts, and the verbs that define these roles. This is done with a proposed semantic structure called "role machine", a proposed semantic abstraction that associates groups of numerous verbs in general formal schemes.

A role machine for some transitive verbs like *eat, drink, kill* and others would be:

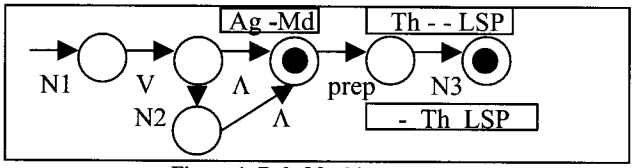

Figure 1. Role Machine example.

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 associated with a list of roles it plays.

In our example we have the nouns: N1=Philosopher, N2=Fork, N3=Table, N4=Side. Automatic generation of roles produces:

N1 = {User, User, Whole, Theme}	N2 = {Used, Used, Theme, Theme}
N3 = {LSP, LSP}	N4 = {Attribute, LSP}

Table 1. Result of analysis of roles.

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) }.

After following the technique, we obtain the probabilities every noun (N) has to become a Class. Results of this particular example are shown in table 2.

Noun	Noun's name	Prob. To be Class
N1	Philosopher	100%
N2	Fork	85%
N3	Table	20%
N4	Side	9%

Table 2. Result of analysis of nouns

Different arrangements of 4W sentences facilitate validation by Trustable choice of roles are based in this role poset scheme.

Figure 2 shows a simple class diagram obtained from our example considering a validation threshold of 50 %.

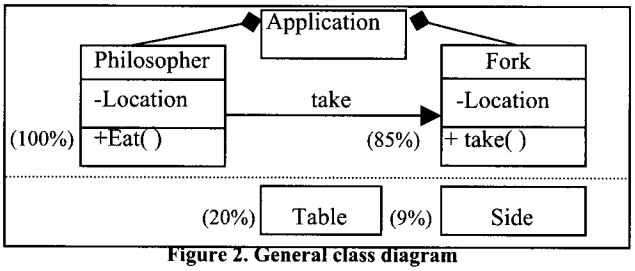

Figure 2. General class diagram

Different arrangements of 4W sentences facilitate validation by analysts and clients. After that step, the tool can produce general dynamic UML diagrams. These diagrams and the 4W sentences may be used to detect and solve ambiguity.

Figure 3 shows a simple sequence diagram from our example.

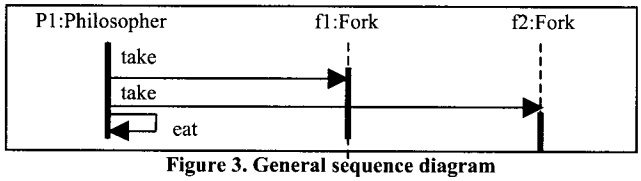

Figure 3. General sequence diagram

5. FUTURE WORK AND CONCLUSIONS

Prototype tool: "GOOAL" (Graphical Object Oriented Analysis Lab.) has produced good results with simple problems. It is being developed to work with complex ones. Observed advantages are formalization, standard notation, validation, traceability, efficiency and early identification of misunderstood requirements.

6. ACKNOWLEDGMENTS

The research project is sponsored by Universidad Autonoma de San Luis Potosi and PROMEP-SEP, Mexico, as part of Phd studies scholarship.

7. REFERENCES

[1] Booch Grady, The Unified Modeling Language User Guide. Addison-Wesley. 1997.

[2] Boyd, N. Using Natural Language in Software Development. Journal Of Object Oriented Programming. February.1999.

[3] Haegeman, Government & Binding theory, Blackwell, 1991.

[4] Overmyer, S. P. et all. Conceptual Modeling through Linguistic Analysis Using LIDA. 23rd international conference on Software engineering July 2001.

[5] Trotter, William T. Combinatorics and Partially Ordered Sets: Dimension Theory, Johns Hopkins Univ. Press, 2002.