Plicons: A Graphic Planning Language for Systems Engineering

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

Abstract:

- A Pictorial language (Planguage Icons = Plicons) for representing systems engineering problems (requirements) and solutions (designs) has been developed, and continues development, by the author. It differs from most all other published software engineering and systems engineering languages in several key respects.
- The main, but not only, differentiating characteristic is that it allows us to model quantified system performance properties and resources *graphically*; whereas most all other graphic languages are limited to things like functions, logic flow, use cases; and invariably avoid any representation at all for quantifiable qualities and costs.

Introduction

- *"Clearly, any model's practical value is directly proportional to its accuracy. If we cannot trust the model to tell us true things about the software system it represents, then the model is worse than useless—it can foster false conclusions."* [UML.Selec].
- *I agree with Selec. And any attempt to model systems engineering products without considering critical performance and cost characteristics results in graphical models we cannot trust.*
- *Here is a list from UML 2.0 [UML.Selic] of the scope of the Unified Modelling Language, and there is no notion of modelling the essential performance (including qualities) and cost attributes of a system.*
    - Language Unit Purpose: URL Categories
        - Actions (Foundation) modeling of fine-grained actions
        - Activities Data and control flow behavior modeling

- Classes (Foundation) modeling of basic structures
- Components Complex structure modeling for component technologies
- Deployments Deployment modeling
- General Behaviors (Foundation) common behavioral semantic base and time modeling
- Information Flows Abstract data flow modeling
- Interactions Inter-object behavior modeling
- Models Model organization
- Profiles Language customization
- State Machines Event-driven behavior modeling
- Structures Complex structure modeling
- Templates Pattern modeling
- Use Cases Informal behavioral requirements modeling

- Here is an example of some of our primary Planguage graphic icons for performance and cost attributes.

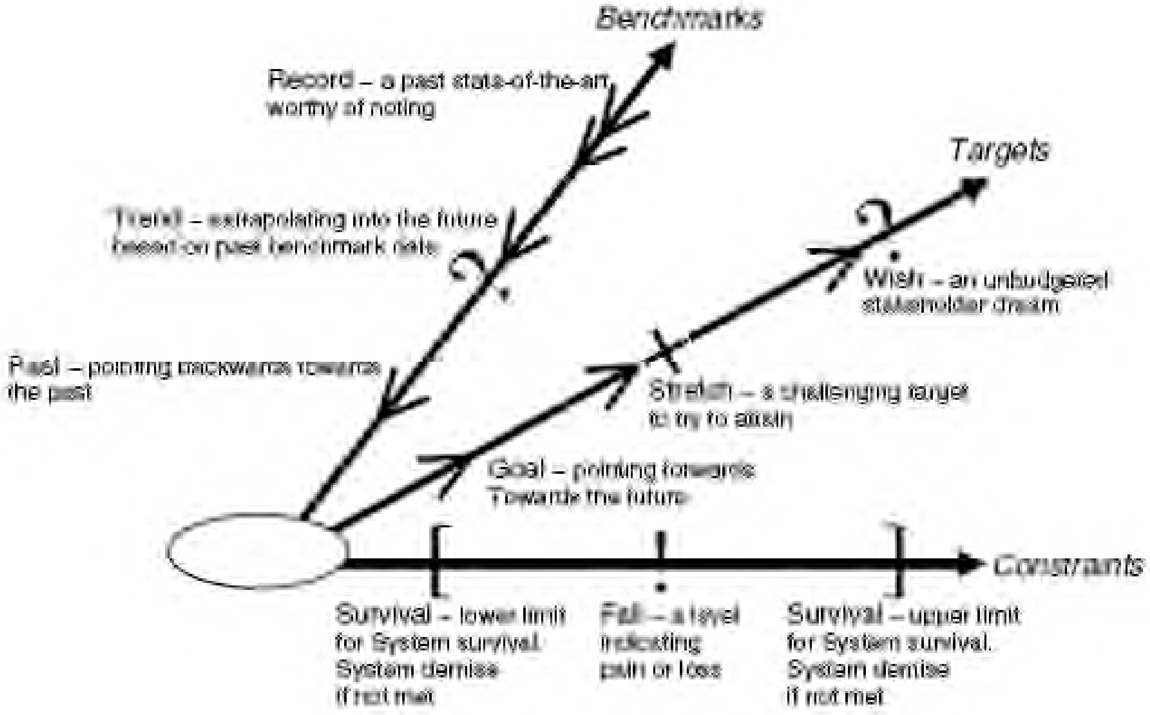

Benchmarks
Record – a past state-of-the-art worthy of noting
Trend – extrapolating into the future based on past benchmark data
Past – pointing backwards towards the past
Targets
Wish – an unbudgeted stakeholder dream
Stretch – a challenging target to try to attain
Goal – pointing forwards Towards the future
Constraints
Survival – lower limit for System survival. System demise if not met
Fail – a level indicating pain or loss
Survival – upper limit for System survival. System demise if not met

Illustration 1: Some fundamental Graphical Icon Concepts, <- CE book Fig. 4.10

- Explanation: A Scale icon is drawn as a line with an arrowhead, connected to a function oval symbol. Performance scales are to the right from the function oval (O->), and resource scales are at the left of the oval with arrowhead connected to the oval (->O). The performance and resource attribute icons must both include a function icon (an oval) to distinguish them from each other. The arrow in a performance attribute points away from the function oval. For a resource attribute, the arrow points towards the function oval. Three graphical performance attributes showing the icons for scalar performance attribute levels: three analytical benchmarks, three future requirement targets and two future requirement constraints, respectively. Usually an attribute would have a mix of whatever benchmark, target and constraint levels were relevant.

Why are Planguage Icons different or a unique contribution to graphical languages

- The primary Plicon distinction is the ability to represent *performance and resource* attributes of a system – most other diagrammatic languages seem entirely focussed on the function of a system, or other UML classes of representation (see list above) and illustrate little or nothing about performance and resource management.

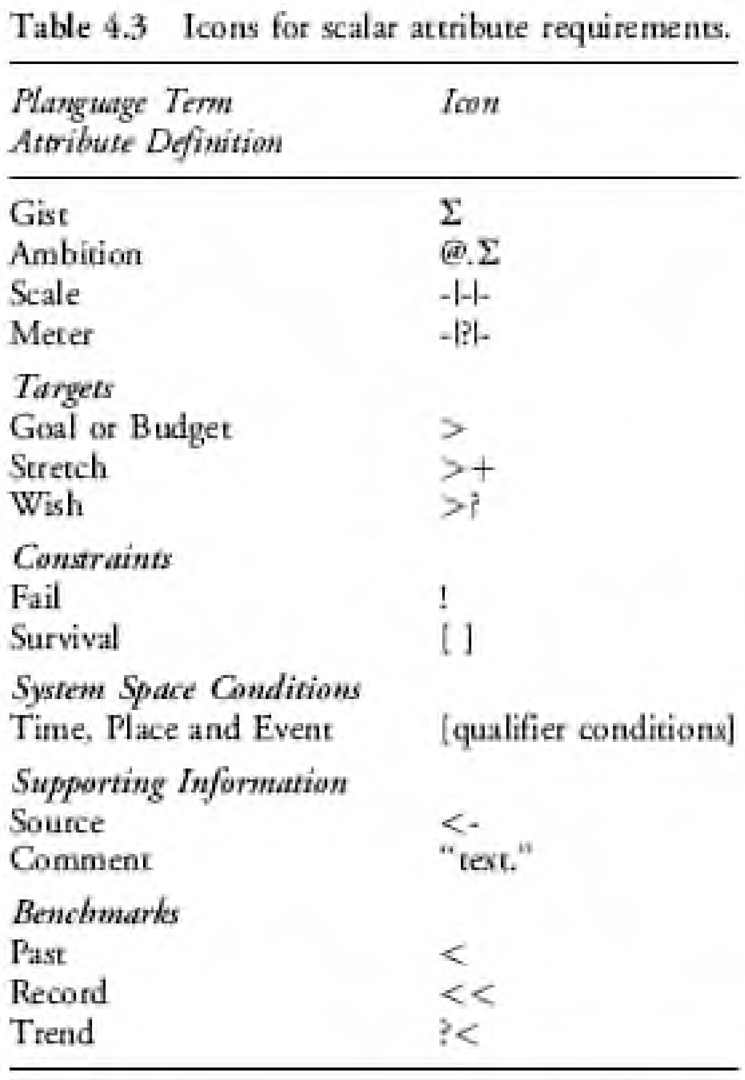


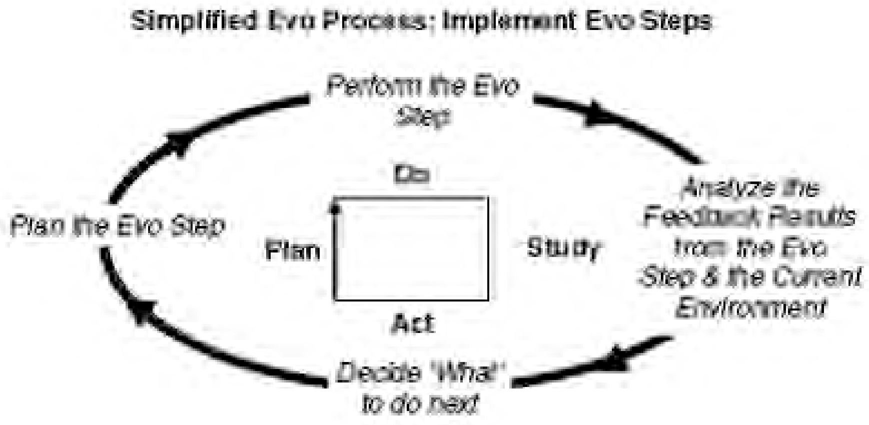
Table 4.3 Icons for scalar attribute requirements.

Planguage Term *Attribute Definition*	*Icon*
Gist	Σ
Ambition	@.Σ
Scale	-|-|-
Meter	-|?|-
Targets	
Goal or Budget	>
Stretch	>+
Wish	>?
Constraints	
Fail	!
Survival	[ ]
System Space Conditions	
Time, Place and Event	[qualifier conditions]
Supporting Information	
Source	<-
Comment	"text."
Benchmarks	
Past	<
Record	<<
Trend	?<

Table: Some fundamental performance and cost attribute keyed icons. <- CE p.134.

Scale '-|-|- is directly derived from Blissymbolics [Bliss]

- A second characteristic of Plicons (Planning Language Icons, or Planguage Icons) is that they are designed to be applicable to a very *broad range* of systems engineering activity – the entire development and operational life cycle. Most other graphical languages seem to focus on *requirements* or *design* stages.

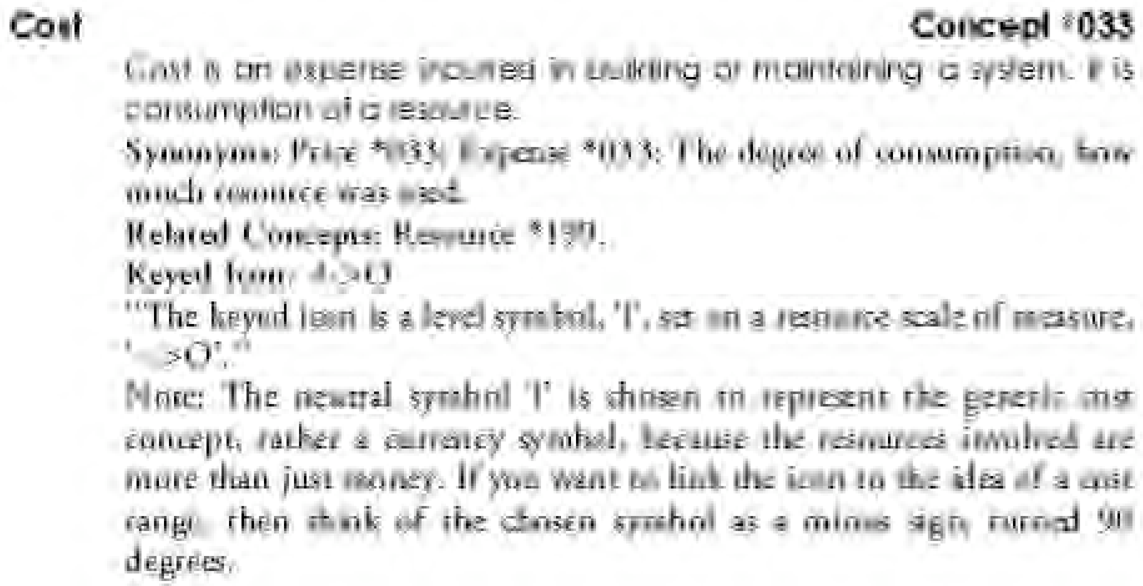


Simplified Evo Process: Implement Evo Steps
Perform the Evo Step
Do
Analyze the Feedback Results from the Evo Step & the Current Environment
Plan the Evo Step
Plan
Study
Act
Decide 'What' to do next

Figure 10.3
A simplified Evo process: implementing Evo steps.

Figure: The process symbol defined the PDSA cycle explicitly. <- CE p.307

- A third characteristic is that the individual icon elements reference a Planguage-defined *concept*, they are clearly defined, and consciously *integrated* with all other defined Planguage concepts.
- Here is an example of definition and integration.



Cost **Concept *033**

Cost is an expense incurred in building or maintaining a system. It is consumption of a resource.

Synonyms: Price *033, Expense *033: The degree of consumption, how much resource was used.

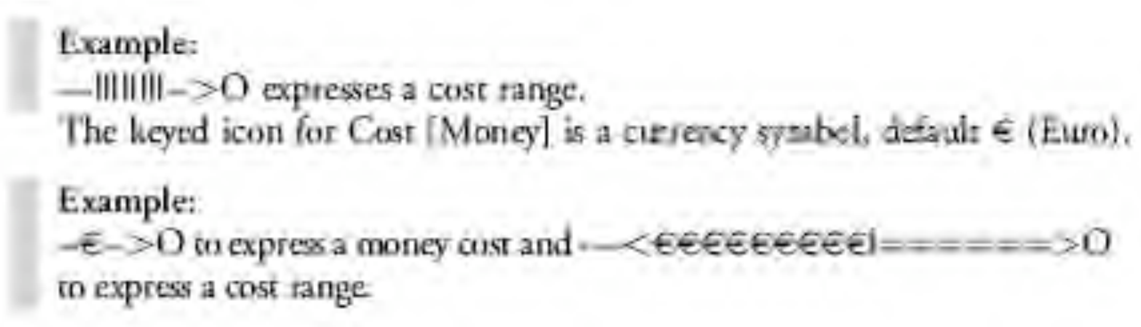
Related Concepts: Resource *199.

Keyed Icon: -I->O

"The keyed icon is a level symbol, 'I', set on a resource scale of measure, '-->O'."

Note: The neutral symbol 'I' is chosen to represent the generic cost concept, rather a currency symbol, because the resources involved are more than just money. If you want to link the icon to the idea of a cost range, then think of the chosen symbol as a minus sign, turned 90 degrees.

-


Example:
—|||||||–>O expresses a cost range.
The keyed icon for Cost [Money] is a currency symbol, default € (Euro).

Example:
–€–>O to express a money cost and ·—<€€€€€€€€€|======>O
to express a cost range.

Example of a Planguage concept with keyed icons [CE, pages 344-5, Concept Glossary].

- Here are some more Plicon characteristics:
    - Drawn and Keyed Plicons: plicons are defined with both a drawn format and a keyable format (keyable from a conventional computer keyboard). The two formats are designed to be as recognizably close as possible; while still considering convenience of keying the icon in practice.

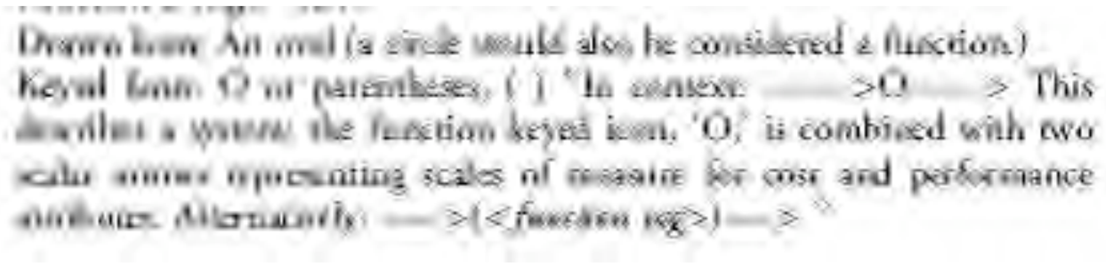

Drawn Icon: An oval (a circle would also be considered a function.)
Keyed Icon: O or parentheses, ( ) "In context: ——>O——> This describes a system: the function keyed icon, 'O,' is combined with two scalar arrows representing scales of measure for cost and performance attributes. Alternatively: ——>(<function tag>)——>"

Illustration from CE, p.361 of defining both Keyed and Drawn Icons for 'Function'.

- Mixed Graphic and other Planguage notation: The Plicons can be integrated with any of the formally defined Planning Language (Planguage) structures, grammar, defined concepts, or text notations available in Planguage; whether it be from Planguage or more locally defined user or project definitions or notation.

Planguage Concept Glossary 431

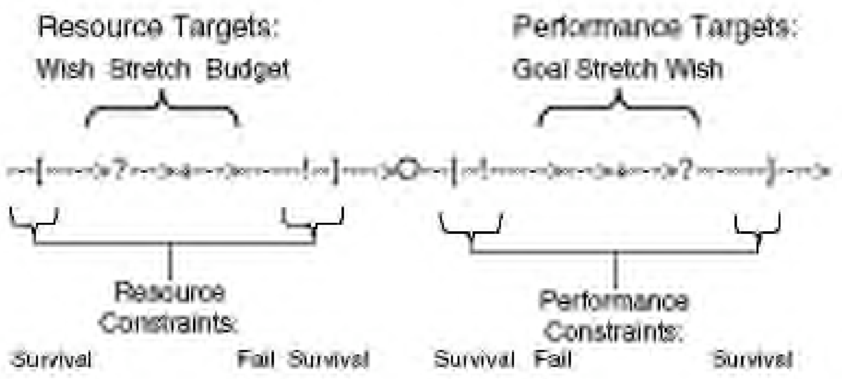


Figure G31
Shows scalar targets can be specified for both performance and resource requirements.

Example of mixed keyed icons and text in CE. [CE p431].

- Optionality: the Plicons can be used at the specification writers discretion, mixed with any other Planguage notation, or not at all.

(Design Specs)---Quality--> ^🡐 (Requirement Specs)---->{Clarity, Detail, Precision}--->
Design spec quality is impacted by these three requirement spec qualities.

(Requirements Spec) ➔ ^[Initial Design: ^[Constraint Filtering] ➔ ^[Feasibility] ➔ ^[Benefit/Cost Optimize]] ➔ (Design Specs)
Designs should go through two initial processes. Constraint filtering and feasibility. Then be optimized for Benefit/Cost.

(Initial Design Specs, Requirements) ➔ ^[Evo Process {Build, Try, Study, Adjust Specs}] ➔ (Field Results).
Initial specs need adjustment early and frequently by being applied in reality.

(Cost Requirements) ➔ ^[Design-to-Cost] ➔ (Design Spec)
We need to 'Design to Cost', not 'Cost a Design'.

(Requirements)➔ ^[Theoretical Design] ➔ ^[Estimate Cost] ➔ ^[Evo] ➔ (Estimates based on Reality)
Future cost estimation is more accurate if based on early Evolutionary delivery realities.

Start: ^[System Operation] ➔ ^[Capacity Expansion] ➔ ^[Redesign to cope with Expansion] ➔Start**.**

Initial successful designs might have to be adjusted for growth and change.

Example: Mixed Keyed icons and text used to express systems engineering relationships. To defined Planguage itself. This is experimental and I have not made it public, for example in the CE book. ^[this is a process symbol]

Objectives of the Planguage icons

- The Plicons are designed to satisfy the following objectives in the Planguage context.
  - Language-neutral notation
    - The icons are designed to not rely on any particular human language.
  - Keyable:
    - The icons are selected because they can be conveniently keyed in from a normal computer keyboard
  - Defined:
    - The icons have a well-considered conceptual definition in the Planguage Concept Glossary. The icon itself is *but one means* of accessing the concept.
  - Consistency:
    - The drawn icons and the keyed icons have reasonable graphical similarity, to aid recognition and learning.
    - The chosen icons are designed to be consistent with each other. For example all 'benchmarks' point leftwards, all targets point rightwards. See Illustration 1 above.
  - Optionality
    - The Plicons are optional in use and alternatives exist. They should be selected for use voluntarily because they offer the user some advantage.

Principles of Plicon design

Here is a set of Keyed Icon Rules:

1. Keyed icons are keyboard character sets with defined meanings.
2. In general they will correspond, as far as possible, with graphic or drawn icons.
3. Their detailed and official meaning will generally be found in the glossary.

4. They should all have a defined term number (*nnn) in the glossary or here.
5. They should be simple to remember.
6. They should consistent in use of terms and sequences.
7. Left side is equivalent to graphic icon top, and right side to bottom ((Input)➔^[…] = Input to Process)
8. A '.' can be used to couple words/terms in a single concept. To give clarity and ambiguity. Optional: (+.➔O.-|-|-.#.± Incremental.cost.scale.estimate.deviation)
9. At least one space shall be inserted between an icon and adjacent terms.(avoid ambiguity!)

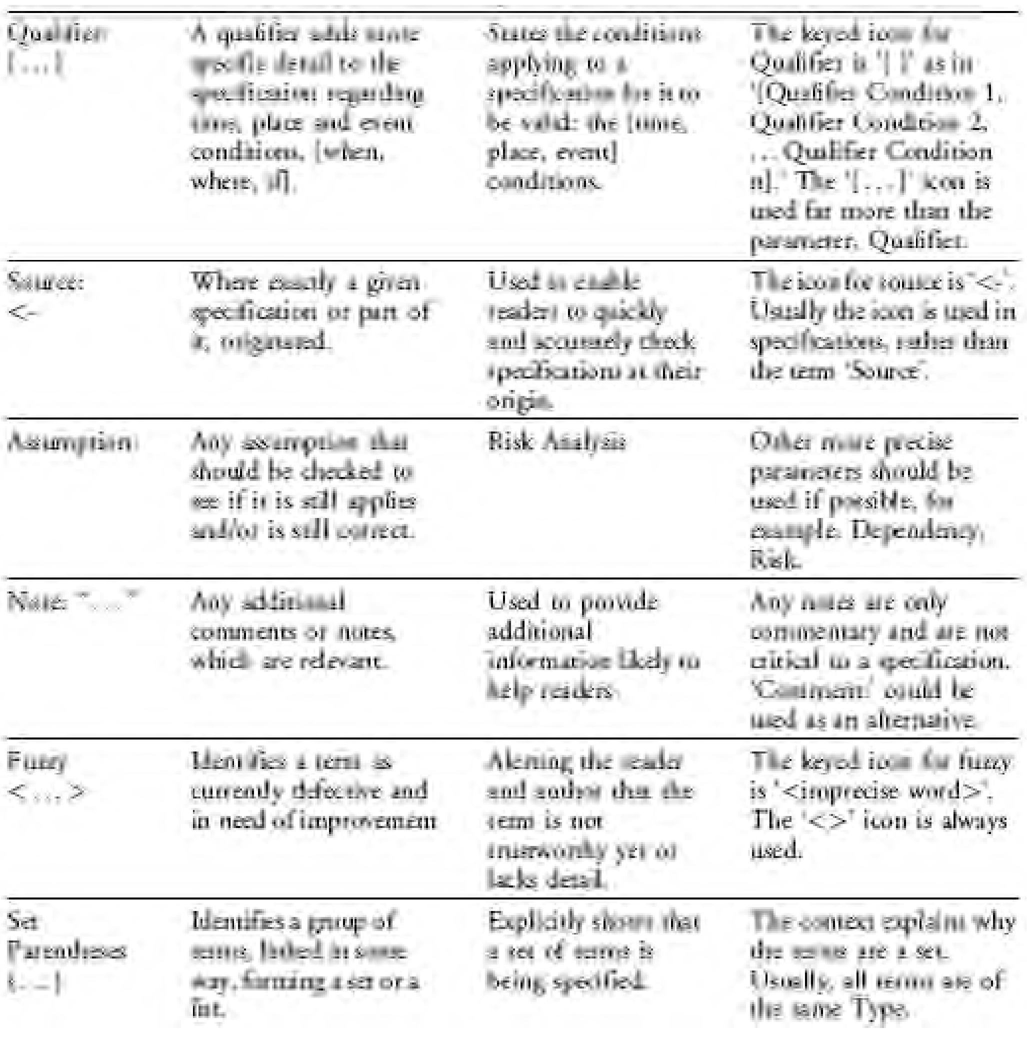
➔[P] means Entry to P (➔[] is Entry symbol), and ➔ [P] means flow to P, or ➔.[P]

'Rules' are good-practice specification guidelines, and can be used to detect 'defects' in a specification. Source: Gilb, Keyed Icons MASTER Specific Rules for Keyed Icons. Version Oct 18 2005.

Basic Plicons

- There are a number of keyed icons that are used regularly in Planguage text, and indeed preferred over human language equivalents because of their brevity and clarity.

- Here are some of them



Qualifier [. . .]	A qualifier adds more specific detail to the specification regarding time, place and event conditions, [when, where, if].	States the conditions applying to a specification for it to be valid: the [time, place, event] conditions.	The keyed icon for Qualifier is '[ ]' as in '[Qualifier Condition 1, Qualifier Condition 2, . . . Qualifier Condition n].' The '[. . .]' icon is used far more than the parameter, Qualifier.
Source: <-	Where exactly a given specification or part of it, originated.	Used to enable readers to quickly and accurately check specifications at their origin.	The icon for source is '<-'. Usually the icon is used in specifications, rather than the term 'Source'.
Assumption:	Any assumption that should be checked to see if it is still applies and/or is still correct.	Risk Analysis	Other more precise parameters should be used if possible, for example: Dependency, Risk.
Note: " . . . "	Any additional comments or notes, which are relevant.	Used to provide additional information likely to help readers.	Any notes are only commentary and are not critical to a specification. 'Comment:' could be used as an alternative.
Fuzzy < . . . >	Identifies a term as currently defective and in need of improvement	Alerting the reader and author that the term is not trustworthy yet or lacks detail.	The keyed icon for fuzzy is '<imprecise word>'. The '<>' icon is always used.
Set Parentheses { . . . }	Identifies a group of terms, linked in some way, forming a set or a list.	Explicitly shows that a set of terms is being specified.	The context explains why the terms are a set. Usually, all terms are of the same Type.

Example of some of the keyed icons used on a regular basis in Planguage. Source CE, page 15.

- Here is a real client Planguage example of use of several of these keyed icons to define a function specification:

Emergency Stop:

Type: Function.

Description: <Requirement detail>.

Module Name: GEX.F124.

Users: {Machine Operator, Run Planner}.

Assumptions: The User Handbook describes this in detail for all <User Types>.

User Handbook: Section 1.3.5 [Version 1.0].

Planned Implemented: Early Next Year, Before Release 1.0.

Latest Implementation: Version 2.1. "Bug Correction: Bug XYZ."

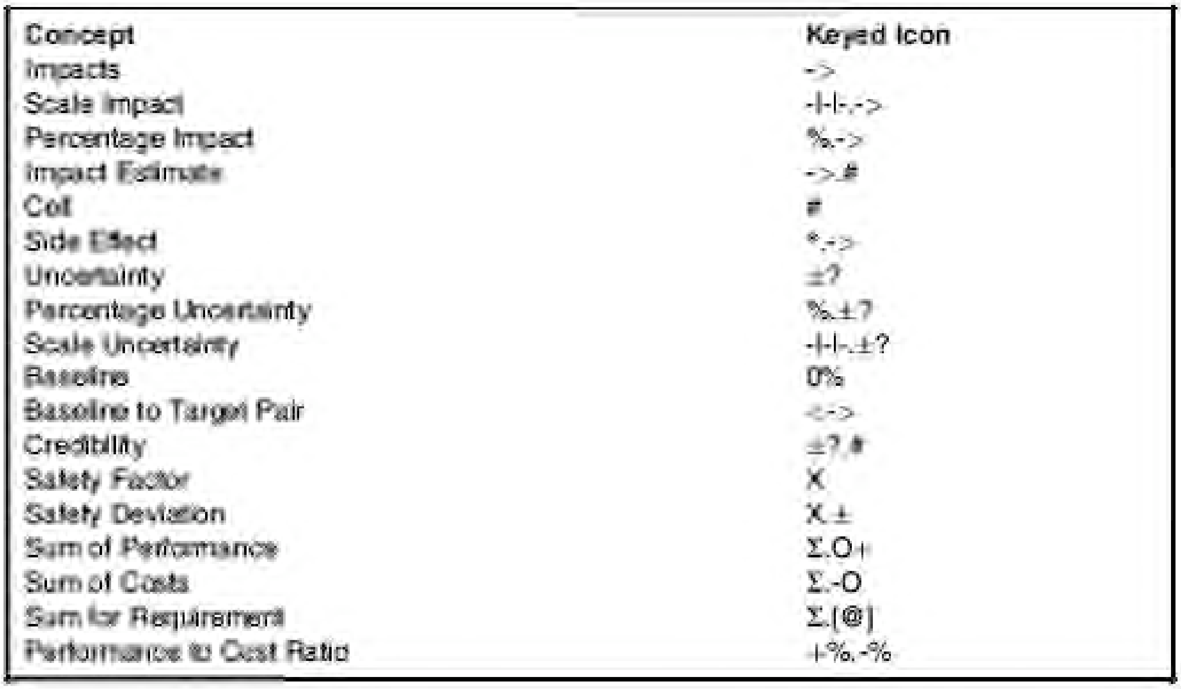
Test: FT.Emergency Stop. <- Carla

Test [System]: {FS.Normal Start, FS.Emergency Stop}.

Hardware Components: {Emergency Stop Button, Others}.

Owner: Carla

Source CE p.91. Note the Set, Qualifier, Source, Note, and Fuzzy brackets being used.



Concept	Keyed Icon
Impacts	->
Scale Impact	-|-|-.->
Percentage Impact	%.->
Impact Estimate	->.#
Cost	#
Side Effect	*.->
Uncertainty	±?
Percentage Uncertainty	%.±?
Scale Uncertainty	-|-|-.±?
Baseline	0%
Baseline to Target Pair	<->
Credibility	±?.#
Safety Factor	X
Safety Deviation	X.±
Sum of Performance	Σ.O+
Sum of Costs	Σ.-O
Sum for Requirement	Σ.[@]
Performance to Cost Ratio	+%.-%

Here is a set of keyed icons relating to the Impact Estimation method (Source Figure 9.11, in CE page 287, Impact Estimation Chapter). Notice how the symbols '->', '±', 'Σ' and '?' are used to build concepts.

Practical Application thus far

- The frequently used subset of the keyed icons are regularly used amongst engineers using Planguage.
- The more exotic defined keyed icons are hardly used by anyone at all, except the author for the purpose of defining them at all – for example the Impact estimation set above (CE p.287). But

then Planguage is a relatively young language, and I would expect the use of icons to grow with the use of the language in time.

- The graphical icons are used frequently and regularly in our teaching slides to explain concepts, especially those to do with performance attributes.

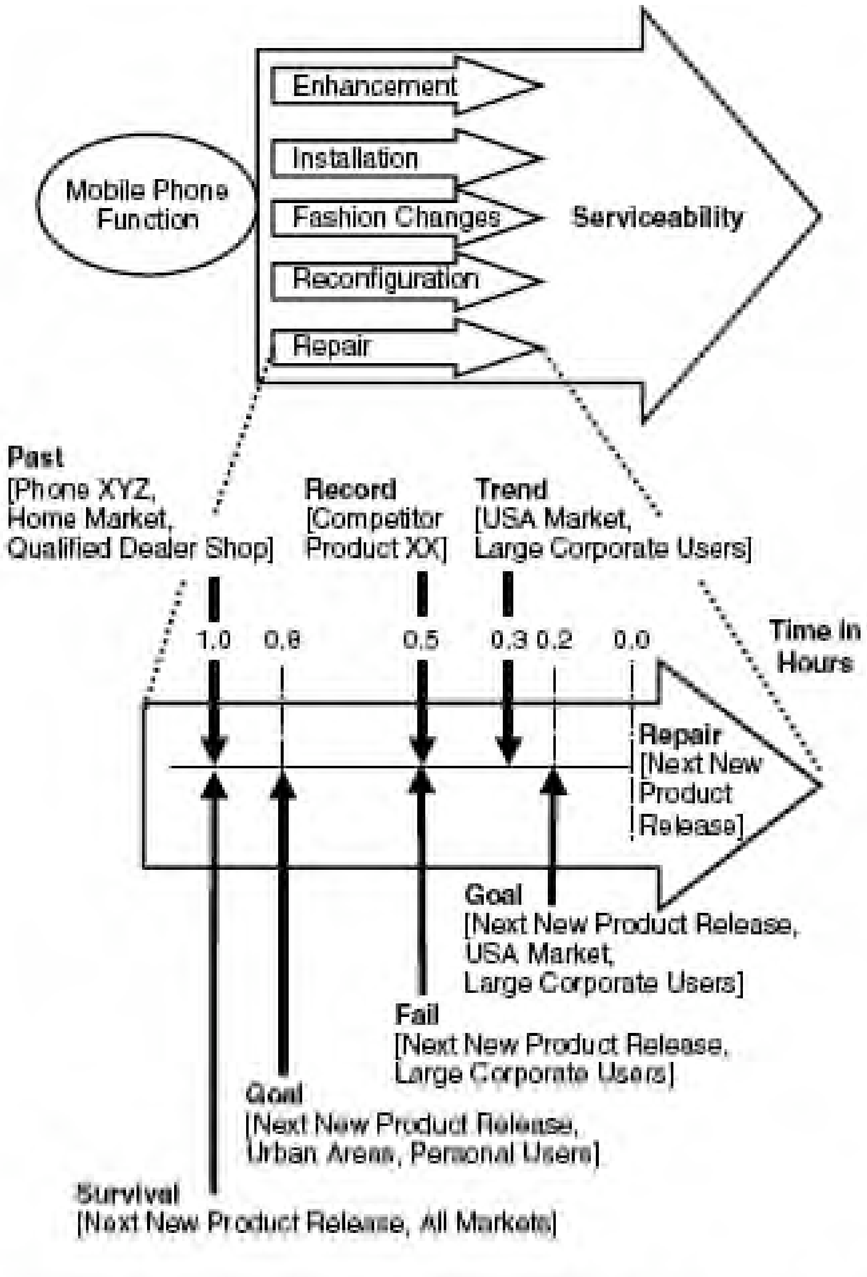

Enhancement
Installation
Mobile Phone Function
Fashion Changes
Serviceability
Reconfiguration
Repair
Past [Phone XYZ, Home Market, Qualified Dealer Shop]
Record [Competitor Product XX]
Trend [USA Market, Large Corporate Users]
1.0
0.8
0.5
0.3
0.2
0.0
Time in Hours
Repair [Next New Product Release]
Goal [Next New Product Release, USA Market, Large Corporate Users]
Fail [Next New Product Release, Large Corporate Users]
Goal [Next New Product Release, Urban Areas, Personal Users]
Survival [Next New Product Release, All Markets]

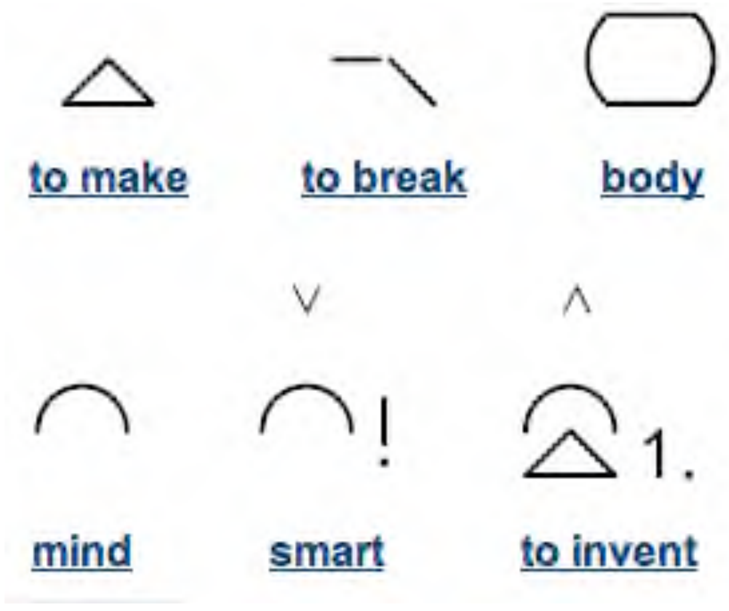
Figure 4.2
Serviceability, a complex performance requirement, decomposes into numerous performance attributes. One of these, the quality, Repair is expanded to show its targets and constraints and, the supporting benchmark information.

Illustration: drawn icons used to illustrate my book. <- CE p.115

- It would seem difficult to teach the quantification of quality concepts without using the drawn icons.

Work Remaining to do to develop Plicons

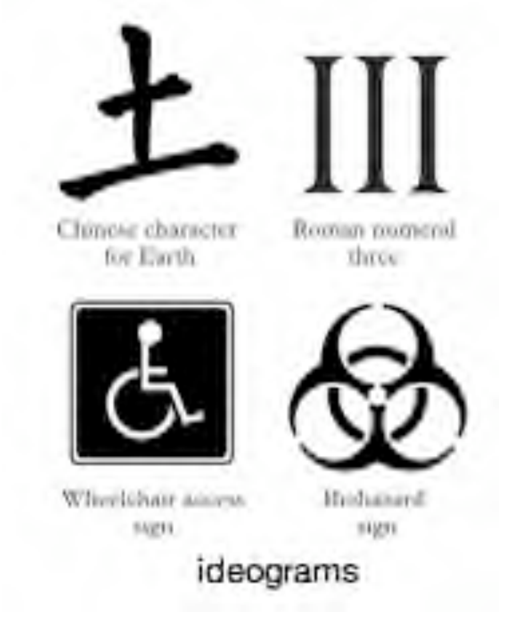
- I have done a lot of work defining icons on my own, but I saw no point in publishing things that were not in fact in frequent use by me and my clients. I initially went quite far in defining Planguage concepts with the help of long strings of Planguage keyed icons. But then I removed all that from the final CE book, as it seemed academic for the intended audience.
- I still have a dream, that requires lots of hard work, by someone, to define a systems engineering discipline primarily by using the icons. This can be with minimum human text, or; in my dreams, with none whatsoever.
- The ideal is a fairly complete language like mathematics or Blissymbolics [Bliss]. The excellent work of Charles Bliss, who I corresponded with many times, was a major inspiration. His language is so well developed, even for scientific and engineering purposes, that it is an open question of whether it would be a good base for a very comprehensive systems engineering language.



Bliss Symbols

- In my own case I saw the need for a few dozen frequently-used symbols for everyday notation, interspersed with human text (as in electronic and music), and a few hundred symbols in total to make a useful systems engineering symbolic language.

- In spite of a large number of graphic notations in the software engineering industry, such as UML, and starting with conventional flowchart symbols about half a century ago, I have been surprised that the software symbols kept closely to the idea of describing logic, but did not ever include graphical notions of system performance and cost. I am afraid that this reflects the narrow education of the programmer. This cultural lack of concern about quality and cost is, in my opinion, a major reason for the high widely-reported IT-system failure experience. When the software community wakes up to the need to act like systems engineers instead of coders, then hopefully the corresponding need for a graphical language to communicate about quality and cost will arrive.
- My personal position is that *real need* must dictate the development work effort that should be put into developing an iconic systems engineering language. But there is some academic fun, as Bliss' life illustrates, in just seeing what we can create. Should any energetic soul wish to develop this I wish them luck and would like to give them a base, some advice, and encouragement.
- Who knows? Maybe I am forgetting the Chinese ideogram's inspiration of Bliss, and that the necessary symbols will simply be an extension of Chinese Ideograms?



-

- The web gives plenty of symbol ideas! [http://www.symbols.net].

Summary

- Planguage, a systems engineering language, strong in performance and cost attribute modelling, also contains a complimentary and optional set of graphical symbols. These planning language icons differ from other graphic modelling languages in their ability to describe variable system performance and cost attributes.
- The graphic language is in practical daily use, both for systems engineering purposes, and teaching purposes.
- It is partly, but not completely, described in 'Competitive Engineering', the Planning Language Handbook.
- The purpose of this paper is to generate awareness that there is a graphical modelling way to describe systems engineering information. And it is far more realistic for both software and system engineering purposes than the currently popular modelling techniques such as UML.

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