NL-OOPS: a requirements analysis tool based on natural language processing

L. Mich[1], R. Garigliano[2]
[1]*Department of Information and Telecommunication Technologies, University of Trento, Italy.*
[2]*SoulSim Ltd, Unit 3M - Mountjoy Research Centre, Durham, United Kingdom.*

Abstract

The main goal of the NL-OOPS (acronym for Natural Language – Object-Oriented Production System) project is to develop a tool supporting object-oriented analysis using natural language (NL) processing. Requirements documents are analysed with LOLITA, a large-scale NL processing system, developed at Durham University and 3F Ltd. Both, the knowledge in the documents and that already stored in the knowledge base of LOLITA can then be used to produce requirements models at different levels of detail. Object oriented modelling is based on a two-phase algorithm for the identification of classes and associations. Moreover, the latest version of NL-OOPS supports traceability between the original input texts, their representation in LOLITA and the final models. To illustrate the main features and the performance of the tools we refer to the Automated Teller Machine study case described by the authors of the Object Modeling Technique.

1. Introduction

The NL-OOPS project started in 1994 as a collaboration between the University of Trento (I) and the University of Durham (UK). The main goal of this project is to develop a CASE tool supporting requirements analysis by generating object oriented conceptual models from requirements documents in NL. Many authors have proposed the idea of extracting object from NL documents, starting from Abbot in 1983, and for entity-relationship diagrams it is older still. Our approach is based on the consideration that requirements are often written in unrestricted

NL and in many cases it is impossible, or likely to produce counter effects, to impose customer restrictions on the language used. Given the complexity of the task, a CASE tool supporting NL requirements analysis, demands a "real" Natural Language Processing System (NLPS). Of course, according to a sound engineering approach, we have to assume that whenever more formalised or structured documents are given, it is advisable to use less "sophisticated" NLP system (see for example [3] or [4]). A review of the projects related to the development of CASE tools using NL processing is given in [9]. As we do not impose restriction on the vocabulary or the grammar, to process requirements documents in current English we use LOLITA, a very sophisticated NLPS. In this way, the knowledge in the documents is stored in the knowledge base of LOLITA, adding new nodes to its semantic network. All these nodes can then be used to produce the requirements models. The core of the tool implements an algorithm for the extraction of classes and associations. Moreover, the latest version of NL-OOPS support traceability between the original input texts, their representation in the knowledge base of LOLITA and the class models. To exemplify the performance of NL-OOPS we refer to the Automated Teller Machine requirements text described by the authors of the Object Modeling Technique [13].

2. The Natural Language Processing System

The system we are using is LOLITA, Large-scale Object-based Linguistic Interactor, Translator and Analyser. In this section we present the main features of LOLITA as regards the NL-OOPS project. LOLITA has been under development since the mid 80s' at Durham University. It has been used for many different applications and took part to the last two editions of the ARPA Message Understanding Competition (MUC) [11]. LOLITA has been built according to the principle of the NL Engineering [5] and many of its features are relevant for the NL-OOPS project. We enumerate here the most important of them. In particular, as it concerns the feasibility of a requirements analysis tool accepting in input NL texts, we have that in the NLP system LOLITA:

Knowledge is represented in a kind of conceptual graph [14, 6], independently from the superficial linguistic structures, i.e. independent from the grammatical form of NL requirements. This overcomes the problem of the incomplete isomorphism between syntax and semantics. In fact, an approach looking for nouns and verbs to identify classes and operations, respectively, would fail to consider that "any nouns can be verbed, and any verbs can be nouned" [2]. Compare, for example, the sentences, "John *kissed* Mary" - "John gave a *kiss* to Mary" and "A guest *reserve* a room ..." – "A guest make a *reservation* ..." (see [9]).

Requirements documents are morphologically, syntactically, semantically and pragmatically processed. It means that information in the requirements documents can be added to the knowledge base of LOLITA and used for the requirements modelling. Thanks to all these analysis levels, LOLITA is able to tackle with the complexities of free language. For example, it can infer that the sentence "Banks are open on weekdays" is semantically equivalent to the sentence "Banks are closed on Sundays". Another critical point for the correct disambiguation of NL sentences is the quantification problem. In the sentences cited in the previous paragraph, we could use "guests" instead of "a guest" and both stand for a set and not for an element.

The conceptual graph, called SemNet, is quite large compared to those of similar systems and contains more than 100K connected nodes. It has already been merged with WordNet (http://www.cogsci.princeton.edu/~wn/). The knowledge in SemNet has been used to support object oriented modelling for the NL-OOPS project.

LOLITA is able to automatically analyse about 90% of sentences. The level of accuracy depends on the quality of the texts input and also on the length of the sentences, as problems arise for sentences with more than 40 words. It follows that usually the output of the requirements documents processing contains most of the information in the original texts, that can be used for the modelling activities of NL-OOPS. These performances are going to improve even more with the fulfilment of the reengineering of LOLITA that recently has been undergone to a conversion for Windows.

The semantic network of LOLITA contains two kinds of nodes: entity nodes and event nodes. Event nodes represent complex relationships between concepts that cannot be represented using an arc. Each node has a set of controls, some of which are very important for requirements modelling in NL-OOPS. For example, the *rank* control gives quantification information: *universal* for general sets, *individual* for anonymous instances of a concept, *named individual* for named instances. Also very important is the rank *family*, which is used to classify nodes into the semantic and pragmatic group to which they belong. Values for family are *living*, *human*, *human organisation*, *inanimate*, *man-made*, *inanimate*. Critical for this tool is the classification of the event nodes. In fact, there are four categories of nodes – static, cyclic, dynamic, and instantaneous. For example, we have a static event with the action "to own something", a cyclic event in "to manage a company", a dynamic event for "to run a race" and an instantaneous event for "to win a race". As the class model describes the static structure of a system, our class identification algorithm exploits this classification.

Each node in SemNet has an identifier and event nodes have a frame-like structure representing their components: subject, action, object (if the action is transitive), source, date, etc. In this section we have reported the main issues about LOLITA as a core for a requirements modelling. Given its, there were many others interesting aspects. However, due to space reasons we do not go into the linguistic aspects further, for which reader can refer to [6,11]. Now we can describe the NL-OOPS.

3. The NL - Object Oriented Production System

The main goal of the NL-OOPS project is to develop a tool supporting NL requirements analysis. A full linguistic analysis of the documents is obtained using LOLITA. In this way we have an intermediate knowledge representation, which can be searched for object oriented modelling. Given the fine granularity of this representation, and taking into account that not all the information in the requirements texts is important (you can have irrelevant or redundant information), we have that the object oriented analysis algorithm has to filter nodes in LOLITA's SemNet to identify classes and associations for the requirements models.

In the first step of the model generation process, the requirements texts are analysed by LOLITA, which add new nodes in its semantic network to represent the knowledge in the documents. All the nodes, both, new and old, used by LOLITA to process the requirements are the main input to NL-OOPS. The class models give the main output. The user can interactively change some thresholds to obtain models at different levels of details. This is possible thanks to the hierarchies in SemNet. The tool provides some functions that allow the analyst to browse the semantic network at different steps of the processing. Traceability functions are provided between, both, final classes in the requirements models and nodes in the semantic network, and the original text.

3.1 The object oriented analysis algorithm

The Object-Oriented modelling activity is based on an algorithm for the extraction of the objects and their associations. The abstract structures of the algorithm implemented in NL-OOPS are shown in figure 1.

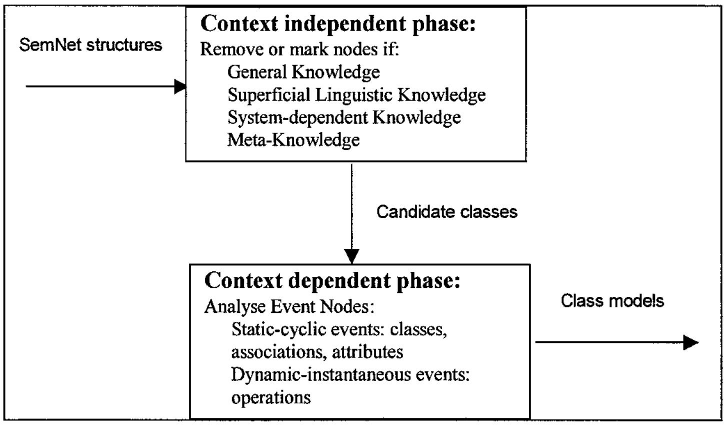

SemNet structures
Context independent phase:
Remove or mark nodes if:
General Knowledge
Superficial Linguistic Knowledge
System-dependent Knowledge
Meta-Knowledge
Candidate classes
Context dependent phase:
Analyse Event Nodes:
Static-cyclic events: classes, associations, attributes
Dynamic-instantaneous events: operations
Class models

Figure 1: The object oriented analysis algorithm

In the first phase - context independent - a list of candidate classes is produced. The second phase performs a context-dependent analysis and uses LOLITA's events classification to extract classes, associations, attributes (or values of attributes) and operations (or arguments of operations) from the list obtained in the first part of the algorithm. From a conceptual point of view, we have that:

– The input of the algorithm is given by the structures of SemNet used to process the NL requirements.
– The steps in the context independent phase are genuine filters, to eliminate those nodes that are not useful for a domain analysis. For example, nodes used by LOLITA to resolve anaphoric references or to identify proper nouns, etc. The output of this phase is a list of object nodes, which are the candidate classes for the class model.
– The second phase of the algorithm exploits the classification of the events in LOLITA's SemNet. Candidate classes become final classes in the class model if they are involved in a given number of events. On the one hand, the static nature of the class model requires static or cyclic events related to the classes, on the other hand instantaneous or dynamic events are used to disambiguate requirements that contain useful information.
– Changing the thresholds for the events in the second phase of the algorithm make it possible to have different versions of the class model. Due to the set-based nature of SemNet, candidate classes are hierarchically connected and different conditions for the events analysis allow to see and use also nodes previously stored in SemNet.

As it is the core of NL-OOPS, it is worth going into the criteria used to design the modelling algorithm.

In the **context independent phase** we have 19 rules, 3 of which are consistence rules. These rules are characterised by different levels of complexity, both, conceptual and computational. Most of the rules are in the General Knowledge group, where we have to remove:
– Nodes at the highest levels of hierarchies (e.g., *groups*, *something*, *things*): this rule is based on the assumption that these nodes do not give useful information about the problem domain. Its implementation would require an information measure [8]. In the current version we delete old, nodes permanently stored in SemNet, which are used for generalisations to add new nodes.
– Nodes used to process spatial knowledge, such as nodes used to specify direction, location, etc. With this criteria we are not cancelling spatial information, but nodes used by LOLITA to process this kind of knowledge. For example, to disambiguate the sentence "The hotel is by the sea", LOLITA uses an event node position, to specify the kind of position (by), and two more events to represent of the sea and of the hotel.
– Nodes used to process temporal knowledge: this rule is like the previous one. LOLITA has to correctly interpret the actual time, the time of the action and the reference time (for the sentence "I would have liked to have gone to the cinema" is in the past).

Then we have to mark event nodes. In this way we can separate entity nodes to build a candidate classes list and to identify information that is useful for the second phase of the algorithm. As SemNet contains both semantic and lexical information, in this step we can use, e.g., the control type *attribute*, *relation*, etc. to mark adjective, numeric information, adverbs, etc.

The second group of rules, related to superficial linguistic knowledge, allows us to eliminate nodes like those used to resolve plural anaphoric references. In fact, if we have two or more subjects for an event node, LOLITA creates a *pair* or *set* node that allows to correctly disambiguate, e.g. *they* in "John went to the supermarket with Mary. *They* both (…)". Another kind of nodes to be deleted is most of the *internal event* nodes. They are used for example to manage expressions like "information system", "car park", "baby oil", etc. The implementation of these rules is straightforward.

Also rules of the third group, delete system dependent nodes, are quite simple. An example is given by nodes used to process proper nouns. In fact, LOLITA creates a set of all the elements with a same name and the corresponding node can then be deleted. Meta-knowledge nodes can give useful information to guide requirements modelling, but they cannot be part of a requirements model. E.g., for the event in figure 2 we have a link source "Roberto": it means that the source of the information is Roberto, so for further explanation we could ask Roberto. The source is even more important in the presence of contradictions or inconsistencies. Another kind of meta-knowledge nodes are those connected with the goals of the applications under development. The criterion to identify them is the following: check for events having status hypothesis and involving nodes instantiated independently on the use of the information system. Each node in SemNet has a status control, which value is hypothesis for epistemic verbs. To discriminate "nodes instantiated independently on the use of the information system" is not easy, so the current version of the tool check the value for status and let the analyst the possibility to dis-activate or not this rule. Events marked goals are then "candidate" goals for the system. To eliminate isolated nodes some consistency rules are then applied, for example, to delete event nodes which subject or object has been cancelled.

The **context dependent phase** is uses LOLITA's event classification to identify classes and associations. One point to stress here is that we assume that the temporal framework used by LOLITA to classify the events and that involved in the requirements documents are coincident. For the cases so far analysed this hypothesis was satisfied, but for more complex projects, it has to be checked. For each candidate class we have to check all the events in which it is involved: to be a final classes it has to be subject or object of an established number of events both, static-cyclic and dynamic-instantaneous. In others words, the main assumption are that:

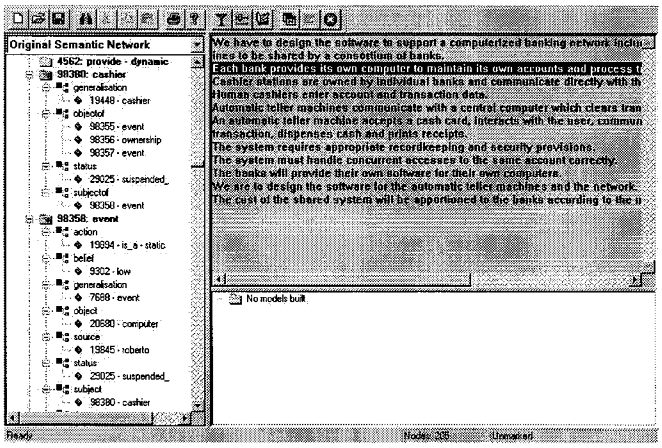
- Static and cyclic events are related to class associations or to attributes;
- Dynamic and instantaneous events are related to operations in the classes.

The first constraint is due to the static nature of class models, as static and cyclic events are related to class associations or to attribute. But also nodes involved in dynamic-cyclic events can be classes if the first condition that is already satisfied be the knowledge in SemNet. Changing the numbers of events for the two categories, the NL-OOPS creates class models at different level of detail. For example, we could have *three* and *four star hotel* as attribute in a class *hotel*, but with a lower thresholds, they could both be sub-classes in another version of the model. Candidate classes that do not enter in the current version of the class model are used to identify attributes, if they take part only in static or cyclic event, or operations, if involved only in dynamic or instantaneous events.

LOLITA's analysis helps to identify instances of classes, which are identified with the following rule: *subject* nodes with action *is_a* and rank *named individual* are instances of the class playing the role object in the event. If a class has at least one instance, an attribute name is added to it.

3.2 The prototype

In this section we illustrate the performance of the tool, its interface and its main functions referring to the ATM study case described in [13] (a demo of the tool is given at http://nl-oops.cs.unitn.it). In the following figure we can see the NL-OOPS's interface (figure 2). We have three frames. The top right frame contains the text of the requirements for the ATM example. The left frame gives a representation of the SemNet structures used by LOLITA for the analysis of the document. In this way the analyst has an interface to SemNet that is useful both to check the output of LOLITA for a given text and to see all the nodes created for its analysis. Old nodes already stored in SemNet are in yellow to be distinguished from new nodes that are two-tone. A first traceability function allows the analyst to check for which sentence a node has been created: selecting a node in SemNet, the corresponding sentence is highlighted.



```
Original Semantic Network
  4562: provide - dynamic
  98380: cashier
    generalisation
      19448 - cashier
    objectof
      98355 - event
      98356 - ownership
      98357 - event
    status
      29025 - suspended_
    subjectof
      98358 - event
  98358: event
    action
      19894 - is_a - static
    belief
      9302 - low
    generalisation
      7688 - event
    object
      20680 - computer
    source
      19845 - roberto
    status
      29025 - suspended_
    subject
      98380 - cashier
```

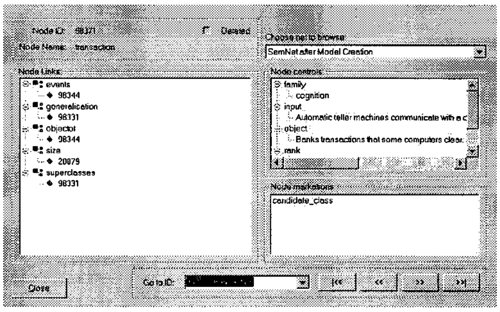
```
We have to design the software to support a computerized banking network inclu
ines to be shared by a consortium of banks.
Each bank provides its own computer to maintain its own accounts and process t
Cashier stations are owned by individual banks and communicate directly with th
Human cashiers enter account and transaction data.
Automatic teller machines communicate with a central computer which clears tran
An automatic teller machine accepts a cash card, interacts with the user, commun
transaction, dispenses cash and prints receipts.
The system requires appropriate recordkeeping and security provisions.
The system must handle concurrent accesses to the same account correctly.
The banks will provide their own software for their own computers.
We are to design the software for the automatic teller machines and the network.
The cost of the shared system will be apportioned to the banks according to the n
```

No models built

Ready Nodes: 205 Unmarked

Figure 2: NL-OOPS main window

NL-OOPS gives also a nodes browser, which shows the input sentence related to a chosen nodes, e.g., to a class in the final model (figure 3). After running the modelling module, the third frame, bottom right, contains a first version of the class model (figure 4). In the left frame we can then choose different views of the nodes. It is possible to see, for example, all the nodes deleted by a rule, the list of candidate classes, the list of final classes, etc. Statistical information about the analysis is also given by NL-OOPS. For the ATM example we have that the input text contains 161 words; LOLITA used 205 nodes, 39 of which have been cancelled in the context dependent phase. The first version of the class model has been created from a list of 94 candidate classes, organised in 28 hierarchies, and the final classes are 14 .



Node ID: 98371
Node Name: transaction
Choose net to browse: SemNet after Model Creation

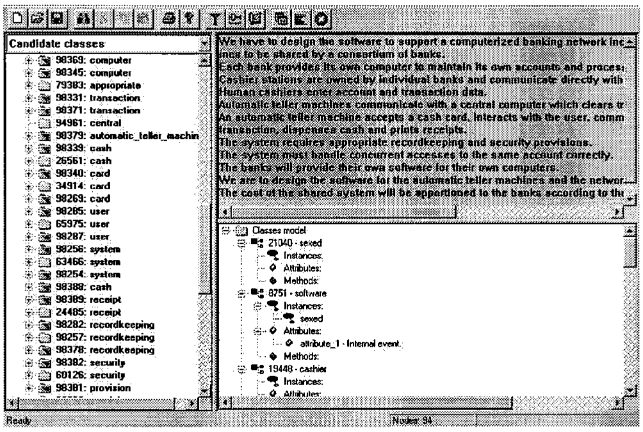
Node Links:
```
events
  98344
generalication
  98131
objectof
  98344
size
  20079
superclasses
  98331
```

Node controls:
```
family
  cognition
input
  Automatic teller machines communicate with a c
object
  Banks transactions that some computers clear.
rank
```

Node markations: candidate_class

Go to ID: [redacted]

Figure 3: NL-OOPS nodes browser


Candidate classes
```
98369: computer
98345: computer
79383: appropriate
98331: transaction
98371: transaction
94961: central
98379: automatic_teller_machin
98339: cash
26561: cash
98340: card
34914: card
98269: card
98285: user
65975: user
98287: user
98256: system
63466: system
98254: system
98388: cash
98389: receipt
24485: receipt
98282: recordkeeping
98257: recordkeeping
98378: recordkeeping
98382: security
60126: security
98381: provision
```

```
We have to design the software to support a computerized banking network inc
ines to be shared by a consortium of banks.
Each bank provides its own computer to maintain its own accounts and proces
Cashier stations are owned by individual banks and communicate directly with
Human cashiers enter account and transaction data.
Automatic teller machines communicate with a central computer which clears t
An automatic teller machine accepts a cash card, interacts with the user, comm
transaction, dispenses cash and prints receipts.
The system requires appropriate recordkeeping and security provisions.
The system must handle concurrent accesses to the same account correctly.
The banks will provide their own software for their own computers.
We are to design the software for the automatic teller machines and the networ
The cost of the shared system will be apportioned to the banks according to th
```

```
Classes model
  21040 - sexed
    Instances:
    Attributes:
    Methods:
  8751 - software
    Instances:
      sexed
    Attributes:
      attribute_1 - Internal event.
    Methods:
  19448 - cashier
    Instances:
    Attributes:
```

Ready   Nodes: 94

Figure 4: Candidate classes and final classes for the ATM example

The final classes identified automatically by NL-OOPS are the following:

21040: sexed
8751: software
19448: cashier
119967: automatic_teller_machine
11800: consortium
20680: computer *
67787: bank *
293686: account
64833: transaction
65975: user
24613: station (cashier station)
67789: bank *
63466: system
37710: cost

Classes in bold correspond to those given in [13], while there is one class missed: “cash card”. For the class “computer”, we do not have here the two subclasses – “bank computer” and “central computer”. A careful examination of these classes reveals that some of them are inadequately processed by LOLITA; for example, the node 21040, that has been used to disambiguate the first sentence in the requirements text or nodes 67789 and 67787 correspond to different sets of Banks. Evaluating this output with recall and precision measures, we have: recall = 80% and precision = 64%. Using the rules tuning function of NL-OOPS, the analyst can change the number of events in which a candidate class has to be involved in order to become a final class, or intervene on the application of a particular rule. For the ATM example, switching on the goals rule, we have that precision increase to 67%, but recall decrease to 72%.

3.3 Implementation issues

The current version of NL-OOPS is the third one and has been re-designed for Windows and implemented in C++ (about 10K lines of code). The first version of NL-OOPS runs in the UNIX operating system, and was developed in Tcl/tk to test the steps of the algorithm. A second version, more integrated with LOLITA, has been written in Haskell, the functional language used for LOLITA. As the NLP system has recently undergone and reengineering and a conversion to Windows in C++, the requirements analysis tool has also been redesigned. The new version of NL-OOPS is characterised by a higher level of interactivity, based mainly on graphs browsing and on rules tuning functionalities. Besides, it would also easier to integrate the tool with others CASE tool supporting lower development phases.

4. Conclusions and future developments

In this paper we have described NL-OOPS, a requirements analysis tool based on the NL processing LOLITA. The object-oriented modelling module implements an algorithm that filters entity and event nodes in the its knowledge base to identify classes and associations. To exemplify the performance of the tool we used the ATM case study, showing the behaviour of the functions used for traceability and to obtain different versions of the class models. The future developments of the project are related to (1) experimentation with different levels of support, as there is a trade-off between automatic modelling process and checking of the resulting models. A fully automated analysis requires a senior analyst to control the output; (2) the redesign and the conversion of LOLITA and the extension of its knowledge base This will improve the quality of the input for NL-OOPS. As regards the interface of NL-OOPS, we are working to improve its integration with the NLP system and to produce graphical diagrams of the class models based on UML notation. The current version of NL-OOPS will also be used as a discovery prototype to investigate the domain knowledge needed in real project.

References

[1] Ambriola V., Gervasi V., "Experiences with Domain-Based Parsing of Natural Language Requirements", in G. Friedl, H.C. Mayr (eds) Application of NL to Information Systems, OCG, 1999, 145-148.

[2] Booch G., "Object Oriented Analysis and Design with Application", Redwood City, CA, Benjamin/Cumming.

[3] Delisle S., Barker K., Biskri I., "Object-Oriented Analysis: Getting Help from Robust Computational Linguistic Tools, in G. Friedl, H.C. Mayr (eds) Application of Natural Language to Information Systems, OCG, 1999, 167-172.

[4] Fliedl G., Kop C., Mayr H.C., Mayerthaler W., Winkler C., "Linguistically Based Requirements Engineering – The NIBA-Project, in G. Friedl, H.C. Mayr (eds) Application of NL to Information Systems, OCG, 1999, 177-182.

[5] Garigliano R., Boguraev B., Tait J., "Editorial", Journal of Natural Language Engineering, Cambridge University Press 1 (1): 1-7

[6] Long D., Garigliano R., "Reasoning by Analogy and Causality: Model and Applications", Chichester, UK, Ellis Horwood, 1994.

[7] Mich L., Garigliano R. "The NL-OOPS project: Object-Oriented Modelling using the Natural Language Processing System LOLITA", in G. Friedl, H.C. Mayr (eds) Application of NL to Information Systems, OCG, 1999, 215-218.

[8] Mich L., Garigliano R. "Measuring Information in a Semantic Network", in Proc. 7th Int. Conf. IPMU'98, Paris, July 6-10, 1903-1904.

[9] Mich L., "NL-OOPS: From Natural Language to Object Oriented Requirements using the Natural Language Processing System LOLITA", Journal of NL Engineering, Cambridge Univ. Press, 2(2): 161-187, 1996.

[10] Mich L., Garigliano R. "A Linguistic Approach to the Development of Object Oriented Systems using the Natural Language System LOLITA", Int Symp. - ISOOMS'94, PA, I, Sep. 21-23, 1994, in E. Bertino, S. Urban (eds), Springer-Verlag, LNCS, 858: 371-386.

[11] Morgan R., Garigliano R., Callaghan P., Poria S., Smith M, Urbanowicz A., Collingham R., Costantino M., Cooper C. and the LOLITA Group, Description of the LOLITA System as used in MUC-6, in Proc. of the 6th ARPA Message Understanding Conf., Morgan Kaufmann, 1996.

[12] Nurcan S., Grosz G., Souveyet C., describing Business Processes with a guided Use Case Approach", in Proc. 10th Int. Conf. CAiSE'98, Springer-Verlag, 1998.

[13] Rumbaugh J., Blaha M., Eddy F., Premerlani W., Lorensen W., "Object Oriented Modeling and Design", Englewood Cliffs, NJ, Prentice Hall, 1991.

[14] Sowa J., "Conceptual Structures", Reading, MA, Addison Wesley, 1983.