A Conceptual Glossary for Systems Engineering:
Define the Concept, don't quibble about the terms.

Iver Holtersvei 2, NO-1410 Kolbotn, Norway, Tom@Gilb.com, www.Gilb.com, +47 66801697

Abstract.
We seem to spend a lot of our time defining terms, arguing over terms, standardizing terms, and misunderstanding terms. In connection with my development of a planning language, and its basic textbook I had to make a decision regarding my glossary. One of the formal design requirements I had set for myself was that the planning language, and its textbooks were easy to translate into other international languages. It was primarily in order to satisfy that requirement that I came up with the concept of a *concept*-centered glossary, rather than the conventional *term*-focused glossary (like a conventional dictionary).

The basic idea is that we focus on defining a *concept*, no matter how many synonyms it might have, or how many different opinions there are about what the concept should be called.

Introduction

In order to have a position of both 'term' neutrality and 'human language' neutrality, the *primary and universal reference* to a Planguage concept is through an *assigned number* (like *001). The textbook in which the glossary is placed will generally use a single *primary term* to refer to the concept. The terms are sequenced in the Glossary according to this primary term, as in a normal dictionary. And, all concept definitions, in books, papers, or slides, that need to refer to the concept, will use that primary term *consistently*.

There are many interesting aspects to this class of glossary that will be brought out in this paper, and we hope it will provide a model for similar work in organizing knowledge about concepts, for educational and institutional purposes.

Example:
Here is a simple example of a concept definition:

Designer **Concept *190 May 6, 2006**

A designer is a person or group who specifies design ideas, in an effort to satisfy specified requirements.

Notes:

1. This is a generic term. More specific terms include systems engineer, planner, or systems architect.

Synonyms [Designer *190]:
• Systems Designer *190

Related Concepts [Designer *190]:
• Systems Architect *193

*Type [Designer *190]: Role.*

Notice the basic structure:
• the primary term (designer) leads, as in conventional glossaries
• the concept reference number is listed immediately (*190)
• a version date for the term is given (May 6, 2006)
• the primary definition is given
• notes are made to give more background about the term
• synonyms are listed, if any
• related concepts are listed, if any
• the concept is classified by type. (a 'Role')

This is not a complete list of all possible elements of a glossary entry, but it is reasonably representative.

Pointing out that terms are defined concepts.

The initial idea of my handbook was that I could be very terse about any technical subject, because all they key words I used were so thoroughly defined in the glossary. It was a matter of 'define once excellently', 'reuse the term many times'. The term meaning was precise. The device used to make sure that the reader was aware that the term was 'Capitalization' of the term. For example: *"The Design Requirements were not Constraints"*.

This is explicit enough, but publishing advisors convinced me that this device in normal text (as opposed to technical specifications) was more disturbing than useful. So, as with most texts, the reader is left to know that the term probably does have a definition, and that they are obliged to interpret the text in that light.

On the deviation from convention in definitions.

I have long since understood that all people will never sincerely try to agree on the use of a single term to refer to a concept. I am also equally resigned to the fact that all people will never completely agree on a concept definition. In fact it is quite difficult to get any two senior opinionated people to do so, as I know from this experience. But I can at least, as a teacher, author and consultant, consistently use my chosen term, while having

respect for other peoples choices – and not imposing my term on them. I can at least develop concept definitions that are the best I, and my circle of advisors, can agree on.

My initial bent is to stick to convention and standards wherever practical. And even when the systems or software engineering culture has corrupted a conventional term, I try to check ordinary dictionaries, and consider moving the concept back to its normal roots. On many other occasions my concept definitions have been created by me in order to fit in with the larger scheme of things (especially my tendency to insist on multi-dimensional quantification); and many definitions I work out so that they are 'deeply useful' in the systems engineering processes.

*For example, it was one of my clients (Los Alamos national labs) that pointed out the necessity of defining a '**Page'** of technical specification in terms of a defined number of 'non-commentary' words, in order to normalize the volume of specification.*

*Another example. When describing technical specification defects, the concepts of major and minor have been used since at least the early 1970's. But they were not usefully and clearly defined in the software culture that used them. Much confusion ensued in classification and estimation as a result. It took me years to move towards my current definitions of '**major** specification defect'. Apparently small, but I argue, powerful distinctions are made now when using the concept definition of 'rule violation' that could 'possibly (cause)' (consequences downstream).*

Major Defect	**Concept *091 November 20, 200x**
A Major defect is a specification defect (a rule violation), which if not fixed at an early stage of specification, then it's consequences will possibly grow substantially, in cost-to-fix and/or damage potential. A Major defect has on average approximately an order of magnitude more downstream cost potential than it's cost to remove immediately.	

I have tried to provide the best set of systems engineering concept definitions my team and I could come up with, after years of hard work. But I also recognize that the process of evolving the concept glossary is an eternal one. Others will finally pass judgment on our choices, hopefully by quoting and reusing them!

Translations and Partial Translations

Most of my work is for large multinationals in a wide variety of countries. Most of this work is done in English. But most people speak to each other in their native language, if they get a chance, even in these environments. And only the multi-nationals use English in their specifications. So there is a need, for several reasons and at several levels for translation of the planning language into other native languages. It is worth pointing out the even in the English language culture, there are preferences for different terms to describe the same concept (synonyms).

The simplest way to get a 'translation' is to simply pick a term and equate it to a concept.

For example:

Need **Concept *599 March 17, 200x**

A 'need' is something desired by a defined stakeholder. The implication is that satisfying that need would have some value for some stakeholder.

A need might not be agreed as a formal requirement, and it might not be prioritized such that it is actually acted upon (designed and implemented).

It is a term often used as a stakeholder view of a problem before requirements specification is carried out.

Synonym [Need, *599]:
• Behov *599 [Norwegian, Bokmål]

Related Concepts [Need *599]:
• Problem Definition *598
• Requirement *026
• Design *047

*Type [Need *599]: Problem Definition Concept.*
======== End of Example ==========

*The term 'Behov' (a relative of the English behooves!) can now be used orally or in writing amongst Norwegians, with the formal (still in English) definition of the concept now valid for the concept. I have already done a trial run on this for a set of Norwegian terms. For 640 concepts you need to pick 640 non-english terms and define the concept they refer to. Presto you have access to the hard work in the concept glossary without needing to actually translate the whole glossary (about 300 pages of very fine-tuned definitions) into the native language. If you need to remind a reader about this you can use selected devices such as capitalization ("Kundenes Behov oversettes til Krav") or reference to the numeric tag ("Kundenes behov (*599) oversettes til krav (*026)".) . i.e. 'The customer's needs are translated into requirements'.*

This partial term-only translation solution may be useful for teaching purposes and rapid adoption of the Planning Language. My initial design was primarily intended to ensure efficient translation of the full Language handbook and associated writings (such as technical papers).

Conversion to dialects: national, and corporate.
The concept glossary, as is true of the entire Planning Language, is intended to be tailored to local use of technical or commercial English. It is neither wise nor necessary to impose my preferred terms on anybody else. So groups have a long tradition of using certain terms, and it will be more comfortable for them to continue doing so, while possibly getting the advantage of more precise operational concept definitions, and the advantage of getting the entire set of concept definitions as a compliment. So all you have to do is to change the primary term to one you prefer. You can note the Planning Language normal term as a synonym if you like.

For example:

Author **Concept *004 February 5, 200x**

An author is the person, who writes or updates a document or specification of any kind.

Note:
This is a generic term, which depending on the specific document type, is usually replaced by specific roles, such as {engineer, architect, manager, technician, analyst, designer, coder, test planner, specification writer}.

Synonyms [Author *004]:
• Writer *004
• Specification Writer *004 [Normal Planguage Term]
Note 'Author' is the XYZ Corporation preferred term.

Related Concepts [Author *004]:
• Owner *102

Type [Author *004]: Role.
========== end of example *004==========

Relationship to technical specifications, such as requirements and design

The glossary concepts and terms are used very directly and heavily in the planning language. They are the basic set of defined concepts to which a particular project or Author can add other defined concepts.

For example if I specify a type of specification:

Type: Complex Quality Requirement.

Then all four words are found as terms, and defined as corresponding concepts in the Planguage Glossary, and have very specific technical meanings. Their individual Capitalization signals this explicitly and is required in the specification.

The method user can add anything they want to add to make up their own local Planguage Glossary. An Author can in addition define their own terms at will, locally in a specification. For example:

Task Effectiveness: Scale: Time to Complete Local Tasks. Complete: defined as: real time minutes continuous duration of task from task doer initiation until successful completion is verified. Local Tasks: defined as: tasks that must be performed on site of product location. Goal [High Schools, Computers, Task = Replacement] <5 minutes.

The terms {Scale, defined as, Goal, Task} are predefined in the Glossary. 'Complete' and 'Local Tasks' are defined locally in this specification. They could also be defined globally for a project or larger set of specifications. 'High Schools', 'Computers', and 'Replacement' are somewhere defined (or need to be), outside of this specification, but not in the larger concept glossary.

Thoughts on the power of concept focused education

I believe that a large consistent well-defined set of systems engineering concepts such as these form a proper partial basis for professional training and higher education programs. Indeed I regularly use them weekly in my professional training courses in requirements and other subjects. I will shortly be preparing a M.Sc. program for software engineering, and it is my intent to use these concepts as a fundamental basis. Concepts are the building blocks of understanding a discipline. It is my experience that most everybody is confused about the meaning of very basic concepts such as 'requirement' and 'design'. They may not initially acknowledge that they are confused. But if you analyze their practices with them, as I have to regularly, they will themselves admit that a large portion of what they called 'Requirements' are really technical design, for meeting undefined requirements.

A simple (real) example: 'Password' is in the non-functional requirement section, but there is no specification for the degree or types of security required, so it is even logically impossible to decide if 'password' is a good design, or should be required as a design.

So, when I find real engineers confused about the meaning of the simplest concepts, it is also clear that they are not enlightened about more-detailed ones such as 'Design Constraint' (*181) or 'Scalar Requirement' (*198 + *026).

I must also conclude from observation that their higher education has not done much for their conceptual understanding

Graphic reference: keyed icons and graphical icons.

Decades ago I became acquainted with Charles Bliss, by correspondence, and his unique work on Blissymbolics, or Semantography. (http://www.symbols.net/blissre.htm)The concept of a highly general 'pictorial language' for technical things became a clear concept to me then.

As a teacher, when I explain things, or make slides for my courses, I know I have to try to help students visualize new ideas. As a result of these influences a set of pictorial representations of the concepts began to grow. So in addition to many other devices for accessing a concept, we have purely graphical possibility.

For example, the symbol -|-|- (directly from Bliss) is the graphic for 'Scale of Measure' or *132. All these options, and any synonyms can be used to reference the conceptual definition. All serve as 'names' to find and use the conceptual definition. The symbols are either graphical icons or keyed icons. Keyed icons car easily be created on a keyboard, and in-flight or integrated with text.

The important thing about the icons in this context is that they represent a language independent notation for systems engineers, like music notes or electrical schematic symbols.

In theory the entire planning language can be defined and explained in these symbols. The reality is that I have not focused on this aspect and pushed it very far. I am putting that work off, which the basic concepts are more mature, published and stabilized. They should be the subject of a separate paper. The most significant single aspect of these symbols compared to all other known symbolic diagramming languages is that they are especially good at representing, and integrating multidimensional performance and cost aspects of a system. Other symbolic methods, for example UML (Universal Modeling Language) does not try to do this at all, and therefore they are incapable of representing the most critical aspects of any system (performance and cost variation).

Here is an example of keyed icons in the concept glossary:

Survival Range: Concept *585 March 3, 200x

A Survival Range is the numeric range of a scalar attribute that indicates survival of the system, with respect to that attribute alone.

Keyed icon [Survival Range, *585] ---[======]-------->O------[=====]---->

Note: the '[' being a lower limit and the ']' being an upper limit.
The '[=====]' is the Survival Range.

Related Concepts:
- *Catastrophe Range [*603 ••••] the range just outside the Survival Range*

*Type [Survival Range, *585]*

========== end of example *585==========

Note on the example: the 'O' represents system function. The input arrow (--->O) represents a cost dimension, like human effort, the output arrow (O--->) represents a a single performance dimensions (like Usability) for that function.

Thoughts on standards

As you can observe, our Anglo-fixated culture, defines English words. For example:

Architectural Description [IEEE] Concept *618 July 18, 200x

Architectural description is "a collection of products to *document* an architecture."
(This definition is identical with IEEE Draft Standard 1471, December 1999)

This concept is generic and can apply to any specific architecture type.

Maybe international, including INCOSE, standards should apply the principle of defning concepts. And then indicate the synonyms in both English and other languages that map to that conceptual definition?

The power of non-trivial definitions

Here is a set of characteristics regarding 'non-trivial definitions'; this has been our guide when working on the concept glossary.

• gives insights that are not obvious to most people
• they are clear enough to support quantification, measurement, testing
• they are consistent with all other definitions in a related set
• they clearly show known synonyms
• they are powerful enough to correct bad practices (such as calling designs 'requirements')
• the focus is on a useful technical concept, not a term alone.
• rich notes are given about each concept
• examples are given where applicable

The problem of consistency in hundreds of conceptual definitions

Making a concept definition, or even a few in a book or a paper is common and useful practice. But in the definition of the Planning Language we are currently up at about 650 concepts. On the one hand, the more concepts that are well defined, the better you can build a great definition by re-using defined concepts.

But, the more terms that cumulate in a related set, the greater the problems in making clean conceptual definitions and in choosing appropriate primary terms seems to pop up.

For example at a very early stage of development, I used the term 'Plan' for the major planned level of a performance characteristic. No harm there. But as time went by and we developed other scalar *target* and scalar *constraint* concepts; the term 'Plan' became unnecessarily general. All the newer scalar target concepts (Ideal, Stretch, Wish, Goal) were more-refined notions of a 'Plan', and in addition we were busy integrating requirements disciplines with Evolutionary Project Management concepts – and there were plenty of 'plan'-like notions in those areas too. In addition, as the number of related concepts grows, the need for terms to describe sets of them grow – and that often causes conflict in a limited set of appropriate terms. In short the conceptual growth cause me to refine my selection of terms on the one hand, and to sharpen the conceptual distinction on the other.

Sometimes making distinctions between objectives, requirements, targets and goals seemed difficult, since many ordinary people do not systematically distinguish between them. Yet at the end of the exercise, I felt that very useful conceptual distinctions had been made. For example, an Objective *100, is a synonym for a Performance Requirement. A complex Objective (like Availability) can be defined by a set of scales of measures. Each elementary objective, having a single scale of measure (like Mean Time Between Failure) could still have any number of target levels of performance, and any other number of constraint levels for performance. The simple innocent 'Plan' level, had exploded into a potentially large number of plans of different nature. The set of all of these sub-requirements is the expression of one larger objective. The conceptual power of being able to do this at all, is still currently mind boggling in a world where so many have not quantified their performance requirement at all ('Graceful Degradation' was one such requirement I worked with last week).

Some projects are not in need of more sophistication in concepts than is common. But the demands put on the real systems engineer demand the most articulate set of concepts and specification tools we can make available.

References

Gilb05: Gilb, Tom, Competitive Engineering, A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage, ISBN 0750665076, 2005, Publisher: Elsevier Butterworth-Heinemann.

The published book has about 168 Concepts in the Concept Glossary. A total of 650 defined concepts are available from the full Concept Glossary (www.gilb.com). Some concepts have been submitted to the INCOSE Glossary. The web glossary is kept up to date with improvements and changes to the published concepts.

Author Bio

Tom has been an independent consultant, teacher and author, since 1960. He mainly works with multinational clients; helping improve their organizations, and their systems engineering methods.
Tom's latest book is 'Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage' (Summer 2005).
Other books are 'Software Inspection' (with Dorothy Graham, 1993), and 'Principles of Software Engineering Management' (1988). His 'Software Metrics' book (1976, OoP) has been cited as the initial foundation of what is now CMMI Level 4.
Tom's key interests include business metrics, evolutionary delivery, and further development of his planning language, 'Planguage'. He is a member of INCOSE and is an active member of the Norwegian chapter NORSEC. He participates in the INCOSE Requirements Working Group, and the Risk Management Group.

Email: Tom@Gilb.com
URL: http://www.Gilb.com

Version Nov 9 2005