Requirements capture in natural language problem statements

Ke Li, R.G.Dewar, R.J.Pooley

Department of Computer Science
School of Mathematical and Computer Sciences
Heriot-Watt University
Riccarton, Edinburgh, EH14 4AS, UK
{kl21, rick, rjp}@macs.hw.ac.uk
http://www.macs.hw.ac.uk

Abstract. Requirements elicitation and analysis remains a stubbornly intractable problem to automate. This paper looks at the use of natural language analysis techniques in the requirements capture process. It begins with a review of past work and then develops an algorithmic approach to analysing and restructuring natural language text. We begin by marking the text up as parts of speech, before we restructure this mark-up into subject-verb-object clauses. After a user-assisted pronoun replacement step, a relatively comprehensive UML class diagram can be generated. Further analysis then allows us to generate use cases. To illustrate the algorithm in practice we use a case study.

1. Introduction

Over the years much work has been done to improve the productivity and quality of the early stage of the software lifecycle. However, requirements engineering has remained relatively impervious to attempts at automation. This is primarily due to the inherently human-centred nature of this process. If, somehow, we could merely semi-automate the process of eliciting the requirements of a system from a domain expert, we could potentially improve the speed, accuracy and utility of this activity. This would have knock on, downstream benefits for the whole lifecycle.

With this in mind, we propose an algorithmic approach for the requirements elicitation process based on case study material. This approach is a tentative start and focuses on analysing extant transcripts in order to elicit Object-Oriented (OO) system elements, such as classes, objects, methods, properties and relationships. At this stage we should add that our definition of requirements elicitation is not merely to extract the "said" elements of a transcript, but to imply/deduce/induce the "unsaid" from the narrative.

For instance, we could potentially show incompleteness or ambiguity that would warrant further interviews.

This paper begins with a review of relevant research into Natural Language Processing (NLP) for requirements analysis in the OO paradigm. Section 3 describes our algorithm for processing extant natural language texts, whereas section 4 illustrates the algorithm's application to a case study. Finally, we draw conclusions and outline future work.

2. Related Work

2.1 Requirements Engineering

Requirements exist in variety of textural sources, which could be any documentation involved in a customers' daily work (e.g. forms, tables, reports) or user requirements documented by customers. However, more requirements exist in the surrounding real world, which can only be discovered by communicating with customers. Transcripts of such sessions are another source, from where requirements can be derived. The primary goal of the Requirements Engineering (RE) process is to discover all the requirements that the future system needs to satisfy to be successful. To achieve that goal, two major activities, requirements elicitation and requirements analysis are carried out in an iterative and incremental manner, involving an informal Natural Language (NL) and a formal modelling language.

Pohl [24] introduces three dimensions of RE. One of them is the representational issue raised from the involvement of range of representation methods categorized into informal, natural language specification and formal specification. The use of Natural Language Processing (NLP) in the analysis of requirements has been studied by a number of researchers and found to yield good, if incomplete, results. Developing a mapping between two categories might help to address the issue and might also offer a better understanding of problems for stakeholders. We focus, therefore, on the use of Natural Language Processing (NLP) to direct elicitation of requirements.

RE is mostly concerned with communicating with customers to gather the essential and relevant domain information that forms the base of requirements and which is the key area for research on how to gather and record information from stakeholders (customers, end-users, domain experts). The main obstacle here is the ambiguity of Natural Language (NL). The research on addressing this issue extends the traditional questionnaire and interviewing techniques [11,26] to Apprenticing [1], Soft System [] and narrative [] techniques to support disambiguated and productive communication.

Unlike the above techniques relying on NL based communication, the Mind Mapping [5] technique suggests that pictures and symbols help stakeholders to share the domain knowledge and understand each other. Use Case[] and Scenarios[] are similar, and have been widely accepted and applied. However, such visual language can only be complementary to textural expression; in the real world NL is always the

primary medium of communication both oral and written. Yet, the ambiguity of NL still remains the main issue in communications among stakeholders.

Using the appropriate modelling techniques, information is obtained and analysed to discover the requirements; these are decomposed into system building blocks to model the static and dynamic aspects of the developed system. However, especially given the massive amount of information involved, RE is not easy. A number of projects [25], [8], [31], [128] demonstrate that NLP can benefit requirements analysis by automatically disambiguating some meanings, and, for OO systems, automatically extracting objects and properties to generate the system static view. However, none of these approaches has so far tackled the support of directly eliciting requirements.

2.2 Natural Language Processing (NLP)

The fundamental issues of NL are [25]:

(i) Lexical ambiguity, Alternative parts of speech (according to circumstances) and Word class: article, determiner, preposition, adjective, number, noun, pronoun, adverb, verb, etc.

(ii) Syntactic ambiguity due to the complexity of sentence Structure (to avoid redundancy) i.e. parallel, clause.

Specifically, NLP for requirements analysis aims at removing ambiguity.

Lexical / word-tagging level analysis is the first analysis level in NLP and allocates a Part Of Speech (POS) to each word. The resulting parse tree then is used for the next level - syntactic analysis. The categories of POS are, for example, singular/plural common noun, third person singular present/past/present continuous/past continuous indicative of verb, preposition, and singular/plural article. Tagging methods and techniques have been developed, such as Rule-based tagger [2], ENGCG [18], Memory-based learning taggers [7], statistical Xerox tagger [6] and CLAWS4 [9].

Syntactic analysis (*syntactic annotation* or *grammatical annotation*) uses a phrase marker or labelled bracketing techniques. It brackets segments as phrases (Prepositional phrase, verb phrase, adjective phrase, noun phrase), clauses (relative clause) and sentences, using techniques such as parse trees and dependency grammars [25].

Semantic analysis (*semantic annotation* or *sense resolution*) analyses open-class (content) words and closed classes (prepositions, conjunctions, pronouns). POS links each word to their counterparts in the “real world”, analysing the relationship of each POS, with any alternatives. Examples of relations are negation, modifier plus adjective and adjective plus noun combinations. Furthermore, methods and techniques used include the anaphoric (discourse) annotation scheme [29].

Generally, NLP systems are Rule based, i.e. context-free grammar, GPSG [9] notations. Some are more flexible, notably statistically based systems, which use probabilistic methods, such as UCREL [15]. The issue with rule based systems is size of the rule-base.

Many methods and techniques have been proposed for OO analysis using NLP. Some suggest Nouns indicate class [19] ,[3], objects and properties; some suggest verbs can denote behaviours. However, nouns are very complex and can play multi roles. For instance,

- "*name*" is more about a property rather than a class
- "*registration*" and "booking" is more about an event or a method rather than a class. This is because of word share the polymorphism character of object-orientation. With a little change at the end, (*n.*) "*registration*" can be (*v.*) "*register*" and (*adj.*) "*registered*"; or with no change at all, "*book*" can be (*v.*) (*n.*)

And the use of Pronouns increases this ambiguity.

Some suggest structures can denote relationships, for instance, hierarchies can be derived from the use of the grammatical structure "is a"[27], [32]; partial relationships or composition can be derived from the structure "is part of"; Burg [4] assigns OO elements to the elements of a proposed functional grammar based intermediate language, while Juristo and Moreno[17] also propose that NL structures can be related to OO concepts. Examples of the suggested NL structures are bottom up "A is type of B" (*piano is a type of instrument*), top down "B can be A" (*instrument can be piano*) which are similar to those above. On the other hand, the structure of the sentence in NL is also complex and ambiguous. For instance,

- *"People eat meat and vegetables."*
- *"People eat and drink."*

Above parallel structures, sharing the same subject verb / subject. The clause structure is even more complex and ambiguous.

Overall, NL is highly informal in nature, with speakers and writers inventing new forms and combinations of language. This inventiveness can be seen especially in the continuous evolution of nouns. Existing NLP systems can provide general assistance in analysing requirements. However, such approaches very much depend on the quality of input documents, and lack the ability to capture unconscious requirements [26], i.e. requirements that are too obvious to need documenting. However, NL will certainly continue to play a crucial role, as long as customers are involved in the RE process.

3. Proposed Processing Algorithm

Our intention is to provide thorough guidance for communicating to customers, with machine assistance, throughout an incremental and iterative process of analysis and design. This guidance employs OO concepts that are closer to the real world, and integrates requirements elicitation and analysis to lead customers to express their problems and wishes. Furthermore, this approach supports UML diagram generation for requirements analysts.

The guidance consists of following parts:

(i) Providing guidance so that customers know what information to provide;
(ii) Ensuring that the system knows what customers have said and mean;
(iii) Ensuring that what customers say is well organized, unambiguous and structured;
(iv) Providing guidance about what else customers need to tell the system.

This guidance is implicit in an algorithm shown in Fig 1 to support the process, and rule set for mapping NL and OO, and for standardization of NL structure, and dialog for communicating to customers to disambiguate. The process starts from step 1, POS tagging on NL based domain description, which identifies the Nouns, Verbs and Pronouns; step 2, the POS identification is simplified in standard sentence structure (Subject – Verb - Object) based on our NL structure standardization rules; step 3, the initial class diagram can be generated accordingly from the simplified NL description; users are involved in step 4 and step 5 to confirm the class of Pronouns and Actors using our dialog.

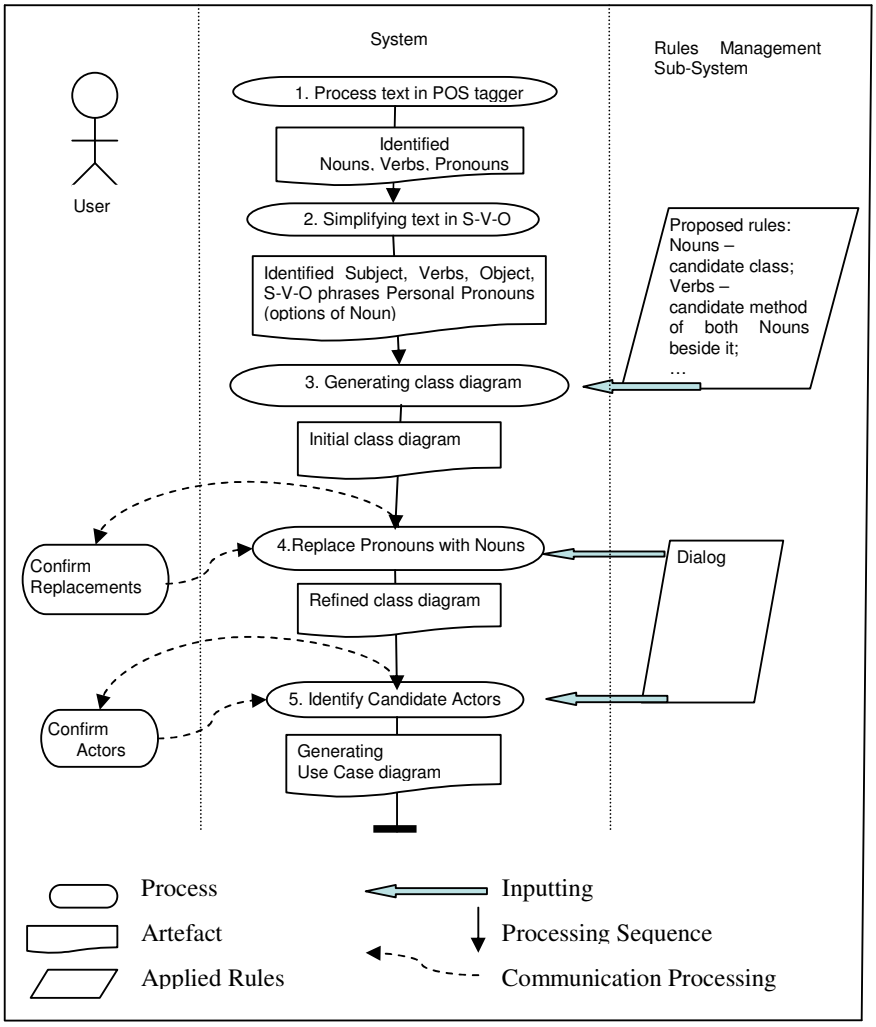

Fig. 1: Proposed Processing Algorithm

We believe that both NL and an object-oriented modelling langue (OOML) contain representations of concrete objects from the real world. The former supports customers to express their problem domain informally and vaguely; the latter supports software engineers in modelling the problem domain formally and precisely. This investigation starts from exploring the mapping between them. Unlike other approaches introduced in section 2, this investigation involves analysing both the mappings between NL elements and OOML elements and the mappings between NL relationships (sentence structure) and OOML relationships.

3.1 Mappings on Elements (MoE)

The elements of NL are seen in terms of word types, including Noun, Pronoun, Personal Pronoun and Verb in this paper. The elements of OOML which are used include class, method, and property. Object is not itself included, since an object is a concrete instance of a class, while we focus on the abstract level.

To explore the mappings between elements of NL and OOML, we propose the following mappings that can be used to generate Class diagrams:

MoE 1: Translating *Nouns* to Classes. This is a long standing step in OO analysis, but needs to be handled very carefully. Additional information must be used to define which nouns really represent useful classes. For instance, the word "*shoe*" on its own does not provide any detailed information of its features (known as properties in OO), but we may be able (as the example below shows) to match properties to it.

MoE 2: Translating *Noun-Noun* to Class-Property according to position. When two nouns appear in sequence in the text, the first Noun is translated to Class and the following Noun is translated to properties of this Class. For instance, "*shoe size*" can be assumed to mean that "*size*" is a property of class "*shoe*".

MoE3: Translating *Subject (S) – Verb(V) – Object(O)* structure to a class diagram with the Subject and Object as classes both sharing the verb as a candidate method.

MoE 4: Translating the lexical *Verb* of a non-personal noun to a Method of this noun.

MoE 5: Translating the lexical *Verb* of a personal noun to a use case (or part of a use case) linked with an actor defined by this noun.

MoE 6: Matching a Noun to a Personal Pronoun as the nouns of previous sentence. This does not provide an unambiguous match and will require user input to disambiguate the actual match. Where one of these Nouns is the same POS , that Noun has highest priority. E.g. where the pronoun is the subject, match it to the subject Noun in the preceding sentence first.

MoE 7: For "S-V-O-O" structure, convert to "S-V-OO", i.e. treat compound nouns as one entity.

3.2 Standardization of NL Structure

In order to simplify the final mapping onto classes and use cases, it is helpful to create a unified structure, in the form of S-V-O triples, throughout the input text. This detracts from the human readability of the text, but helps the machine readability. This is achieved by applying the rules below.

1. Where the Subject precedes an equal parallel structure, split into two (or more) simpler sentences, where the subject is shared in turn by the following parts. Therefore, convert "S-V1-O1-V2-O2", to "S-V1-O1" "S-V2-O3". As an example, "The baker kneads the dough and bakes the bread" becomes, "The baker kneads the dough" "The baker bakes the bread".

2. Where both Subject and Verb precede an equal parallel structure, split into two (or more) simpler sentences, where the Subject -Verb is shared in turn by the following parts. Therefore, convert "S-V-O1-,/and-O2-,/and,-O3" to "S-V-O1" "S-V-O2" "S-V-O3". As an example, "The baker bakes cakes and bread" becomes "The baker bakes cakes" "The baker bakes bread".

3. Verb lead equal parallel structure, Verb is shared by the following parts that also share the potential Subject. Therefore, convert "(Subject)-V-O1-,/and-O2-,/and,-O3…" to "(Subject)-V-O1" "V-O2" "V-O3"

4. Where the sentence contains a verb in a continuous tense, regard this as a complementary structure, where the Subject is shared. Therefore, convert "S-V1-O-V2ing" to "S-V1-O" " S-V2". For instance, "Bakers make bread by baking" becomes "Bakers make bread" "Bakers bake".

3.3 Mapping to Relationships

Having identified the nouns and verbs and created a simplified, canonical noun-verb-noun structure we are in a position to map this information onto classes and their relationships with each other. Two key steps are:

1. If the sentence uses the passive voice, convert "S-Ved" to "V-O(S)". For instance, "*the shoe is ordered by customers*" is reformulated as "*customers order the shoe*"

2. Translating the lexical *Verb* to a Method of the classes formed from the two *Nouns* either side of this *Verb,* so that a relationship can be created between the classes. Thus for the example "The baker bakes bread", we create classes "baker" and "bread" and include the method "bake" in both initially. We note that a relationship exists between these classes.

4. Case Studies: Shoe Company

Here we shall illustrate the application of our proposed algorithm by applying it a case study extracted from text in Lunn [33]. For brevity, we have not included the text as it appears in the original publication, but the unabridged, albeit obfuscated, text is visible in a marked up form in Table 1.

We follow the proposed algorithm, and apply the proposed rules for mappings from section 3. In Step 1, the initial text is processed by the POS tagging system [34] to extract Nouns, Verbs and Pronouns.

Table 1. Identification of POS. Illustrates partially that Nouns are identified by tagger NN1/2, Pronouns are identified by tagger PNP/S) and lexical Verbs are identified by tagger VVI/G/N

Customers[NN2] will[VM0] need[VVI] to[TO0] register[VVI] with[PRP] the[AT0] Odd[AJ0] Shoe[NN1] Company[NN1] to[TO0] make[VVI] orders[NN2].[.]
On[PRP] registration[NN1] ,[,] they[PNP] need[VVB] to[TO0] provide [VVI] name [NN1] and[CJC] address[NN1] ,[,] payment[NN1] details[NN2] ([(] credit[NN1] card[NN1] etc.[AV0]) [)] ,[,]shoe [NN1] sizes[NN2] ,[,] gender[NN1] ,[,] and[CJC] any[DT0] special [AJ0] details[NN2] .[.]
To[TO0] order[VVI] ,[,] customers[NN2] will[VM0] select[VVI] from[PRP] the[AT0] shoe[NN1] range[NN1] .[.]
An[AT0] order[NN1] can[VM0] be[VBI] waiting[VVG] for[PRP] delivery[NN1] to[PRP] the[AT0] Odd[AJ0] Shoe[NN1] Company[NN1] ,[,] waiting[VVG] for[PRP] despatch[NN1] ,[,] waiting[VVG] for[PRP] credit[NN1] clearance[NN1] or[CJC] despatched[VVN] .[.]
When[AVQ] supplies[NN2] arrive[VVB] the[AT0] database[NN1] will[VM0] need[VVI] updating[VVG] .[.]

In the Step 2, the original description is rewritten with the standard structure "*Subject-Verb-Object*", to simplify each sentence. Nouns, Pronouns (options of Noun), Verbs are captured, see Table 2.

Table 2. Simplified NL structured.

S1
Customers[NN2] register[VVI] Shoe[NN1] Company[NN1]
Customers[NN2] make[VVI] orders[NN2]
MoE6
(Customers/Shoe Company)
S2
They[PNP] provide[VVI] name[NN1]
they[PNP provide[VVI] address[NN1]
they[PNP] provide[VVI] payment[NN1]
…
they[PNP] provide[VVI] gender[NN1]
they[PNP] provide[VVI] details[NN2]
S5
Customers[NN2] select[VVI] shoe[NN1] range[NN1]

S3
order[NN1] wait[VVG] delivery[NN1] Shoe[NN1] Company[NN1]
order[NN1] wait[VVG] despatch[NN1]
order[NN1] wait[VVG] credit[NN1] clearance[NN1]
S4
Supplies[NN2] arrive[VVB] database[NN1]
supplies[NN2] updating[VVG]

In the Step3, the initial class diagram is generated applying our MoE rules (Fig. 2.) It shows some classes have no methods no properties and no relationship with other classes, which give us an idea of either those are not a class or not domain relevant. Removing all such classes is a viable strategy. The refined class diagram, which partially exemplifies step 4 in our algorithm, is illustrated in Fig. 3. Step 4 also requires us to replace pronouns with proper nouns. This results in the class diagram shown in Fig. 4. Finally, step5 dictates that we can identify candidate actors for a use case diagram. One such use case is shown in Fig. 5.

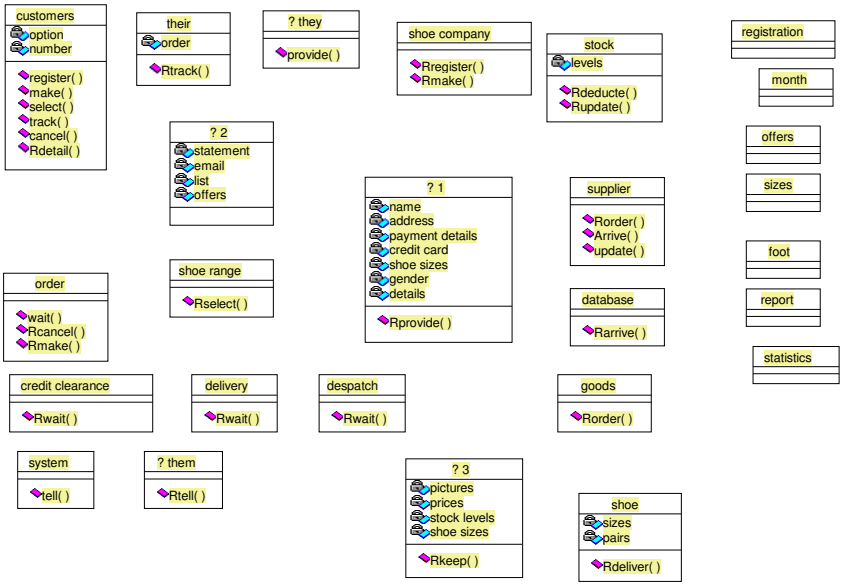

Fig. 2. Initial Class Diagram

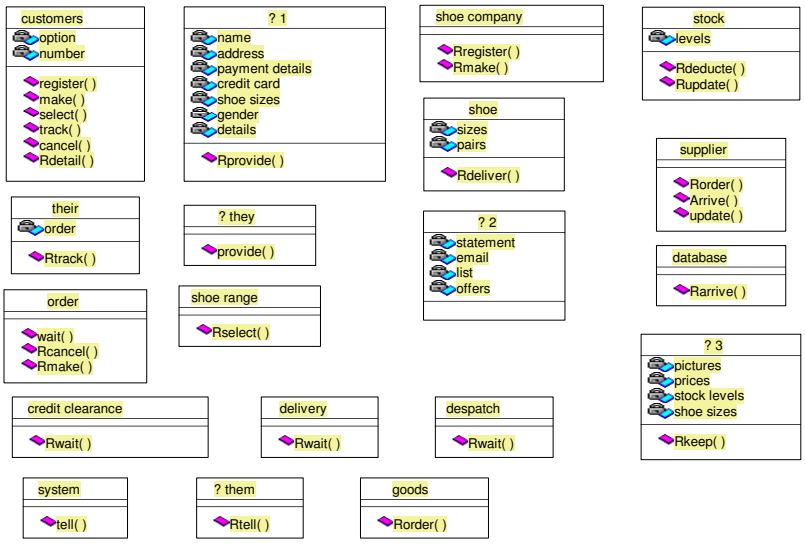

Fig. 3. Refined Class Diagram

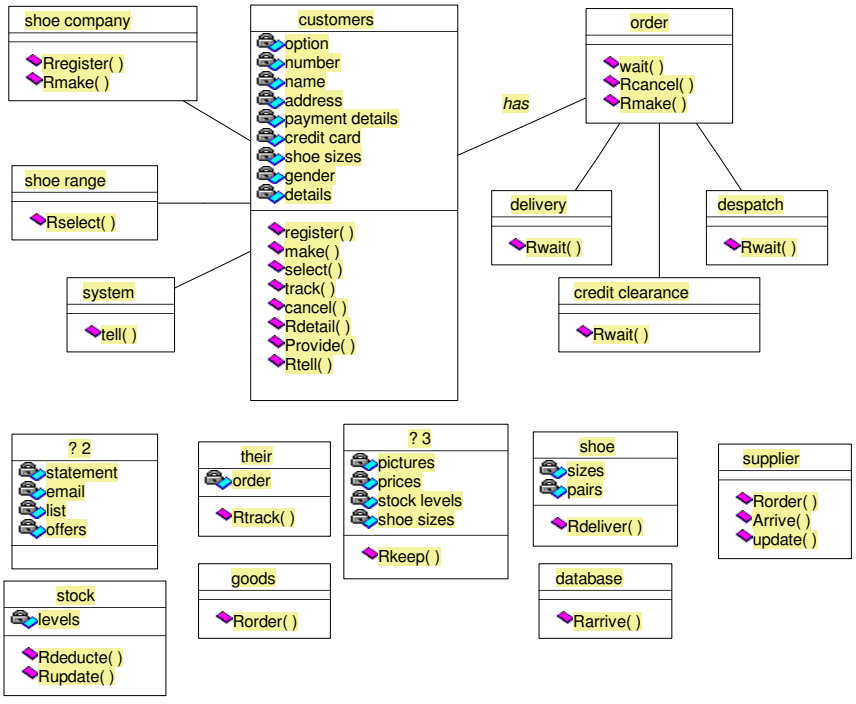

Fig. 4. Class diagram with pronouns replaced

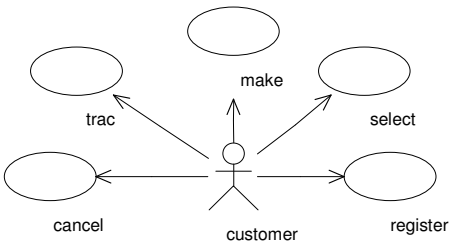

Fig. 5. Use Case Diagram

5. Conclusions

We have presented an approach that will help to structure NL text so that formal modelling becomes possible. Although our investigations are at an early stage, our vision is to implement our ideas in a system that would generate dialogue with a domain expert in order to automatically produce static and dynamic UML representation of the system under consideration. At the same time, we would hope to have some intuitive system representation for the potentially non-UML literate expert to work with and understand.

References

1. Beyer H.R., Holtzblatt K. (1995) Apprenticing With the Customer. Communications of the ACM. Vol. 38(5). Pp. 45-52.
2. Brill E. (1992) A simple rule-based part-of-speech tagger. Proceeding on Third Conference on Applied Natural Language Processing. Trento, Italy.
3. Buchholz H., Dusterhoft A., Thalheim B. (1996) Capturing Information on Behaviour with the RADD_NLI: A Linguistic and Knowledge Base Approach. Proceeding, Second Workshop Application of Natural Language to Information System. IOS Press, Amsterdam. Pp. 185-196.
4. Burg J.E.M. (1997) Linguistic Instruments in Requirements Engineering. IOS Press, Amsterdam.
5. Buzan T., Buzan B. (1995) The Mind Map Book. London: BBC Books.
6. Cutting D., J Kupiec., Pederson J., Sibun P. (1992) A practical part-of-speech tagger. Proceeding on Third Conference on Applied Natural Language Processing. Trento, Italy.
7. Daelemans W., van den Bosch A., Zavrel J., Veenstra J., Buchholz S., Busser B. (1998) Rapid development of NLP modules with memory-based learning. Proceeding on ELSNET in wonderland (ELSNET98). Utrecht. Pp. 105-113
8. Delisle S. Barker K. Biskri I. () Object-Oriented Analysis: Getting Help From Robust Computational Linguistic Tools
9. G. Gazdar, E.H. Klein , G.K. Pullum , & I.A. Sag (1985) Generalized Phrase Structure Grammar. Oxford: Blackwell, and Cambridge, Ma.: Harvard University Press.

10. Garside R., Smith N. (1997) A hybrid grammatical tagger: CLAWS4, in R. Garside, G. Leech, A. McEnery (eds.) Corpus annotation: Linguistic information from computer text corpora. Longman. Pp. 102-121
11. Goguen J.A., Linde C. (1993) Techniques for Requirements Elicitation. In Proceedings, Requirements Engineeing'93, IEEE Computer Society. Pp. 152-164.
12. Harmain H.M. Gaizauskas R. (2000) CM-Builder: An Automated NL-based CASE Tool
13. Hoffman L. (1981) Foundations of Family Therapy. New York: Basic Books.
14. Holtzblatt K., Beyer H.R. (1995) Requirements Gathering: the Human Factor. Communications of the ACM. Vol. 38(5). Pp. 31-32
15. http://www.comp.lancs.ac.uk/computing/research/ucrel/
16. Hutchings A.F., Knox S.T. (1995) Creating Products Customers Demand. Communications of the ACM. Vol. 38(5). Pp. 75-80.
17. Juristo N., Moreno A.M. (2000) How to Use Linguistic Instruments for Object-Oriented Analysis. IEEE Software, May/June, Pp. 80-89
18. Karlsson R., Voutilainen A., Heikkila J., Anttila A. (1995) Constraint Grammar, a language-independent system for parsing unrestricted text. Mouton de Gruyter.
19. Kristen G. (1994) Object Orientation: The KISS-Method: From Information Architecture to Information System. Addison-Wesley, Reading, Mass.
20. Lamsweerde A.V. (2000) Requirements Engineering in the Year 00: A Research Perspective
21. Moore J.M. (2003) Communicating Requirements Using End-User GUI Constructions with Argumentation.
22. Moore J.M. Shipman III F.M. (2001) Requirements Elicitation using Visual and Textual Information
23. Nuseibeh B. Easterbrook S. (2000) Requirements Engineering: A Roadmap.
24. Pohl K. (1993) The Three Dimensions of Requirements Engineering. Proceedings of Fifth International Conference on Advanced Information System Engineering (CaiSE'93), Paris. Pp. 275-292.
25. Rayson P., Garside R., Sawyer P. (1999) Language engineering for the recovery of requirements from legacy documents. REVERE project report, Lancaster University, May 1999
26. Robertson S. (2001) Requirements trawling: techniques for discovering requirements. Int. J. Human-Computer Studies Vol. 55, Pp. 405-421.
27. Rolland C., Proix C. (1992) A Natural Language Approach for Requirements Engineering. Proceeding on Conference Advanced Information Systems Engineering. Springer-Verlag, Manchester, UK. Pp. 257-277.
28. Ryan K. (1993) The Role of Natural Language in Requirements Engineering.
29. S. Fligelstone (1992) Developing a Scheme for Annotating Text to Show Anaphoric Relations. In: G. Leitner (ed.) New Directions in Corpus Linguistics. Mouton de Gruyter. Pp. 153-170
30. Saiedian H., Dale R. (2000) Requirements Engineering: making the connection between the software developer and customer. Information and Software Technology. Elsevier Science.Vol. 42. Pp. 419-428.
31. Sawyer P. Rayson P. Garside R. (2002) REVERE: support for requirements synthesis from documents
32. Tjoa A.M., Berger L. (1993) Transformation of Requirements Specification Expressed in Natural Language into an EER Model. Proceeding: 12th International Conference on ER Approach. Springer-Verlag, Berlin. Pp. 206-217.
33. Ken Lunn (2003) Software Development with UML. PALGRAVE MACMILLAN.
34. http://www.comp.lancs.ac.uk/ucrel/claws/trial.html