The Role of Natural Language in Requirements Engineering

Kevin Ryan

Dept. of Computer Science & Information Systems
University of Limerick
Ireland

Abstract

It is argued that the potential role of natural language processing in the requirements engineering process has been overstated in the past, possibly due to fundamental misunderstandings of the requirements engineering process itself. Since more realistic ambitions are likely to lead to less disappointment in the future, an effort is made to identify some phases and tasks where natural language processing may usefully be applied. It is suggested that the validation of requirements must remain an informal, social process.

1. Introduction

The history of natural language processing (NLP), in relation to the specification of systems and programs has been bedevilled with many unrealistic suppositions and presumptions. Given the critical and expensive nature of current approaches to requirements engineering (RE) the prospect of a support system that would automatically understand a user's needs is, naturally, very appealing. Numerous research projects have proposed to derive and validate system requirements knowledge by means of a natural, or "near natural" conversation with the prospective client [eg 11, 12]. This facility, it is fondly believed, is both feasible and desirable and would make specification of systems both easier and more accurate. Unfortunately this belief is incorrect on both counts. In this brief paper, I wish to assert that natural language processing does not now, nor will it in the foreseeable future, provide a level of understanding that could be relied upon, and even if it could, it is highly questionable that the resulting system would be of great use in requirements engineering.

2 Communication and language

One cause of these misconceptions about NLP may be the view of RE as being essentially a problem in interlanguage communication. The task of the systems analyst, or latterly the knowledge engineer, is portrayed as one of translation between the two specialist worlds of computing and the application domain. There is no doubt that the language of the application domain, and its specialist jargon, is important in systems specification and so the specifier/designer must exhibit some fluency in it if the client is to have confidence in the result. But mere understanding of the syntax or even the specific semantics of a specialised language is not the most crucial factor in bridging the communications gap. Of far greater significance are the unstated assumptions that reflect the shared ("common sense") knowledge of people familiar with the social, business and technical contexts within which the proposed system will operate [eg 1]. The intrinsic difficulty of modelling common sense knowledge is well illustrated by the painfully slow progress in the Cyc project [8]. Neither is it realistic to expect the client to learn another language, the language of computing, so that he or she may understand fully some arcane specification language. This is not realistic for two reasons. Firstly, because there is not just one but many languages to be learned. Secondly, and mainly, because the clients of other professionals (eg lawyers, architects) rely on the professional to interpret their wishes and to translate it as necessary into the specialised jargon. For all these reasons, and with the wisdom of hindsight, we would be well advised to avoid promises of systems that will "understand" language in any meaningful way. However, although the computing professional will not be replaced by a super-intelligent natural language interface, there are still a number of realistic uses for NLP in the RE process.

3. NLP in the initial phases

The supposition, implicit in many NLP-based requirements capture proposals [eg 13], that all requirements for a future system normally exist in textual form, is not borne out in reality. It is true that some information occurs naturally as text, typically process descriptions or predefined procedures, but much more is to be found in diagrams or in the physical reality surrounding the client. To rely solely on the text as a source of knowledge or to expect the client to reduce all his or her demands to a textual form is clearly impracticable. Assuming however that the requirements definition task is

0-8186-3120-1/92 $3.00

being performed by an intelligent human and that a substantial body of machine readable text is available, there is no doubt that tools to scan, search, browse and tag that text could assist in developing a full and accurate statement of needs [eg 9]. This would definitely not imply the automatic understanding of free text.

4. NLP in systems specification

The benefits of formal languages for requirements specification have been well established and, increasingly, formal verification is being required for critical systems [3]. Natural language therefore can not be relied upon in the development of a system specification. At the same time however we must recognise that some important requirements are difficult to quantify and may be impossible to express in current formal specification languages. For example, demands that a user interface be "user-friendly", that a piece of code be "easily maintained" or that a future alternative application be "borne in mind" during a design can be crucial in meeting the business needs of a client. Such demands are likely to be expressed early on in the lifecycle but can easily be "refined out" during the specification phase, mostly because they can not easily be formalised. It would be preferable however, to provide a requirements tracing facility that would tag such requirements at an early stage, to guard against their being lost, and would allow them to be incorporated in the formal specification as natural language comments or links in hyperdocuments. [see eg 7]

It is also the case that descriptive textual and graphical material greatly assists human understanding, not least when it provides an alternative viewpoint on a formal specification as advocated by Finkelstein [6]. Textual material from the early phases of a project, when suitably tagged and indexed, can provide the background information and contextual clues that a human developer or maintainer requires so as to understand the design goals and decisions implicit in the finished system [14].

5. NLP in requirements validation

Validation of requirements is not the inverse of requirements capture. That is to say that the objective is **not** to generate a natural language script that fully describes the system being specified and still less to impute the corporate strategic goals that ultimately motivate the proposed development. Instead, the objective must be a social one, namely to demonstrate convincingly the conformance of the specification to the client's needs. Indeed it can be argued that all proofs are grounded in social processes [8]. By definition an NLP system can only be part of this process. However, we can envisage systems that assist in various ways. A support system might generate scripts of sample cases for the client's approval. The cases could be chosen to reflect both extreme (limiting) and average (expected) situations. Graphics and animation, as used in prototyping systems, would normally be needed to supplement any natural language generated. A second promising approach is critiquing, currently finding favour in expert systems development. In the critiquing approach a system could draw on previously processed cases, possibly stored as schemas, and compare them to the emerging specification. Where partial matches are found the differences are queried and the specifier may then decide whether or not to accept, and act on, the critique [see eg 5, 10]. This approach would be greatly enhanced if a limited NL question and answer facility were provided. This might deal, in a way similar to rule-based expert systems, with questions such as why, what-if, and why not. The answers to such questions could be used by the client, or another domain expert, to test and to validate the knowledge embodied in the formal requirements specification.

6. The future of NLP in requirements engineering.

The complexity of large scale systems is not a result of specifying them accurately and completely but is, rather, a reflection of their inherently complex nature. In this regard, proposed "Just tell me" systems are a dangerous illusion. Neither informal speech nor natural language text is capable of expressing unambiguously the myriad facts and behaviours that are included in large scale systems and this would be true even if we had "solved" the problem of natural language understanding, which we have not.

In fact, understanding in NLP seems to be a fractal-like problem. As each small piece is closely examined it turns out to be even harder than expected and to embody in it many of the problems that were found at the earlier macro stages. While it is conceivable that narrow domain understanding of natural language may be achieved in the medium term it would be foolish to depend on it to solve the RE bottle-neck.

Instead we must accept that systems are, and will increasingly be recognised to be, social organisms, embodying everything from the deterministic microchip to the emotional and personal needs of the people involved, as argued, for example, by Checkland [2]. For future systems we can expect that their technical performance will be mathematically stated and verified but their conformance to need will be judged, over time, within a dynamic and essentially undefinable social context. It is in supporting that social process, and not in supplanting it, that natural language processing will have its proper role.

References

[1] B Adelson & E Soloway, *" The Role of Domain Experience in Algorithm Design"*, IEEE Trans. on Software Eng. V SE-11 No11 Nov.1985 pp 222-241.

[2] P Checkland, *Systems Thinking, Systems Practice* , John Wiley, 1981

[3] B Cohen, *"Justification of Formal Methods for System*

Specification", IEE Software Engineering Journal, (1)1989.

[4] R A de Millo, A J Lipton and A J Perlis, *"Social Processes and Proofs of Theorems and Programs"* , Communications of the ACM, May 1979.

[5] S F Fickas, *"Automating the Analysis Process"*, 4th International Workshop on Software Specification and Design, Calif. 1987

[6] A Finkelstein, *"Multi-party Specifications"*, Proceedings of 5th International Workshop on Software Specification and Design", IEEE Pittsburg.

[7] J Kramer, K Ng, C Potts, K Whitehead, *"Tool Support for Requirements Analysis"*, IEE Software Engineering Journal, (3)1988.

[8] D B Lenat, R V Guha, K Pittman, D Pratt and M Shepherd, *"Cyc: Toward Programs with Common Sense"*, Communications of the ACM, [33][8] August 1990.

[9] P Loucopoulos P & R E M Champion, *"Concept Acquiisition and Analysis for Requirements Acquisition"*, IEE Software Engineering Journal, (2)1990.

[10] B Mathews & K Ryan, *"Requirements Specification using Conceptual Graphs"*, 2nd International CASE Conference, London UK, 1989.

[11] P Punchello, P Torrigiani, F Pietri, R Burlon,B Cardile, M Conti, ; *"ASPIS : A Knowledge-Based CASE Environment"*, IEEE Software, (5)(2) March 1988.

[12] C Rolland & C Proix, *"A Natural Language Approach to Requirements Engineering"*, 4th International CAiSE Conference, Manchester UK, 1992

[13] M Saeki, H Horai, H Enomoto, *"Softtware Development Process from Natural Language Specification"*, 11th International Conference on Software Engineering, 1989.

[14] D Wile, *"Program Developments: formal explanations of implementations"*, Communications of the ACM V26 No 11, November 1983.