4th Asia-Pacific Conference on Systems Engineering (APCOSE 2010)
Keelung, Taiwan October 4-6, 2010

Languages and Models in Systems Engineering

Erik W. Aslaksen

Sinclair Knight Merz

100 Christie Street, St. Leonards 2065, Australia

Tel.: +612 9928 2436, easlaksen@skm.com.au

ABSTRACT

The purpose of this paper is to examine some important characteristics of languages and models, as they are used in engineering, and to show that for much of the work carried out under the heading of systems engineering, languages close to natural language are required in order to preserve the multi-disciplinary and holistic character of systems engineering. Two such types of languages are identified; pattern languages and functional element languages.

KEYWORDS: : language, model, pattern, functional element

1. INTRODUCTION

Systems engineering has been undergoing an evolution ever since its inception as a means of improving the creation of large technology-based systems, such as the telephone system and various aerospace and defense systems, and for the last fifteen years or so INCOSE, as the international organization dedicated to the advancement of systems engineering, has been in the forefront of that evolution. It may be seen as an evolution from technology-based systems supporting human-based systems to technology-based systems performing their functions with the support of humans returning again to human-based systems, or enterprises, performing their functions with the support of technology-based systems. As engineers it is natural for us to focus on the technology, but we recognize that, and with our understanding of systems we are increasingly taking a wider, more holistic view, and testimony to this shift in the relationship between technology and humans (and their environment) is provided by the themes of the recent INCOSE Symposia; *Systems Engineering for the Planet* in 2008 and *The Human Dimension to Systems Engineering* last year. What is, perhaps, not so well recognized is our focus as humans; each one of us sees the world as something external to ourselves. We view the external world as something separate from us, with given characteristics that we can observe, not considering that those observations and our concept of the world is very much dependent on our faculties. A worm might have a very different view of the world, and so might a life form far advanced on ours somewhere in the universe.

This paper explores some aspects of the relationship between the individual and the external world and in particular, of course, the implications for systems engineering. We first look at the nature of our perception and the role of "system" as a central concept in forming it. But how do we know what that perception is, how do we express it, how do we process it? Through the use of language, the feature that, together with the faculty of speech, is most significant in defining what a human being is. Much of our thinking is done in terms of language, and speech gives us the ability to communicate those thoughts directly between humans. There are many languages, not just the

spoken ones, but symbolic languages, patterns, etc., but when we view them all under the unifying concept of a system, we shall gain an understanding of how the core of systems engineering may be seen as a search for the most appropriate language for a given engineering task. In this view, the problems involved in the relationship between technology and society appear as language problems.

Any description in any language, be it of a problem situation or of a solution, can be considered to be a model of reality, and in this sense it is really doubling up to be speaking of a modeling language. All languages express conceptual models of aspects of reality, with the aspects being such things as cost, reliability, safety, etc., and with only natural language being able to cover them all. Conversely, a model exists within a particular language; in order for a model to convey information from creator to user it is necessary for both of these actors to be proficient in the language. Therefore, when we, in the following, make observations about language, we are also making observations about models.

2. PERCEPTION AND OUR ABILITY TO RECOGNIZE RELATIONSHIPS

When we observe external objects through our senses, the faculty of representation turns these sensations into empirical intuitions. In addition, the faculty of representation is able to generate pure intuitions; intuitions to which there corresponds no object that is perceived through the senses, such as an angel. The faculty of understanding processes the intuitions and generates concepts. Concepts are classes of intuitions, and when we say we understand what an object is, it means that we know to which concept it is related. Generating concepts and recognizing relationships are the two main capabilities of the faculty of understanding, and it is through consideration of this latter capability that we shall gain some useful insight.

The space-time nature of our perception also allows us to immediately form such concepts as “tall” and “fast”, and our senses of light intensity color, pitch and loudness allows us to form a vast number of further concepts relating to single objects. But the space-time nature of our perception also allows us to form particular *relationships* between different objects; our three-dimensional perception of space gives rise to such relationships as “next to” and “on top of”, and our perception of time gives rise to such relationships as “before” and “after”, “earlier” and “later”. We are so used to the space-time nature of our perception that we hardly ever give a thought to how it is connected with our faculties and senses. For example, if we did not have memory, there could be no concept of time, and if we lived on a surface (e.g. as sightless microbes) our perception of the world might be different, as described in *Flatland: A Romance in Many Dimensions*, the brilliant book by Edwin A. Abbott [1].

On the next level of mental processing, we are able to carry out a certain amount of abstraction, in that we can see similarities between objects by disregarding detailed differences, giving rise to such concepts as “same as” and “larger than”, and, as a consequence, to classes of objects, such as “house” and “dog”. The relationship between an intuition and its concept may be considered the primary form of relationship; it is simply a *classification* of objects, leaving us with a vast set of concepts without any structure or ordering, except the space-time ordering that is inherent in the perception of the objects.

We now come to a level of mental processing that is of particular interest to systems engineering; the ability to form relationships between classes of objects of the form “consists of” and “make up”. When we think of a house, we think of something that consists of walls, roof, windows, doors, stairs and rooms, and this ability is much more than that of a relational database; it is the ability to conceive of the house as an entity and as a composite of these elements *at the same time*. That is, the elements are *interacting* to form the whole. This ability has its limitations, as was pointed out by George Miller in his seminal paper *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information* [2].

The foregoing related to physical objects, but we are also able to perceive actions, such as "to lift" and "to push", and to process them in much the same manner. So, what we have at this point is a vast collection of intuitions and their concepts, and it is on this collection that the next level of mental processing takes place; the processing that is independent of perception and that we generally call "thinking". If we view that processing as consisting of two distinct sub-processes, recalling elements from the collection and operating on these elements, then the operations are carried out using natural language, the language we speak every day. For example, if someone asks you how many windows there were in your childhood home, you recall a picture of this home and then count, "one", "two", and so on. Or if you are thinking of what to cook for dinner, you might recall that you bought some sausages and put them in the fridge, "I could have some sausages", by association you recall that your mother told you to always have a vegetable, "I guess I should have a vegetable", then you recall that you have some cabbage in the pantry, "well, cabbage would be all right", and so on; all these words and sentences go through your mind. And it is no different when we are doing engineering; calculations, assessments, selections, etc. are all done in terms of language. When you enter 52.7 on your calculator, you say "fifty-two point seven" in your mind as you punch in the digits. Natural language is the language of the mind; every thought we think and every concept we form are in terms of this language. All other languages, including those we use in engineering, are derived from it

3. NATURAL LANGUAGE

The natural language used for this paper is English, but any other natural language, such as French or Chinese, can be seen as more or less equivalent; this equivalence arises, of course, from the fact that they all relate to the same concepts; the classes of objects or actions perceived by us through our senses. However, this fact also accounts for a degree of "fuzziness" inherent in language, which is best discussed by looking at a sentence. We are all familiar with the subject-predicate type of sentence structure, such as "Lucy is tall". Here "Lucy" is the subject term and "is tall" the predicate term; the sentence predicates tallness of Lucy. This type of sentence structure is perhaps the most important one for our purposes, but there are other structures. A sentence such as "All cars have wheels" does not fit the subject-predicate structure, because "all cars" is not a subject. To make sense of this sentence, and to extend the analysis of the logic of a sentence to all types of sentences, we introduce the concept of a *variable*, say *x*, and the notion of sets. Let Y be the set of all objects that have wheels and X the set of all cars, then the meaning of the sentence is that $x \in X \Rightarrow x \in Y$. This same approach applies to the subject-predicate structure; if X is the set of all objects that are tall, the meaning of the sentence is that there exists an *x* that is identical with Lucy and $x \in X$. It also applies to a sentence like "Lucy exists"; the meaning is that there exists an *x* such that *x* is identical with Lucy, or $\exists x : \; x = \text{Lucy}$.

Returning to the subject-predicate sentence; now that we have looked at it meaning in terms of a variable, we should look at the components of the sentence. First, the subject-term. A name like Mary is a member of a class called *singular terms*, which also includes such items as "a man" or "the Prime Minister", and all singular terms *refer* to objects (including classes of objects). What exactly is meant by "reference"? We shall be satisfied with our intuitive understanding of it as the relationship that holds between a singular term, such as "Lucy", and Lucy herself. We realise that the main purpose of language is to be able to refer to objects and make statements about them.

Now to the predicate term. When a predicate is combined with a singular term, it makes a statement about the singular term, and the complete sentence has a *meaning*. Meaning has two dimensions, a *sense* and a *reference*. The sense of the sentence "Lucy is tall" is the idea that someone can be tall, the reference is the truth-value of the sentence - either true or false, depending on the object (person) to whom the singular term refers. In mathematical terms, the concept of "tall" is a function from object to truth-value of the sentence.

This can be summarised as follows:

a. For a singular term, its sense is the understanding that it refers to an object, and its reference is the understanding of what particular object it refers to.

b. For a predicate term, its sense is the understanding that it refers to a concept, its reference is the understanding of what particular concept it refers to.

c. For the sentence, its sense is our understanding of the relationship between the singular term and the predicate term. It is composed of the sense of the singular term and the sense of the predicate term, but it is more than just the sum of them; *the sense of a sentence is an emergent property.* The interaction between the terms is governed by the rules of *syntax*. - The reference of a sentence is its truth-value.

The issue we want to recognize and consider out of this brief and simplified detour into grammar (a good discussion is given in [3]) is that the meaning of a sentence has a degree of uncertainty associated with it, arising out of a number of sources. First of all, the understanding of what a singular term refers to may differ somewhat from person to person. That is obvious when we consider how we learn our natural language; it is by sensing the same object or action at the same time as we hear the spoken word, and again, it is a feature of our brains that we can recognize and process such acoustic signals and relate them to what we are sensing. Of course, not all concepts arise through observation; it is an essential feature of civilization that we transmit knowledge from one generation to the next by teaching and learning, but in any case, what we understand by a singular term, its reference to us, is related to our individual experience and education.

Secondly, and in addition to the uncertainty inherent in the meanings of its constituent words, there are two sources of uncertainty in the meaning (both sense and reference) of a sentence. One is that the *literal meaning* of the sentence, which is an inherent property of the sentence, independent of the context in which it finds itself, is not necessarily completely defined by the syntactic rules. For example, a syntactically correct sentence can be ambiguous. The second, and much more significant, is that the meaning is dependent on the context, the so-called *pragmatic implications*. The context includes the sentences surrounding the sentence, as well as the education, cultural background, and experience of both the person that wrote (or uttered) the sentence and the person that read (or heard) the sentence.

The purpose of this detour into the uncertainty associated with information transfer using natural language is not to provide a basis for pursuing that problem and possible solutions to it any further; it is to remind us of its inherent nature and to make us realize that, as any other language must be based on natural language, this characteristic will be inherited by any such language.

4. OTHER LANGUAGES

If we define a language roughly as a means of expressing intellectual content, and consisting of a set of symbols representing concepts and a set of rules for combining them, then, as engineers, we are familiar with a number of other languages, although we might not normally think of them as such. The oldest and best known type of engineering language is the engineering design drawing, which comes in numerous variants, from architectural drawings to circuit diagrams, more recently programming languages, and now such general languages as typified by UML.

These languages all have a common purpose; to improve what we might call the "cost-effectiveness" of the language, where the effectiveness is the accuracy of the expression and cost is the effort involved in both generating it and using it (i.e. cost to both sender and receiver). Natural language is unbounded; there is no limit to the accuracy we can achieve if we only make the expression long and detailed enough, but the effort is correspondingly unbounded (as is demonstrated by the cost of legal work). Also, the development of each of these languages is based on a common condition; the applicability is restricted to a defined

context, i.e. interest or user group and subject matter.

We can distinguish several different approaches to achieving the common purpose has. Firstly, one can restrict both the lexicon and syntax of natural language by eliminating all those elements that add little or nothing to the expression of the subject matter of the user group. An example of such a controlled natural language is the *Simplified Technical English* (Specification ASD-STE100 of the Aerospace and Defence Industries Association of Europe). Another example is a high-level programming language, although there is a subtle difference between these two examples, which it might be useful to comment on here. Language is a means communication, and if this is between two persons, the accuracy of the communication is the degree to which what was in the mind of the sender is also reproduced in the mind of the receiver. Besides any distortion in the physical communications channel, there are two sources of uncertainty involved; the meaning the sender ascribes to the words uttered and the meaning the receiver ascribes to the words heard. A controlled natural language reduces both of these. In the case of a computer language there is no uncertainty in the receiver's interpretation; to a computer there are no pragmatic implications of a received message, but on the other hand the programming language is severely constrained by the literal interpretation capability of the compiler. The uncertainty in the message transfer is solely due to the sender's interpretation of the meaning of the message.

A second, somewhat related approach is that taken in many applications of natural language to reduce the uncertainty by specifying the particular meanings ascribed to certain words in that particular application. A typical example of this are legal documents, which often have a section headed "Interpretation" containing precise definitions of the meaning of words, such as defining "Intellectual Property Rights" as "any intellectual or industrial property right including, without limitation, any patent, design right, copyright, confidential information, trade mark, and all other rights of intellectual property defined in Article 2 of the Convention Establishing the World Intellectual Property Organisation of July 1997". Wherever the words "Intellectual Property Rights" appear in the document, on must consider that definition to be inserted. The purpose of having the definitions up front in the document is therefore purely a sort of shorthand, a means of reducing the bulk, and therefore the cost, of the document; it is not in any way a simplification in the sense of a controlled natural language.

A third approach is to create a language of additional words relevant to a particular subject matter, such as mathematics, medicine, the various engineering disciplines, etc., with each word a shorthand for a more detailed description. This is an example of the *chunking* introduced by Prof. Miller in the previously cited paper. For example, to an electrical engineer, the word "resistor" is short for "an object that provides a linear relationship between the current through it and the voltage across it, characterized by resistance value, power dissipation rating, accuracy, and temperature coefficient". And associated with this conceptual shorthand, or discipline-based language, are two of the underpinnings of engineering; graphic languages, as e.g. in circuit diagrams, process diagrams, or mechanical drawings, and standardized construction elements, such as resistors, capacitors, nuts, bolts, shafts, bearings, and steel profiles. So, we now have a *class* of objects called "resistors", using the word "resistor" we can speak about such objects using natural language, we have a symbol for this class and rules for how to manipulate this symbol in order to express intellectual content, and we have standardized elements forming *subsets* within this class (thus creating a hierarchical ordering of classes). This has been the state of affairs in engineering (and other professions) for hundreds of years; more recently it was also "rediscovered" by software engineers.

Finally, in an extension of this third approach, one can use natural language to define a limited set of application-specific concepts as the "words" of a new language, together with an associated set of rules for how to combine the concepts in order to express intellectual content in this language, but where the "words" do not refer to any physical object or even class of objects. Of course, this is

nothing new in natural language - the word "happiness" does not refer to any physical object - but it is relatively new in engineering. The need for this type of language arises due to a shift in the focus of engineering from what an object is to what is does; to its purpose and how well it fulfills this purpose, and two classes of such languages are *patterns* and *functional elements*.

For completeness, we should mention the use of signs, as in traffic signs or the sign for a toilet or a telephone. Such signs have a very wide user group; wider than any one natural language, but they cannot be combined to form new meanings and so do not make up a language. They simply form isolated chunks of information restricted to particular contexts.

Patterns and pattern languages were introduced by Christopher Alexander [4] and co-authors as the elements in a methodology for finding solutions to problems in creating cities, towns, neighborhoods, and buildings, and more recently Cecilia Haskins [5] has promoted the use of patterns as a language for systems engineers that captures what is known within the community about socio-technical systems. A pattern is a description of the features and the relationships between them (hence the name "pattern") that should be present in order for a solution to be satisfactory; that is, for it to fulfill its purpose, and a description of that purpose is included as part of the pattern. A set of patterns which is complete within a given context constitutes a pattern language, and the solution to any problem within that context is described by a subset of these patterns. That is, the patterns are the "words" in the description of the solution. Such a subset is, however, not simply a collection of patterns; just as the words in a sentence are structured by the syntax of natural language, it has a hierarchical structure that arises from the fact that the pattern language itself is structured in levels of increasing detail. The definition of a solution (or better, the requirements on a solution) starts with the most general statement of the problem (and thereby the purpose of the solution) and the selection of the corresponding pattern; this then calls up patterns that describe the features of this solution in more detail, and so on, forming a system in a top-down fashion. The development of an appropriate system of patterns for a given problem is a process of unfolding, and the quality of the whole is an emergent property.

In the case of Alexander's work, the context is the architecture of cities, towns, and buildings, and the overarching purpose, i.e. the most general formulation of the purpose of any architectural design, is the harmony of the events taking place in the built environment, and the patterns in space are determined by the patterns of events. This harmony does not exist in the sense of cause and effect, but as a congruence between patterns of events and patterns in space. This is "the timeless way of building", and the understanding of the harmony arises out of our own nature, as well as our relationship to the rest of Nature, and it can be thought of as an absence of conflict and tension between the events taking place.

In a number of previous publications, the author has promoted a somewhat similar approach to engineering design, introducing a language of *functional elements* [6]. These elements describe the requirements on what a solution must be able to do in order to fulfill its purpose, and the most significant similarities are that there is a hierarchical ordering determined by the ultimate purpose, in this case maximizing a generalized Return on Investment, and that any particular solution is described in terms of an unfolding sequence of elements selected from the language [7]. A full account of the application of this language to system design is given in [8]; here we only want to point out that, despite these and other similarities, there is a very significant difference between a pattern language and a functional element language, and that is that in a pattern there is a direct connection with the purpose (which is to overcome or suppress a problem situation), whereas a functional element only provide a means for characterizing (and thereby optimizing) the degree of fulfillment of the purpose. Patterns include information on good solutions in the form of rules and heuristics; functional elements provide a framework within which any solution must be described in order to allow it to be evaluated against its purpose.

5. LANGUAGES IN SYSTEMS ENGINEERING

In this last part of the paper, we compare some of the features of a few of the languages employed in systems engineering, as listed in Table 1, and make some observations on their use. There is nothing absolute or unique about the subdivision into features or the naming of these features, but it allows us to make some comparisons.

Language	Mathematics	Block diagrams	SysML	Functional elements	Patterns	Natural language
Purpose	Unlimited	Single-issue purpose	Single purpose limited applicability	Single purpose universal applicability	Context-dependent purpose	Unlimited
Elements	Symbols (numbers)	Blocks	Objects, classes	Functional elements	Patterns	Words (lexicon)
Rules	Operators	Purpose defined	Application specific			Syntax
Descrip-tions	Expressions (equations)	Diagrams	Diagrams	Functional systems	Pattern systems	Sentences
Scope	Calculations					Stories

Table 1 Comparison of some features of various languages. The last row illustrates the unbounded nature of mathematics and natural language.

A pattern language is always tied to a context-dependent purpose; e.g. Alexander's patterns are tied to architecture as the context and the harmony between event and space as the purpose. Functional elements support a single purpose (maximizing a generalized ROI), but can be applied to any engineered object. SysML also has the single purpose of modeling behavior, and is therefore mainly applicable to active systems. Block diagrams have a single purpose, such as describing reliability, interconnection (circuit diagrams), process flow, etc. Mathematics and natural language are both unlimited as far as purpose is concerned.

The combination of elements to form models is determined by operators in the case of mathematics, is defined by the purpose in the case of block diagrams (e.g. the interconnection of blocks in a reliability block diagram), is determined by the application in the case of SysML, and by the rules of syntax in the case of natural language. Functional elements and patterns have their relationships defined in the elements themselves.

Both mathematics and natural language are unbounded in the sense that there is no limit to how complex a calculation can be or how many words and sentences a story can contain. The models in the other languages are bounded by the physical reality they are modeling.

These languages all have their place within systems engineering, as do a number of specialized languages connected with software programs, and selecting the most appropriate language in each particular case is itself an important task. Two of the main considerations are as follows.

5.1 The audience (or user group)

Who is going to use the information? They all have to be familiar with the language used to convey it. An important characteristic of systems engineering is its applicability in a multidisciplinary environment and, even more importantly, its ability to present a holistic view of a project, understandable to a range of stakeholders. This can perhaps be illustrated by the diagram in Fig. 1

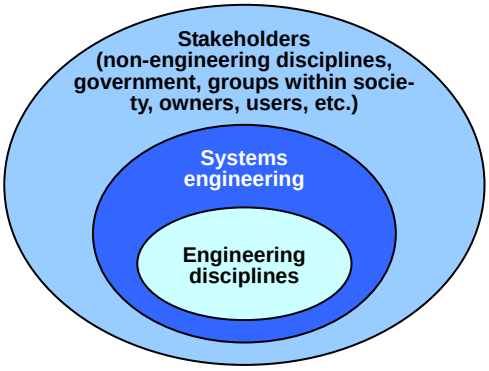

Fig. 1 The dual role of systems engineering as an integrator of engineering disciplines and as an interface between engineering and project stakeholders.

In this view, systems engineers are involved in three groups of activities; activities requiring communication only with other members of the systems engineering community, activities requiring communication with non-systems engineers within the engineering disciplines, and activities requiring communication with non-engineers (i.e. the rest of society). The languages appropriate for each of these activities differ; clearly, for communication with society in general only natural language is appropriate. For activities taking place wholly within systems engineering, specialized languages, such as SysML and functional elements, are appropriate to the extent that they improve the cost-effectiveness of the engineering process. It is the third group, which requires an interface between systems engineers and engineers in general, that raises an interesting question. Should all engineers be familiar with a systems engineering language? The situation is very much the same as in the case of software; being able to program in one or more software languages has become a necessity for every engineer, with the most common one being Excel (and yes, this is a language, with its own elements and syntax, albeit at a fairly low level). But engineers who program as part of their work would not normally use or have any familiarity with a higher level modeling language such as UML, and it is highly doubtful if SysML will find general acceptance and use in the engineering community.

5.2 The purpose of the communication

The choice of language will depend very much on what the purpose of the communication is. If it is to convey the result of an engineering activity to another person, i.e. a fact or requirement, such as the statement “The container shall be fabricated from grade 304 stainless steel”, then natural language is the obvious choice. If it is to convey a requirement to a machine, such as the drilling pattern in a steel beam to a numerically controlled machine tool, then there are various industry standard languages, in particular under the umbrella of STEP [9]. If it is to convey information for further processing by a computer, then there are again a number of languages, such as the Web Ontology Language (OWL) endorsed by the World Wide Web Consortium and, for systems engineering, AP233 under STEP. In general, the more specialized the application, the more restrictive the language, and the further away from natural language.

Perhaps the most significant difference in purpose is between communicating facts or data and communicating knowledge. In addition to data, knowledge requires an understanding of how the data was obtained, of its meaning in relation to its context. To represent knowledge, the language must not only be able to represent data, but also relations between data, and this divides languages into two corresponding groups; those that are used for communicating data only (usually between equipment), and those that can be used for communicating knowledge in the form of models. So while it is true that all languages are models (of reality), not all languages can be used to build models (in the sense of representing relationships). Another, somewhat imprecise way of expressing this dichotomy is to say that all languages provide a static model of reality, but only some are able to provide a dynamic (or executable) model. For example, if in a 3-D model of a plant the height of a column is increased, the rest of the plant does not automatically adjust to this; whereas if an input parameter, such as the cost of labor, is changed in a cost-benefit model, the values of all associated parameters, including the Return on Investment, are adjusted accordingly.

In systems engineering, an important purpose at the front end of a project is to act as the interface between a wide range of stakeholders and the engineering team dedicated to creating whatever object is required. This means building a model of the stakeholder requirements and the environment in which they exist (what Warfield calls "the problematique" [10]) that is understandable and adequate for the uses of everyone involved in the process, and this implies that the language must be close to natural language. This is where such languages as pattern languages and functional elements come in; languages that are, in effect, simply forms of chunking within natural language, with a hierarchical ordering of the chunks. Of course, creating new chunks requires adherence to certain sets of rules, but e.g. the rules for creating functional elements are very simple [8]. And then, using the language, as in creating, understanding, and using a model created in that language, requires very little special knowledge.

6. CONCLUSION

All languages are derived from natural language, and their common purpose is to improve the cost-effectiveness of their application, as compared to using natural language, by restricting their area of applicability. The smaller the area of applicability and thereby the greater the degree of specialization and removal from natural language, the smaller the user group. Many specialized languages are used in engineering, and they are all very effective within their discipline-specific application. However, a central characteristic of systems engineering is a multi-disciplinary and holistic approach to projects, and a significant activity is to develop a common understanding among all stakeholders of the criteria for a successful project. Consequently, we conclude that for this activity languages are required that are close to natural language, and that two suitable types of languages are pattern languages and functional element languages.

REFERENCES

1. Abbott, E.A, *Flatland: A Romance in Many Dimensions*, Basil Blackwell, Oxford, 1884. The fifth edition was reprinted by HarperCollins in 1983, and a Dover Publications thrift edition was published in 2007.

2. The seminal work in this area is a paper by G.A. Miller, "The Magical Number Seven, Plus of Minus Two: Some limits on Our Capacity for Processing Information", *The Psychological Review*, vol. 63, pp. 81-97, 1956.

3. Smith, N. and D. Wilson, *Modern Linguistics, The Results of Chomsky's Revolution*, Penguin Books, 1979.

4. Alexander, C. et al, *The timeless way of building*, vol.1, Oxford University Press, New York 1977, and vol.2, *A pattern language: Towns buildings, construction*, 1979.

5. Haskins, C., "Using patterns to transition systems engineering from a technological to a social context", *Systems Engineering*, vol. 11, no. 2, 2008. Also "Using patterns to share best results – a proposal to codify the SEBOK", *Proc. 13th Annual INCOSE Int. Symp*, Washington DC, June 30-July3, 2003, and "Application of patterns and pattern languages to systems engineering", *Proc. 15th Annual INCOSE Int. Symp*, Rochester, NY, July 2005.

6. Aslaksen, E.W., "A Leadership Role for NCOSE", *Proc. 4th Int'l Symp, National Council on Systems Engineering*, San Jose, August 1994 and also in *The Changing Nature of Engineering*, McGraw Hill, 1996.

7. Henney, K., *Context Encapsulation – Three Stories, a Language, and Some Sequences*, at http://www.two-sdg.demon.co.uk/curbralan/papers/europlop/ContextEncapsulation.pdf. While this paper is specialised to software design, it contains a generally valid discussion of sequences and stories.

8 Aslaksen, E.W., *Designing Complex Systems – an Introduction to Design in the Functional Domain*, Taylor & Francis, 2008.

9. ISO 10303 *Industrial automation systems and integration – Product data representation and exchange*.

10. Warfield, J.N., *An Introduction to Systems Science*, World Scientific Publishing, 2006.