UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

IMPROVING THE QUALITY OF CONCEPTUAL MODELS WITH NLP TOOLS: AN EXPERIMENT

Luisa Mich, John Mylopoulos and Nicola Zeni

2002

Technical Report # DIT-02-0047

Improving the Quality of Conceptual Models with NLP Tools: An Experiment

Luisa Mich[1], John Mylopoulos[2], Nicola Zeni[3]

Abstract. *Conceptual models are used in a variety of areas within Computer Science, including Software Engineering, Databases and AI. A major bottleneck in broadening their applicability is the time it takes to build a conceptual model for a new application. Not surprisingly, a variety of tools and techniques have been proposed for reusing conceptual models (e.g., ontologies), or for building them semi-automatically from natural language descriptions. What has been left largely unexplored is the impact of such tools on the quality of the models that are being created. This paper presents the results of an experiment designed to assess the extent to which a Natural Language Processing (NLP) tool improves the quality of conceptual models, specifically object-oriented ones. Our main experimental hypothesis is that the quality of a domain class model is higher if its development is supported by a NLP system. The tool used for the experiment -- named NL-OOPS -- extracts classes and associations from a knowledge base realized by a deep semantic analysis of a sample text. Specifically, NL-OOPS produces class models at different levels of detail by exploiting class hierarchies in the knowledge base of a NLP system and marks ambiguities in the text. In our experiments, we had groups working with/without the tool, and then compared and evaluated the final class models they produced.*

1. Introduction

Conceptual models are used in a variety of areas within Computer Science, including Software Engineering, Databases and AI [1]. A major bottleneck in broadening their applicability is the time it takes to build a conceptual model for a new application. A variety of tools and techniques have been proposed to address this problem. Some of these tools support the reuse of conceptual models (e.g., ontologies), while others support their semi-automatic construction from natural language descriptions. What has been left largely unexplored in this research is the impact of such tools on the quality of the models that are being created.

In this paper we present the results of an experiment designed to assess the extent to which a Natural Language Processing (NLP) tool that supports the semi-automatic construction of a conceptual model improves the quality of the result. The tool used for the experiment -- named NL-OOPS -- extracts classes and associations from a knowledge base realized by a deep semantic analysis of a sample text. In particular, NL-OOPS produces class models at different levels of detail by exploiting class hierarchies in the knowledge base of the NLP system and marks ambiguities in the text. In our experiments, we had groups working with/without the tool, and then compared and evaluated the final class models they produced.

Section 2 of the paper describes the features of the NL-OOPS tool and the knowledge base upon which the system of natural language processing is based. Section 3 outlines the stages of the experiment, beginning with the experimental hypotheses and operational decisions taken. Section 4 contains an evaluation of the models produced by the experiment, highlighting the effect that NL-OOPS had on their quality. The concluding section summarises the findings of the experiment and describes directions for future research.

2. The NL-OOPS Tool

To build a tool that is able to extract from textual descriptions the elements necessary to design and build conceptual models, it is possible to adopt two complementary approaches. The first limits the use of natural language to a subset that can be analysed syntactically. Various dialects of "Structured English" do just that. The drawback of this approach is that it won't work for existing text. The second approach adopts NLP systems capable of understanding the content of documents by means of a semantic, or “deep,” analysis. The obvious advantage of such systems is that they work for arbitrary natural language text. Moreover, such systems can cope

[1] Department of Information and Telecommunication Technologies, University of Trento; email: mich@dit.unitn.it.
[2] Department of Computer Science, University of Toronto; email address: jm@cs.toronto.edu.
3 Department of Computer and Management Sciences, University of Trento; email: zeni@cs.unitn.it.

with ambiguities in syntax, semantics, pragmatics, or discourse (see also [2]). Clearly systems of this type are much more complex, require further research, and have a limited scope compared to those in the first category[4].

NL-OOPS is based on LOLITA (Large-scale Object-based Language Interactor, Translator and Analyser), a NLP system whose development required approximately one hundred man-years. In the sequel we focus on those features that are relevant to the experiment at hand. The first aspect to note is that NL-OOPS is a general-purpose system that was not designed with any specific purpose or domain in mind. As such, the nucleus of LOLITA includes all the functions for analysis of natural language: morphology, parsing with respect to a 1500-rule grammar, semantic and pragmatic analysis, inference, and generation. The knowledge base of the system consists of a kind of conceptual graph [3], called SemNet, which contains about 150,000 nodes. Thus LOLITA is among the larger implemented NLP systems. Since the concepts in SemNet are represented in a form independent of surface linguistic structures, it is possible to overcome a serious problem faced when extracting information useful for conceptual modeling. In particular, heuristic rules for "structural" models (e.g., the Entity-Relationship or the UML class model) suggest looking for nouns to identify entities/classes, and verbs to identify relationships and associations. In practice, this type of heuristic fails frequently because nouns can be verb-phrased, and verbs can be noun-phrased [4]. Compare, for example, the phrases: "John kissed Mary" - "John gave a kiss to Mary" and "Guests reserve a room …" – "Guests make a reservation …" (see [2]).

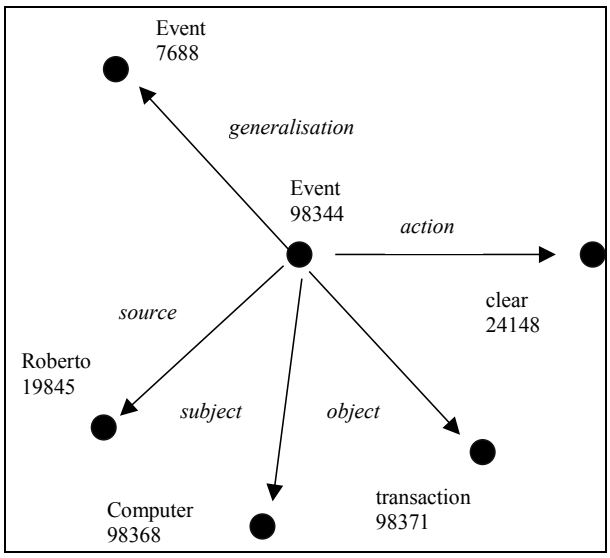

Figure 1 – Simplified Event node

In terms of structure, SemNet contains two basic types of nodes: entity nodes and event nodes. While entity nodes represent concepts organised into hierarchies, event nodes represent complex relationships among concepts, which cannot be fully represented by arcs. Event nodes have a frame-like structure which makes it possible to display their components: subject, action, object (if the verb is transitive), source, date, etc. An example of an event node is shown in figure 1.

In addition, every node has an associated group of control variables (or *controls*). The control *rank*, for example, gives information regarding quantification: *universal* for concepts of general sets, *individual* for anonymous instances of a concept, and *named individual* for those having a name. Another important control is *family*, used by NL-OOPS to identify actors for UML use case models. Some of the possible values for family are: *living, human, human organisation, inanimate, man-made*. Critical for the construction of the support tool for conceptual modeling is the classification of event nodes. There are four categories of event nodes: *static*, *cyclic*, *dynamic*, and *instantaneous*. For example, a static event would be "to own something", a cyclic event would be "to manage a company", a dynamic event "to run a race" and an instantaneous event "to win a race".

LOLITA is capable of analysing automatically about 90% of encountered phrases. The degree of accuracy depends on the quality of the text for input and on the length of the sentences[5]. Consequently, the output of the tool contains most of the information from the original text. NL-OOPS implements an algorithm for the

[4] The choice of a NLP system may be guided by an evaluation of its capabilities for the tasks defined by the Message Understanding Competition (MUC) Conference, organised by DARPA: http://www.itl.nist.gov/iad/894.02/related_ projects/muc/.

[5] Problems often arise for sentences containing more than 40 words.

extraction of classes and associations from the semantic network of LOLITA. Documents (in English) are analysed by LOLITA and their content is stored in its knowledge base, adding new nodes to its semantic network. All these nodes can then be used to produce a conceptual model (figure 2).

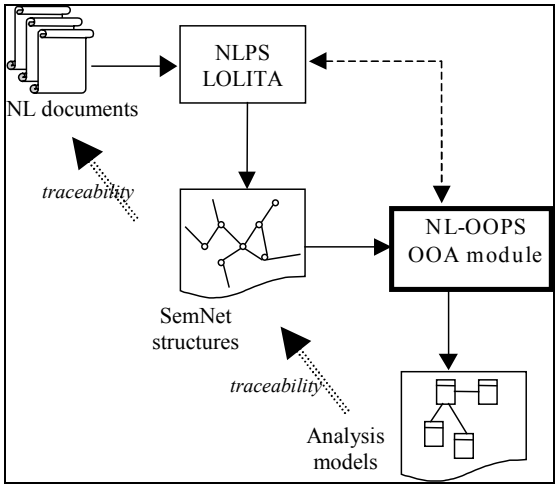

Figure 2 – The generation process of the model

In particular, the algorithm for identifying classes and associations is based on two phases, illustrated in figure three and described in [2].

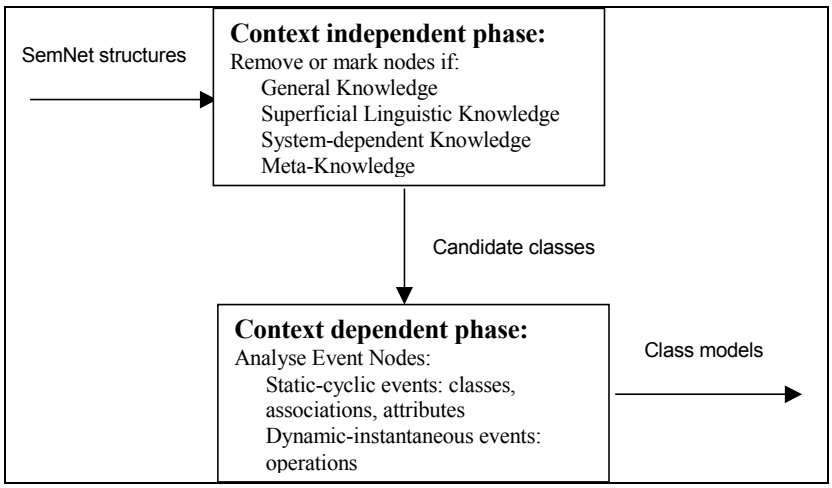

Figure 3 – The model-generation algorithm

The steps in the first (context independent) phase function as filters that choose which nodes are useful for domain analysis. These filters create a list of classes possible for the class model (candidate classes). The second phase of the algorithm is based on the classification of the events of LOLITA in order to identify the final classes of the model. This phase also determines what associations will be included in the class diagram and inserts methods and attributes. The basic assumption is that the classes and associations describe the static structure of a domain. Thus they will be identifiable using the information on the type of event linked to candidate classes, particularly those events classified as static or cyclic. Dynamic or instantaneous events can be useful in identifying methods. Moreover, by varying the number of these events it is possible to have diagrams of classes at different levels of abstraction, taking advantage of the hierarchy of nodes of the system's semantic network (figure 4).

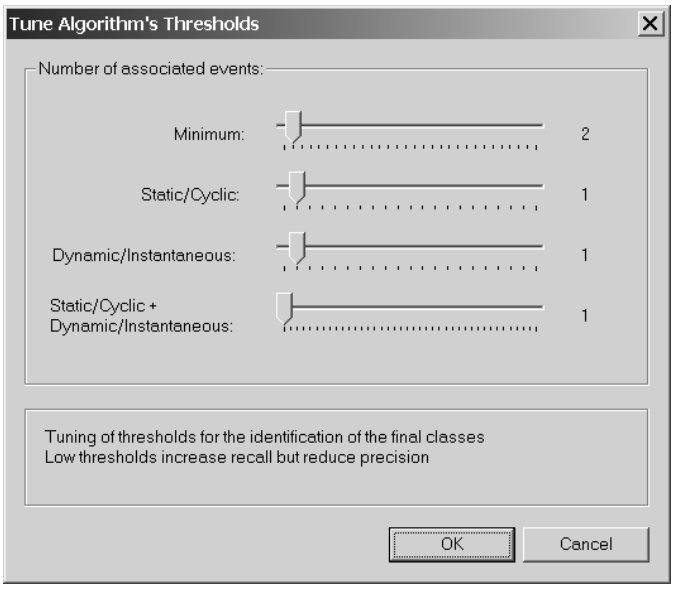

Figure 4 – Thresholds for obtaining different versions of the class models

Figure 5 shows the NL-OOPS's interface, which consists of three frames. The top right frame contains the text being analysed for the *ScoreSystem* case used for the experiment (see appendix A for the full text.) The left frame gives a representation of the SemNet structures used by LOLITA for the analysis of the document. Old nodes already stored in SemNet are in yellow to be distinguished from new nodes that are two-tone. After running the modeling module, the third frame, bottom right, contains a version of the class model. To maintain consistency, all the elements used and created by the tool use notations similar to those of the file system and of widely used operating systems.

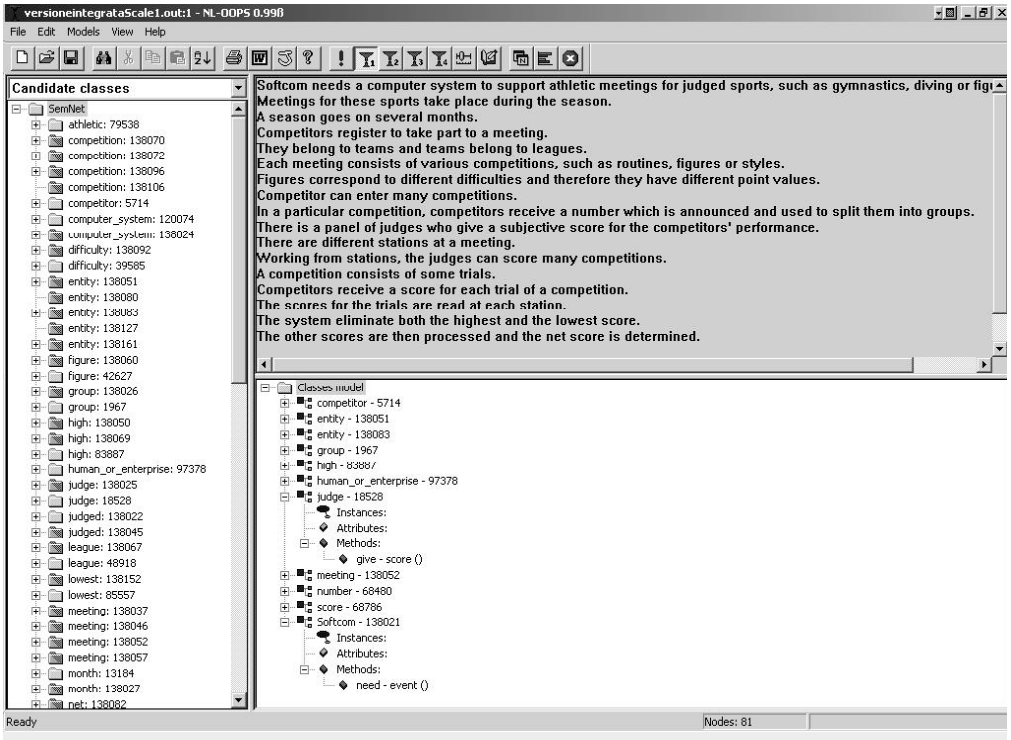

Figure 5 - The NL-OOPS interface

The tool can also show intermediate outputs, some corresponding to nodes marked by individual steps of the algorithm, while others are useful in identifying elements of the conceptual model (associations, attributes, methods, use cases, etc.). In addition, the tool can export intermediate results to a Word file. Moreover, a traceability function allows the user to check what nodes were created for a given sentence. The same

information can be obtained using the nodes browser of NL-OOPS, which makes available further information related to a specific node, such as the marking of the algorithm (figure 6).

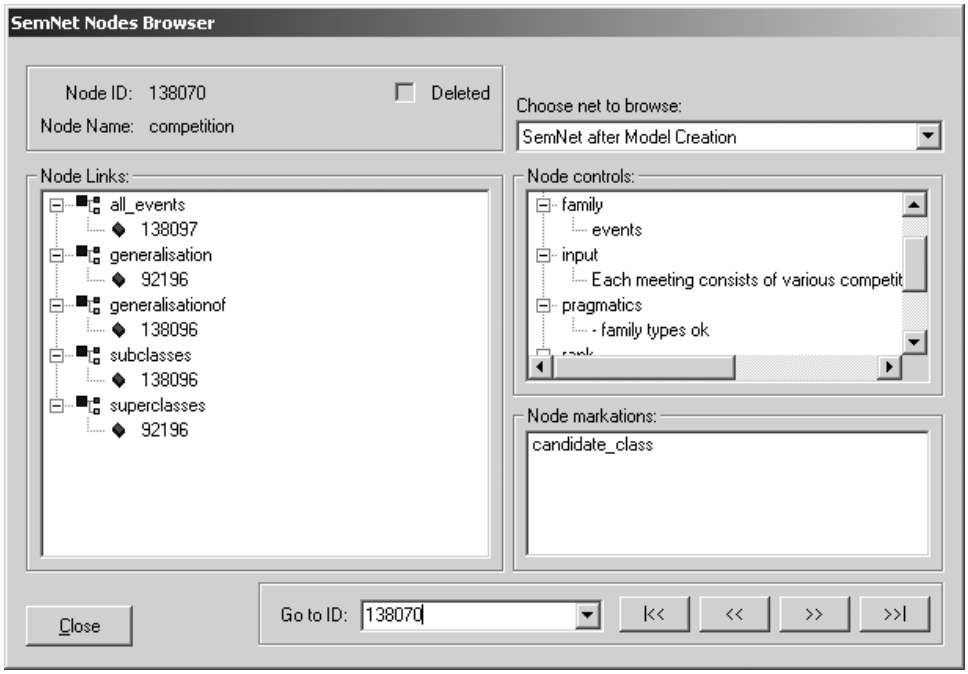

Figure 6 – The NL-OOPS nodes browser

The functions that support traceability between the text, the nodes of the knowledge base in LOLITA, and the generated models make it possible to identify semantic ambiguities in the text [5]. In a case, for example, where it is impossible to identify the subject and the object of a given action, LOLITA reacts by introducing generic nodes such as entity or something. An ambiguity of this type was actually found in the text used in our experiment. Specifically, the sentence "In a particular competition, competitors receive a number which is announced and used to split them into groups.", leaves it unclear who or what created the groups.

NL-OOPS also provides statistical information. In the case of *ScoreSystem*, the initial text was found to contain 178 words. LOLITA used 199 nodes to make it less ambiguous, while for the default values of the thresholds the list of candidate classes contains 81 elements, organised into 35 hierarchies.

3. The Experiment

Our main experimental hypothesis for the experiment is that the quality of a domain class model is higher if its development is supported by a NLP tool such as NL-OOPS. To this end, we designed an experiment involving twelve students of the University of Trento who are academically comparable and who have the same general level of competence in analyzing and modeling with UML. Their competence was comparable to that of a recently employed junior analyst. This was verified by means of a questionnaire, which also evaluated the subjects' command of English (a relevant issue, given that the subjects were Italian.)

The students were subdivided into six groups according to the results of the questionnaire (table 1). The first three groups worked with NL-OOPS. The text used was adapted from a case study found in [6], and deals with a problem that requires some familiarity with sports. The language used is quite simple, but does contain some ambiguities whose clarification requires some assumptions regarding the domain. Each group had access to a PC with Microsoft Office 2000 while carrying out the experiment. The instructions were given in a brief PowerPoint presentation, and these files were also made available on each PC. The most important of these are displayed in figure 7.

	1 Tool	2 Tool	3 Tool	1	2	3
English	intermediate/ elementary	intermediate/ intermediate	advanced/ advanced	intermediate/ advanced	intermediate/ intermediate	intermediate/ intermediate
Specialized Training	no/no	no/no	no/no	no/no	no/no	yes /no
Work Experience in IT	no/no	yes/no	no/no	no/no	yes/yes	no/no
University exposure to ERD	IS/IS,DB	IS/DB	IS,DB/IS,DB	DB/IS,SWE	IS/IS,DB	IS,DB,SWE/ IS;DB
University experience with UML	IS,OS/IS,OS	IS,DB/IS,DB	IS,DB/IS,DB	DB/IS,SWE	DB/SWE	DB,SWE/ DB,SWE
Use of UML for real projects	no/no	no/no	no/no	no/no	no/yes	no/no
Familiarity with UML diagrams	development/ development	read/ development	development/ development	read/ development	read/ development	development/ development
Use of CASE tool for Univ. coursework	yes/yes*	no/yes	yes/yes	no/yes	yes/yes	yes/yes
Existence of Project NL-OOPS	no/no	no/yes	no/yes	no/yes	no/no	no/no
OO programming	development/ development	read/exist	exist/exist	exist	read/ development	development/ development
OO languages	Java, VB/Java, VB	Java/null	Java/null	null/null	Java/Java	Java/Java

* IS=Information Systems, OS=Operating Systems, DB=Data Base; SWE= Software Engineering

Table 1 – Results of the questionnaire

Instructions

- Read carefully the text that describes the problem for which you desire to build a class model (slide: "Problem statement")
- Build the class model using terms in English: if you have problems with the English, please check the translation; (slide: "Terms of the problem")
- The slide "Notation for class diagrams" contains UML notation for class diagrams.

Figure 7 – Main instructions for the experiment

The groups were asked to develop a model of the domain classes for the case study, identifying classes, associations, attributes, multiplicity and methods. To minimise any confusion resulting from the language, everyone was provided with a paper copy of the text with the literal translation in Italian on the reverse side. The length of the experiment was initially set for 75 minutes but was later extended to 90 minutes.

The groups that used NL-OOPS received brief training in the use of the tool. The training focused on the functions that support the identification of classes, the ability to vary the threshold for the class-building algorithm, also to see the list of candidate classes. The groups also received some training in using the node browser.

It is important to remember that the class models produced automatically by NL-OOPS derive from nodes in SemNet used to model the content (meaning) of the text. This means that their quality is linked to the quality of the input text and the quality of the analysis carried out by LOLITA. The intervention of the modeller is necessary at two levels: (1) to identify spurious classes that result from errors in the analysis or from text ambiguities; (2) to decide on the level of detail at which to describe the class hierarchies. For short text such as the one used in our experiment, the first type of problem is more relevant.

To avoid additional noise, it was decided to not use a specific tool to build the class diagrams. For the six diagrams produced, two groups used PowerPoint, one used Excel, while all three groups working without a NL-OOPS chose Word. Before giving an evaluation of the models, we present the different classes suggested by the tool for different thresholds (table 2). The classes contained also in the model given in [6], which we adopted as reference model, are indicated in bold.

NLOOPS-1	**NL-OOPS-2**	**NL-OOPS-3**	***Reference classes***
competition			*competition*
competition			*competitor*
competitor	**competitor**	**competitor**	*figure (styles,routines)*
entity (work)	entity (work)		*judge*
entity (announce)	entity (announce)		*league*
group	group	group	*meeting*
high	high		*score*
human_or_enterprise	human_or_enterprise		*season*
judge	**judge**		*station*
meeting	**meeting**		*team*
number	number	number	*trial*
score	**score**	**score**	
Softcom	Softcom		
R=.45; P=.42	R=.36; P=.36	R=.18; P=.5	

Table 2 – Classes identified by NL-OOPS

As shown, the table does not contain classes such as season, station, team, and trial. Instead, these are instead present in the list of candidate classes. In the first two cases three classes are indicated: entity (works), entity (announce), human_or_enterprise. These correspond to ambiguities in the text. In the first case, entity was introduced by the NLP for the sentence “Working from stations, the judges can score many competitions”. Here, it cannot be assumed that the subject of working is “judges”. The second class resulted from an analysis of the sentence “In a particular competition, competitors receive a number which is announced and used to split them into groups”. In this sentence, the subject of the verb "announce" is ambiguous. Finally, “human_or_enterprise” was derived from an attempt to disambiguate the first sentence where the tool could not determine whether SoftCom is a company or a person. Here, a company title (e.g., "Ltd.", "Co.", or "Inc.") would have resolved the ambiguity. For all of these nodes the use of the node browser of NL-OOPS makes it possible to go back to the sentence from which the nodes were derived.

4. Evaluation

The evaluation of the quality of conceptual models is a difficult task and is still largely under-investigated (there are not many contributions addressing this problem; one of the few is [7]). For our experiment we adopted a mixed approach. Specifically, we used the measures of recall and precision to evaluate the classes identified with respect to the reference model [6]. We also asked two experts to mark and comment on the solutions proposed by the different groups.

Table 4 displays the classes identified by the six groups, along with the evaluations of the recall and precision measure. R (recall) counts the number of classes correctly identified with respect to the total number in the model. P (precision) counts the number of correct classes with respect to the total number identified in the model. In addition, we used the F-measure that is a function of both R and P, and is high if both measures are good [8]. Obviously, we are dealing with quantitative evaluations that provide information on the quality of the models.

Reference Class List [6]	**1 tool**	**2 tool**	**3 tool**	**1**	**2**	**3**
Competition (1)°	x	x	x	x	x	x
Competitor (1,2,3)°	x	x	x	x	x	x
Figure			x*	x*	x*	x (Sport)
Judge (1,2)°	x	x	x	x	x	x
League	x		x	x	x	x
Meeting (1,2)°	x	x	x	x	x	x
Score (1,2,3)°	x	x	x	x	x	x
Season				x	x	
Station				x	x	x
Team	x		x	x	x	x
Trial	x	x	x	x	x	
Other	System	Group Number Prize	Group Vote	Number	Group Month Number	Prize
	R=73% P=89% F=80.2	R=54% P=67% F=59.8	R=81% P=81% F=81	R=100%; P=92% F=95.8	R=100% P=79% F=88.3	R=81% P=90% F=85.3

° Classes identified by NL-OOPS for the three different thresholds in table 2.

* Routines,Styles

Table 3 – Classes in the models

The table suggests that some classes were correctly identified by all the work groups (competition, competitor, judge, meeting, score), and also that these were classes identified by the tool also. For the classes that were not included in the models, it is worth noting that the tool seemed to have an inertia effect. On one hand, it introduced less acceptable classes (e.g., Group). On the other hand, it resulted in the failure to introduce some indispensable classes (e.g., Season, Team). A comparison with questionnaire results suggests that this may be related to a greater level of dependence on the tool for less-experienced participants. Of course, these results are also a function of the capabilities of the tool itself. Another observation regarding the classes Figure (which refers to different types of sport) Station and Season is that these are associated with domains that require a scoring system. This makes them critical and at the same time problematic for the creation of models.

To evaluate the overall quality of the models while taking into account all other elements (associations, multiplicities, attributes, and methods), we asked for the help of two experts. One used a more structured evaluation method -- introducing parameters for form and content -- while the other was less rigid. However, the final classification was identical. The results are shown in table 4, which compares the models produced by the six terams using the F-measure.

	1 tool	**2 tool**	**3 tool**	**1**	**2**	**3**
Class identification (F-measure)	5°	6°	4°	1°	2°	3°
Quality of the model (Expert evaluation)	4°	5°	2°	3°	1°	6°

Table 4 – Classification of Models

The experts judged the best model to be the one produced by a group in which two of the students had used UML for other projects[6]. NL-OOPS did not seem to help significantly for the identification of classes. However, the experts thought that the model of group 2 was the second best model overall. NL-OOPS. Another observation regards the influence of the tool concerns the names of elements in the models. The names of classes produced by NL-OOPS were more consistent compared to those proposed by students. Similar considerations apply to a consistent use of the UML's notation.

From the observations of those participating in the experiment (question 14 of the questionnaire and in a brief final interview) some considerations emerged:

- Those who used NL-OOPS would have preferred further training;

[6] In fact, one of the experts asked if the model was developed by a professional.

- Three groups (group 1 and 2 working with the tool and group 2 without the tool) noted the presence of ambiguity in the text and would have liked more information on the domain or a longer text;
- All the groups that used NL-OOPS would have preferred to have a tool to design the diagrams, while groups working without the tool did not voice this preference.

5. Conclusions

We have reported a preliminary experiment regarding the use of NLP systems to produce class diagrams. To the best of our knowledge, no other experimental results exist on this topic. The experiment brought to light useful elements for further research into the possibility of effectively supporting conceptual modeling with NLP tools, as well as for research in improving the tool itself.

In particular our experiment suggests the need for enhanced NLP systems that are capable of producing high quality analyses. In addition, we need to broaden the scope of our experiments to evaluate the impact on productivity of NLP tools. Another point that emerged from the study was the importance of personal experience of the subjects involved. Senior analysts could take full advantage of this type of instrument, making more effective use of the models produced semi-automatically even if the initial quality of the models produced by the tool is not high.

As for the planning and design of further studies, it is vital to take into account the fact that for a more effective and efficient use of the tool, an extended training period is required. This is not surprising, given that the tool is founded on a rather complex semantic network which requires exploration and interaction in a kind of "hypertext" mode -- a potentially disorientating exercise. The use of intermediate results generated by the tool is useful only when accompanied by the use of the node browser and the traceability functions. It is also probably necessary to ensure a better understanding of the rules used by NL-OOPS when filtering the nodes used to build the models.

Acknowledgements

We would like to thank Prof. Roberto Garigliano for his permission to use LOLITA.

References

1. Mylopoulos, J.: "Information Modeling in the Time of the Revolution", *Information Systems 23(3-4),* May (1998) 127-156.
2. Mich, L.: "NL-OOPS: From Natural Language to Object Oriented Requirements using the Natural Language Processing System LOLITA", Journal of Natural Language Engineering 2 (2): Cambridge University Press, (1996) 161-187.
3. Sowa, J.: Conceptual Structures, Reading, MA, Addison Wesley (1983).
4. Booch, G.: Object Oriented Analysis and Design with Application, Redwood City, CA, Benjamin/Cumming (1984).
5. Mich, L, Garigliano, R.: "Ambiguity measures in Requirements Engineering." International Conference on Software Theory and Practice. In: Feng, Y., Notkin, D., Gaudel, M. (eds), House of Electronics Industry (2000) 39-48.
6. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-Oriented Modeling and Design, Prentice-Hall (1991).
7 Moody D.L., Shanks G.G: "What makes a good data model? A framework for evaluating and improving the quality of enityt Relationship Models", The Australian Computer Journal 30 (3), August (1998) 97-110.
8. van Rijsbergen, C.J.: *Information Retrieval*, Butterworths (1979).

Appendix

ScoreSystem Problem statement

SoftCom needs a computer system to support athletic meetings for judged sports, such as gymnastics, diving or figure skating. Meetings for these sports take place during the season. A season goes on for several months.

Competitors register to take part to a meeting. They belong to teams and teams belong to leagues.

Each meeting consists of various competitions, such as routines, figures or styles. Figures correspond to different difficulties and therefore they have different point values.

Competitor can enter many competitions. In a particular competition, competitors receive a number which is announced and used to split them into groups.

There is a panel of judges who give a subjective score for the competitors' performance. There are different stations at a meeting. Working from stations, the judges can score many competitions.

A competition consists of some trials. Competitors receive a score for each trial of a competition. The scores for the trials are read at each station. The system eliminates both the highest and the lowest score. The other scores are then processed and the net score is determined. Final prizes are based on the net scores.