The REVERE project: experiments with the application of probabilistic NLP to systems engineering

Paul Rayson[1], Luke Emmet[2], Roger Garside[1] and Pete Sawyer[1]

[1]Lancaster University, Lancaster, UK. LA1 4YR
{paul, rgg, sawyer}@comp.lancs.ac.uk
[2]Adelard, Coborn House, 3 Coborn Road, London UK. E3 2DA
loe@adelard.co.uk

Abstract. Despite natural language's well-documented shortcomings as a medium for precise technical description, its use in software-intensive systems engineering remains inescapable. This poses many problems for engineers who must derive problem understanding and synthesise precise solution descriptions from free text. This is true both for the largely unstructured textual descriptions from which system requirements are derived, and for more formal documents, such as standards, which impose requirements on system development processes. This paper describes experiments that we have carried out in the REVERE[1] project to investigate the use of probabilistic natural language processing techniques to provide systems engineering support.

1. Introduction

Despite natural language's well-documented shortcomings as a medium for precise technical description, its use in software-intensive systems engineering [1] (henceforth referred to simply as systems engineering) remains inescapable. The products of the systems engineering process (requirements specifications, acceptance test plans, etc.) almost always have to employ natural language in order to describe the desired system properties for a heterogeneous readership. Recognising this, several researchers [2, 3, 4, 5, 6, 7] have used natural language processing (NLP) tools for the analysis, paraphrasing and quality evaluation of documents produced by the systems engineering process. There has been little attention paid to the main focus of this paper, however: tools to assist the analysis of natural language inputs to systems engineering.

In developing the requirements for a system, systems engineers face the need to understand large volumes of information. This information is almost always expressed in natural language. This is true of information elicited from human stakeholders (users, customers, domain experts, marketing experts, etc.), and information derived from technical and business documents. Systems engineers

[1] REVerse Engineering of REquirements. EPSRC Systems Engineering for Business Process Change (SEBPC) programme project number GR/MO4846. Further details can be found at: http://www.comp.lancs.ac.uk/computing/research/cseg/projects/revere/

collect this information to build up their understanding of the system requirements and the constraints on the system and on the development process. Hence, for example, requirements information elicited from stakeholders may need to be analysed in the context of information about the system's operational environment and standards that regulate the system's application domain. A systems engineer for a railway signalling system, for example, might be faced with having to understanding documents describing complex signalling control processes, statements of requirements from many different stakeholders, specifications of equipment to which the system will have an interface, engineering standards and railway legislation.

Whatever the source of the information, its analysis is a difficult task that requires skill, domain knowledge and experience and is almost completely unsupported by tools. This is a particular problem where the volume of the information is large and the sources of information diverse.

In this paper we describe our work in the REVERE project, where we are investigating the feasibility of using probabilistic NLP techniques to improve this situation. The work is illustrated with two examples: analysis of a large corpus of transcripts and field notes of an ethnographic study of an air-traffic control (ATC) application (reported elsewhere [8]), and analysis of a new safety-critical systems standard. These both serve to illustrate the principles involved. In combination, the two examples illustrate the promise of the approach and highlight some issues that require further attention for the approach to be widely exploitable.

2. Natural language in Systems Engineering

Systems engineering is concerned mainly with identification and analysis of the system requirements, identification of a configuration of components (the system architecture) that will satisfy the requirements, and verification that once completed and integrated, the configuration of components does meet the requirements. To do this, the system requirements must be acquired and analysed. These requirements are always constrained by many factors that include budgetary limits, the operational environment, technical feasibility, the need for standards compliance [9] and many others. Hence, many factors must be understood and balanced by a systems engineer. Natural language invariably plays a large part in the systems engineering process and this fact has attracted sporadic interest in the employment of NLP techniques in systems engineering. Work has focused on both the products of the process and inputs to the process.

2.1 Natural language products of the systems engineering process

Products of the process include specifications of the requirements and test plans. The use of natural language is needed to enable these documents to be read by a heterogeneous readership that includes both engineers and customers. However, it is hard to describe complex concepts simply, clearly and concisely in natural language. In recognition of this, several research projects [2, 4, 6, 7] have investigated the synthesis of conceptual models (object models, data-flows, etc) from natural language descriptions. These have employed rule-based NLP and so require documents written using natural language subsets as their input.

2.2 Natural language inputs to the systems engineering process

The inputs to the process include human-sourced information and formal documents. Human-sourced information is information elicited directly from system stakeholders. It may comprise, for example, unstructured textual description and transcripts of user interviews. It is one of the raw materials from which the system requirements are derived. Formal documents include standards, process descriptions, user manuals and specifications of other systems. They are often needed to complement human-sourced information as sources of additional requirements and constraints. They are also a crucial source of domain knowledge needed to interpret the requirements and constraints. The crucial point is that a great deal of intellectual work is needed to identify a coherent and cohesive set of system requirements. These are never pre-formed but have to be synthesised from a variety of sources that contain information which may be poorly structured, contradictory, at varying levels of detail and of uncertain relevance.

The systems engineer must use whatever information resources are available to derive the requirements [10]. This typically entails an iterative process of inferring key abstractions (stakeholders, roles, tasks, domain objects, etc.) and verifying these against the structure and behaviour of existing systems. Hence, while work concerned with products of the systems engineering process relies upon a pre-existing formulation of the system requirements, work on inputs to the process must cope with much messier natural language text.

Dan Berry and colleagues have studied this problem over a number of years [11, 12, 13]. Their work is based on the use of pattern matching techniques to extract abstractions. The frequency with which the abstractions occur within the text is taken as an indication of the abstractions' relevance. The authors of the work recognise this assumption has been challenged by work on automatic abstraction in other domains, but argue that it appears to be valid in the context of requirements analysis.

Both Berry's work, and similarly motivated work by [14], explicitly recognise that NLP cannot automate the system engineer's job. The system engineer still has to read the myriad documents. Instead, the tools seek to mitigate the problems of information overload. They do this by compensating for human weakness (such as attention lapses due to tiredness) by helping to flag abstractions that would repay detailed manual analysis. This is a refreshingly realistic view of the potential for NLP that is informed by a real understanding of the problems of system engineering, and one that forms the starting point for our own work.

3. The REVERE project

Like Berry's work, the REVERE project is concerned with supporting systems engineers faced by the need to extract information from large volumes of unstructured text. We do not believe that it is possible to fully automate this. We do, however, believe it is feasible to help provide a 'quick way in' to unfamiliar documents, in order to help focus system engineers' attention on candidates for important abstractions. An important requirement for REVERE is that the approach must scale. This immediately rules out purely rule-based NLP because of the diversity of the documents that may have to be analysed. We do not believe that the pattern-matching alone can offer sufficient coverage or be sufficiently efficient for the kind of interactive support that we aim to provide.

Our approach is therefore to exploit *probabilistic* NLP techniques. Instead of attempting to model the grammar of a natural language as a set of rules, the class of probabilistic tools that we are interested in classifies words on the statistical likelihood of them being a member of a particular syntactic or semantic category in a particular context. The probabilities are derived from large corpora of free text which have already been analysed and 'tagged' with each word's syntactic or semantic category. Extremely large corpora have been compiled (the British National Corpus consists of approximately 100 million words [15]). For some levels of analysis, notably part-of-speech tagging, probabilistic NLP tools have been able to achieve levels of accuracy and robustness that rule-based techniques cannot approach.

Probabilistic tools do not attempt to automate understanding of the text. Rather, they extract interesting properties of the text that a human user can combine and use to infer meaning. Evidence from other domains suggests that such tools can effectively support analysis of large documents. For example, in [16] probabilistic NLP tools were used to quickly confirm the results of a painstaking manual discourse analysis of doctor-patient interaction. In this application, they were also able to reveal information that had not been discovered manually.

Probabilistic NLP techniques meet the requirement for scalability. The execution time of the tagging process varies approximately linearly with the document size. Once the text has been tagged, retrieval and display tools are needed to allow the user to interact with the document. These use the tags to provide views on the document that reveal interesting properties and suppress the bulk of text. They do this in a way that is largely independent of the size of the document. Hence the user is protected from information overload by being able to be selective about the information they want to extract.

4. The REVERE tools

We have adapted and experimented with a set of existing NLP tools developed at Lancaster for the processing of English language text. The most important of these is CLAWS [17]. CLAWS uses a statistical hidden Markov model technique and a rule-based component to identify the parts-of-speech (POS) of words to an accuracy of 97-98%. One obvious application of this in a system engineering context is the identification of modal verbs such as 'shall', 'must', 'will', 'should', etc. Expressions of need, desire, etc., consistent with user or system requirements can therefore be located in a document very easily and without the need to construct complex regular expressions or search templates. Even this basic level of analysis goes beyond what is provided by the current generation of requirements and document management tools that are becoming widely used by systems engineers.

A semantic analyser [18] uses the POS-tagged text to assign semantic tags that represent the general sense field of words from a lexicon of single words and an idiom list of multi-word combinations (e.g. ‘as a rule’). These resources contain approximately 52000 words or idioms and classify them according to a hierarchy of semantic classes. For example, the tag *A1.5.1* represents words or idioms meaning *Using*, which is a subclass of *general and abstract terms*. Words that would be assigned this tag (in the appropriate POS context) include *user*, *end-user* and *operator*. Similarly, the tag X2.4 is a subclass of *Psychological actions*, *states and processes* and would be assigned to terms meaning *Investigate*, such as *search*, *browse* and *look for*. The tagset has been crafted from analysis of a large corpus of English text. One of the things we are investigating is the extent to which it would be applicable to the technical domain(s) of systems engineering.

These tools are integrated into a retrieval and display tool called WMATRIX (a development of XMATRIX [19]). This embodies a process model that leads the user through a sequence of steps needed to apply the POS and semantic tagging and other types of analysis that, once complete, allow the user to interact with abstractions of the text. Many of these abstractions are provided by frequency profiling. At the most basic, this produces a simple concordance of individual words and displays them in the context of their surrounding text. Frequency profiling becomes more useful when a semantically tagged document can be compared against a *normative corpus*: a large representative body of pre-tagged text. Comparison with the normative corpus allows information to be extracted from a document by searching for statistically significant deviations from the frequency norm suggested by the corpus. This exploits the tendency for words or expressions to have different semantic profiles in different domain contexts. A general usage normative corpus is likely to reveal many of the dominant domain entities and roles as words or expressions that occur with a frequency that deviates grossly from the norm. These are the kinds of abstractions that an engineer must identify when analysing the requirements for a system, since they help build up a picture of what the system must do and of the people and objects in the system's environment.

With complex systems, there are often so many diverse requirements that some separation of concerns needs to be imposed on the problem space in order to understand the requirements and roles of different stakeholders. This is sometimes supported by the use of viewpoints on a system. A viewpoint can be thought of as a *partial specification* [20]; a subset of the system requirements that represent those requirements unique to a particular stakeholder's perspective of what the system must do. The engineer can use WMATRIX to search a document for roles, since roles often correspond to stakeholders. By finding the set of candidate roles and viewing where they occur in the body of the text, it is possible for the engineer to verify whether the roles are important and build up an understanding of how they interact with the system's environment.

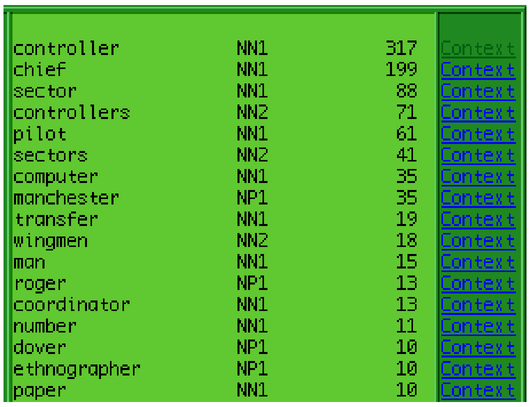
To provide a snapshot of the results so far, and illustrate some of the key issues, we now briefly describe two examples. These are an analysis of the requirements for an air traffic control system and an evaluation of a new standard for the development of safety-critical systems.

5. Example 1: Air traffic control

The target documents are field reports of a series of ethnographic studies at an air traffic control (ATC) centre. This formed part of a study of ATC as an example of a system that supports collaborative user tasks [8]. The documents consist of both the verbatim transcripts of the ethnographer's observations and interviews with controllers, and of reports compiled by the ethnographer for later analysis by a multi-disciplinary team of social scientists and systems engineers. The field reports form an interesting study because they exhibit many characteristics typical of information. The volume of the information is fairly high (103 pages) and the documents are not structured in a way (say around business processes or system architecture) designed to help the extraction of requirements. Two stages in the analysis are shown: a search for candidate roles; and an analysis against a normative corpus.

5.1 Role analysis

Roles are likely to emerge from several kinds of analysis using WMATRIX. Corpus analysis will often reveal a role as a noun with a semantic category that has a frequency distribution that varies significantly from that of the normative corpus. In formal documents certain parts of speech are often associated with roles (and other entities of the domain). In a standards document, for example, modal verbs often reveal roles by their context, such as: "The **Developer shall** define the safety test to be conducted…"

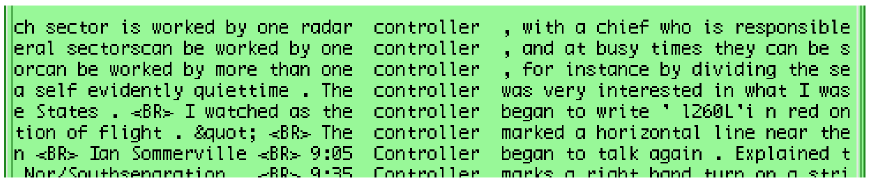


controller	NN1	317	Context
chief	NN1	199	Context
sector	NN1	88	Context
controllers	NN2	71	Context
pilot	NN1	61	Context
sectors	NN2	41	Context
computer	NN1	35	Context
manchester	NP1	35	Context
transfer	NN1	19	Context
wingmen	NN2	18	Context
man	NN1	15	Context
roger	NP1	13	Context
coordinator	NN1	13	Context
number	NN1	11	Context
dover	NP1	10	Context
ethnographer	NP1	10	Context
paper	NN1	10	Context

Fig. 1. Candidate roles in air traffic control

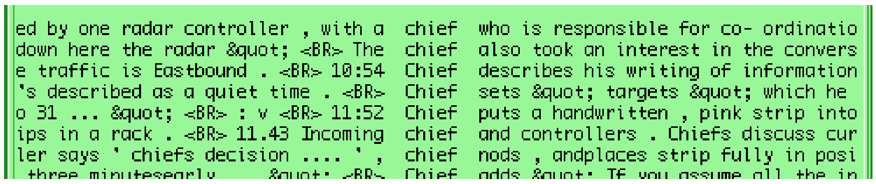
In this early stage of our analysis of the ATC field reports, an initial role analysis was performed by a simple combination of POS analysis and regular expressions. These are often revealed as human agent nouns with common endings for job-titles (such as 'er' or 'or', 'et' or 'ot', 'man' or 'men', etc) and adjectives that are commonly used without their accompanying noun ('head', 'chief', 'sub', etc.). Using this, the candidate roles that occur most frequently in the ATC document (and hence imply significance) are shown in figure 1.

Figure 1 shows a mixture of words that are clearly not roles (e.g. sector, computer, manchester, roger), but also some that are: controller, chief, wingmen, coordinator and ethnographer. Ethnographers are roles in the analysis rather than the application domain, but the other three are all roles or stakeholders, with their own requirements or viewpoint on ATC. Theories about whether the candidate roles are significant or not can be tested by viewing their context. Figures 2 and 3 show examples of the contexts in which occurrences of 'controller' and 'chief' occur.



```
ch sector is worked by one radar  controller  , with a chief who is responsible
eral sectorscan be worked by one  controller  , and at busy times they can be s
orcan be worked by more than one  controller  , for instance by dividing the se
a self evidently quiettime . The  controller  was very interested in what I was
e States . <BR> I watched as the  controller  began to write ' 1260L'i n red on
tion of flight . &quot; <BR> The  controller  marked a horizontal line near the
n <BR> Ian Sommerville <BR> 9:05  Controller  began to talk again . Explained t
```

Fig. 2. References to the role name *controller*



```
ed by one radar controller , with a  chief  who is responsible for co- ordinatio
down here the radar &quot; <BR> The  chief  also took an interest in the convers
e traffic is Eastbound . <BR> 10:54  Chief  describes his writing of information
's described as a quiet time . <BR>  Chief  sets &quot; targets &quot; which he
o 31 ... &quot; <BR> : v <BR> 11:52  Chief  puts a handwritten , pink strip into
ips in a rack . <BR> 11.43 Incoming  chief  and controllers . Chiefs discuss cur
ler says ' chiefs decision .... ' ,  chief  nods , andplaces strip fully in posi
```

Fig. 3. References to the role name *chief*

The examples illustrate that by browsing the roles, the systems engineer can impose a viewpoint on the mass of information that allows them to build up a picture of the corresponding stakeholders' activities within the system's application domain. The first lines in each of figure 2 and 3, for example, include an explanation of both roles' responsibilities. Of course, there is also much 'noise' and sometimes synonyms are used for a single role. However, by using this technique, we have isolated sets of requirements for each of the roles identified.

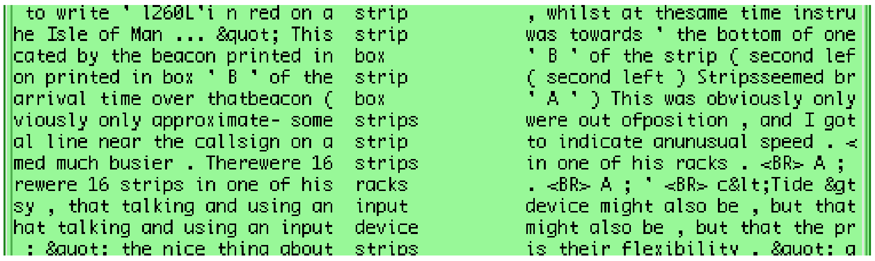
5.2 Corpus analysis

The motivation for corpus analysis is that entities that are significant to the application domain will be revealed by the relative frequency of their appearance in the text when compared against a normative corpus. The normative corpus that we used was a 2.3 million-word subset of the BNC derived from the transcripts of spoken English. Using this corpus, the most over-represented semantic categories in the ATC field reports are shown in table 1. The log-likelihood figure is a statistical measure of deviation from the word's frequency deviation from the normative corpus. The higher the figure, the greater the deviation.

Table 1. Over-represented categories in ATC field reports

Log-likelihood	Semantic tag	Word sense (examples from the text)
3366	S7.1	power, organising (‘controller’, ‘chief’)
2578	M5	flying (‘plane’, ‘flight’, ‘airport’)
988	O2	general objects (‘strip’, ‘holder’, ‘rack’)
643	O3	electrical equipment (‘radar’, ‘blip’)
535	Y1	science and technology (‘PH’)
449	W3	geographical terms (‘Pole Hill’, ‘Dish Sea’)
432	Q1.2	paper documents and writing (‘writing’, ‘written’, ‘notes’)
372	N3.7	measurement (‘length’, ‘height’, ‘distance’, ‘levels’, ‘1000ft’)
318	L1	life and living things (‘live’)
310	A10	indicating actions (‘pointing’, ‘indicating’, ‘display’)
306	X4.2	mental objects (‘systems’, ‘approach’, ‘mode’, ‘tactical’, ‘procedure’)
290	A4.1	kinds, groups (‘sector’, ‘sectors’)

With the exception of Y1 (an anomaly caused by an interviewee’s initials being mistaken for the PH unit of acidity), all of these semantic categories include important objects, roles, functions, etc. in the ATC domain. The frequency with which some of these occur, such as M5 (flying), are unsurprising. Others are more revealing about the domain of ATC. Figure 4 shows some of the occurrences of the semantic category O2 (general objects) being browsed by a user of WMATRIX. The important information revealed here is the importance of 'strips' (formally, 'flight strips'). These are small pieces of cardboard with printed flight details that are the most fundamental artefact used by the air traffic controller to manage their air space. Examination of other words in this category also reveal that flight strips are held in 'racks' to organise them according to (for example) aircraft time-of-arrival.



```
 to write ' 1260L'i n red on a  strip       , whilst at thesame time instru
he Isle of Man ... &quot; This  strip       was towards ' the bottom of one
cated by the beacon printed in  box         ' B ' of the strip ( second lef
on printed in box ' B ' of the  strip       ( second left ) Stripsseemed br
arrival time over thatbeacon (  box         ' A ' ) This was obviously only
viously only approximate- some  strips      were out ofposition , and I got
al line near the callsign on a  strip       to indicate anunusual speed . <
med much busier . Therewere 16  strips      in one of his racks . <BR> A ;
rewere 16 strips in one of his  racks       . <BR> A ; ' <BR> c&lt;Tide &gt
sy , that talking and using an  input       device might also be , but that
hat talking and using an input  device      might also be , but that the pr
 : &quot; the nice thing about  strips      is their flexibility . &quot; a
```

Fig. 4. Browsing the semantic category O2

Similarly, browsing the context for Q1.2 (paper documents and writing) would reveal that controllers annotate flight strips to record deviations from flight plans, and L1 (life, living things) would reveal that some strips are 'live', that is, they refer to aircraft currently traversing the controller's sector. Notice also that the semantic categories' deviation from the normative corpus can also be expected to reveal roles. In this example, the frequency of S7.1 (power, organising) confirms the importance of the roles of 'controllers' and 'chiefs', identified by the role analysis described above.

Using the REVERE tools does not automate the task of identifying abstractions, much less does it produce fully formed requirements that can be pasted into a specification document. Instead, it helps the engineer quickly isolate potentially significant domain abstractions that require closer analysis. It cannot guarantee completeness. For example, some important abstractions may be overlooked because their frequency of occurrence in the document being analysed is close to that of the normative corpus. In addition, word semantics may be domain-specific leading to them being tagged wrongly (in the domain context), making it easy to overlook their significance. This issue is explored in the next example.

6. Example 2: A Standards Document

This example explores the REVERE tools' utility for assessing the impact of standards in systems engineering. The example is based upon the publication of a new national standard for the procurement of safety-critical military systems (approximately 21000 words). In all domains, systems engineers are constrained by regulations, standards and best operating practice, all of which may be documented in a number of sources. Safety-critical systems engineering processes tend to be particularly tightly regulated and document-centric. Systems engineers working in safety critical domains therefore need to acquire a good awareness of the standards landscape and how individual standards apply to different projects. This obviously requires a lot of reading and interpretation of standards documents. This might be to:

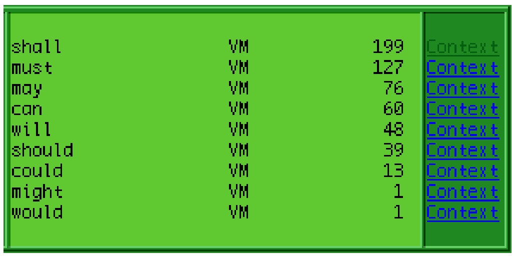
- keep abreast of emerging standards in their domain in order to monitor international best practice;
- anticipate the effect of new standards on future international, national and sector (such as defence) standards;
- build competence for possible future work in the market for which the standard was written;
- identify a set of key attributes to assess against the standard to establish compliance.

Systems engineering standards are like *meta* requirements documents - they specify generic requirements on the development processes and their products within a domain. In contrast to the class of documents used in the ATC experiment, standards tend to be strongly structured and highly stylised in the lexical and syntactic conventions used, and in the semantics attached to certain terms. In particular, modal verbs are frequently used to signify where system properties or development practices are mandatory or advisory.

Our analysis of the standard had two goals: to determine the weight given to different development practices mandated or advised by the standard; and to identify the roles of people who would use or be regulated by the standard. These were performed using POS and role analysis.

6.1 POS analysis

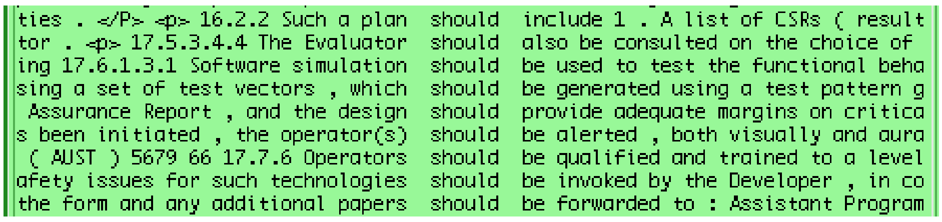
WMATRIX allows the engineer to isolate words that are assigned any given POS tag. In standards documents, words given the modal verb tag ('VM') are of particular interest. Figure 5 illustrates the frequency profile of all the standard's modal verbs.



shall	VM	199	Context
must	VM	127	Context
may	VM	76	Context
can	VM	60	Context
will	VM	48	Context
should	VM	39	Context
could	VM	13	Context
might	VM	1	Context
would	VM	1	Context

Fig. 5. Modal verbs' occurrence in the standard

The most common modal verb in the standard is 'shall'. In standards (and other types of specification documents) a convention is often adopted in which 'shall' is used as a keyword in clauses that denote mandatory requirements. The convention often extends to using other modal verbs to denote weaker obligations. Any of the modal verbs may also appear in normal English usage in informal, descriptive sections of the document.



```
ties . </P> <p> 16.2.2 Such a plan  should  include 1 . A list of CSRs ( result
tor . <p> 17.5.3.4.4 The Evaluator  should  also be consulted on the choice of
ing 17.6.1.3.1 Software simulation  should  be used to test the functional beha
sing a set of test vectors , which  should  be generated using a test pattern g
 Assurance Report , and the design  should  provide adequate margins on critica
s been initiated , the operator(s)  should  be alerted , both visually and aura
 ( AUST ) 5679 66 17.7.6 Operators  should  be qualified and trained to a level
afety issues for such technologies  should  be invoked by the Developer , in co
the form and any additional papers  should  be forwarded to : Assistant Program
```

Fig. 6. Occurrences of 'should' in the standard

Once identified, the modal verbs were browsed in their context within the standard to build up a picture of the conventions used in the standard. This was essentially to see if their usage complied with our expectations. We started by browsing the occurrences of 'shall', to distill a view of the mandatory requirements of the standard. As expected, most of the occurrences of 'shall' in the standard occur in formal clauses representing mandatory requirements. However, documents cannot always be relied upon to use modal verbs consistently. This is illustrated in figure 6, which shows a subset of the 39 occurrences of 'should'. Normally, where 'should' appears in a formal clause of a standard, it is used to denote an advisory requirement. However, to avoid any ambiguity, it is common for a standards document to contain a definition of the convention used to differentiate between mandatory and advisory requirements. Our suspicion was aroused when browsing the lists of modal verb contexts failed to reveal such any such definition. We then discovered the following occurrence of 'should' in the standard: "*17.7.6 Operators **should** be qualified and trained …*". This turns out to represent a mandatory requirement and hence represents a violation of the lexical convention that we had assumed for the standard.

Our exploration of the document's use of modal verbs revealed mistaken assumptions about the conventions used by the document. We eventually isolated the clause in the standard that defined the conventions used: mandatory requirements were indicated by bold text and advisory requirements were written in plain text. Our tools were unable to detect this easily because the tools currently do not include any formatting analysis.

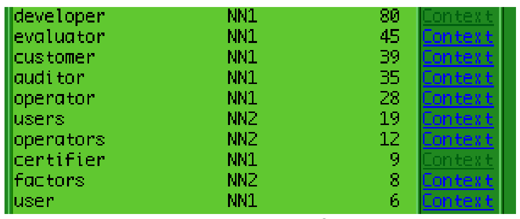
Identifying the paragraph that defined the convention was complicated because we had to find words with semantic tags that corresponded to *mandatory* and *advisory*. This requires experimentation with several tags:

- S6 'Obligation and necessity'
- S8 'Helping/hindering'
- X7 'Wanting'

The terminology in the standard for a mandatory requirements was simply the word 'requirement'. This was tagged X7. The terminology used for an advisory requirement was the word 'guidance'. This was the tag S8. Clearly, in a standards document context, these two terms, as well as others such as 'mandatory' and 'advisory' should all be given the same tag. This revealed a problem in the use of a tagset derived from the analysis of general English for the analysis of technical or formal documents.

6.2 Role analysis

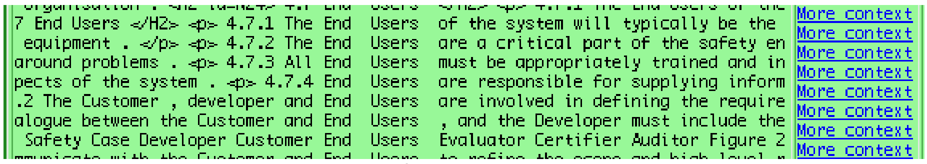
In this stage of the analysis we were interested in discovering the roles identified for people who would apply or be constrained by the standard. This is important because different roles will have different viewpoints on the standard. For example, developers are typically interested in how to comply with a standard, while assessors are interested in how to evaluate a developer's work against a standard. In its introduction section, the standard identifies a set of roles as follows: "This standard applies to the Customer, Developer, Auditor and Evaluator of the system". By applying the REVERE tools' role analysis we identified several other candidates for roles in the text. The most commonly occurring are illustrated in figure 7.



developer	NN1	80	Context
evaluator	NN1	45	Context
customer	NN1	39	Context
auditor	NN1	35	Context
operator	NN1	28	Context
users	NN2	19	Context
operators	NN2	12	Context
certifier	NN1	9	Context
factors	NN2	8	Context
user	NN1	6	Context

Fig. 7. Candidate roles identified in the standard

Of course, many of these will be synonyms or special cases of the primary roles explicitly identified by the document. For example, the (prime) contractor is treated as the developer. However, others that are not explicitly identified do appear to be significant. For example, Figure 8 illustrates occurrences of references to users (or end users) in the document. While these may not be directly affected by the standard, there is an implication that they will be involved in a development project that complies to the standard.



```
[illegible]                                                                     More context
7 End Users </H2> <p> 4.7.1 The End  Users  of the system will typically be the  More context
 equipment . </p> <p> 4.7.2 The End  Users  are a critical part of the safety en  More context
around problems . <p> 4.7.3 All End  Users  must be appropriately trained and in  More context
pects of the system . <p> 4.7.4 End  Users  are responsible for supplying inform  More context
.2 The Customer , developer and End  Users  are involved in defining the require  More context
alogue between the Customer and End  Users  , and the Developer must include the  More context
 Safety Case Developer Customer End  Users  Evaluator Certifier Auditor Figure 2  More context
[illegible]                                                                     More context
```

Fig. 8. Occurrences of the role 'users' in the standard

7. Conclusions

Our work on the REVERE project is motivated by the potential to provide the means for rapid analyses of complex and voluminous free text that often forms an input to systems engineering projects. The need for rigour in systems engineering means that a deep understanding of key information has to be acquired by the systems engineer. However, faced with a large volume of information of uncertain relevance and quality, tools that supported rapid but fairly shallow analysis would be of potential value to systems engineers. Although shallow, the analysis supported by such tools would help the systems engineer to identify where they needed to focus their attention. This would mitigate attentional losses caused by information overload.

The paper has described two experiments using a set of tools that we have developed. These include POS and semantic taggers and are integrated by an end-user tool called WMATRIX. The experiments (a third is reported elsewhere [21]) reveal both promise for the approach and limitations of our existing tools.

The principal defects appear to be caused by the need to tailor our semantic tagger to a technical domain. Despite this, the results of our work to date lead us to believe that, just as probabilistic NLP has emerged in commercial products in other domains (notably word processing and speech recognition), it also has the potential to form a key component of next-generation systems engineering tools.

It is crucial to the understanding of our work that we do not aim for completeness; systems engineering will always rely upon human skill and expertise. However, by rejecting as impossible the use of NLP for fully automating any aspect of systems engineering, we are able to focus on our goal of supporting systems engineers' manual analysis of documents. Initial results in a variety of systems engineering domains suggests that the REVERE tools are effective in helping engineers identify crucial domain abstractions and test theories about what abstractions exist, their importance and how they are inter-related in the domain.

References

1. Stevens, R., Brook, P., Jackson, K., Arnold, S.: Systems engineering: coping with complexity, Prentice-Hall, 1998.

2. Rolland, C., Proix, C.: A Natural Language Approach for Requirements Engineering, Lecture Notes in Computer Science, Vol. 593, 1992.
3. Burg, J., van de Riet, R.: COLOR-X: Object Modeling profits from Linguistics, Proc. Second International Conference on Building and Sharing of Very Large-Scale Knowledge Bases (KB&KS'95), Enschede, The Netherlands, 1995.
4. Cyre, W., Thakar, A.: Generating Validation Feedback for Automatic Interpretation of Informal Requirements, in Formal Methods in System Design, Kluwer, 1997.
5. Rosenburg, L., Hammer, T., Huffman, L.: Requirements, Testing & Metrics, Proc. 15th Annual Pacific Nothwest Software Quality Conference, Utah, USA, 1998.
6. Ambriola, V., Gervasi, V.: Experiences with Domain-Based Parsing of Natural Language Requirements, Proc. 4th International Conference NLDB '99, Klagenfurt, Austria, 1999.
7. Steuten, A., van de Reit, R., Dietz, J.: Linguistically Based Conceptual Modeling of Business Communication, Proc. 4th International Conference NLDB '99, Klagenfurt, Austria, 1999.
8. Bentley R., Rodden T., Sawyer P., Sommerville I, Hughes J., Randall D., Shapiro D.: Ethnographically-informed systems design for air traffic control, Proc. CSCW '92, Toronto, November 1992.
9. Emmerich, W., Finkelstein, A., Montangero, C., Antonelli, S., Armitage, S., Stevens, R.: Managing Standards Compliance, IEEE Trans. Software Engineering, 25 (6), 1999.
10. Butler, K., Esposito, C., Hebron, R.: Connecting the Design of Software to the Design of Work, Communications of the ACM. 42 (1), 1999.
11. Berry, D., Yavne, N., Yavne, M.: Application of Program Design Language Tools to Abbott’s method of Program Design by Informal Natural Language Descriptions, Journal of Software and Systems, 7, 1987.
12. Aguilera, C., Berry, D.: The Use of a Repeated Phrase Finder in Requirements Extraction, Journal of Systems and Software, 13 (9), 1990.
13. Goldin, L., Berry, D.: AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation, Automated Software Engineering, 4, 1997.
14. Fliedl, G., Kop, C., Mayr, H., Mayerthaler, W., Winkler, C.: Linguistically Based Requirements Engineering - the NIBA Project, Proc. 4th International Conference NLDB '99, Klagenfurt, Austria, 1999.
15. Aston, G. and Burnard, L.: The BNC Handbook: Exploring the British National Corpus with SARA, Edinburgh University Press, 1998.
16. Thomas, J., Wilson, A.: Methodologies for Studying a Corpus of Doctor-Patient Interaction, in Thomas, J. and Short, M. (eds.) Using Corpora for Language Research, Longman, 1996.
17. Garside, R., Smith, N.: A Hybrid Grammatical Tagger: CLAWS4, in Garside, R., Leech, G., and McEnery, A. (eds.) Corpus Annotation: Linguistic Information from Computer Text, Longman, 1997.
18. Rayson, P., and Wilson, A.: The ACAMRIT semantic tagging system: progress report, Proc. Language Engineering for Document Analysis and Recognition (LEDAR), Brighton, England. 1996.
19. Rayson, P., Leech, G., and Hodges, M.: Social differentiation in the use of English vocabulary: some analyses of the conversational component of the British National Corpus, International Journal of Corpus Linguistics. 2 (1), 1997.
20. Jackson, D. and Jackson, M.: Problem decomposition for reuse, BCS/IEE Software Eng. J., 11 (1), 1996.
21. Rayson, P., Garside, R., Sawyer, P.: Recovering Legacy Requirements, Proc. Fifth International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ’99), Heidelberg, Germany, 1999.