

OBJECT-ORIENTED ANALYSIS: GETTING HELP FROM ROBUST COMPUTATIONAL LINGUISTIC TOOLS

Sylvain Delisle ¹, Ken Barker ², Ismail Biskri ¹

Abstract

In object-oriented (OO) software engineering, objects, attributes and processes are often specified in natural language descriptions. Following a precise methodology, we here look at the analysis phase of OO software engineering and show that robust computational linguistic tools initially developed for knowledge extraction from text can provide useful support during this initial but crucial phase of any OO software development project.

1. Introduction

1.1. The role of text and narrative descriptions in OO analysis

The use of nouns, pronouns and noun phrases to identify objects and classes during the early stages of OO analysis is not a new idea. As reported in [10], it was first proposed by Russell J. Abbott then popularised by Grady Booch, both in 1983. This might seem to be a restrictive way to look at OO analysis, and there is still disagreement as to what extent noun phrases should influence the output of this phase. Nonetheless, it is fair to say that a majority of modern OO methodologies include an equivalent notion in one form or another. As we will see in section 1.2, some OO methodologies have generalised this idea to consider verb phrases as well. If the software engineer is going to consider noun or verb phrases, the first interesting question is where to look for them. The answer varies according to each OO methodology but some typical examples are: in a concise summary of the subject matter ([4]); in a description of the problem space ([4],[11]); in a description of the users' needs ([4],[12]); in use cases ([13]); or in a summary paragraph describing the problem ([5]). The fundamental idea here is that these texts and narrative descriptions contain the basic elements to identify the building blocks of the OO analysis of the problem. The second interesting question is how one should go about the analysis of relevant textual material. This is what we consider next.

1.2. A representative OO analysis methodology

Chapter 11 of S.A. Conger's book on software engineering ([5]) is dedicated to OO analysis. The first four steps of Conger's OO analysis methodology are:

¹ Département de mathématiques et d'informatique, Université du Québec à Trois-Rivières, C.P. 500, Trois-Rivières, Québec, Canada, G9A 5H7. email: {Sylvain_Delisle, Ismail_Biskri}@uqtr.quebec.ca

² School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada, K1N 6N5. email: kbarker@site.uottawa.ca

1. Develop summary paragraph (pp.464-5: "The first, and most important, step of object-oriented analysis ... The guidelines for writing the paragraph are as follows: 1. Write only declarative sentences of the form: Noun-Verb, Noun-Verb-Object, Verb-Object; 2. For ease of quality assurance, write each sentence on its own line; 3. Review the paragraph carefully to ensure: all desired functions are represented, all major information and processes are identified, all sentences are at the same level of abstraction, detail, and importance.");
2. Identify objects of interest (p.468: "1. Underline all nouns in the summary paragraph; 2. List the underlined nouns on a separate sheet of paper, using the exact same sequence and spelling as in the paragraph; 3. Evaluate each noun to make sure it is an object (...); 4. Determine whether the object is in the solution space (...) or the problem space (...); 5. Name each unique object in the solution space. (...).");
3. Identify processes (p.473: "1. Circle all verbs in the summary paragraph; 2. List the circled verbs on a separate sheet of paper, using the exact same sequence and spelling as in the paragraph; 3. Evaluate each verb to make sure it is a process. (...); 4. Determine whether the process is in the solution space or the problem space; 5. Name each unique process in the solution space. (...); 6. Assign objects to verbs if the object is transformed by the process or if the object data is read by the process; 7. Evaluate the object assignments (...).");
4. Define attributes of objects (p.479: "To define the attributes of an object, we identify all of the information about objects. ... The original description of the project is rechecked to identify any adjectives or adjectival phrases describing nouns that are objects in the solution space. (...).").

In the remainder of this paper, we will focus on steps 2 to 4 which we think are good candidates for automation. Subsection 1.3 presents text analysis tools well adapted to such a task; section 2 will show how these tools perform on Conger's detailed example of a video rental system.

1.3. DIPETT & HAIKU: robust computational linguistic tools for text analysis

Syntactic analysis of textual material is performed by DIPETT (Domain-Independent Parser of English Technical Texts). The output of this broad-coverage parser consists of a single parse tree, the best analysis according to its surface-based heuristics. Tests on a number of actual, unedited texts have produced either full parses or fragmentary parses (into sequences of phrases) for up to 95% sentences. A sentence, DIPETT's unit of processing, contains a main clause and possibly subordinate clauses. Before semantic analysis takes place, parse trees of structurally complex sentences are decomposed into parse trees for clauses. We do not assume that input texts are free of ambiguity, but the user is expected to help the system select only one parse tree and only one semantic analysis for each input string analysed. Semantic analysis is performed by a separate module called HAIKU. Each clause (HAIKU's unit of processing) is built around a main verb which is essential in case analysis ([3], [7], [8]). Case relations represent semantic relationships between a clause's main verb and its syntactic arguments (subject, objects, prepositional phrases, adverbials). Links labeled by cases usually correspond to roles in the activity denoted by the verb; for example, the agent case identifies an entity that acts. Cases appear in texts as predicate-argument structures, each case denoted by a syntactic marker and occupied by a particular filler. For example, in "The thief broke the

window with a stone”, with is the marker and a stone is the instrument of the break activity. DIPETT and HAIKU are robust components of a fully implemented text analysis system, called TANKA, originally designed for knowledge acquisition—details appear in [2], [8] and [9].

2. Applying the text analysis tools to the task of OO analysis

It would have been too easy to apply our text analysis tools to any textual material associated with an OO analysis task and claim that, whatever results they produce, these should be useful to the software engineer. Not only were we motivated to follow a precise methodology, but we were even more motivated to evaluate our results as objectively as possible. What we decided to evaluate was how close the results produced by the computational linguistic tools would be to the results of (steps 2 to 4 of) the OO analysis performed by human experts in the video rental example mentioned in section 1.2. We submitted, verbatim, the summary paragraph of the video rental example ([5]:p.468) to our text analysis system and conducted automatic syntactic analysis with DIPETT and semi-automatic semantic analysis with HAIKU. Figure 1 below presents an excerpt of the summary paragraph (the complete paragraph contains 25 sentences):

```
To rent tapes.
Customers select one to n videos for rental.
Customer phone number is entered to retrieve customer data either to
  create a rental or to retrieve an active rental.
Any outstanding video rentals are displayed with the amount due on
  each tape and a total amount due.
Bar code IDs for each tape are entered.
Video name and rental price from inventory are displayed.
When all tape IDs are entered, the system computes the total.
Money is collected and the amount is entered into the system.
Change is computed, displayed, and further processed by the clerk
  as required.
When the clerk presses an 'order-complete' option key (to be defined
  by the system), this rental is complete and the video inventory file
  is updated (decrease the count of available copies by one).
The rental is stored and printed.
The customer signs the order form, takes the tape, and leaves.

To return a tape.
The video bar code ID is entered into the system.
The rental is displayed and the tape is marked with the date of return.
If past-due amounts are owed, they can be paid at this time; or the clerk
  can select the 'order-complete' option which updates the rental with the
  return date and calculates past-due fees.

Etc.
```

Figure 1: Excerpt from the summary paragraph describing the video rental system.

Let us now take one sentence out of the above excerpt and explain how it is analysed by our text analysis system. Our example sentence is The video bar code ID is entered into the system. It is first correctly parsed by DIPETT and then submitted to HAIKU which, subject to user approval, produ-

ces a structure (simplified here) equivalent to:

```
Enter ( dobj / obj / "the video bar code ID" , into / lto / "into the system" )
```

which means that the main verb (or process, or activity) of this example sentence is enter, that its direct object is the video bar code ID which holds the role of semantic object of enter, and that its complement into prepositional phrase holds the *location-to* (or destination) semantic role of enter. It is clear that the results obtained from our text analysis system allow the software engineer to easily collect an initial list of candidate objects (from noun phrases) and candidate processes (from verbs in verb phrases). As shown by Conger, it is from this initial list that the software engineer will make strategic choices and iteratively refine the OO model. Case analysis can be applied to help accomplish steps 2 and 3 of Conger's methodology. Step 4, the definition of the objects' attributes requires more sophisticated noun phrase analysis.

Conger simply names adjectives as identifying object attributes. It is true that adjectives and adjectival phrases tend to identify such attributes. However, the English language is also well-known for its tendency to convey a lot of information in more complex noun phrases, including concatenated noun sequences (noun compounds), such as video bar code ID in the above example. HAIKU's noun modifier relationship (NMR) analyzer [1] identifies semantic relationships between nouns and the elements modifying them in complex noun phrases. For video bar code ID, the NMR module uncovers the information that a video has a video bar code ID, that a bar code ID is made of bar code and that a bar code contains bars. This detailed analysis of noun compounds, useful for step 4, reinforces the results for step 3 as it ensures that appropriate candidate objects have been identified.

The complete processing of the full summary paragraph of the video rental example led to these results: 76% (34/45) of the objects, with most of their attributes, and 66% (23/35) of the processes identified in Conger's initial lists were identified. These results are encouraging considering that i) they were obtained with a text analysis system **not** specifically designed for computer-assisted OO analysis, and ii) they were obtained for a text **not** adapted or modified for our text analysis system.

3. Conclusion

We have shown that computational linguistic tools, such as those in the TANKA text analysis system, are appropriate for preliminary computer-assisted OO analysis. They allow a software engineer to concentrate on the collection or preparation of an accurate textual description of the problem at hand without having to resort to specialised OO tools, diagrams or techniques during the initial stages of analysis. The first draft obtained from such a text analysis system can then be used as the foundation of a more elaborate OO model. We conducted a small-scale experiment that produced interesting results suggesting that this line of research deserves more attention. In particular, we have started to work on a closely related problem: what happens when the text available for OO analysis contains much more than what the software engineer strictly needs to conduct its analysis?

How does one reliably identify the relevant subset for the current task? To this end, we are considering the use of statistical tools as a filtering mechanism prior to in-depth text analysis ([6]). More precisely, mathematical (or numerical) tools are generally quite robust but can only provide coarse-granularity results; such tools are usually appropriate for the processing of very large inputs. On the other hand, computational linguistic tools are able to provide fine-granularity results but are less robust, often semi-automatic, and usually appropriate for the processing of relatively much shorter inputs. As a result, a synergistic combination of both types of tools should allow us to get the best of both worlds. A connectionist classifier is first used to locate potential documents, or segments thereof, that could be of interest to the user. The user then selects segments that will be forwarded to the linguistic processor in order to semi-automatically analyze their textual data and extract relevant information or knowledge elements.

4. References

- [1] BARKER, K. & S.SZPAKOWICZ, "Semi-Automatic Recognition of Noun Modifier Relationships", *Proceedings of COLING-ACL '98*, Montréal, (Canada), August 10-14 1998, 96-102.
- [2] BARKER, K., S.DELISLE & S.SZPAKOWICZ, "Test-Driving TANKA: Evaluating a Semi-Automatic System of Text Analysis for Knowledge Acquisition", *Proceedings of the 12th Canadian Artificial Intelligence Conference—CAI-98*, Vancouver, BC (Canada), June 18-20 1998, 60-71.
- [3] BARKER, K., T.COPECK, S.DELISLE & S.SZPAKOWICZ, "Systematic Construction of a Versatile Case System." *Journal of Natural Language Engineering*, Cambridge University Press, 3(4): 279-315, 1997.
- [4] COAD, P. & E.YOURDON, *Object-Oriented Analysis*, Yourdon Press/Prentice Hall, 1990.
- [5] CONGER, S.A., *The New Software Engineering*, Wadsworth Publishing Company, 1994.
- [6] DELISLE, S. et I.BISKRI (1999), "Textual Data Mining through the Synergistic Combination of Classifiers and Linguistic Processors", *Proceedings of The 1999 International Conference on Imaging Science, Systems, and Technology (CISST'99)*, June 28 - July 1st, Las Vegas (Nevada, USA), to appear.
- [7] DELISLE, S. & S.SZPAKOWICZ, "Extraction of Predicate-Argument Structures from Texts", *Proceedings of the 2nd Conference on Recent Advances in Natural Language Processing—RANLP-97*, Tzigov Chark (Bulgaria), September 11-13 1997, 318-323.
- [8] DELISLE, S., "Text processing without A-Priori Domain Knowledge: Semi-Automatic Linguistic analysis for Incremental Knowledge Acquisition", Ph.D. thesis, TR-94-02, Department of Computer Science, University of Ottawa, 1994.
- [9] DELISLE, S., K.BARKER, T.COPECK & S.SZPAKOWICZ, "Interactive Semantic analysis of Technical Texts." *Computational Intelligence*, 12(2): 273-306, 1996.
- [10] FIRESMITH, D.G., *Object-Oriented Requirements Analysis and Logical Design (A Software Engineering Approach)*, John Wiley & Sons, 1993.
- [11] PRESSMAN, R.S., *Software Engineering (A Practitioner's Approach)* (4th edition), McGraw-Hill, 1997.
- [12] RUMBAUGH, J., M.BLAHA, F.EDDY, W.PREMERLANI & W.LORENSEN, *OMT 1 : Modélisation et conception orientées objet*, Masson/Prentice Hall, 1997.
- [13] WHITTEN, J.L. & L.D.BENTLEY, *Systems Analysis and Design Methods* (4th edition), Irwin/McGraw-Hill, 1998.