NATURAL LANGUAGE PROCESSING FOR REQUIREMENTS ENGINEERING: APPLICABILITY TO LARGE REQUIREMENTS DOCUMENTS

Leonid Kof
Fakultät für Informatik, Technische Universität München,
Boltzmannstr. 3, D-85748 Garching bei München, Germany
kof@informatik.tu-muenchen.de

August 23, 2004

Abstract

This paper describes a case study on application of natural language processing in very early stages of software development. At this early stage it is very important for the domain expert (who is, most probably, the future user) and the software expert to define a common language, understood by both of them. To define such a common language, we extract terms from the text written by domain expert, classify these terms and build a domain ontology using them.
In our previous work [11] we experimented with applicability of the approach to small requirements documents. It turned out that manual work is also necessary to extract information from requirements document. This gave rise to the question whether the same approach also works for larger documents.
In this paper we present a case study on the extraction of the ontology from a larger document. The case study shows that the approach works for larger documents as well and the cost (in the sense of time consumption) is reasonable.

1. Introduction

Precise specification is a key success factor for a software project. Formal specification is ideal for the software developer, but it is not reasonable to require the author of the requirements document, who is seldom familiar with formal methods or even with the concept of "specification", to provide a formal description.

According to Berry [4] and Luisa et al. [13], the overwhelming majority of requirements are written in natural language. For the reader of such a requirements document it is very important to identify the concepts used by the writer (=domain expert) and relations between them. These concepts and relations build the basis of the common language understood both by the requirements engineer and the domain expert. On the basis of these terms we also can construct a domain ontology. Domain ontology is itself a valuable requirements engineering product, as stated by Breitman et al. [5].

The necessity to extract information from natural language documents motivated a lot of research on application of text analysis in requirements engineering.

1.1. Related Work

Although Kevin Ryan claims in [19] that natural language processing (NLP) is not ripe enough to be used in requirements engineering, we can nevertheless identify several useful applications:

- According to [19], NLP can not understand the text. But it is not the goal of most NLP-using approaches to understand the text, the goal is often to extract concepts contained in the document.
- Kevin Ryan [19] claims also that the domain model produced by NLP means may be incomplete, because some information that is thought to be "common domain knowledge" is omitted in the requirements text. But this is exactly one of the tasks of the requirements engineer to detect such omissions. So, an incomplete extracted model would be an indicator for some omissions in text.

To put it in a nutshell, there are some sensible NLP applications in requirements engineering even though NLP is not yet capable of proper text understanding.

In [3] Ben Achour classifies the linguistic methods as either lexical or syntactic or semantic. Lexical methods, as for example AbstFinder [10] are the most robust ones. AbstFinder extracts the terms (lexica) that occur repetitively in the specification text. This method is extremely robust, because it does not rely on part–of–speech analysis, parsing or something like that. It just considers sentences as character sequences and searches for common subsequences in different sentences. It does not perform any term classification.

In the syntactical approaches, like in those by Abbott [1], Chen [7] and Saeki et al. [20], one analyzes either parts of speech (substantives, verbs etc.) or looks for special sentence constructions. Abbott [1] analyzes used substantives to produce the data types for the construction of the program. Saeki et al. [20] additionally analyze verbs and declare them to operations. Chen [7] goes further and produces an E/R–diagram as analysis result. To construct the E/R–diagram, one searches for special sentence constructions that become the basis for certain relations. The set of constructions that can be analyzed is finite and the sentences that do not fit into one of the predefined types can not be analyzed.

Although semantic approaches are related to our work, they are not really comparable: Our goal is to *extract* ontology from text, whereas semantic approaches *use* ontologies to convert text to formulas.

There are also some other related approaches, as for example approaches by Chris Rupp [18], Natt och Dag et al. [16] and Fabbrini et al. [8]. They are related in the sense that they also analyze requirements documents written in natural language, but they do not fit into the above classification scheme, because their goal is not to automate the extraction of the information from text.

Chris Rupp [18] defines a set of writing rules (writing templates). The templates define which arguments are necessary for which verbs. For the sentences written according to templates, one can manually extract the actors, actions and objects. The goal of the approach is to unify writing, which allows to produce better requirements documents. The approach does not offer automated support for text analysis. Fabbrini et al. [8] introduce categories of key words that are undesirable in requirements documents (like "if needed", "as ... as possible" etc.) and measures document quality by counting the undesirable constructions. Natt och Dag et al. [16] search for similar and related requirements

by finding common concepts (words) in different requirement phrasings. These approaches are interesting for producing qualitative requirements, but they are not comparable with our approach, whose primary goal is to extract domain information from the document.

1.2. Goals of our Work

Daniel Berry addressed in his talk "Natural language and requirements engineering – nu?" [4], what could help to reduce the disadvantages of natural language specifications:

1. Learn to write less ambiguously and less imprecisely
2. Learn to detect ambiguity and imprecision
3. Use a restricted natural language which is inherently unambiguous and more precise

Additionally to the extraction of the domain ontology, we want to address the first two points with our approach. "Learn to write less ambiguously and less imprecisely" means that we introduce a set of writing rules. These rules make the text less ambiguous from the point of view of the human reader and at the same time make computer–based analysis better applicable.

We also detect ambiguities (= inconsistencies in term usage) in the specification text and eliminate them. When the specification text is good enough (= consistent in term usage and grammatically correct), we can extract a domain ontology from the text.

The goal of our work is also to further develop syntactical approaches mentioned in the previous section. We want to support the extraction both of the terms and the relations between them, aiming at building an application domain ontology, thus providing more than purely syntactical approaches. Nevertheless, we do not require firm sentence structure, like semantical approaches.

In our previous work [11] we experimented with applicability of the approach to small requirements documents. It turned out that certain manual work is necessary to extract information from requirements document. This gave rise to the question whether the same approach also works for larger documents.

The paper is organized in the following way: In section 2 we sketch the techniques used for ontology extraction. Section 3 presents the results of the first small case study that are necessary for understanding the potential scalability problems. In section 4 we present the actual scalability case study and its results. In section 5 we evaluate these results.

2. Ontology Extraction: the Approach

To make the paper self-contained, we sketch here the techniques used for ontology extraction. A more in-depth introduction can be found in [11].

Extraction of domain knowledge consists of three basic steps:

1. term extraction
2. term clustering and taxonomy building
3. finding associations between extracted terms

The terms with the associations between them build the domain model that we want to extract.

Term Extraction: The aim of this step is to extract predicates with their arguments (subjects and objects). Subjects and objects are the terms we want to extract, predicates are used to classify them.

In the very first step of term extraction, each sentence is properly parsed. "Proper parsing" means that a parse tree is built, as opposed to Part–of–Speech (POS) tagging, which just marks every word as a substantive / adjective / verb / ... Proper parsing eases the extraction of predicates and their arguments.

Term Clustering: Extracted terms are clustered according to grammatical contexts (=verb subcategorization frames) they are used in. Primary clusters are built by either subjects or objects of the same predicate. Cluster overlaps are used to find related clusters and to join them. We use the tool ASIUM [9] for term classification. The result of this classification is an initial domain taxonomy. (See also [11] for more details.)

Association Mining: This step takes the taxonomy generated in the second step as input and enriches it by general associations between extracted terms. The idea is borrowed from data mining, as described by Maedche and Staab in [14]. Text is considered as a set of database transactions and the terms occurring often in the same transaction are assumed to be related. Additionally, the relations are lifted to the right ontology level, as described by Srikant and Agrawal in [21]. (See also [11] for more details.)

After the last step we get an initial application domain model, represented as a tree of terms and binary relations between these terms.

3. First Case Study: Lessons Learned

This section sketches the lessons learned from the first case study, described in [11]. In this first case study we discovered that certain manual work is required to extract the ontology from a requirements document. The document that we used for the first case study [2] was rather small (just 6 pages), so the amount of manual work was small as well.

As a result of the case study we came to the conclusion that ontology extraction results can be improved if the analyzed text obeys to certain rules. These rules include:

- Rules that make sense anyway:
  - Always use the same name for the same concept
  - In the case of compound names, either use names that, put in the sentence, remain grammatically correct (e.g., "normal mode" instead of "mode normal") or mark the compound names as such (i.e., "mode-normal").
  - Always use the complete form in the case of compound names: i.e., "stop message or start message" instead of "stop or start message".
  - Do not use the verbs "be" and "have". They do not provide much information even for the human reader. For the computer–based analysis they produce large clusters of unrelated concepts. Nevertheless, it is allowed to use these verbs to build passive form or perfect tenses. In this case they are easy to filter out.

- Rule due to technical deficiencies:
  - Do not use cross–sentence references like “Message X is sent by unit Y. This message indicates . . . ” In future versions of our approach we could try to use anaphora resolution algorithms, like those presented by Lappin and Leass in [12] and Mitkov in [15]. Such an algorithm could match “Message X” in the first sentence and “This message” in the second one.

Rewriting specification texts according to these rules can be done only manually, which also gives raise to the question whether the text analysis technique scale.

4. Scalability Case Study: Instrument Cluster Specification

For the scalability case study we took a larger specification [6], consisting of approximately 80 pages. This case study describes a car instrument cluster, showing the current speed, RPM (motor revolutions per minute), outside temperature and so on. The instrument cluster communicates via CAN bus with other ECUs (electronic control units).

As in the first case study, our goal was to extract the application domain ontology from the document. In the scalability case study we also documented the time that was necessary for different process steps. This enabled us to identify time consuming steps that potentially do not scale.

4.1. Document Preparation

Text analysis starts with document preparation. There is a set of purely technical issues that are unimportant for smaller documents, but can become time consuming for larger ones. For the analysis we need the text in one-sentence-per-line format. There are tools that recognize sentence boundaries, as for example the one by Ratnaparkhi [17]. However, it turned out that this approach does not work well if the text also contains incomplete sentences.

So, in the first step of text preparation we manually transformed the text into one-sentence-per-line format. This took (together with the first reading of the specification text) one working day.

At this stage we did not reformulated grammatically wrong sentences and we did not convert item lists and tables to fully-fledged sentences. Although the subcategorization frame extraction in its current form works only for grammatically correct sentences, we wanted to see how much noise data is produced in such a way and whether it is really necessary to rephrase incorrect sentences manually.

4.2. Parsing and Information Extraction

After reformatting the text we were able to parse it and to extract syntax information. We extracted the predicate, the subject and objects from each sentence. Extraction results showed that rephrasing of incorrect sentences was necessary.

When analyzing the extracted predicates and their arguments, we discovered a lot of wrong verbs and objects. For example, the operations “=”, “$<$” and “$>$” were classified as verbs, as they often occurred in the specification text in the verb position:

- If Ig-Lock = 1 then the ignition key is in position ignition on.

- If Current-Speed-V $<$ 30 km/h and the Internal-Temp values are sinking, then Outside-Temp-Shown = Internal-Temp.
- If Current-Speed-V $>=$ 50 km/h the rising Internal-Temp values are ignored for 1,5 minutes.

There was an additional problem with the text containing incomplete and grammatically incorrect sentences: The term extraction looks for the sentence predicate and then extracts predicate's arguments (terms). For grammatically incorrect sentences this is not always possible, so incorrect sentences are just ignored during term extraction. If our requirements document contains incorrect sentences, we can not be sure that we extract all the relevant concepts. It could happen that some concepts occur only in incomplete sentences, so that they are completely ignored.

For these reasons our next step was to rewrite incomplete sentences into grammatically correct ones.

4.3. Lists and Tables: Proper Phrasing

It turned out that lists and tables were the main source of incomplete sentences. For example, input signals of the instrument cluster were described like this:

Ig-Lock: Describes the position of the ignition key. If Ig-Lock = 1 then the ignition key is in position ignition on. Sent by the ignition lock control unit. Scope: {0,1}. Received every 100 ms. Transferred by the CAN bus.

We completed each phrase of such constructions so that it became a grammatically correct sentence. In most cases it could be done schematically, but the rephrasing still required manual work. For example, we transformed the above construction into

- Ig-Lock describes the position of the ignition key. If Ig-Lock equals 1 then the ignition key is in ignition-on-position. Ig-Lock is sent by the ignition lock control unit.[1] Ig-Lock can equal 0 or 1. Ig-Lock is received every 100 ms. Ig-Lock is transferred by the CAN bus.

We also performed some transformations according to our writing rules (see also section 3). Such transformations allowed us to perform syntax–based analysis. All these transformations took 1.5 working days, which is justifiable for a 80-pages document. The overall time cost for document preparation (up to this point) amounted to 2.5 working days.

4.4. Taxonomy Extraction

Taxonomy extraction is based on the analysis of cluster intersections. The first ASIUM run showed that there were more than 600 cluster intersections produced by the text. To build a taxonomy we have to analyze cluster intersections, so this step could become time consuming.

During taxonomy building we also analyzed single clusters to detect wrong usage of terms: Every time we came across a cluster containing unrelated concepts, we could detect the (textual) source of this inconsistency and eliminate it.

In the instrument cluster case study we detected relatively small number of inconsistencies:

[1] It is not necessary to mark "ignition lock control unit" as a compound term, because this term is grammatically correct.

- The verb "denote" produced a huge concept cluster containing unrelated concepts. This was due to the fact that the verb "denote" occurred both in constructions like "⟨`some-signal`⟩ `denotes...`" and in "⟨`some-parameter`⟩ `denotes...`" This problem could be corrected for example by replacing "denote" by "influence" when talking about system parameters. In the case study we did not do so because both signals and system parameters could be clustered using other verbs. We just ignored the "denotes"-cluster.

- During the clustering we discovered some concept names that had to be replaced. The replacement was necessary, because several different names were used for the same concept.

With the corrections described above we were able to build a sensible taxonomy. Due to space limitations we do not present the taxonomy here.

Analyzing the whole plethora of cluster intersections and building a taxonomy (concept and cluster hierarchy) took approximately 1.5 working days. The overall time cost up to this point amounted to 4 working days.

4.5. Association Mining

To explain scalability problems potentially posed by association mining, we want to start by introducing some data mining definitions. For an item set A, let $trans(A)$ be the set of transactions containing A and let N be the total number of transactions. The *support* of the association $A \Rightarrow B$ is defined as $\frac{|trans(A \cup B)|}{N}$. The *confidence* of the association $A \Rightarrow B$ is defined as $\frac{|trans(A \cup B)|}{|trans(A)|}$. There is a tool, KAON [14], performing association mining on the basis of these concepts. It computes the support and confidence measures for every association and sorts the associations according to these measures. If the analyst finds an association sensible, she can include it into the ontology. This tool was used in our case studies.

In our case studies we performed the analysis on the per–sentence basis and we defined a transaction as a pair of concepts occurring in the same sentence. For the instrument cluster case study this definition led to more than 1000 potential associations. To cope with this plethora of potential associations, we sorted them lexicographically by $(absolute\ frequency,\ confidence)$. Absolute frequency of the association $A \Rightarrow B$ is defined as $|trans(A \cup B)|$. *Formally*, two associations with the same support have also the same absolute frequency, so we could have used the standard measure $support$. Due to rounded support values presented by KAON to the user, $absolute\ frequency$ gives more information. Lexicographical sorting means that we sorted the associations by $absolute\ frequency$ and in the case of equal $absolute\ frequency$ they were sorted by $confidence$.

For the ontology building we used the associations with $absolute\ frequency \geq 5$. I.e., we considered only pairs of concepts occurring 5 or more times in the same sentence. This corresponded approximately to the most frequent 25% of associations. It took us about one working day to (manually) validate these associations and to include the relevant ones into the ontology. The overall time cost up to this point amounted to 5 working days.

4.6. Instrument Cluster Case Study: Results and Lessons Learned

Starting the instrument cluster case study, we wanted to see whether the ontology extraction approach presented in section 2 still works for large documents and whether the amount of manual work necessary for the extraction is still justifiable.

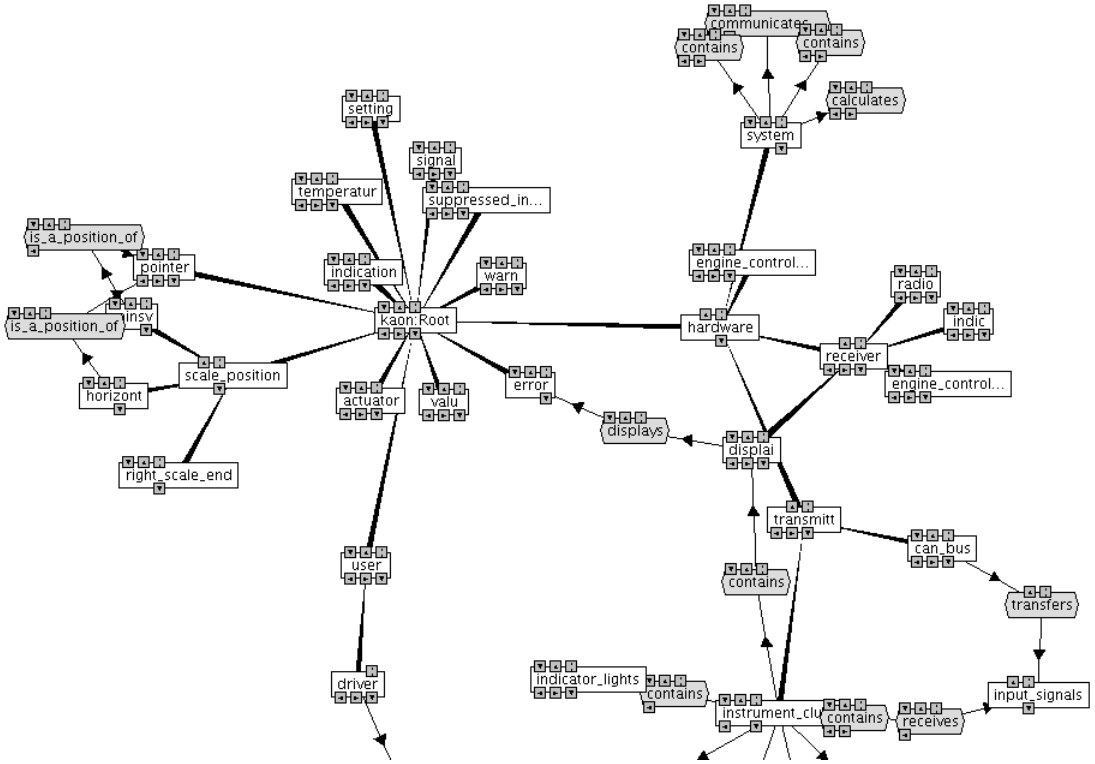

Figure 1: **Instrument Cluster: part of the extracted ontology**

Figure 1 illustrates that ontology extraction worked for this case study as well: Figure 1 shows an excerpt of the extracted ontology. It shows the top ontology class (`kaon:Root`), its subclasses (actual ontology classes) and relations between them. Colors and arrows have the following meaning: Variable–width lines denote "is-a"–relations, the thin end pointing to the more general concept and the thick end pointing to the more special one. White boxes denote concepts and classes of concepts, dark boxes are properties (associations) connecting concepts. Arrows are directed in such a way that a property and two concepts that it connects build a sensible sentence, if read in the arrow direction. For example, in figure 1 one can read:

- Can_bus transfers input_signals
- Instrument_cluster receives input_signals
- Instrument_cluster contains display
- ...

The other goal (additionally to pure feasibility test) of the instrument cluster case study was testing scalability of the approach. During this case study we extracted:

- 123 concepts and concept classes, organized in 13 top–level classes and further subclasses
- 61 associations between different concepts

Additionally to the extraction of concepts and associations we discovered and corrected inconsistencies in term usage. We find the time cost of 5 working days justifiable for an 80–pages document,

1.	Format the text (one sentence per line)	partially automatic
2.	Tag each word (Part–of–Speech)	automatic
3.	Parse the tagged text	automatic
4.	Extract predicates and their arguments	automatic
5.	Build concept clusters	automatic
6.	Look for cluster intersections and build a taxonomy	**interactive**
7.	Transfer the taxonomy to the associations mining tool	partially automatic
8.	Look for potential associations	automatic
9.	Decide which associations are sensible	**interactive**

Table 1: Overview: Steps of Ontology Construction

given that we detected and corrected inconsistencies and constructed a domain ontology. The steps where manual work is necessary consume approximately the same time and have therefore the same influence on scalability.

5. Summary

In this paper we presented an approach for extraction of domain ontology from text documents. Here we want to give an overview of the complete approach and of the produced results.

5.1. Summary of the Approach

The ontology extraction approach consists of several steps, most important of which are introduced in section 2. The list in the table 1 gives an overview of the whole approach and shows which steps are performed completely automatically and which ones require human interaction.

The steps in table 1 correspond to the principal approach, they do not show detection and correction of inconsistencies. Inconsistencies are detected in interactive steps: concept clustering (step 6) and decision about sensible associations (step 9). After the correction of inconsistencies one have to restart with the tagging (step 2).

As one can see, some steps are marked as “partially automatic”, while others are “interactive”. The difference is fundamental: “partially automatic” steps are not completely automatic yet because of some technical problems. In the case of text formatting (step 1), there are problems with incomplete or grammatically incorrect sentences that are often present as bullet points in specification texts. In the case of taxonomy transfer (step 7) it is a mere problem of tool integration.

For the steps that are marked as “interactive” complete automation is not desirable. As Goldin and Berry state in [10], complete automation is not desirable if it could lead to information loss or wrong results. In the case of taxonomy building (step 6) and association ascription (step 9) we can find inconsistencies, which often manifest themselves in senseless term clusters or senseless associations. It is impossible for an automatic tool to decide which clusters/associations are sensible. Even after elimination of inconsistencies not every cluster intersection leads to a *sensible* larger cluster defining a more general concept and not every potential association is a sensible one. So, even for a perfectly consistent text a completely automatic tool would not be feasible.

5.2. Scalability, Qualitative Complexity Analysis

Here we estimate the *qualitative* complexity of the used approach, to be able to predict the time consumption when the input document grows. We estimate the complexity for every document step, to identify potential bottlenecks that remained hidden in the quantitative case study.

Let n be the number of sentences in the text.

Document formatting & reading is obviously linear in document length ($\mathcal{O}(n)$).

Tagging, parsing & predicate extraction work on the per–sentence basis and are linear in document length as well ($\mathcal{O}(n)$).

Clustering & cluster intersection analysis: The number of concept clusters is proportional to the number of verbs used in the document. If we assume that the number of verbs is proportional to document length, the number of clusters is directly proportional to the document length as well. The number of cluster *intersections* is then $\mathcal{O}(n^2)$. Analysis of cluster intersections can thus become a bottleneck for larger documents.

Elimination of inconsistencies: Time necessary to eliminate terminology inconsistencies depends on the number of inconsistencies and *not* on the text length.

Association mining & validation: Time necessary for interactive association validation is proportional to the number of potential associations. A potential assotiation is a pair of terms occurring in the same sentence. A sentence containing m terms produces $m(m-1)/2 = \mathcal{O}(m^2)$ potential associations. The total number of associaions is $\mathcal{O}(n\overline{m^2})$, where $\overline{m^2}$ means the average square number of concepts per sentence. This means that a text containing short sentences is easier to analyze than a text containing long ones (even if the total size measured in characters is the same).

5.3. Lessons Learned

In the first case study (presented in [11]) we wanted to see the actual applicability of the approach. For this purpose we chose a relatively small document, consisting of only 6 pages. On the basis of this first case study we saw that the approach is applicable as long as the text is consistent in term usage. Inconsistencies were detected on the basis of "strange–looking" term clusters and associations. Such inconsistencies can be eliminated only manually, which was not a problem for a short text. However, this gave rise to the question whether the approach is applicable to larger documents.

The goal of the second case study (presented in this paper) was testing scalability of the approach, which was performed on an 80–pages document. The overall time cost for both inconsistency elimination and ontology building amounted to 5 working days. Given the fact that mere skim reading of the document took almost one working day, we find the total of 5 days justifiable.

Summing up, we can say that the presented approach is promising for requirements engineering. It can be used in RE–process both for quality assurance on the document level and for bridging the gap between documents and domain–specific ontology.

Acknowledgements

Here we want to thank Jewgenij Botaschanjan and Markus Pister who provided the instrument cluster specification document and made this case study possible.

References

[1] R. J. Abbott. Program design by informal english descriptions. *Communications of the ACM*, 26(11):882–894, 1983.

[2] J.-R. Abrial, E. Börger, and H. Langmaack. The steam boiler case study: Competition of formal program specification and development methods. In J.-R. Abrial, E. Borger, and H. Langmaack, editors, *Formal Methods for Industrial Applications*, volume 1165 of *LNCS*. Springer–Verlag, 1996. `http://www.informatik.uni-kiel.de/~procos/dag9523/dag9523.html`.

[3] C. Ben Achour. Linguistic Instruments for the Integration of Scenarios in Requirement Engineering. In P. R. Cohen and W. Wahlster, editors, *Proceedings of the Third International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'97)*, Barcelona, Catalonia, June 16-17 1997.

[4] D. Berry. Natural language and requirements engineering - nu?, 2001. `http://www.ifi.unizh.ch/groups/req/IWRE/papers&presentations/Berry.pdf`, accessed 09.01.2003.

[5] K. K. Breitman and J. C. Sampaio do Prado Leite. Ontology as a requirements engineering product. In *Proceedings of the 11th IEEE International Requirements Engineering Conference*, pages 309–319. IEEE Computer Society Press, 2003.

[6] K. Buhr, N. Heumesser, F. Houdek, H. Omasreiter, F. Rothermehl, R. Tavakoli, and T. Zink. Daimler-Chrysler Demonstrator: System Specification Instrument Cluster. `http://www.empress-itea.org/deliverables/D5.1_Appendix_B_v1.0_Public_Version.pdf`.

[7] P. Chen. English Sentence Structure and Entity-Relationship Diagram. *Information Sciences*, 1(1):127–149, May 1983.

[8] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool. In *26th Annual NASA Goddard Software Engineering Workshop*, pages 97–105, Greenbelt, Maryland, 2001. IEEE Computer Society.

[9] D. Faure and C. Nédellec. Asium: Learning subcategorization frames and restrictions of selection. In Y. Kodratoff, editor, *10th European Conference on Machine Learning (ECML 98) – Workshop on Text Mining*, Chemnitz Germany, April 1998 1998.

[10] L. Goldin and D. M. Berry. Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation. *Automated Software Eng.*, 4(4):375–412, 1997.

[11] L. Kof. An Application of Natural Language Processing to Requirements Engineering — A Steam Boiler Case Study, 2004. Contribution to SEFM 2004.

[12] S. Lappin and H. J. Leass. An algorithm for pronominal anaphora resolution. *Comput. Linguist.*, 20(4):535–561, 1994.

[13] M. Luisa, F. Mariangela, and N. I. Pierluigi. Market research on requirements analysis using linguistic tools. *Requirements Engineering*, 9(1):40–56, 2004.

[14] A. Maedche and S. Staab. Discovering conceptual relations from text. In W.Horn, editor, *ECAI 2000. Proceedings of the 14th European Conference on Artificial Intelligence*, pages 321–325, Berlin, 2000. IOS Press, Amsterdam.

[15] R. Mitkov. *Anaphora Resolution*. Longman, 2002.

[16] J. Natt och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. A feasibility study of automated natural language requirements analysis in market-driven development. *Requirements Engineering*, 7(1):20–33, 2002.

[17] A. Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, Institute for Research in Cognitive Science, University of Pennsylvania, 1998.

[18] C. Rupp. *Requirements-Engineering und -Management Professionelle, iterative Anforderungsanalyse für die Praxis*. Hanser–Verlag, second edition, 05 2002. ISBN 3-446-21960-9.

[19] K. Ryan. The role of natural language in requirements engineering. In *Proceedings of IEEE International Symposium on Requirements Engineering*, pages 240–242. IEEE Computer Society Press, 1992.

[20] M. Saeki, H. Horai, and H. Enomoto. Software development process from natural language specification. In *Proceedings of the 11th international conference on Software engineering*, pages 64–73. ACM Press, 1989.

[21] R. Srikant and R. Agrawal. Mining generalized association rules. *Future Generation Computer Systems*, 13(2–3):161–180, 1997.