Experimenting with Linguistic Tools for Conceptual Modelling: Quality of the models and critical features

Nadzeya Kiyavitskaya[1], Nicola Zeni[1], Luisa Mich[1], John Mylopoulos[2]

[1] Department of Information and Communication Technologies, University of Trento
Via Sommarive 14, 38050, Povo, Trento, Italy
{nadzeya.kiyavitskaya, nicola.zeni, luisa.mich}@unitn.it
[2] Department of Computer Science, University of Toronto
jm@cs.toronto.edu

Abstract. This paper presents the results of three experiments 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. In our experiments, we had groups working with and without the tool, and then compared and evaluated the final class models they produced. The results of the experiments give insights on the state of the art of linguistics-based Computer Aided Software Engineering (CASE) tools and allow identifying important guidelines to improve their performance, highlighting which of the linguistic tasks are more critical to effectively support conceptual modelling.

1. Introduction

According to the results of a market research whose aim was to analyse the potential demand for a CASE tool integrating linguistic instruments as support for requirements analysis, 79% of requirements documents are couched in unrestricted NL. Also the majority of developers (64%) pointed out that the most useful thing to improve general efficiency in modelling user requirements would be a higher level of automation [20]. However, there is still no commercial NLP-enabled CASE tool. There have been many attempts to develop tools that support requirements engineering since the '80s. The objective of this work was to evaluate how the linguistic CASE tools can support the modelling process, thereby speeding up requirements formalisation. This research makes use of linguistic techniques which were considered state-of-the-art at that time, although newer technologies have now been developed.

In this work we present the results of a set of experiments designed to investigate the extent to which a NLP tool that supports the semi-automatic construction of a conceptual model improves their quality. The tool used for the experiments – named NL-OOPS – extracts classes and associations from a knowledge base realized by a

deep semantic analysis of a sample text [14]. 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 [16], [17], [18]. In our experiments, we had groups and individuals working with and without the tool, and then compared and evaluated the final class models they produced. The results of the experiments give some insight on the state of the art of linguistic CASE tools and identify some important parameters for improving their performances.

Section 2 of the paper presents the main research projects related to the use of linguistic tools of different complexity to support conceptual modelling. Section 3 describes the main features of the NL-OOPS tool and the knowledge base upon which the system of NLP is based. Section 4 outlines the stages of the experiments and contains an evaluation of the models produced, focusing on 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. Theoretical background

Recently an increasingly number of studies have proposed the use of linguistic tools to support conceptual modelling. A description of the early related works can be found in [16] and an updated list of the different approaches is given on the web site of the NL-OOPS project.[1] We will report here only some of the most important to illustrate the efforts toward the development of linguistic based CASE tools.

In the early 1980's, Abbott [1] proposed an approach to Ada program design based on linguistic analysis of informal strategies written in English. He suggested a methodology that produces static analysis and design products obtained by an informal technique requiring high participation of users for making decisions. This approach was further developed by Booch [4], who proposed a syntactic analysis of the problem description (nouns suggest objects and classes of objects, and verbs suggest operations). However, both Abbott and Booch recognised the importance of semantic and real-world knowledge in the analysis process. Saeki, Horai, and Enomoto [27] were the first to use linguistic tools for a comprehensive analysis of the software process. They described a process of incrementally constructing software modules from object-oriented specifications obtained from informal NL requirements. Dunn and Orlowska [10] described a natural language interpreter for the construction of NIAM (Nijssen's Information Analysis Method) conceptual schemas. The construction of conceptual schemas involves allocating surface objects to entity types, i.e. semantic classes, and the identification of elementary fact types. The system accepts declarative sentences only and uses grammar rules and a dictionary for type allocation and the identification of elementary fact types. Another relevant work was the expert system ALECSI-OICSI [6] that allowed the user to express requirements both in NL and in graphic form. ALECSI uses a semantic network to represent domain knowledge [25]. Along similar lines, Cockburn [7] investigated the application of linguistic metaphors to object-oriented design by associating relational

[1] http://nl-oops.cs.unitn.it

nouns with objects and adverbs with polymorphisms. Cordes and Carver [9] proposed one of the first attempts to apply automated tools to requirements analysis and the automatic generation of object models from requirements documents. While the translation of the initial requirements into a suitable knowledge base requires human interaction to resolve ambiguities, the subsequent translation of the domain knowledge into object models is automated. The authors acknowledged that the translation of formalized knowledge into object models is sensitive to the quality of the initial requirements specification (as one would expect). Goldin and Berry [15] in their work introduce a new approach for finding abstractions in NL text using traditional signal processing methods. Burg and Van de Riet [5] launched an interesting and ambitious project that attempts to minimize the participation of the user in the job of extracting classes and relationships from the text. Their system, COLOR-X, aims at including a large lexicon to aid in semantic model validation. Osborne and MacNish [22] present a method whose objective is to eliminate ambiguity in NL requirements. The problems associated with processing unrestricted NL are established and the impact of multiple word senses for terms used in a requirements document is reduced by using a Controlled Language (CL). Ambriola and Gervasi [3] developed a prototype to produce interactively conceptual models of NL requirements using a domain dictionary and a set of fuzzy-logic rules to match a NL construction. Among the most recent projects, Overmyer, Lavoie and Rambow [24] present an interactive method and a prototype, LIDA, to provide linguistic assistance in producing a subset of UML results. However, the text analysis remains in good part a manual process. It provides reliable tools for the user to analyse texts but it does not analyse text itself. Another research area highly connected both with the conceptual modelling and NLP is investigation of the quality of requirements' language. In particular, some authors focused on the support of writing requirements [2], [30], [23], [17].

In this context, NL-OOPS presents a higher degree of automation, because it is based on a large NLP System [14], that made it possible to produce completely automatically a draft of a conceptual model starting from narrative text in unrestricted NL (English) [16]. NL-OOPS is the tool used in the experiments and is described in the next section.

3. The NL-OOPS Tool

To develop 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 NL 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 NL text. Moreover, such systems can cope with ambiguities in syntax, semantics, pragmatics, or discourse. Systems of this type are much more

complex, require further research, and have a limited scope compared to those in the first category.

NL-OOPS is an NLP-based CASE prototype. It is a general-purpose system that was not designed with any specific purpose or domain in mind. It was founded on LOLITA (Large-scale Object-based Language Interactor, Translator and Analyser) NLP system, which includes all the functions for analysis of NL: morphology, parsing with respect to a 1500-rule grammar, semantic and pragmatic analysis, inference, and generation [13]. The knowledge base of the system – SemNet – consists of a kind of conceptual graph, which contains about 150,000 nodes. Thus LOLITA is among the largest implemented NLP systems. 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. Consequently, the output of the tool contains most of the information from the original text. NL-OOPS implements an algorithm for the 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. In particular, the algorithm for identifying classes and associations is based on two phases [15]. Figure 1 shows the NL-OOPS's interface, which consists of three frames. The top right frame contains the text being analysed for the SoftCom case used for the experiments. The left frame gives a partial 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 modelling module, the third frame, bottom right, contains a version of the class model. 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 or a Java source file; 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 for instance, the hierarchies in which it is involved.

4. The Experiments

Our main experimental hypothesis was that if model development is supported by a NLP-based tool, then the quality of the domain class model is higher and the design productivity increases. So, the goal of the experiments was firstly to confirm or refute this assumption and then to identify the features and the linguistic tasks of an NLP-based CASE system that are more relevant to the design of an effective CASE tool.

4.1 Realization of the experiments

In each experiment, we assigned a problem, a software requirements document, to the participants and we asked them to develop in a given time a class domain model,

identifying classes, associations, multiplicity, attributes and methods. Half of the participants were supported by the NL-OOPS tool. They were also given some training in the use of NL-OOPS. The training focused on the functionalities that support the identification of classes and in particular: how to change the threshold for the algorithm to produce models at different levels of detail; how to see the list of candidate classes, and how to use the nodes browser to navigate the knowledge base. The chosen class model could then be deployed in a java file, which was reverse engineered into PoseidonCE[2] or Rational Rose[3] class diagrams. Both these tools create the list of classes and the analyst has only to drag the classes in the diagram from the source list and then to check and complete the diagram. Before the experiment, we administered a short questionnaire to assess the experience of the participants.

Fig. 1. The NL-OOPS interface

To compare the results of the experiments, we used the same requirements text in all the experiments. In particular, the text was adapted from a case named Softcom [25]. It deals with a problem that requires some familiarity with judged sports, e.g., gymnastics or diving. The language is quite simple, but also realistic in that it contains all the typical features for requirements: use of the passive voice, etc. The first two experiments involved couples of analysts; the first preliminary experiment focused on the quality of the models obtained with and without NL-OOPS; in the second, we asked the participants to save the diagrams at fixed time intervals, to obtain data also about productivity. In the third experiment, to simplify the context of the experiment, each analyst worked alone; to this end, each was asked to develop two models for two different problem statements, one with the tool and one without it. The second case study named Library [11] had a level of difficulty similar to that of the Softcom case. Both the texts are given in the Appendix. For the first two experiments participants were undergraduate students and their competence in object-oriented conceptual modelling was comparable to that of junior analysts. For the last experiment, participants were PhD students, with higher competence. Before analysing the results of the experiments, we present the classes suggested by the tool

[2] Gentleware: http://www.gentleware.com
[3] IBM – Rational Software: www.rational.com

with different thresholds for both Softcom in Table 1 and Library in Table 2. These classes constitute the main input for the analysts working with NL-OOPS. To interpret the results we refer to the class models proposed with the problem sources [25], [11]. The names of the classes in the reference models are in bold. This choice was made to minimize the subjectivity in the evaluation of the models produced by the participants. We calculated recall, precision, and the F-measure to evaluate the performance for the class identification task.[4]

Table 1. Classes identified by NL-OOPS: SoftCom case (Words in parenthesis correspond to the actual concepts in SemNet)

NLOOPS-1 (12)	NL-OOPS-2 (10)	NL-OOPS-3 (5)	Reference classes (11)
Competition			**Competition**
Competition			
Competitor	**Competitor**	**Competitor**	**Competitor**
Entity (worker)	Entity (worker)		
Entity (announcer)	Entity (announcer)		
			Figure (styles, routines)
Group	Group	Group	
High	High		
Judge	**Judge**	**Judge**	**Judge**
			League
Meeting	**Meeting**		**Meeting**
Number	Number	Number	
Score	**Score**	**Score**	**Score**
			Season
			Station
			Team
			Trial
Softcom	Softcom		
R=45.5%; P=41.7%, F-measure =43.5%	R=36.4%; P=40.0%, F-measure = 38.1%	R=27.3%; P=60.0%, F-measure =37.5%	R_{avg}=36.4%;P_{avg}=47.2% F-measure$_{avg}$=39.7%

The models proposed by NL-OOPS do not contain classes such as season, station, team, and trial; they are instead present in the list of candidate classes. In the first two cases two classes are indicated: entity (worker), entity (announcer), corresponding to ambiguity in the text. In the first case, entity was introduced by the NLP system for the sentence "Working from stations, the judges can score many competitions": it cannot be automatically assumed that the subject of working is "judges". The second class results from an analysis of the sentence "In a particular competition, competitors receive a number which is announced and used to split them into groups", where the subject of announces is unknown. For all of these nodes, the use of the node browser of NL-OOPS makes it possible to go back to the original sentence to determine whether it gives the information necessary for the model.

[4] R (recall) counts the number of correct identified classes divided by total number of correct classes, P (precision) counts the number of correct identified classes divided by total number of classes, the F-measure combines R and P [29]

Table 2. Classes identified by NL-OOPS: Library case (Words in parenthesis correspond to the actual concepts in SemNet)

NLOOPS-1 (17)	NL-OOPS-2 (10)	NL-OOPS-3 (7)	Reference classes (7)
Book	**Book**	**Book**	**Book**
Borrower	**Borrower**	**Borrower**	**Borrower**
Person			
Copy			**Item (Copy-> Book Copy; Magazine Copy)***
Employee	Employee		
Entity (delete, update, create)	Entity (delete, update, create)	Entity (delete, update, create)	
Entity (cancel)			
Entity (register)			
Software_System	Software_System	Software_System	
It (Library)	It (Library)	It (Library)	
Entity (Library)			
Library	Library	Library	
Loan			**Loan**
Magazine	**Magazine**	**Magazine**	**Magazine**
Murderer (remove)			
Purchase			
Reservation	**Reservation**		**Reservation**
Thing (Superclass of Book and Magazine)	***Thing (Superclass of Book and Magazine)***		
Pair (Superclass of Book and Magazine)			
Title			**Title (->Book Title; Magazine Title)***
R = 100%, P= 41.2% F-measure = 58.4%	R=57.1-71.4%, P=40.0-50.0%** F-measure =47.0-58.8%	R=42.9%,P=42.9%, F-measure =42.9%	**R_{avg}=66.7%-71.4%;** **P_{avg}=41.4%-44.7%;** **F-measure$_{avg}$=49.4%-53.4%**

***** The hierarchies for "Copy" and "Title" represent two alternatives used to evaluate the models developed by the students.

****** The maximum values were calculated including the class "Thing".

For the Library case, the measures of the class identification task are higher than for the SoftCom case. However, the quality of the models produced by NL-OOPS is reduced by the presence of classes due to unresolved anaphoric references ("It", "Entity", "Pair"), or to ambiguity in the sentences. For example, the subject in sentence "The reservation is cancelled when the borrower check out the book or magazine or through an explicit cancelling procedure" is omitted. Another spurious class is "Murder", which was introduced by LOLITA as subject of an event related to the "remove"-action (due to the absence of domain knowledge).

4.2 Analysis of the results

Evaluating the quality of the models is a subjective process. The experience gained from the experiments and the analysis of the literature about quality of conceptual model,[5] helped us to define a schema to support their evaluation. The schema take into account the criteria related to both external and internal quality, evaluating:

[5] There are only few papers about this topic; see for example, [21], [28].

- The content of the model i.e. semantic quality: how much and how deep the model represents the problem.
- The form of the model i.e. syntactic quality: the proper and extensive use of UML notation.
- The quantity of identified items: class model, number of classes, attributes, operations, associations, and hierarchies.

Each of these criteria reflects a particular aspect of model quality. To evaluate them we assign a scale with a range from 0 (lowest mark) to 5 (highest mark).
The overall quality of the models is measured basing on mixed approach:

- the application of the quality schema
- the evaluation by two experts, that assessed the models as they usually do for their students' projects.

For the class identification task we calculated recall, precision, and the F-measure.

First Experiment. In the first preliminary experiment the group of twelve students was split into six subgroups [19]. Three groups worked with NL-OOPS. The length of the experiment was 90 minutes. For the six diagrams produced, two groups used PowerPoint, one used Excel, and all groups working without a tool chose Word. The results of the identification task are given in Table 3.

Table 3. Class identification

	1 tool	2 tool	3 tool	1	2	3
Recall	72.7%	54.5%	81.8%	100.0%	100.0%	81.8%
Precision	88.9%	66.7%	69.2%	78.6%	68.8%	90.0%
F-measure	80.0%	60.0%	75.0%	88.0%	81.5%	85.7%

To evaluate the overall quality of class diagrams, we asked two experts to mark and comment on the solutions proposed by the different groups (table 4).

Table 4. Experts' evaluation of overall model quality

Groups	Quality
1 tool	pretty good
2 tool	low
3 tool	good
4	pretty good
5	good
6	low

The experts judged the best model to be the one produced by group 5, in which two of the students had used UML for real projects. So, if on this basis, it was excluded, in order to have comparable level of groups, the best model would be one developed with the support of NL-OOPS. Considering these results with those in table 3, the tool seemed to have an inertial effect that on one hand led to the tacit acceptance of classes (e.g., group); on the other hand it resulted in the failure to introduce some indispensable classes (e.g., season, team). From the analysis of the feedbacks given by the participants some considerations emerged:

- those who used NL-OOPS would have preferred more training;
- each group that used NL-OOPS would prefer to have a tool to design the diagrams, while groups working without the tool did not voice this preference.

All these considerations were used for the realisation of the subsequent experiments.

Second Experiment. We repeated the experiment a year later involving ten students. In this experiment we divided the participants into five groups: two of them used NL-OOPS. The participants had access to Poseidon CE. We asked them to produce the model of the domain classes for the problem assigned. The length of the experiment was set for 1 hour. As we wanted to obtain also information regarding the productivity of conceptual modelling supported by linguistic tools, we asked the students to save every fifteen minutes screen shot of their model.

The performances related to the class identification task are summarised in the following table in which we report recall, precision and F-measure (table 5):

Table 5. Class identification

	15'	**30'**	**45'**	**60'**
Recall	45.5%	69.7%	75.7%	75.7%
tool	50.0%	59.1%	63.6%	81.8%
Precision	33.4%	71.3%	74.2%	76.4%
tool	52.6%	60.3%	72.9%	73.3%
F-measure	38.5%	66.2%	73.5%	74.7%
tool	50.9%	59.2%	67.3%	75.8%

Marks and comments on the overall quality made by two experts are given in table 6.

Table 6. Expert evaluation of overall model quality

Groups	**Quality**
1	low
2	pretty good
3	good
4 tool	pretty good
5 tool	low

The experts judged the best model to be the one produced by group 3 which participants (according to the questionnaire) used UML for real projects.

The application of the quality schema described in section 4.2 gives the following results (table 7):

Table 7. Overall Quality

	Content				**Form**				**Items**				**Total**			
Time	***15'***	***30'***	***45'***	***60'***	***15'***	***30'***	***45'***	***60'***	***15'***	***30'***	***45'***	***60'***	***15'***	***30'***	***45'***	***60'***
no tool	0.0	1.7	2.0	3.3	0.0	1.7	2.1	3.7	1.1	2.6	2.9	3.9	0.4	2.0	2.3	3.6
With tool	3.0	3.3	2.5	3.5	3.0	3.2	2.5	2.8	2.2	2.2	3.4	4.1	2.8	2.9	2.8	3.5

From this table we can see that in total the advantage of tool during modelling process remains substantial till the last time interval.

Third Experiment. In the third experiment we made some more changes. First of all, the participants worked individually. They had to deal with two different problem statements of comparable difficulty, with and without the NL-OOPS prototype. We set the length of the experiment to 20 minutes for each case. As in the second experiment we decide to collect progressive results, so we asked them to save the

model in an intermediate file after the first 10 minutes. The results for the class identification task are presented in the table 8. We should comment that even though the experts chose the requirement texts of comparable level, for the linguistic analysis there was the difference. For instance, Library case turned to be more difficult for the NLP system to understand because it contains many anaphors (table 1-2).

Table 8. Class identification

Parameter		Case*	10'		20'	
Recall		softcom	51.5%	49.6%	70.9%	62.2%
		library	47.6%		53.6%	
	tool	softcom	47.6%	59.5%	53.6%	62.5%
		library	71.4%		71.4%	
Precision		softcom	85.2%	75.9%	72.5%	62.0%
		library	66.7%		51.5%	
	tool	softcom	66.7%	58.3%	51.5%	50.8%
		library	50.0%		50.2%	
F-measure		softcom	64.2%	59.9%	71.7%	62.1%
		library	55.6%		52.5%	
	tool	softcom	55.6%	57.2%	52.5%	55.7%
		library	58.8%		59.0%	

We can assume here existence of some inertial effect because the users tend to keep all the candidate classes provided by NL-OOPS without getting rid of the fake classes ("it", "thing", "entity", etc.).

Table 9. Overall Quality

Parameter		Case	10'		20'	
Content		softcom	1.4	1.3	3.9	3.5
		library	1.1		3	
	tool	softcom	2.1	2.1	3.5	3.9
		library	2		4.2	
Form		softcom	1.5	1.4	3.8	3.4
		library	1.3		3	
	tool	softcom	2.4	2.5	4.2	4.5
		library	2.6		4.9	
Items		softcom	0.7	1	3	3.1
		library	1.2		3.2	
	tool	softcom	1.7	2	4.1	4.4
		library	2.2		4.7	
Total		softcom	1.2	1.2	3.6	3.3
		library	1.2		3.1	
	tool	softcom	2.1	2.2	3.9	4.3
		library	2.3		4.6	

Marks and comments on the overall quality made by two experts are given in table 10.

Table 10. Expert evaluation of overall model quality

Person	Evaluation			
Time	*10'*	***10' tool***	*20'*	***20' tool***
1	low	pretty good	low	good
2	*	-	pretty good	good
3	low	good	low	good
4	low	pretty good	low	good
5	low	pretty good	pretty good	good
6	-	-	pretty good	good
*7***	-	-	-	-
8	-	low	pretty good	low
9	low	good	low	good
10	low	-	low	low

*Grey cells correspond to Softcom

**Person 7 violated the rules of experiment, so the data cannot be considered as correct

In this experiment both quality and productivity had been improved thanks to the support of the NL-OOPS tool, even though the participants were pessimistic about using such kind of linguistic instrument.

5. Conclusions

The empirical results from the three experiments neither confirm nor refute the initial hypothesis of this paper that the quality of a domain class model is higher if its development is supported by a NLP system. There is some evidence, however, that model quality is better for NLP tool users early on during the modelling process. See the results of the third experiment at 10 minutes in Table 9. As to the impact of a NLP tool on productivity, the results of the experiments are uniformly inconclusive, but there is some evidence in Tables 7 and 9 that users work faster when supported by the tool. We interpret these results to mean that at initial steps the tool is helpful in speeding up the work, but by the end of the process, the advantage is lost because the users have to go into details of the text anyway to verify the correctness of list of classes and to derive other elements of the class diagrams.

The prototype was in some ways misused, as users were not able to take advantage of all the functionality provided by the system. Apparently, the groups working with the tool used only the initial draft of the class model and only part of the list of the candidate classes produced by the tool. A user did not go deep into the details of the semantic network constructed by the system and focused his/her attention only on the final list of the most probable classes candidates. To avoid this effect, NL-OOPS should have better integration of the different types of information it generates with the diagram visualization tools.

On the methodological level, the quality evaluation schema and the approach we adopted for the experiments described in this paper for the evaluation of NL-OOPS can be used to evaluate the output produced by any case tool designed to support the modelling process.

Other lessons learned from the experiments regarding features for an effective NLP-based CASE tool, include:

- The knowledge base produced by the linguistic analysis must be presentable in a user-understandable form;
- The most and least probable class and relationship candidates should be highlighted, to help the user modify the final model, either by extending it with other classes or by deleting irrelevant ones;
- The tool should be interactive to allow the analyst to resolve ambiguities and reflect these changes in the semantic representation immediately.

In general terms, the experiments confirm that, given the state of the art for NLP systems, heavyweight tools are not effective in supporting conceptual model construction. Instead, it makes sense to adopt lightweight linguistic tools that can be tailored to particular linguistic analysis tasks and scale up. Moreover, linguistic analysis may be more useful for large textual documents that need to be analysed quickly (but not necessarily very accurately), rather than short documents that need to be analysed carefully. We will be focusing our future research towards this direction.

References

1. Abbott R., "Program Design by Informal English Descriptions". *Communications of the ACM*, 26 (11): 882-894, 1983
2. Aguilera C., Berry D. M. "The Use of a Repeated Phrase Finder in Requirements Extraction". *Journal of Systems and Software,* Vol. 13 p.209-230, 1990
3. Ambriola V., Gervasi V. "An Environment for Cooperative Constraction of Natural-Language Requirements Bases", *In Proc. 8th Conference on SWE Environments,* IEEE, 1997
4. Booch G., "Object Oriented Development". *IEEE Transactions on Software Engineering*, Vol. SE-12, N. 2, p.211-221, 1986
5. Burg, J.F.M., Linguistic Instruments in Requirements Engineering, IOS Press, 1996
6. Cauvet C., Proix C., Rolland C., "ALECSI: An Expert System for Requirements Engineering*", Conf. on Advanced Information Systems Engineering (CAiSE)*, Interlaken, CH, p.31-49, 1991
7. Cockburn A., "Using Natural Language as a Metaphoric Basis for Object-Oriented Modelling and Programming". IBM Technical Report, TR-36.0002, 1992
8. Cockburn A., "Using natural language as a metaphoric base for OO". *In Proc. Conf. on OO Programming Systems Languages and Applications Archive*, Vancouver, p.187-189, 1993
9. Cordes D., CarverD., "An Object-Based Requirements Modelling Method"*, Journal of the American Society for Information Science*, 43 (1): 62-71, 1992
10. Dunn L., Orlowska M. A natural language interpreter for construction of conceptual schemas. *In CAiSE'90, 2nd Conf. on Advanced Information Systems Engineering*, LNCS 436, p. 371–386, Springer-Verlag, 1990
12. Eriksson H-E., Penker M., *UML Toolkit*, John Wiley, New York, 1998
13. Ingalls Daniel H. H., "The Smalltalk-76 programming system design and implementation". *In Proc. 5th ACM SIGACT-SIGPLAN Symp. on Principles of programming languages*, Tucson, Arizona, Jan. 23-25, p.9-16, 1978
14. Garigliano R., Urbanowicz A., Nettleton D.J., "Description of the LOLITA system as Used in MUC 7". *In Proc. Conf. MUC 7*, 1997
15. Goldin L., Berry D. M.: "AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation" *In Proc. First International Conference on Requirements Engineering* Colorado Springs, CO: IEEE Computer Society, 1994.

16. Mich L., “NL-OOPS: From Natural Language to Object Oriented Requirements using the Natural Language Processing System LOLITA”, *Journal of Natural Language Engineering,* Cambridge University Press, 2 (2): 161-187, 1996
17. Mich L., Garigliano R., “Ambiguity measures in Requirements Engineering”, in *Proc. International Conference on Software - Theory and Practice - ICS2000, 16th IFIP World Computer Congress,* Beijing, pp. 39-48, 21-25 Aug. 2000
18. Mich, L. and Garigliano, R. “NL-OOPS: A Requirements Analysis tool based on Natural Language Processing”. *In Proc. 3rd Int. Conf. On Data Mining 2002*, Bologna, Sep. 25-27, 2002, p.321-330
19. Mich L., Mylopoulos J., Zeni N., “Improving the Quality of Conceptual Models with NLP Tools: An Experiment”. *Tech. Report DIT*, University of Trento, 2002
20. Mich L., Franch M., Novi Inverardi P.L., “Requirements Analysis using linguistic tools: Results of an On-line Survey”, *Journal of Requirements Engineering*, Springer-Verlag, online Oct. 2003
21. Moody D.L., Shanks G.G. “What Makes a Good Data Model? A Framework for Evaluating and Improving the Quality of ER Models”. *Australian Computer Journal* 30(3): 97-110, 1998
22. Osborne M., MacNish C.K., “Processing natural language software requirement specifications”. *In Proc2nd IEEE Int. Conf. on Requir. Engineering (ICRE'96)*, IEEE Press, p. 229-236, 1996
23. Fabbrini F., Fusani M., Gnesi S., Lami G. “Quality Evaluation of Software Requirements Specifications”, *In Proc.of the 13th International SW Quality Week Conference,* 2000
24. Overmyer S.P., Lavoie B., Rambow O., “Conceptual Modelling through Linguistic Analysis Using LIDA”. *In Proc. 23rd Int. Conf. on SW engineering (ICSE 2001)*, Jul. 2001, p.401-410
25. Rolland C., Proix C., “A natural language approach for requirements engineering”. *In P. Loucopoulos (ed), Advanced Information Systems Engineering*, LNCS. Springer-Verlag, 593: 257-277, 1992
26. Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., *Object-Oriented Modelling and Design*, Prentice-Hall, 1991
27. Saeki M., Horai H., Enomoto H., “Software Development Process from Natural Language Specification”. *In Proc. 11th Int. Conf. on SW Engineering*, Pittsburgh, PE, Mar. 1989, p.64-73.
28. Teeuw W.B., Van den Berg H., “On the Quality of Conceptual Models”. *In Proc. Work. Behavioral models and design transformations - ER'97*, Los Angeles, CA, USA, Nov. 6-7, 1997, http://osm7.cs.byu.edu/ER97/workshop4/tvdb.html
29. van Rijsbergen C.J., *Information Retrieval*, Butterworths, 1979
30. Wilson, W., Rosenberg, L., Hyatt, L., "Automated Quality Analysis of Natural Language Requirement Specifications", *In Proc. of the 14th Pacific Northwest SW Quality Conf.*, 1996

Appendix

Softcom 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 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. 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.

Library Problem statement
A software system to support a library is to be developed. A library lends books and magazines to borrowers. These borrowers, books and magazines are registered in the system. A library handles the purchase of new titles for the library. Popular titles are bought in multiple copies. Old books and magazines are removed when they are too old or in poor conditions. The librarian in an employee of the library who interacts with the borrowers and whose work is supported by the system. A borrower can reserve a book or a magazine that is not currently available in the library. So that, when it is returned or purchased by the library, that person is notified. The reservation is cancelled when the borrower check out the book or magazine or through an explicit cancelling procedure. The library can easily manage the information about the books. It can create, update or delete the information. The information concerns the titles, the borrowers, the loans and the reservations. The system can run on all popular technical environments such as Windows, UNIX. It has a modern graphical user interface. The system is also easy to extend with new functionality.