Conceptual Modeling through Linguistic Analysis Using LIDA

Overmyer[1], Scott P
Massey University - Albany
Private Bag 102 904, NSMSC
Albany, New Zealand
S.P.Overmyer@massey.ac.nz

Lavoie, Benoit
CoGenTex, Inc.
840 Hanshaw Road
Ithaca, NY 14850, USA
benoit@cogentex.com

Rambow[2], Owen
ATT Labs-Research, B233
180 Park Ave, PO Box 971
Florham Park, NJ 07932, USA
rambow@research.att.com

Abstract

Despite the advantages that object technology can provide to the software development community and its customers, the fundamental problems associated with identifying objects, their attributes, and methods remain: it is a largely manual process driven by heuristics that analysts acquire through experience. While a number of methods exist for requirements development and specification, very few tools exist to assist analysts in making the transition from textual descriptions to other notations for object-oriented analysis and other conceptual models. In this paper we describe a methodology and a prototype tool, Linguistic assistant for Domain Analysis (LIDA), which provide linguistic assistance in the model development process. We first present our methodology to conceptual modeling through linguistic analysis. We give an overview of LIDA's functionality and present its technical design and the functionality of its components. We also provide a comparison of LIDA's functionality with that of other research prototypes. Finally, we present an example of how LIDA is used in a conceptual modeling task.

1. Introduction

Object-oriented analysis and design grows more popular every year. Books on the subject abound [4,7,9,13,16]. Traditional systems analysts and developers are working hard to gain the knowledge and expertise required to take advantage of this relatively new technology. Despite the advantages that object technology can provide to the software development community and its customers, the fundamental problems associated with identifying objects, their attributes, and methods remain. These tasks are largely manual processes driven by heuristics that analysts acquire through experience. The primary tools for object identification and refinement are pencil and paper, with the results being transferred to CASE tools after the analysis is largely completed. There are some software tools associated with some methods of object identification and refinement, such as the CRC card approach [2]; however, the CRC card approach is labor intensive, and difficult for an individual analyst to use.

Very often, an analyst is given a text document that describes the environment for which an information system is to be developed. This document might include business processes, user tasks, information about existing systems, and so forth. Information about system requirements and proposed system use appears in many forms, from rambling discourse on the operational concept of a proposed system, to loosely organized text descriptions of the business environment and the user's tasks, to highly structured step-by-step procedure descriptions. It is the job of the requirements analyst, regardless of development methodology, to understand, develop, and document this information in a form that can be analyzed by developers, and translated into a software design, and subsequently into code. Traditionally, systems analysts have developed the requirements and modeled them utilizing notations such as process models (e.g., data flow diagrams), data models (e.g., entity-relationship diagrams), flow charts, and plain text. For some time, task descriptions in the form of scenarios have also been employed on traditional analysis efforts as well.

The answer to these challenges that is offered by the object-oriented community is the Use Case, and its associated notations, or some variant of that idea. For example, in the Unified Modeling Language (UML), the notations offered are Use Case texts, Use Case Diagrams, and Activity Diagrams [3]. The widespread use of Use Cases as a basis for object-oriented analysis is apparent both in academic literature and in industrial practice on

[1] The contribution of this co-author on the work reported in this paper was done while affiliated with Drexel University.

[2] The contribution of this co-author on the work reported in this paper was done while affiliated with CoGenTex, Inc.

object-oriented development projects.

Still, the process for developing objects from Use Cases and other descriptions is largely manual, and difficult. While a number of methods exist for Use Case development and specification, very few tools exist to assist analysts in making the transition from text descriptions, use cases, or scenarios, to other notations for object-oriented analysis, such as class diagrams and activity diagrams. Without a methodology, and a tool to assist the analyst, it is often very difficult to identify classes, their attributes and methods, and their relationship to other classes in the problem domain.

Section 2 provides a methodology for class identification and elaboration. Section 3 describes LIDA, a tool providing linguistic assistance during the modeling process and used for the refinement of the methodology. Section 4 describes a scenario-based description of the application of the methodology using LIDA. Finally, Section 5 describes related work.

2. The methodology

There are a number of ways in which analysts approach the object-oriented analysis and design process, depending on the analyst's initial state of knowledge. This paper covers the situation in which an analyst is starting with a text domain description, operational concept document, use cases, or other text system and/or task descriptions, and attempts to identify model elements and their relationships.

Starting with a text description of the concept of operation for a proposed system (in any text-based format), the analyst typically reads the entire document, picks out or highlights salient nouns and verbs, then attempts to identify potential objects and methods. Previous work on conceptual modeling based on textual analysis [1,6] already suggested conventions using the part-of-speech of the words for identifying objects and methods; associating classes with nouns, relationships with verbs, and attributes with adjectives and prepositional phrases. Our practical experience [12] also indicates that conceptual modelers often find such conventions to be natural and practical.

Using a case tool such as Rational Rose (www.rational.com), an analyst might start to construct a model in UML, elaborating on it in an incremental manner. This capability of going back and forth between text and graphical model during development of a model can be very helpful, and can minimize errors before the model complexity obscures error states.

Most text documents that describe system functions contain many more nouns, verbs, adverbs, and adjectives than are useful in an object identification effort. These words are necessary in prose and conversation to fluently describe the context and manner in which a system is to be used. However, in requirements engineering parlance, many more words describe the "domain" in which the software is to reside than describe the "machine", or the software system itself. The sheer number of words must be reduced to a subset that is directly applicable to the development of a system using a system of classes. Seemingly irrelevant prose is useful, however, in that it helps define a context of use. In other words, knowing how the system is to be used helps the analyst identify which classes are relevant, and which objects in the environment are usefully identified as system classes. Prior to attempting to identify classes, the analyst should already have prepared a set of use cases or scenarios that represent the operational concept for the proposed system. The analyst should have this information in mind prior to using our methodology. Our methodology is consistent with the Unified Software Development Process, and LIDA uses UML as the preferred notation.

The general flow of the class identification process using LIDA is illustrated in Figure 1. The analyst first imports the document(s) to be analyzed and the software tags the parts-of-speech of the words. Next, as shown in the figure, the analyst first works the noun list marking relevant candidate classes, and iteratively removing those classes that, upon reflection, do not qualify as classes, or that are candidate attributes instead. When the candidate classes have been identified, the analyst moves to the adjectives list to identify candidate attributes of the classes. Proceeding down the activity diagram, the analyst then moves to the verb list to identify candidate methods and roles. Following this identification process, the analyst then proceeds to the LIDA Modeler (see Subsection 3.3) to graphically associate attributes, methods, and roles with the appropriate classes. The analyst may check the semantics of the resulting model by using Model Explainer (see Subsection 3.4). As the figure also shows, this process is iterative, and often results in a return to the class identification process, as the model is refined. Finally, the analyst models interaction and relationships between classes, again using an iterative process.

An example of using the methodology with LIDA is shown in Section 4. Bearing the general flow of activities in mind will help the user make sense of the example provided. This example represents only a single path through the methodology (one scenario) accomplished using LIDA, but we feel that the tool is sufficiently flexible to adapt to most established organizational standards and processes for object-oriented analysis and design. We base this opinion on the notion that at the beginning of class modeling activities, the basic elemental, cognitive tasks associated with model development are the same.

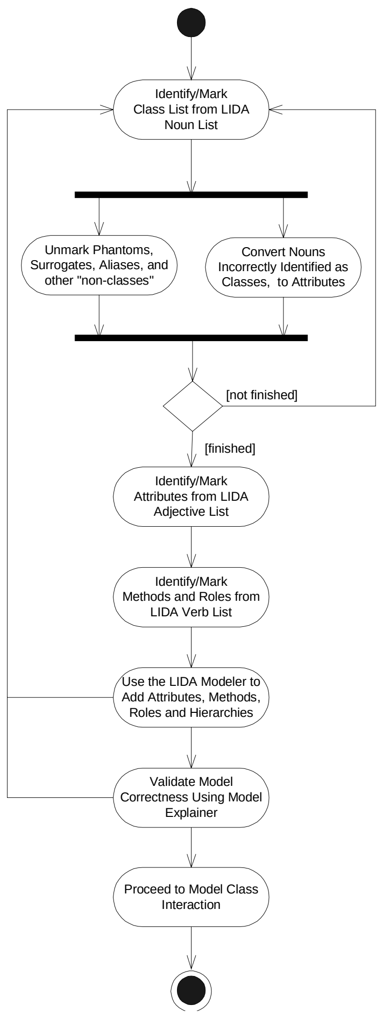

Identify/Mark Class List from LIDA Noun List
Unmark Phantoms, Surrogates, Aliases, and other "non-classes"
Convert Nouns Incorrectly Identified as Classes, to Attributes
[not finished]
[finished]
Identify/Mark Attributes from LIDA Adjective List
Identify/Mark Methods and Roles from LIDA Verb List
Use the LIDA Modeler to Add Attributes, Methods, Roles and Hierarchies
Validate Model Correctness Using Model Explainer
Proceed to Model Class Interaction

Figure 1. LIDA class identification & modeling process

The goal of this method is to utilize existing text descriptions of a problem domain, and from them, produce an initial conceptual class diagram with attributes, methods, and roles for export to an object-oriented CASE tool.

3. LIDA: A Linguistic assistant for Domain Analysis

3.1. Overview

LIDA (LInguistic assistant for Domain Analysis) helps analysts develop object-oriented models of a domain, using a subset of UML. In order to develop such models, the requirements analyst or knowledge engineer often needs to analyze large volumes of text from "legacy documents"— these might include user manuals of legacy systems, company policies, use cases, or transcripts of interviews with domain experts. LIDA facilitates this analysis by compiling a list of the words and multi-word terms in a document, and providing a graphical interface for the user to mark them as corresponding to elements of a model. It also lets the user validate models as they are created, through integration with ModelExplainer tool [12], which generates textual descriptions of a model.

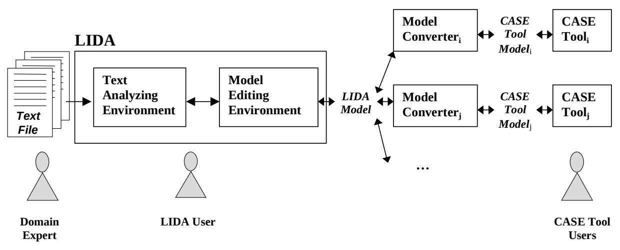


Figure 2. LIDA's overall system architecture

The LIDA prototype currently implemented has the following features:

- Domain-independent linguistic processing is used to group different forms of the same base word together, to determine part of speech (noun, verb, adjective), and to detect multi-word terms.
- The full text, word lists, and evolving UML model are displayed in parallel, letting the user compare different views.
- Words and multi-word terms can be assigned a type in the model (Class, Attribute, Role, etc.) with the click of a button. The corresponding strings are color-coded in the text display and graphically displayed in the diagram window.
- Words and multi-word terms can be sorted alphabetically, by frequency, by part of speech, or by assigned type.
- A context or "KWIC" (Key Words In Context) view displays only those sentences containing a chosen word or group of words.
- Hypertext descriptions of model can be obtained to help validate or document a model.
- Completed models can be exported for refinement within CASE tools, or can be imported from CASE tool for validation with texts within LIDA.

LIDA's overall system architecture is shown in Figure 2. LIDA consists of two main interface components: a Text Analyzing Environment and a Model Editing Environment.

The Text Analyzing Environment (described in further detail in Subsection 3.2) offers LIDA's users the functionality needed for analyzing the lexical content of text files that domain experts may have written for specific domains and assists them in the process of identification and marking of the lexical items corresponding to candidate model elements. Model elements marked in the Text Analyzing Environment propagate to the Model Editing Environment (described in further detail in Subsection 3.3) where they can be assembled in a class model. While building the model using the Model Editing Environment, the textual context of the model elements is directly accessible, and the addition of new elements to the model or the removal of elements from the model directly affect the marking of the text in the Text Analyzing Environment.

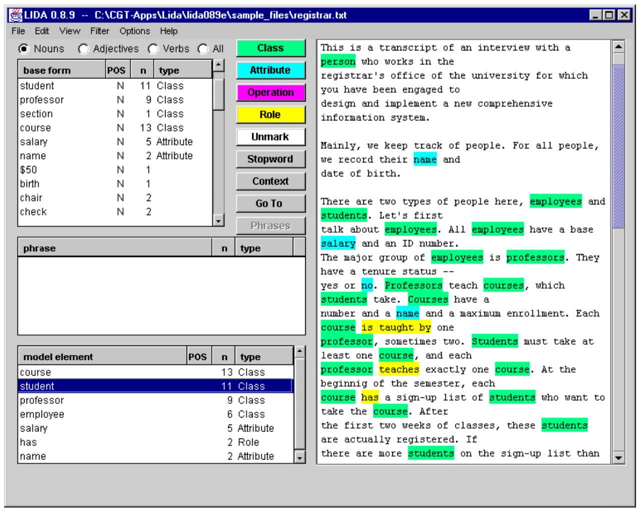
A model developed in LIDA can be saved as a file using a LIDA-specific file format. LIDA model files can either be reloaded into LIDA or converted and exported to a different modeling tool (an example of tool is given below) for further model refinement. When loaded into LIDA, LIDA models can be applied either to the texts that were originally used for the development of the model or to different texts. The lexical content of a text after loading a model is marked following the information found in the loaded model.

Export of LIDA models to modeling tools and import of models into LIDA is made possible by using converters customized for these modeling tools. The current implementation of LIDA supports export to and import from the Visio™ modeling tool (www.visio.com) where the model converter is developed using Visio™ API.

3.2. LIDA Text Analyzing Environment

The Text Analyzing Environment illustrated in Figure 3 represents LIDA's main component that provides its central functionality. The main functional features of the Text Analyzing Environment include:

- Reading text from file (RTF and ASCII formats are currently supported).
- Assigning part-of-speech to words using a broad coverage part-of-speech tagger, MXPOST [14], a software tool developed at the University of Pennsylvania (for example, deciding that *employee* is a noun).
- Retrieving base form of words, and counting word occurrences by their base form.
- Finding multi-word phrases for a given headword (for example, *temporary employee* for the headword *employee*).
- Finding user-supplied multi-word phrases.
- Allowing the user to mark a word or a phrase as a candidate model element; all the occurrences of the marked word or phrase are highlighted in the text in a color associated with a model element type.
- Retrieving textual context of marked words



base form	POS	n	type
student	N	11	Class
professor	N	9	Class
section	N	1	Class
course	N	13	Class
salary	N	5	Attribute
name	N	2	Attribute
$50	N	1	
birth	N	1	
chair	N	2	
check	N	2	

phrase	n	type

model element	POS	n	type
course		13	Class
student		11	Class
professor		9	Class
employee		6	Class
salary		5	Attribute
has		2	Role
name		2	Attribute

```
This is a transcript of an interview with a
person who works in the
registrar's office of the university for which
you have been engaged to
design and implement a new comprehensive
information system.

Mainly, we keep track of people. For all people,
we record their name and
date of birth.

There are two types of people here, employees and
students. Let's first
talk about employees. All employees have a base
salary and an ID number.
The major group of employees is professors. They
have a tenure status --
yes or no. Professors teach courses, which
students take. Courses have a
number and a name and a maximum enrollment. Each
course is taught by one
professor, sometimes two. Students must take at
least one course, and each
professor teaches exactly one course. At the
beginnig of the semester, each
course has a sign-up list of students who want to
take the course. After
the first two weeks of classes, these students
are actually registered. If
there are more students on the sign-up list than
```

Figure 3. LIDA's Text Analyzing Environment

LIDA thus provides reliable tools for the user to analyze texts, but it does not analyze texts itself. The current state-of-the-art in natural language processing does not allow for dependable deep semantic analysis or even for adequate syntactic analysis; we leave it up to the user to find important information in the texts, to notice inconsistent or incomplete information in the texts, and to resolve these inconsistencies and omissions in the modeling phase.

Subsection 4.1 describes a scenario illustrating how these features can be used in LIDA to assist the user identifying the model elements through text analysis.

3.3. LIDA Model Editing Environment

The Model Editing Environment (or Modeler) illustrated in Figure 4 offers the functionality needed to build a model from the candidate model elements marked in LIDA's Text Analyzing Environment. The main functional features of the Model Editing Environment include:

- Displaying lists of candidate model elements marked in LIDA's Text Analyzing Environment or added directly from the Model Editing Environment. In Figure 4, the candidate model elements are displayed in the four graphical tree structures and list structures appearing on the left side of the Window. The model element candidates added in LIDA's Text Analyzing Environment propagate to these structures, and conversely, new model elements added to these structures from the Model Editing Environment propagate to LIDA's Text Analyzing Environment, modifying the marking of the text. This bi-directional propagation of information between the

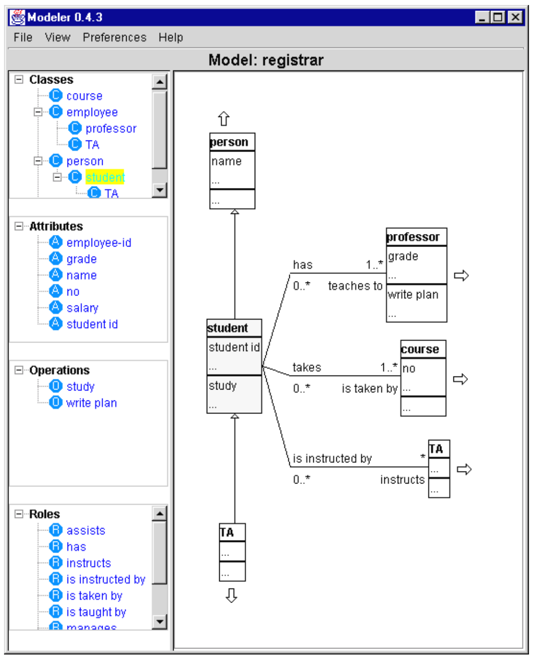


Figure 4. LIDA's Model Editing Environment

Text Analyzing Environment and the Model Editing Environment enables the developer to go back and forth between the text analysis process and the model building process.

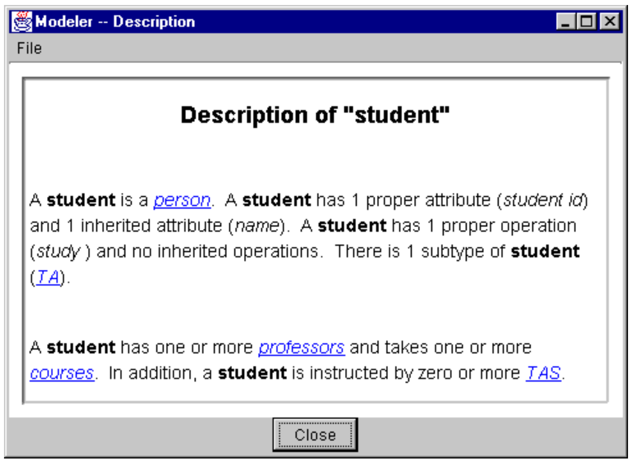
- Offering operations needed for combining model elements into a class diagram.
- Displaying textual contexts useful in the process of model building. (An example of textual context illustrated in Figure 7 will be presented in Subsection 4.1.)
- Generating textual descriptions of model elements to help documenting the model or validating the model with domain experts. (More details on the text descriptions can be found in Subsection 3.4.)

Subsection 4.2 describes a scenario illustrating how these features can be used in LIDA to assist the user associating the model elements into a class diagram.

3.4. LIDA text descriptions

LIDA is integrated with ModelExplainer [12], a tool used to generate hypertext descriptions for object models. Figure 5 illustrates an example of description for the class *student* based on the model shown in Figure 4.

The descriptions are generated from customizable text plans [11] set in the above example to include the following class information: superclasses, class attributes, class operations, subclasses and associations to other



Description of "student"

A **student** is a person. A **student** has 1 proper attribute (*student id*) and 1 inherited attribute (*name*). A **student** has 1 proper operation (*study*) and no inherited operations. There is 1 subtype of **student** (TA).

A **student** has one or more professors and takes one or more courses. In addition, a **student** is instructed by zero or more TAS.

Figure 5. LIDA's text description

classes. Hyperlinks generated with the descriptions allow the user to obtain additional descriptions.

The descriptions can be used for different purposes, including:

- Providing a textual support to the LIDA user for validating the model with domain experts who may not be familiar with the graphical notation used for the modeling.
- Providing a textual support to the LIDA user for documenting a model.
- Allowing the user to compare the generated text to the original document for validation.

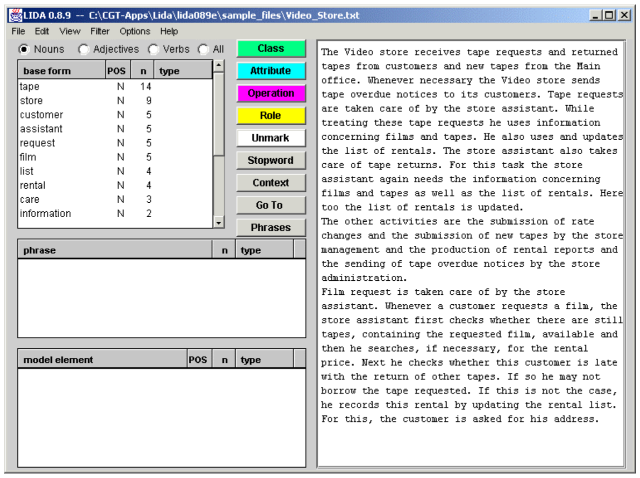
4. An example analysis using LIDA

4.1.Working with the text

First we open LIDA's Text Analyzing Environment, and import the text document to which the parts-of-speech tagger is applied (if not previously tagged). Figure 6 shows the default opening appearance of a text file after tagging: the text file appears in the right-hand window, and the parts-of-speech appear in the upper-left window, with the base form of the words displayed. (Only nouns are visible in Figure 6.)

Most methods recommend the identification of candidate classes first, suggesting that the most frequently occurring nouns are likely candidates. In order to arrange the nouns in the order of occurrence, we click on the "**n**" heading directly above the numbers of nouns. This results in a sort of the nouns (or, properly speaking, of their base forms) in descending order according to their frequency of occurrence in the document also as shown in Figure 6.

Examining the list of nouns, we find "tape" appearing 14 times in the document. Since this is a video rental


LIDA 0.8.9 -- C:\CGT-Apps\Lida\lida089e\sample_files\Video_Store.txt
File Edit View Filter Options Help
Nouns Adjectives Verbs All
Class
Attribute
Operation
Role
Unmark
Stopword
Context
Go To
Phrases

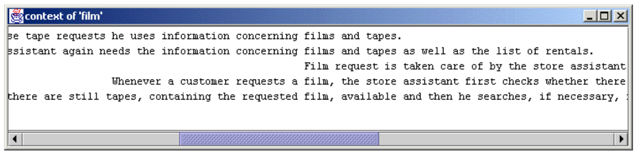
base form	POS	n	type
tape	N	14	
store	N	9	
customer	N	5	
assistant	N	5	
request	N	5	
film	N	5	
list	N	4	
rental	N	4	
care	N	3	
information	N	2	

phrase	n	type

model element	POS	n	type

```
The Video store receives tape requests and returned
tapes from customers and new tapes from the Main
office. Whenever necessary the Video store sends
tape overdue notices to its customers. Tape requests
are taken care of by the store assistant. While
treating these tape requests he uses information
concerning films and tapes. He also uses and updates
the list of rentals. The store assistant also takes
care of tape returns. For this task the store
assistant again needs the information concerning
films and tapes as well as the list of rentals. Here
too the list of rentals is updated.
The other activities are the submission of rate
changes and the submission of new tapes by the store
management and the production of rental reports and
the sending of tape overdue notices by the store
administration.
Film request is taken care of by the store
assistant. Whenever a customer requests a film, the
store assistant first checks whether there are still
tapes, containing the requested film, available and
then he searches, if necessary, for the rental
price. Next he checks whether this customer is late
with the return of other tapes. If so he may not
borrow the tape requested. If this is not the case,
he records this rental by updating the rental list.
For this, the customer is asked for his address.
```

Figure 6. Text Analyzing Environment with loaded text



```
se tape requests he uses information concerning films and tapes.
ssistant again needs the information concerning films and tapes as well as the list of rentals.
                                              Film request is taken care of by the store assistant
                Whenever a customer requests a film, the store assistant first checks whether there
there are still tapes, containing the requested film, available and then he searches, if necessary, 
```

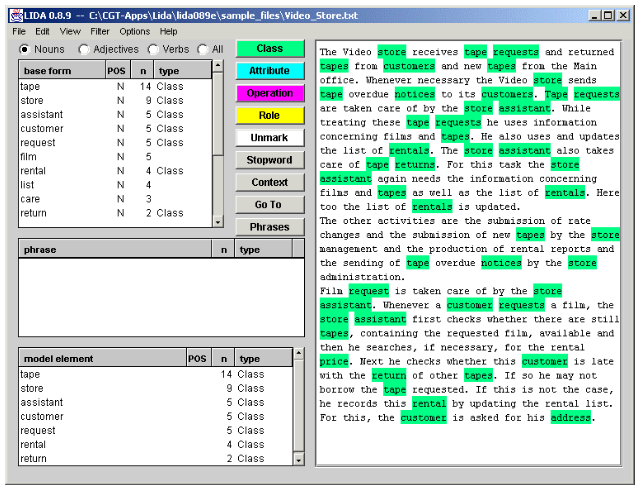
Figure 7. Context (or KWIC) window

store, and tapes are what are rented, this seems like a reasonable candidate for the first class. We first select the word "tape" in the list of nouns, and then click the **Class** button. LIDA highlights the word "tape" everywhere it occurs in the document (in its base form or in an inflected form such as “tapes”). The next most frequently occurring noun is "store", followed by "assistant", "customer" and "request". We highlight all of these in turn, and mark them as possible classes (or candidate classes).

The next word on the list is "film". We suspect that "film" and "tape" might be used to refer the same object (since we know that most video stores don't rent photographic film). In order to verify that this is the case, we highlight the word "film", and click on the **Context** button. This results in the display of the context (or KWIC) window as shown in Figure 7.

Notice that the Context function locates the word of interest, in this case "film", in the center of the window, with the rest of the sentence in which the word appears on either side of the word. As we read through the context, although we notice that there is a possible "containment" relationship indicated in the last sentence ("there are still tapes, **containing** the requested film"), we decide that this is irrelevant to our analysis and that tape and film refer to the same concept, so we do not mark "film" as a class. Ideally, we might mark "tape" and "film" as synonyms; however, LIDA does not currently support this operation.

The next noun in the list that looks like a reasonable


LIDA 0.8.9 -- C:\CGT-Apps\Lida\lida089e\sample_files\Video_Store.txt
File Edit View Filter Options Help
Nouns Adjectives Verbs All
Class
Attribute
Operation
Role
Unmark
Stopword
Context
Go To
Phrases

base form	POS	n	type
tape	N	14	Class
store	N	9	Class
assistant	N	5	Class
customer	N	5	Class
request	N	5	Class
film	N	5	
rental	N	4	Class
list	N	4	
care	N	3	
return	N	2	Class

phrase	n	type

model element	POS	n	type
tape		14	Class
store		9	Class
assistant		5	Class
customer		5	Class
request		5	Class
rental		4	Class
return		2	Class

```
The Video store receives tape requests and returned
tapes from customers and new tapes from the Main
office. Whenever necessary the Video store sends
tape overdue notices to its customers. Tape requests
are taken care of by the store assistant. While
treating these tape requests he uses information
concerning films and tapes. He also uses and updates
the list of rentals. The store assistant also takes
care of tape returns. For this task the store
assistant again needs the information concerning
films and tapes as well as the list of rentals. Here
too the list of rentals is updated.
The other activities are the submission of rate
changes and the submission of new tapes by the store
management and the production of rental reports and
the sending of tape overdue notices by the store
administration.
Film request is taken care of by the store
assistant. Whenever a customer requests a film, the
store assistant first checks whether there are still
tapes, containing the requested film, available and
then he searches, if necessary, for the rental
price. Next he checks whether this customer is late
with the return of other tapes. If so he may not
borrow the tape requested. If this is not the case,
he records this rental by updating the rental list.
For this, the customer is asked for his address.
```

Figure 8. Marking of candidate classes

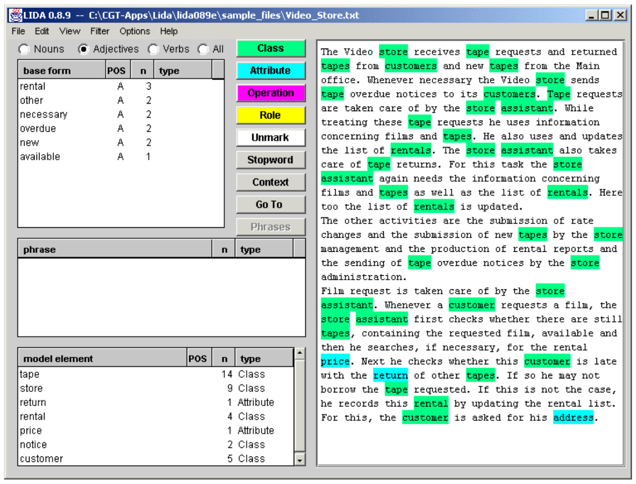
candidate is "rental", since we know that there is likely to be a rental transaction in the system, and that transactions are candidates for classes. Similarly, we tentatively mark "return", "notice", "address", "report", and "price" as candidate classes. Figure 8 shows the appearance of LIDA’s Text Analyzing Environment at this point. Current model elements are shown in the lower-left window. Note that it is also possible to highlight and mark words as model elements directly in the text window. This is very useful when giving the text a final read before moving from one step to the next.

Next, we review the list of model elements, and revise it according to our knowledge (and our customer's knowledge) of the video rental problem domain. For example, we have identified "address" as a class, but upon examining the context by using the Context feature, it is apparent that "address" is really an attribute of "customer" (because of the possessive pronoun "his").

To correct this problem, we select “address” in the list of model elements, unmark it as a class candidate using the **Unmark** button and mark it as an attribute candidate using the **Attribute** button. The word "address" is now highlighted in cyan, indicating that it is modeled as an attribute. We examine the remainder of the list of model elements, and convert other incorrectly modeled elements (e.g., "price") to attributes.

Having completed our initial marking of classes using LIDA, we now turn to possible attributes using tagged adjectives. As shown in Figure 9, this list is rather short, and not very rich in possible attributes. The only 2 possibilities on the list are "overdue", a possible attribute of "rental", and "available", a possible attribute of "tape". The other adjectives don't make sense as attributes in our context.

Finally, we use our list of tagged verbs to help identify methods or operations associated with the classes we have


LIDA 0.8.9 -- C:\CGT-Apps\Lida\lida089e\sample_files\Video_Store.txt
File Edit View Filter Options Help
Nouns [✓ Adjectives] Verbs All
Class
Attribute
Operation
Role
Unmark
Stopword
Context
Go To
Phrases

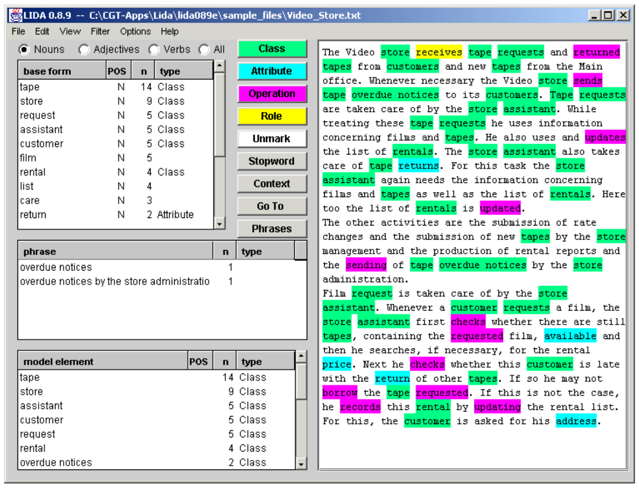
base form	POS	n	type
rental	A	3	
other	A	2	
necessary	A	2	
overdue	A	2	
new	A	2	
available	A	1	

phrase	n	type

model element	POS	n	type
tape		14	Class
store		9	Class
return		1	Attribute
rental		4	Class
price		1	Attribute
notice		2	Class
customer		5	Class

```
The Video store receives tape requests and returned
tapes from customers and new tapes from the Main
office. Whenever necessary the Video store sends
tape overdue notices to its customers. Tape requests
are taken care of by the store assistant. While
treating these tape requests he uses information
concerning films and tapes. He also uses and updates
the list of rentals. The store assistant also takes
care of tape returns. For this task the store
assistant again needs the information concerning
films and tapes as well as the list of rentals. Here
too the list of rentals is updated.
The other activities are the submission of rate
changes and the submission of new tapes by the store
management and the production of rental reports and
the sending of tape overdue notices by the store
administration.
Film request is taken care of by the store
assistant. Whenever a customer requests a film, the
store assistant first checks whether there are still
tapes, containing the requested film, available and
then he searches, if necessary, for the rental
price. Next he checks whether this customer is late
with the return of other tapes. If so he may not
borrow the tape requested. If this is not the case,
he records this rental by updating the rental list.
For this, the customer is asked for his address.
```

Figure 9. Marking of candidate classes and attributes


LIDA 0.8.9 -- C:\CGT-Apps\Lida\lida089e\sample_files\Video_Store.txt
File Edit View Filter Options Help
[✓ Nouns] Adjectives Verbs All
Class
Attribute
Operation
Role
Unmark
Stopword
Context
Go To
Phrases

base form	POS	n	type
tape	N	14	Class
store	N	9	Class
request	N	5	Class
assistant	N	5	Class
customer	N	5	Class
film	N	5	
rental	N	4	Class
list	N	4	
care	N	3	
return	N	2	Attribute

phrase	n	type
overdue notices	1	
overdue notices by the store administratio	1	

model element	POS	n	type
tape		14	Class
store		9	Class
assistant		5	Class
customer		5	Class
request		5	Class
rental		4	Class
overdue notices		2	Class

```
The Video store receives tape requests and returned
tapes from customers and new tapes from the Main
office. Whenever necessary the Video store sends
tape overdue notices to its customers. Tape requests
are taken care of by the store assistant. While
treating these tape requests he uses information
concerning films and tapes. He also uses and updates
the list of rentals. The store assistant also takes
care of tape returns. For this task the store
assistant again needs the information concerning
films and tapes as well as the list of rentals. Here
too the list of rentals is updated.
The other activities are the submission of rate
changes and the submission of new tapes by the store
management and the production of rental reports and
the sending of tape overdue notices by the store
administration.
Film request is taken care of by the store
assistant. Whenever a customer requests a film, the
store assistant first checks whether there are still
tapes, containing the requested film, available and
then he searches, if necessary, for the rental
price. Next he checks whether this customer is late
with the return of other tapes. If so he may not
borrow the tape requested. If this is not the case,
he records this rental by updating the rental list.
For this, the customer is asked for his address.
```

Figure 10. Marking of all candidate model elements

previously identified. Identifying methods at this point is much more difficult, and is typically done at a later iteration of the OOAD lifecycle (during analysis & design). Nonetheless, we can examine the list for candidates at this stage to help get as complete a class diagram as possible, as early in the process as possible. With the classes in mind, we examine the list of verbs in search of behaviors (operations or methods) the classes might exhibit. As before, we can use the Model Element window to view the list of classes more easily. As an example, we start with the "notice" class. Scanning the list of verbs, we find a behavior that a "notice" might exhibit, "send", and mark it as an operation. We follow this same general procedure for the remaining classes, resulting in the following verbs marked as possible operations: "check", "request", "return", "record" and "borrow". It is also critical, especially when differentiating between methods and roles, to use the context feature to view noun-verb collocations.

While looking over the highlighted text, we notice that a phrase is apparent "overdue notice", which may represent a sub-class, or special kind-of notice. We click on the "notice" base form, then on the **Phrases** button to the right. A list of phases will appear in the middle window on the left that contain the word "notice" as their head (i.e., that all describe types of notices). By selecting the phrase "overdue notices", and clicking the **Class** button, we will mark the phrase as a class. The result of this operation is shown in Figure 10. Notice that the first occurrence of the phrase "overdue notices" is highlighted together in the text window.

Finally, we examine the verb list to see if there are any possible roles that one class will play with respect to another class. Notice that a customer can "receive" an overdue notice and "receive" a tape that they have rented, so select "receive", and tentatively mark it as a role using the **Role** button.

Now that we have a set of initial classes, attributes, and operations, we are ready to use them to construct our model. We select the **Edit Model …** option from the File menu, which starts the LIDA Modeler application in a new window, as shown in Figure 11. Notice that there are four sub-windows on the left side of the Modeler window, labeled Classes, Attributes, Operations, and Roles. Each sub-window contains those model elements previously identified in the text.

4.2. Working with the graphical model

Using LIDA's Model Editing Environment, we will now combine these model elements into a preliminary UML class model. Clicking on the candidate model element "customer" in the Text Analyzing Environment shown in Figure 10 displays this element in the Model Editing Environment shown in Figure 11.

The model element "customer" appeared as an UML class without attributes, operations or any associations. We first assign attributes and operations to this class using the candidate attributes and operations previously extracted from the text. The textual context is always available, if a memory aid is needed, to assist with any assignment decisions. An attribute of "customer" that we are certain of is "address". We assign "address" as an attribute of "customer" by clicking in the attribute zone of "customer" in Figure 11 and by selecting the corresponding attribute in the attribute dialog box shown in Figure 12. In a similar way, we assign "request" and "return" as two operations of "customer" by clicking in the operation zone of "customer" in Figure 11 and by selecting the corresponding operations in the operation dialog box shown in Figure 13. The resulting definition for "customer" is shown in Figure 14.

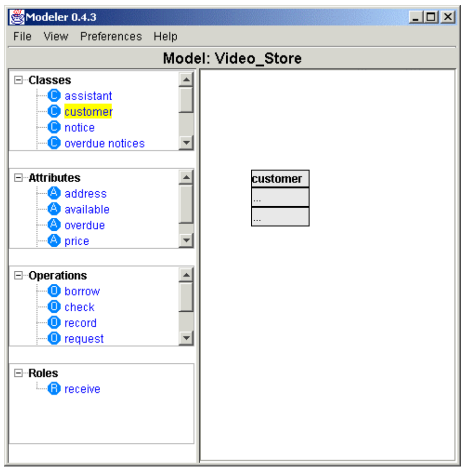
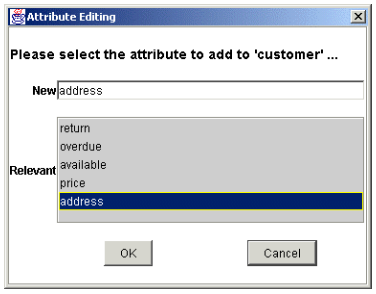


Figure 11. Initial model state for the customer class

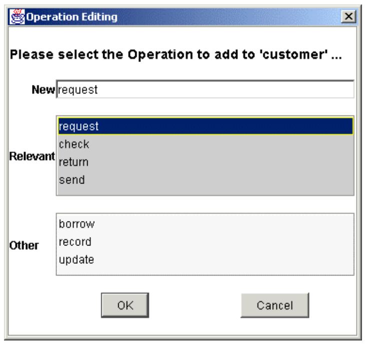


Please select the attribute to add to 'customer' ...

New: address

Relevant: return, overdue, available, price, address

Figure 12. Context sensitive attribute editing



Please select the Operation to add to 'customer' ...

New: request

Relevant: request, check, return, send

Other: borrow, record, update

Figure 13. Context sensitive operation editing

The attributes and operations listed in the attribute and operation dialog boxes shown in Figures 12 and 13 are sorted by their relevance to the class "customer". This relevance is based on the proximity of the occurrences of the words corresponding to attributes and operations to the words corresponding to the class "customer" in the text. New attributes and new operations not available for selection in the attribute and operation dialog boxes can be created by the user simply by typing their names in the attribute and operation dialog boxes. When new attributes and operations are created, they become available for all

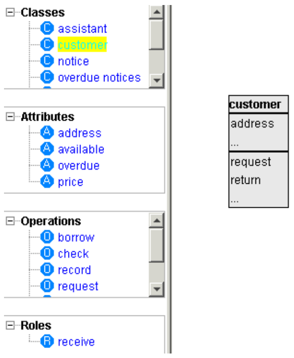


Figure 14. Second model state for the customer class

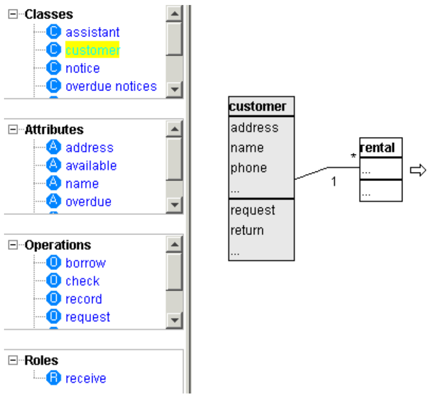


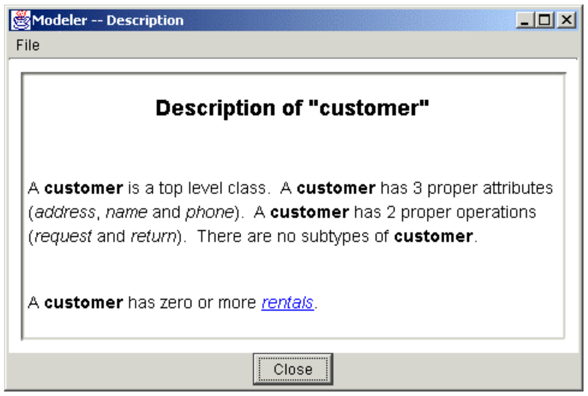
Figure 15. Third model state for the customer class

the components and this can modify the marking in the Text Analyzing Environment.

For example, given our domain knowledge, we know that video stores keep a record of the customers' names and telephone numbers, but this information is not indicated in the text that we analyzed earlier. So, we create two new attributes, "name" and "phone", that we add to "customer". We continue this activity until the class definition is as complete as possible, given current domain knowledge.

The next step is to decide the nature of the associations between the "customer" class and other classes, given our knowledge of the use cases under which the system will be used. LIDA helps with this process as well via functionality for editing associations between classes. We know that "customer" and "rental" are associated, since customers are naturally engaged in rental transactions, so we first create an association between these two classes.

We are still not finished editing the association, since we have not specified the multiplicity of the association. The default multiplicity is *, or many-to-many. From our knowledge of renting videos, and from the base text, we observe that this is incorrect, and that we must change the multiplicity of the customer to 1:1 or, "one-and-only-one", but that one customer can rent many videos. After



Description of "customer"

A **customer** is a top level class. A **customer** has 3 proper attributes (*address*, *name* and *phone*). A **customer** has 2 proper operations (*request* and *return*). There are no subtypes of **customer**.

A **customer** has zero or more *rentals*.

Figure 16. Textual description of the customer class

setting the association multiplicity between "customer" and "rental", we obtain the model diagram shown in Figure 15.

This model diagram can be validated with a domain expert using text descriptions that can be generated from menu items associated with each object in the model. Figure 16 illustrates the text description for the class "customer" as it is defined in Figure 15.

To continue with model development, we associate the remaining classes with their operations and attributes as we did with the "customer" class. When all of the word lists are used, and any additional attributes and operations have been added that are ***clearly needed*** and ***well understood*** by the analyst, we proceed to build the associations between classes. Most of the identification of additional classes, attributes, and operations will be accomplished in cooperation with users, customers and other domain experts. Now, the initial, conceptual level class model is finished.

To continue iterative refinement of the class model, and the development of associated models using UML notation, the basic class model should be exported to a more feature-rich diagramming tools. LIDA supports the export of models from LIDA to Visio™, a diagramming tool that supports the UML notation. Import of models from Visio to LIDA is also supported.

5. Related Research

There is an extensive literature discussing the relation between the linguistic structure of the requirements document and the model that is obtained from it, for example [6,10,15,17]. LIDA is inspired by the observations in these research efforts, but differs from these efforts crucially. These previous efforts are either purely methodological, or aim at completely automatic model extraction, which is currently beyond the state of the art for serious (i.e., non-demo) software engineering efforts. An interesting and ambitious example of this type of research is the COLOR-X system [5], which aims at including a large lexicon to aid in semantic model validation and provides a new modeling language specially developed for this purpose. Perhaps the system that comes closest to LIDA is the Grammalizer system, which is sold by the Dutch consulting group KISS (http://www.kiss.nl) and which is conceptually related to COLOR-X. Because it is a commercial product, information about this system can only be obtained from marketing literature, but it appears to use similar linguistic technology to LIDA (broad coverage part-of-speech tagging and morphological processing).

The current implementation of LIDA differs from these efforts in the following key aspects:

- LIDA provides linguistic tools (part-of-speech tagging, morphological analysis, generator for text descriptions), which are broad coverage instead of being domain specific in order to be immediately usable in new domains.
- LIDA only uses linguistic tools that have a high degree of accuracy; we consider parsing (automatic syntactic analysis), let alone automatic semantic analysis, to be not sufficiently accurate in the current state-of-the-art approaches to be incorporated into a tool intended for software engineering practice.
- LIDA contains morphological knowledge of English, but no semantic knowledge. Creating semantic lexicons is very resource-intensive, and the currently available resources (such as WordNet [8]), while the fruit of years of labor, are not of sufficient quality and consistency to be used in a modeling tool. The task of modeling in some sense represents the task of formally defining the semantics of the domain; using someone else’s rough general-purpose semantics is not likely to be helpful. In LIDA, we make no assumptions about what sort of concepts will be found -- the analyst works with the result of LIDA's linguistic processing to create a model.
- LIDA provides a smooth two-way integration between text analysis and model diagramming, allowing the user to go back and forth between these processes.
- LIDA is flexibly integratable with different approaches to modeling through the use of UML-style models, which also means that LIDA is compatible with UML and UML-based commercial off-the-shelf graphical modeling tools. LIDA is designed to be compatible with actual software engineering practice, rather than impose a new methodology for the entire modeling process.

6. Conclusion

In this paper, we have described a methodology using a prototype tool, LIDA, to provide linguistic assistance in the model development process. This linguistic assistance is important because it facilitates the opportunistic creation of cognitive artifacts for analysis, and is extensible to the creation of many classes of conceptual models. The methodology involves an initial identification of model elements, or conceptualization, through assisted text analysis, followed by a refinement through validation using text descriptions of the model being developed. LIDA offers a good integration between the text analysis process and conceptual modeling process. However, the text analysis remains in good part a manual process that can be cumbersome with complex texts. While there are numerous critics of a linguistic approach to model creation, the fact remains that all modelers start with a greater or lesser knowledge of the problem domain, and varied levels of access to sources of that knowledge. In the event that this knowledge is represented as text, our tool and methodology will provide a useful starting point. LIDA provides not an end, but a beginning to analysis.

7. Further information

For more information regarding the work reported in this paper, contact lida@cogentex.com.

8. Acknowledgements

The work reported in this paper was partially funded by SBIR F30602-97-C-0088 and SBIR F30602-92-C-0015 awarded by USAF Rome Laboratory, and a TRP-ROAD cooperative agreement F30602-95-2-0005 with the sponsorship of DARPA and Rome Laboratory. We are thankful to Richard Kittredge, Tanya Korelsky, Daryl McCullough and David Post for their contributions to the work reported here.

9. References

[1] Barker, Richard, *CASE Method: Entity Relationship Modelling Book* (Addison-Wesley Oracle Series), New York:Addison-Wesley, 1989.

[2] Bellin, David, Suchman, Susan and Booch, Grady, *The CRC Card Book* (Addison-Wesley Object Technology Series), New York:Addison-Wesley, 1997.

[3] Booch, Grady, Jacobson, Ivar, and Rumbaugh, James, *The Unified Modeling Language User Guide* (The Addison-Wesley Object Technology Series), Addison-Wesley, 1998.

[4] Booch, Grady, *Object-Oriented Analysis and Design With Application*s (Addison-Wesley Object Technology Series), New York: Addison-Wesley, 1994.

[5] Burg, J.F.M. and van de Riet, R.P., "Analyzing Informal Requirements Specifications: A First Step towards Conceptual Modeling", *Proceedings of the 2nd International Workshop on Applications of Natural Language to Information Systems*, Amsterdam, The Netherlands. IOS Press, 1996.

[6] Chen, P.P-S., "English Sentence Structure and Entity-Relationship Diagrams", *Information Systems*, 29, 1983.

[7] Coad, Peter and Yourdon, Edward, *Object-Oriented Analysis* (Yourdon Press Computing Series), New York:Yourdon Press, 1991.

[8] Fellbaum, C, *WordNet: An Electronical Lexical Database*, MIT Press, 1998.

[9] Jacobson, Ivar, *Object-Oriented Software Engineering: A Use Case Driven Approach* (Addison-Wesley Object Technology Series), New York:Addison-Wesley, 1994.

[10] Kristen, G., *Object Orientation: The KISS-method: From Information Architecture to Information System.* Addison-Wesley, 1994.

[11] Lavoie, B., and Rambow, O., "A Framework for Customizable Generation of Multi-Modal Presentations", *Proceedings of the 36th Meeting of the Association for Computational Linguistics*, Montreal, Canada, 1998.

[12] Lavoie, B., Rambow, O. and Reiter, E., "Customizable Descriptions of Object-Oriented Models", *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington, DC, 1997.

[13] Odell, James J. and Fowler, Martin, *Advanced Object-Oriented Analysis and Design Using UML* (SIGS Reference Library , No 12), SIGS Books & Multimedia, 1998.

[14] Ratnaparkhi, A., "A Maximum Entropy Part-Of-Speech Tagger", *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania, 1996.

[15] Rolland, G. and Proix, C., "A Natural Language Approach to for Requirements Engineering", *Proceedings of the 4th International Conference on Advanced Information Systems Engineering*. P. Loucopoulos, ed., Springer-Verlag, Manchester, 1992.

[16] Schneider, Geri, Winters, Jason P., and Jacobson, Ivar, *Applying Use Cases : A Practical Guide* (Addison-Wesley Object Technology Series) New York:Addison-Wesley Pub Co., 1998.

[17] Tseng, F.S.C., Chen A.L.P., and Yang, W-P., "On Mapping Natural Language Constructs into Relational Algebra through E-R Representation", *Data and Knowledge Engineering*, 9, 1992.