Requirements Definition - A Plea For a Return To English

Erik W. Aslaksen
Sinclair Knight Merz
100 Christie Street, St. Leonards Nsw 2065
Australia

Abstract. The formulation of requirements continues to be a problem in systems engineering, despite considerable efforts over the last several decades and the emergence of a number of formalised languages. In this paper, we argue that a natural language, i.e. English, is still the most appropriate for most cases, and that it would be beneficial to concentrate more effort on ensuring the proper use of English rather than on inventing new formalisms to circumvent the perceived weaknesses of English.

Introduction

Requirements are still a major topic in systems engineering, both within INCOSE and in other engineering organizations. Within INCOSE we have a separate Working Group, the Requirements Working Group, which has produced a number of papers over the last ten years and conducts a lively on-line discussion group (see the corresponding web site at www.incose.org/), each Symposium has seen one or more papers or panels concerned with the topic, and the INCOSE Journal *Systems Engineering* contains several related papers every year (see e.g. Gambhir [2001], which contains a list of recent references). Other engineering organizations also devote considerable effort to the formulation of requirements, as is evident from e.g. Lamsweerde [2000] and Nuseibeh [2000], both of which also contain a very substantial list of references.

The problems encountered in writing good requirements can be generally grouped into those associated with achieving completeness and those associated with making the requirements unambiguous to its intended readership. And there are two aspects to both of these characteristics - we require a methodology for developing requirements to satisfy them, and we need a process for verifying whether a given set of requirements meets them or not. It is not our purpose to enter any further into these problems here, but a review of the literature over the last fifty years shows a significant shift in emphasis. In the 50ies and 60ies, considerable effort was spent on developing a usage of English suitable for technical writing, both in educational institutions (a typical textbook for a two-semester subject was Thomas [1957]), in industry (for example, the Bell Laboratories Graduate Study program, mandatory for all engineers below PhD level), and, in particular, in the US Department of Defence (as evidenced by e.g. [MIL-ST-961], [MIL-STD-490], and [MIL-HDBK-63038-2]). But with the rise of software engineering, and with the significant problems experienced in converting requirements into code in a controllable and efficient manner, there has been a gradual shift in emphasis, to the extent that, at least in electrical engineering and systems engineering, requirements specification has almost become synonymous with software requirements specification. In the case of systems engineering this is particularly ironic, because if there is one engineering discipline that is concerned with a holistic view and with engaging a wide segment of society, it is systems engineering, and it needs to use a means of communication that is acceptable to all within this segment (see also [Alexander 2002]). In the following section we take a closer look at this phenomenon, before going on to discuss a possible remedy.

It should be noted that, throughout this paper, English represents any natural language; this shorthand is not meant as a claim for the superiority of English over other natural languages.

Software Engineering as a Special Case

The process of engineering is a step-wise process of converting user requirements into requirements for the fabrication or construction of the equipment or works that will meet the user requirements. The requirements are written in a language that is appropriate to their use, and therefore, towards the fabrication end of the process, the language becomes specialised to the fabrication process and relies heavily upon graphics (drawings, diagrams, and the like). Mechanical engineering, power engineering, electronics, civil engineering, etc. all have their specialised languages that have developed over a long time, in some cases centuries.

When computers arrived on the scene, a completely new and different situation arose. To design and fabricate the computer, the existing language for electronics was adequate (with appropriate extensions). But in addition, one now had to communicate with this entity and tell it what to do in order to perform its intended function. Until then, all communications in the process of engineering had been between humans; there had never been any question of communicating with a bridge or a diesel engine. The problem now was that the computer understood only an extremely simple language, and so needed specialised people who could translate between English and machine language. For well-known reasons this was a highly unsatisfactory situation, and mitigation took place on two fronts. On the one hand, the computer was enabled to do part of the translation itself, thereby narrowing the gap between its programming language and English. On the other hand, the process of bridging this gap was subdivided into two (or more) steps, with the intermediate result formulated in a language somewhere in between English and the programming language. As in the other engineering disciplines, these intermediate languages often made considerable use of graphics, and a recent example is UML [Rational 2001].

When a new situation arises, or a new invention or theory is put forward, it is natural to at first become so absorbed in the novel aspects that one completely overlooks the aspects it has in common with existing practice and knowledge. Software engineering has been no exception, and it has been nothing if not amusing to observe the glee (even, at times, pompousness) with which software engineers have rediscovered such concepts as objects and reusable modules; concepts that have been fundamental to engineering for a very long time. Just take a normal machine screw; it is an object belonging to the class “screws” which belongs to the class “fasteners”, it has attributes (diameter, length, thread length, head type, material, surface finish, etc.), and is certainly reusable. If we did not have standardised components, but had to design and specially manufacture each one as we needed it, engineering would be hopelessly inefficient. We are now just waiting for the software engineers to catch up, so that when we need a piece of software to perform a particular function, we go out and buy a couple of hundred standard software modules, link them up in the right order, and that’s it.

But, back to reality and on a more serious note, the current trend towards applying what has been developed for the special case of software engineering to the general case of engineering, and in particular to the most general case of systems engineering, must be viewed with some scepticism. Graphics can be a very useful illustration of relationships and structure, but to display in a one-page use-case diagram what is much more adequately described in about ten lines of text makes no sense, and the little stick man popping up everywhere is frankly beginning to look a bit ridiculous.

A Return to Basics

Let us now return to the basic case, where the human mind is the “computer” that processes data between the input and the output of a step in the engineering process and then transmits the data to the person carrying out the next step, using English as the transmission language. What is the machine language of this computer? What is the nature of its variables, and what is its instruction set? The “final” answers to those questions are still shrouded in mystery, but at a

reasonably high level (say, corresponding to a high-level computer language) the variables are words. A word is the smallest unit to which one can give a meaning, and thus words provide the basic elements of semantic interpretation.

The two principal ways in which two humans can communicate using English is shown in Fig. 1. They are distinguished by the physical nature of the signal transmitted between the two persons; in the one case it is an acoustic signal, in the other case an optical signal. Aside from this difference there is, however, another one which is much more significant - in the case of the acoustic path the human organism contains the two transducers required. The vocal tract produces sounds that are immediately recognisable by the ear, whereas in the case of the optical path, one needs the intermediate artefacts of pencil and paper (or their equivalent) in order to achieve a reasonably efficient encoding of the signal. (Sign language being so inefficient as to be uninteresting in the present context.) The visual sense has a much higher information acquisition capability than the auditory sense, but it is the presence of matched (in the sense of encoding) receiving and transmitting capabilities that makes the spoken language the primary representation. This may be what makes the human species unique. And, to continue this line of thought a little further, if humans had been provided with integral visual display units that directly put thoughts into pictures, it could well have been the optical channel that would have been the primary one, and transmission speeds could have been orders of magnitude higher.

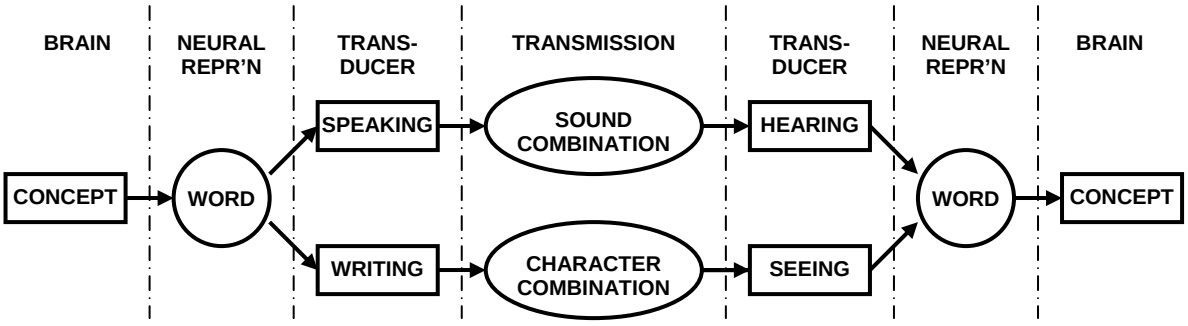

Figure 1 Block diagram of linguistic interaction, showing the two physical channels between two brains. The brain transforms concepts into neural representation of words, this is converted to a physical signal by the transducer, and on the receiving end this process is reversed.

While the encoding of the neural representation into a physical signal and the corresponding decoding in both the acoustic and visual channels is an interesting subject, we need not consider it any further here. It is in the process of putting the concepts, i.e. what we understand, into words and vice versa where the problems associated with writing requirements come in. Whereas computer languages are built up in a hierarchical fashion, with the concepts on one level being defined in terms of the simpler concepts on the level below (e.g. macros in an assembler language being defined by machine language instructions), English has a completely different structure. There are really no "levels" beyond or within the set of words; there is no visible structure that expresses the relation of simple words to complex words (measured by the complexity of their meaning). Except for certain combinations, such as downfall, interrelation, overcompensate, etc., there are no rules for constructing complex word from simpler words (although this varies considerably from one natural language to another, with e.g. German containing a lot more combinations of the type listed above than does English). The length of the word is not even significant; the noun "God" and the verb "to think" both represent highly complex concepts, whereas the noun "tomorrow" represents a simple and immediate concept. Of course, words are combined into sentences, sentences combined into paragraphs, and so on in order to express more detail and provide a greater depth of definition, but this is done by

creating “new” relations between the same type of elements - the words - whose meanings are again defined by such relations, and so on. It is a circular process, much subtler than the straightforward logic of computer programming.

There are also some further problems associated with the translation of concepts into words, such as the lack of boundedness in English. There is in principle no limit to the number of words in a lexicon, nor to the length of a sentence. And, perhaps most importantly, the meaning of a sentence, its *semantic interpretation*, is dependent on the wider context in which it is embedded and on the background of the reader. In addition to the *literal meaning*, an inherent property of the sentence that is independent of the context in which it finds itself, there are the *pragmatic implications*, information conveyed by the sentence when it is combined with all other knowledge available to the reader at the time of reading. So, there it is no wonder that writing requirements in English is problematic, but that is in itself no reason to give up on trying to overcome or, at least, reduce these problems.

The Benefits of English

Before going on to outlining how some of the problems encountered when using English to express requirements might be reduced, we need to be clear about what the advantages are, so that we do not inadvertently negate them.

a. It is the only language (aside from a simple sign language used e.g. for traffic signs) that is understood to a significant degree by the whole population.

b. English has a long history, and there is a great deal of information available about it in the form of dictionaries and thesauri, numerous books and papers on its grammar and semantics, and on its use in various applications (such as technical writing).

c. It is much more powerful than any other language. Its richness in words and in the ways these words can be combined is unparalleled, and its ability to develop along with our expanding knowledge and intellectual capability ensures that it is never outdated.

d. It is universally applicable, used both in specialised applications (e.g. to express technical requirements) and in all areas of everyday life. Therefore, as we all get constant practice, maintaining our proficiency requires relatively little additional effort.

There are probably a number of other advantages that could be put forward, but already the ones listed above suffice to demonstrate what an extraordinary means of communication English constitutes. It should, therefore, not be surprising that, in order to get the full benefit out of it, one needs a considerable amount of instruction and training, but somehow this is not a truth that enjoys much popularity these days. There is a feeling that a natural language should be just that; children pick it up by listening and practice, and any further formalisation only serves to stifle their individuality and creativeness. This is not the place to discuss educational policy, but this attitude must surely be quite at odds with the way we, as engineers, would approach the matter. If we have a complex piece of equipment - say, a sophisticated, numerically controlled machine tool, or perhaps an aeroplane - would we let somebody operate the equipment just from looking at how a skilled operator does it? Without any understanding of its internal workings, the principles underlying its operation, the designer’s intent, its inherent limitations, and so on? Of course not; the “sit by Jenny” approach to training went out the door long ago. But this is, to a large extent, the situation with English today, and the frustration expressed with the “limitations” and “imprecision” of English within the engineering community is due more to an inability to exploit the full power of English than to any shortcomings of English itself.

The Way Forward

The proposed way forward when it comes to using English to express requirements is really just a continuation of the work done in the 50ies and 60ies, but with some guidance from the work done in linguistics in the meantime, not least the work of Chomsky [Chomsky 1957, Chomsky 1972, and Smith 1979]. It can be considered to consist of two parallel paths, a syntactic path and a semantic path, and they are outlined very briefly in the following.

If one is asked to define a semantic element, say X, the answer is usually a sentence of the form "X is a Y which {}". Here Y is the *class* to which X belongs; it is a more general element. To it are related a large number of qualifying or specialising elements, and {} is the appropriate subset of these. Thus, the definition process consists of three subprocesses: First, the *generalisation*, which places the element into its class, then the *analysis*, which determines all the characteristics of this class, and then the *specialisation*, which chooses the appropriate characteristics for the present case. This is illustrated in Fig. 2, which also indicates that in order to be able to carry out the specialisation properly, i.e. to make the right choice among the possibly very large set of characteristics, the context in which the element is used may have to be considered.

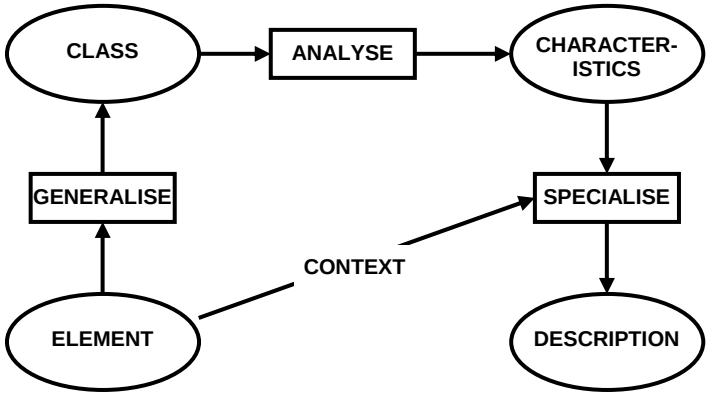

Figure 2 The concept definition process, consisting of three subprocesses and involving four entities.

The following definition provides an example:

> The *transfer function* of a two-port is a complex-valued functional, F, of a complex-valued function, the input amplitude spectrum g(ω), where ω is a positive, real variable called the (angular) frequency, such that a voltage source with amplitude spectrum g(ω) and internal impedance z_0 connected to the input port results in an output voltage spectrum F(g(ω)) across a load impedance z_0.

The generalisation is making the element a functional, the analysis is characterising a functional as a mapping from one set of functions to another, and the specialisation consists of specifying the type of functions and the boundary conditions, i.e. the impedance level. Within the area of electrical engineering, this is a context-free definition.

Thus, a definition, as it appears in a requirements definition, is the result of this definition process, and the subprocesses described above find their expression in different parts of the

definition, leading to an ordering or *higher-level syntax*. The two main parts of the definition are:

- The *classification*, which describes the class to which the element to be defined belongs, and
- the *relations*, which relate the element to other, already defined elements.

It should be noted that the relations mentioned here serve a purely semantic purpose - to define the *meaning* of an element, as discussed further below. They must not be confused with the relations that arise as a result of the top-down systems engineering process, and which express the structure of the system being designed. There is no engineering involved in or implied by the definition process at all; it is a secondary process providing a necessary support to the engineering process by defining the concepts with which the latter operates.

The previous process of defining new concepts, and which was reflected in a higher-level syntax, leads one intuitively to a semantic ordering. It would be natural to say that a concept defined in terms of a number of other concepts is in some way more *complex* than the latter. In this manner, concepts become ordered by the number of definition processes they are removed from some initial set of words or concepts. But how is such an initial set determined? An obvious choice would be the set of words used frequently, in everyday speech. That set would differ greatly between a mathematician, a priest, and a longshoreman, but taking the intersection of all such sets for different users, on could come up with a basic set of words.

On the other hand, one could look at the set of words whose semantic components or meanings are, at least to some high degree, independent of the context in which they are used. The intersection of this set with the basic set is a *primitive set*; it is made up of all those words well known to and assigned the same meaning by all persons regarded as having some defined minimum degree of competence in English.

This ordering makes the set of all words appear as an unbounded sphere, with the primitive set as a central core and then layer upon layer of words of increasing complexity. The direction in which one progresses outward represents the particular profession or area of application, and within such a cone a there is a further subdivision into the particular application or context. This is illustrated in two dimensions in Fig. 3. However, as one progresses outward in any direction, the meanings of the words become more and more context-dependent; the sphere is decomposed into finer and finer cones or fibres, between which there are significant differences in the meaning given to the same word. This is on the one hand a result of increasing specialisation and lack of communication between professions, but on the other hand, and more importantly, it is an inherent function of the increasing complexity, of the increasing richness of the concepts. Even for the specialist it becomes impossible to give an abstract, context-free meaning to concepts far removed from the primitive set. This point was discussed in [Aslaksen 1987].

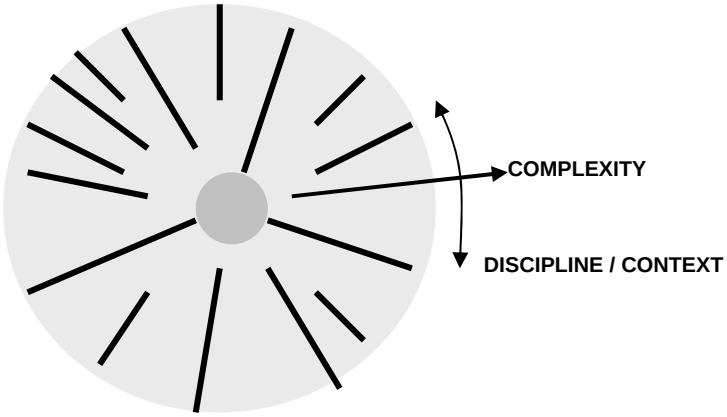

Figure 3 A two-dimensional representation of a lexicon, parameterised by the degree of complexity and discipline (or context). The radial lines separate different disciplines, and the darker grey area in the middle represents the primitive set.

Introducing an arbitrary measure of context dependency, κ, normalised to $0 \le \kappa \le 1$, then κ will be a function of the distance, *r*, from the centre of the sphere, as shown in Fig. 4. That is, this figure is meant to demonstrate what is the general trend (but not an absolute rule) for complexity and context dependency to be related and increase together. Up to a certain radius, r_0, there is no context dependency; this is the primitive set. (The value of r_0 should be increasing steadily with time due to an increase in the level of general education, but there is some doubt as to whether that is true or not.) From here on the value of κ rises slowly until, at $r = r_k$, it rises rapidly towards 1. As the value of *r* increases in the range $r_0 \le r \le r_k$, persons with increasingly specialised knowledge will automatically infer the correct pragmatic implications, thus allowing virtually context-free definitions. In this range it therefore becomes a question of correctly identifying the degree of specialisation of the person who is to read the document. For *r* greater than r_k even the specialist will need to know the context in order to give precise meaning to the concepts; using abstract concepts alone will lead to ambiguity.

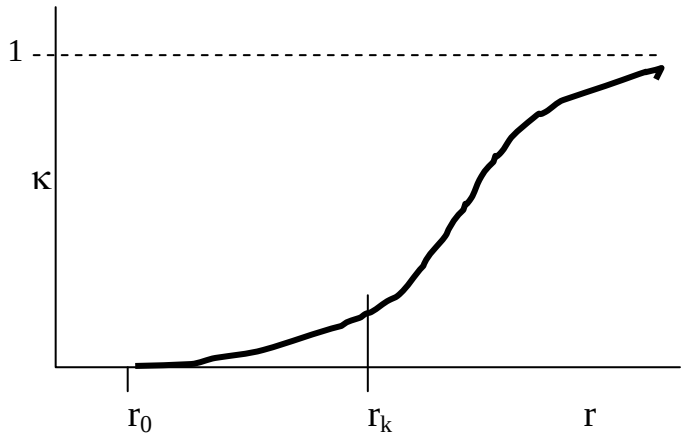

Figure 4 The increase in context dependency, κ, (arbitrarily normalised to unity) as one moves away from the primitive set, with complexity $r < r_0$. For $r < r_k$ context-free definitions are possible within areas of specialisation; beyond r_k the context will need to be specified along with the requirements.

For systems engineering, which often deals with concepts in the region $r > r_k$, the implications are clear:

a. Before producing a requirements definition document, we need to be clear about the intended readership.

b. Where the meaning of a concept depends on the context, *and* the readership cannot be reasonably assumed to automatically infer and have an understanding of that context, it has to be described as part of the document.

It is the second implication that seems to cause systems engineers the most frustration; on the one hand they would like to have very minimalist formulations of the requirements, on the other hand they would like their requirements to be understood by and useful to a wide audience, which is appropriate to such a multidisciplinary activity as systems engineering. The complaints about the "fuzziness" of English are often a reflection of the inability (or unwillingness) to write a good description of the context, and the expectation of using brief, formal formulations of requirements is a reflection of confusing humans with computers. We *tell* a computer what to do, but a human needs to *understand* what to do, and that takes a few extra words.

References

I. Alexander, The Glamour of Formalisation - A Warning, easyweb.easynet.co.uk/~iany/consultancy/glamour.htm.

E.W. Aslaksen, Systems Engineering and System Specification, J. Electrical and Electronics Eng., Aust., Vol. 7, No. 3, 1987, pp. 159-165.

P. Blackledge, Specification languages, IEE Proceedings, Vol.130, Pt. A, No. 4, June 1983.

N. Chomsky, Syntactic Structures, Houton & Co., S'Gravenhage, 1957 (2nd printing 1962).

N. Chomsky, "Language and Mind", Enlarged Edition, Harcourt Brace Jovanovich, Inc., 1972.

S. Gulu Gambhir, An Investigation of Facilitor-Assisted and CONOPS-Based Requirements Elicitation Methods Using a 2x2 Factorial Experimental Design, Systems Engineering, Vol.4, No.4, John Wiley & Sons, 2001, pp. 272-286.

A. van Lamsweerde, Requirements Engineering in the Year 00: A Research Perspective, Proc. 22nd Int'l Conference on Software Engineering, Limerick, June 2000, ACM Press.

MIL-STD-961, Military Specifications and Associated Documents, Preparation of

MIL-STD-490, Specification Practices

MIL-HDBK-63038-2, Technical Writing Style Guide

B. Nuseibeh and S. Easterbrook, Requirements Engineering: A Roadmap, ICSE-2000 Future of Software Engineering, A Finkelstein (ed.), 4-11 June 2000, Limerick, Ireland, ACM Press

OMG Unified Modeling Language Specification, Version 1.4, Rational Software Corporation, September 2001

N. Smith and D. Wilson, Modern Linguistics, Penguin Books Ltd., 1979

J.D. Thomas, Composition for Technical Students, revised, Charles Scribner's Sons, New York, 1957