

Architectural Framework and Modeling Innovations for Attribute Based Testing & Evaluation

Eric Smith, Ricardo Pineda, Justin Kieser

ESmith2@UTEP.edu

Systems Engineering Program

Research Institute for Manufacturing and Engineering Systems

University of Texas at El Paso

Abstract

Architectural frameworks and modeling languages have traditionally supported the graphing and description of components of the architectural synthesis, system functions, and most recently, system requirements. The history of testing and evaluation of government acquisition systems shows, however, that fitness in global attributes plays a key role in system success. It is therefore imperative that architectural frameworks and modeling languages be modified to illustrate, to the testing and evaluation, acquisition, and design communities, specifically and precisely what global attributes are being expected from a system. Attributes are characteristics of value inherent in situations or objects, and have been defined as “complex intangible characteristics that are difficult to capture, but yet are crucial in directing systems engineering activities” [Smith and Bahill, 2010, p. 3]. “Attributes” are global qualities that the customer needs.

Attributes such as Testability, Inspect-ability, Affordability, Configurability, Extensibility, and Safety should be as clearly defined as are the requirements that specify capability behavior, functions, or low-level features. Attribute characterization for testing and evaluation is increasingly important because expectations and mandates for complementary systems are now the norm.

Attributes can be connected all the way down to a design parameter space through various means. Accepted frameworks such as the Department of Defense Architectural Framework (DoDAF), modeling languages such as the Systems Modeling Language (SysML), and simulation, as well as general hierarchical structures such as tradeoff studies, or less structured means, such as architecting principles and heuristics, are aids for design. However, these means are sometimes insufficient. In all approaches, definitions and measures are essential.

1, Attribute Hierarchical Architecture: Attribute and Capability Characterization for Testing and Evaluation

Attributes remain unaffected and unchanging in the face of changing technologies of implementation. The same can be said, but in a lesser degree, of capabilities, requirements, and functions. Note that capabilities can be defined as required functionality, or, a required set of functions. "Capability" is a required functionality that includes one or a set of functions with prescribed performance. Wasson defines a capability as "an explicit, inherent feature activated or excited by an external stimulus to perform a function (action) at a specified level of performance until terminated by external commands, timed completion, or resource depletion" [2006, p. 26].

Abstract, unchanging attributes establish stable global reference qualities, while allowing evolutionary interfacing of system of systems (SoS) level, even as requirements and functions evolve at the system level. Figure 1 illustrates the transition from the top-level attribute Effectiveness, to the capabilities Deployability, Lethality, Mobility, and Durability, to the different functions performed by various military units.

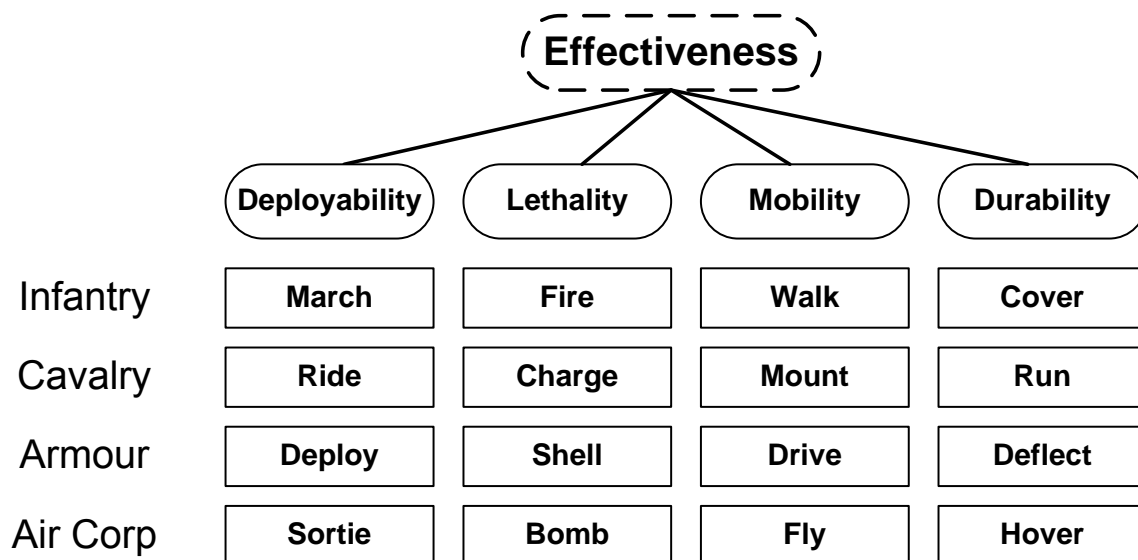


Figure 1: Attribute, capability and functional allocation for various military units

A focus on attributes and capabilities allows stability in objectives, even in the face of changing requirements and functions. The differentiation and separation of basic systems engineering elements at different levels [Bahill, Szidarovszky, Botta and Smith, 2008] is fundamental to the clarity of this transition hierarchy.

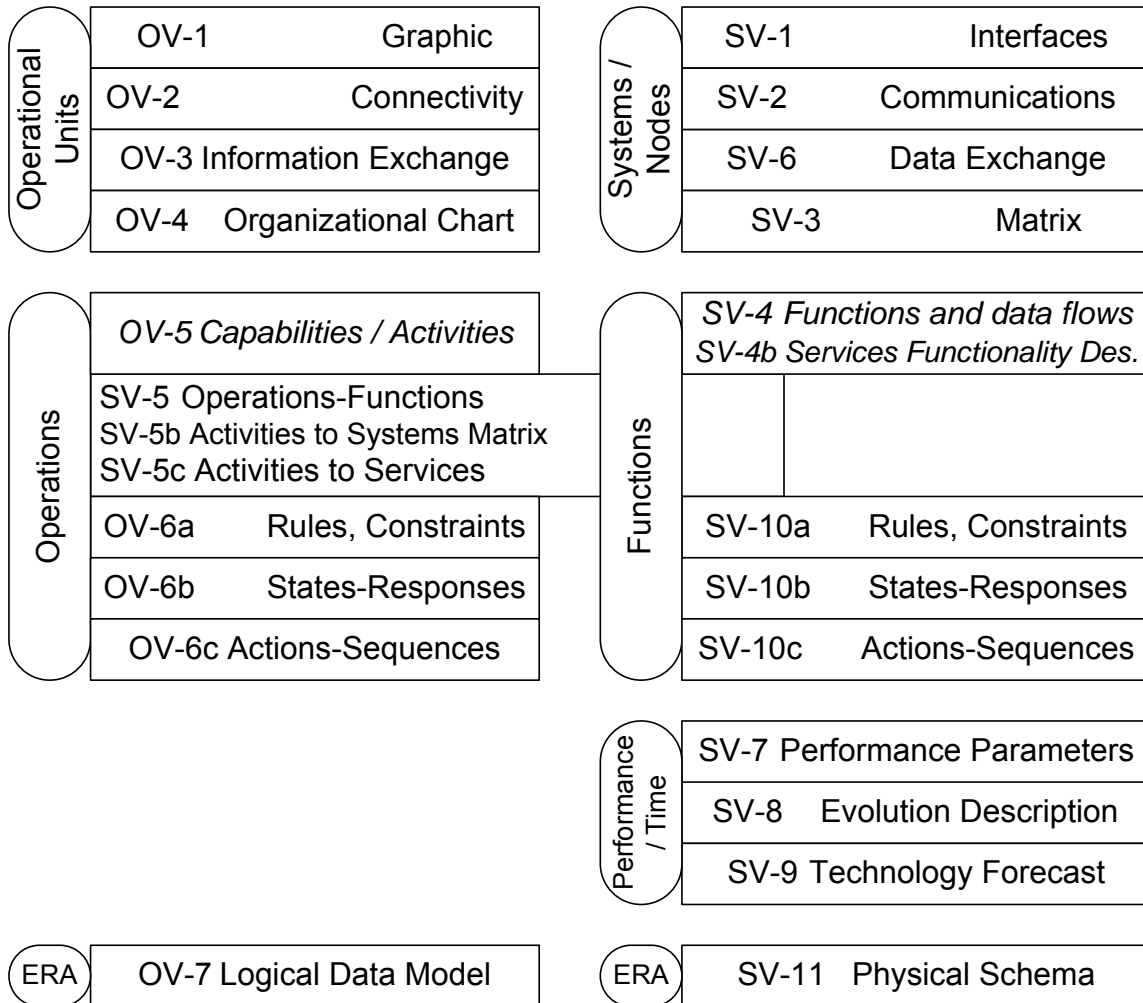
2, DoDAF

Architectural frameworks such as the Department of Defense Architectural Framework (DoDAF) have not yet evolved to adequately describe attributes in architectures. DoDAF only mentions capabilities. DoDAF, like the Systems Modeling Language (SysML), is suited to diagram functions and more concrete components in an architectural synthesis. Recall that the development of physical design methods has traditionally lagged behind in the areas of the graphical representation of system functions and states, and most recently, system requirements and capabilities. For example, SysML only recently added a Requirement Diagram. Limitations in DoDAF are similarly apparent.

Figure 2 shows the alignment and interfacing of DoDAF views. Note that there is a good general correspondence between the Operational Views and the System Views; however, the numbering of the views and their titles can bear with improvements for the sake of clarity and ease of use. In the area of describing the performance of architectures in time, there is a lack of balance between the systems views, and the missing corresponding operational views. Specifically, SV-7 Performance Parameters, SV-8 Evolution Description, and SV-9 Technology Forecast should be mirrored by operational views that describe *Capability Performance Parameters*, *Capability/Operational Evolution*, and a *Forecast of Capability Changes*. Further, the same views should be extended for global attribute characterization; for example, with *Attribute Characteristics*, *Attribute Evolution*, and a *Forecast of Attribute Changes*.

**Department of Defense Architectural Framework (DoDAF)
Alignment & Interfaces of Views**

AV-1 Overview & Summary
AV-1 Dictionary



TV-1 Standards Profile
TV-2 Standards Forecast

Figure 2: Alignment and Interfacing of DoDAF Views

Note that summary descriptions of the essence of the views have been provided, and that ERA stands for Entity-Relationship-Attribute diagram.

3, Capability Decompositions and Allocations

Capabilities have recently provided relatively stable reference hierarchies for allocation, allowing changeability at the functional and architectural levels, as implementing sub-systems and components have evolved. Figure 3 shows how the capability of *Playing Music* has remained constant, even as implementing technologies have evolved.

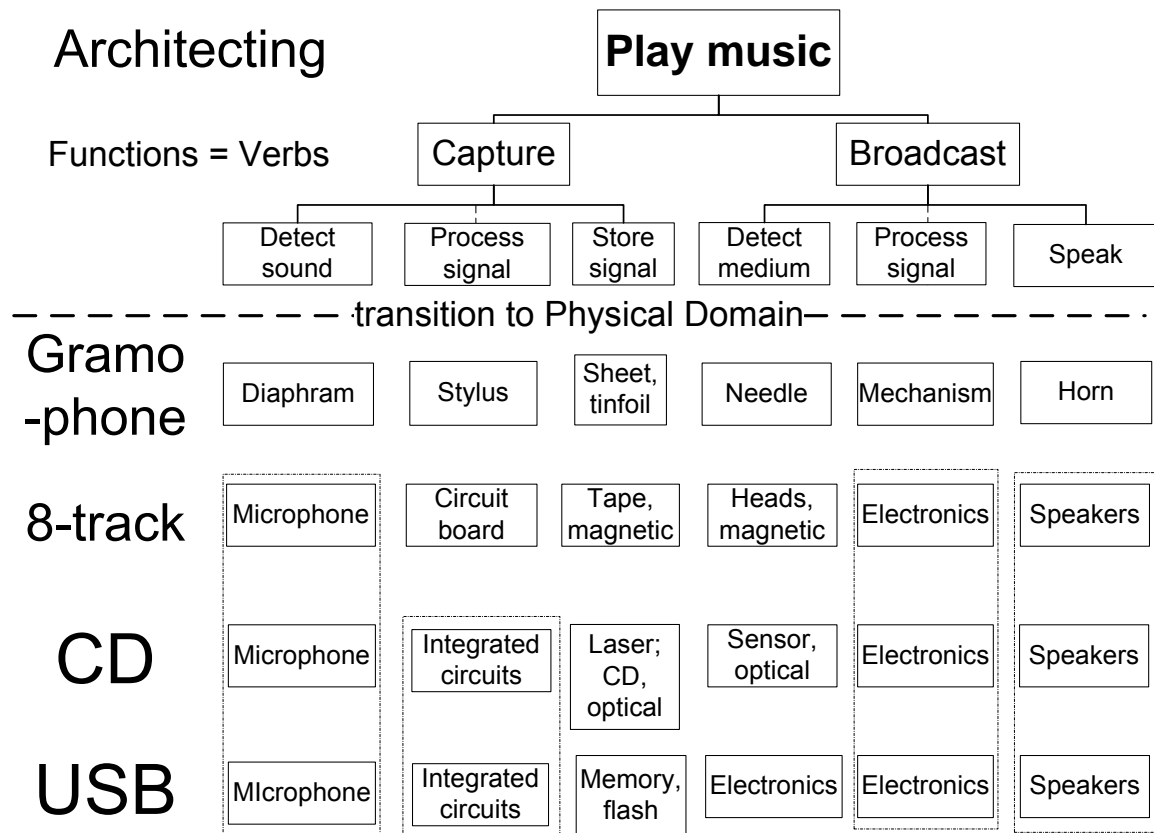


Figure 3: *Play music* capability decomposed into functions and implemented in changing technologies

Figure 4 shows how the capability *Fly* has remained constant, even its decomposed functions have been implementing in different architectural sub-systems and technologies.

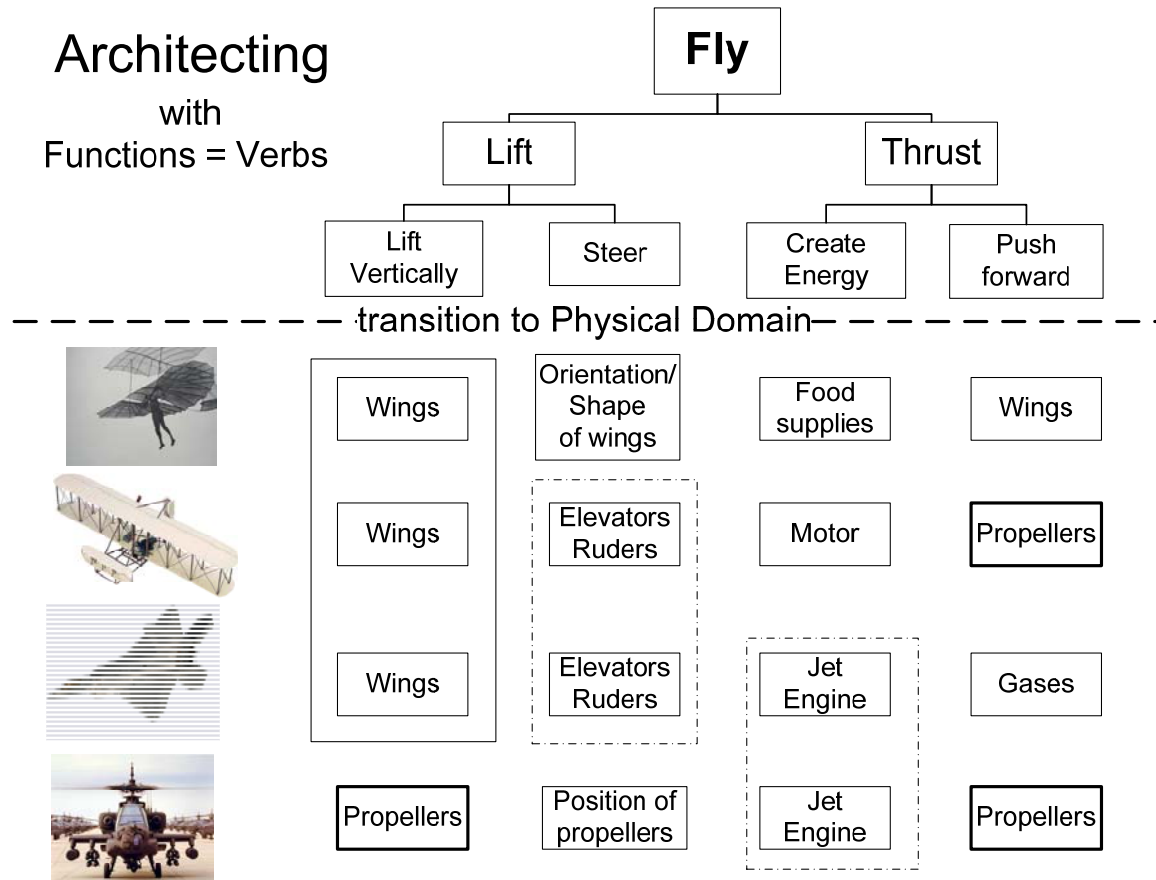


Figure 4: *Fly* capability decomposed into functions and implemented in changing sub-systems and technologies

4, Connecting Attributes and Design Parameter Spaces

There are various traditional ways to formally connect attributes to low-level design parameter spaces.

Tradeoff studies connect attributes to design parameter spaces through hierarchical structures. Figure 5 shows how a tradeoff study connects an Overall Score of the attribute of customer satisfaction with lower-level attributes and measurable parameters, in a tradeoff among car racing formats in a reduced version of Dr. Terry Bahill's Pinewood Derby Tradeoff Study [Bahill, 2006; Bahill and Karnavas, 2000].

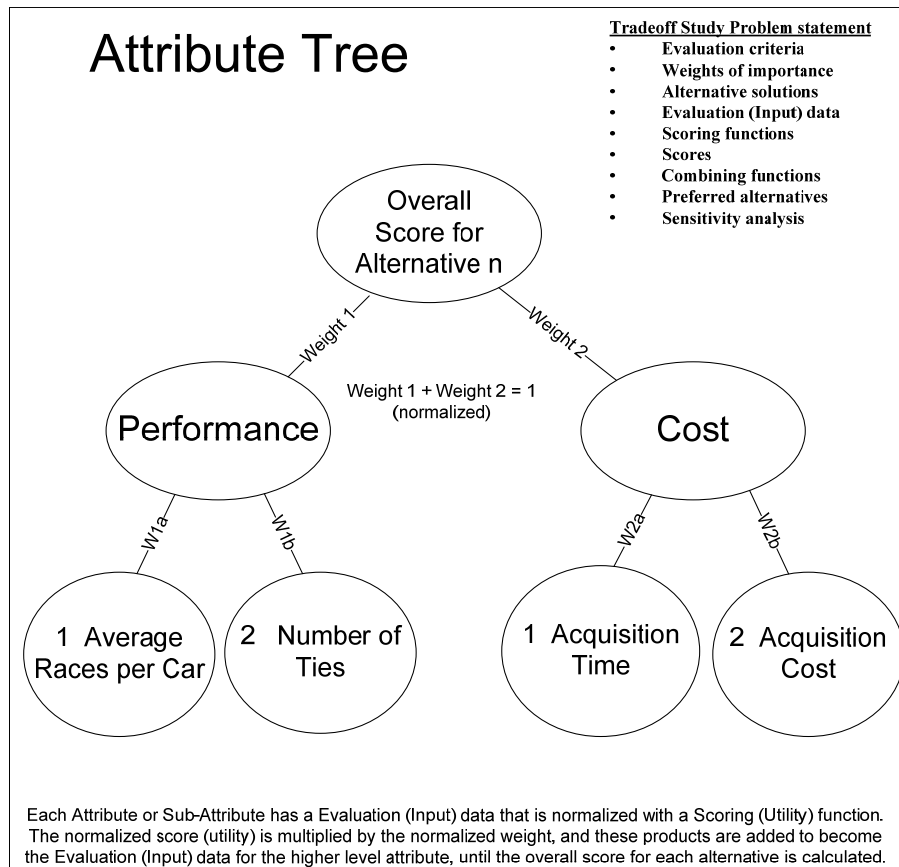


Figure 5: Tradeoff study tree connecting a capability to design parameters

Figure 6 shows the elements of a tradeoff study, and visualizes the parallel application of the same attribute-based tradeoff study to the various alternatives under consideration.

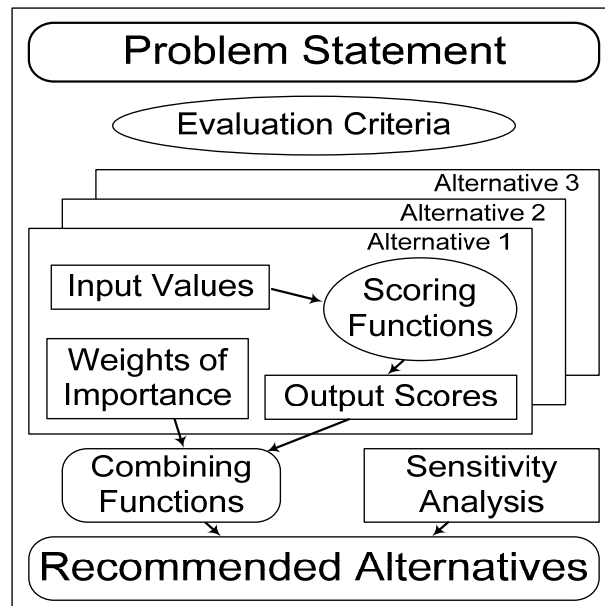


Figure 6: Elements of a tradeoff study

Architecting principles and heuristics for good design also connect high-level attributes and capabilities with lower-level design parameters. An example of an architecting principle, 'Work on high risk items first,' connects the high-level attribute of risk with the low level components that may cause the risk. Rechtin [1991] lists a number of architecting principles that essentially connect high-level attribute needs with low-level design parameters:

- "Occam's Razor: "The simplest solution is usually the correct one"
- Keep It Simple, Stupid (KISS)
- Simplify, Simplify, Simplify
- Relationships among the elements are what give systems their added value
- Leverage is greatest at the interfaces
- Partitioning: choose elements that are as independent as possible, with minimal communication between elements
 - Low external complexity; high speed
 - High internal complexity; slow change
- Do not partition or slice where high rates of information exchange are necessary
- Support element architecture must fit supported architecture
- Amid a wash of paper, a small number of documents become the pivot of project mang.
- "Don't every stop talking about the system"

- There's no such thing as Immaculate Communication!"

Blaauw [1972] gives principles of good information systems architecture:

- 1, "Consistency
- 2, Orthogonality (elements relatively independent of each other)
- 3, Propriety (proper to functions, no unnecessary functions)
- 4, Parsimony (no functional redundancy in different forms)
- 5, Transparency (functions introduced in implementations not imposed on the user)
- 6, Generality (multi-pupose)
- 7, Open-Endedness (alternate uses of needed functions)
- 8, Completeness (in solving needs and desires of user)."

5, Mathematical Hierarchies Relating Attributes to Design Spaces

Mathematically, connecting attributes to a design space involves considering a large number of low-level design parameters, which allows the calculation of a limited number of top-level attributes, with the possible use of intermediary characteristics and measures. Good definitions for these elements are essential.

Procedurally, defining a design space involves constraining each design parameter, either intrinsically or by relations to other design parameters, and constraining top-level attributes, both intrinsically and by relations or weights of importance that the customer has provided. A design space of contiguous point solutions requires:

- 1, Refining a solution space by reduction of a larger space
2. Hierarchical decomposition and integration of attribute-based or capability-based definitions and measures
3. Mathematical formulation, integration and analysis

Simulations

Simulations can be built relatively easily given today's simulation software. Simulations that seek to characterize global attributes or capabilities from the collective behavior of low-level components require the building of a structure that accomplishes just such a transition. The benefit of simulations is that the stochastic and mathematical consequences of low-level actions on high-level attributes or capabilities need not be predicted or even known to be observable. Simulations may provide higher fidelity representations of a system, and also allow inherent synergies between system parts to emerge in the collective behavior of a system.

Multi-Objective, Linear and Non-Linear Formulations

Mathematically, this general design problem can be modeled as a multi-level network with multiple objectives that can be approximated and solved as a linear program. The beginnings of a linear mathematical formulation for multi-attribute or multi-capability optimization, where the attributes or capabilities are denoted as x_1 through x_n , and weighting constants are denoted a_1 through a_n , is found in the objective function:

$$\text{Maximize: } f(\mathbf{a}, \mathbf{x}) = a_1x_1 + a_2x_2 + a_3x_3 + \dots + a_nx_n = \sum a_i x_i$$

This objective function allows the unlimited and unbalanced increase of any one of the objectives, as long as the objective function as a whole is maximized. Assuming baseline values for x_i and x_k , and the customer's preferences for increasing the attributes or capabilities by Δx_i and Δx_k , it is possible to formulate a self-balancing objective function with the following mathematics:

$$\text{Maximize: } 1 / [\text{abs} [(\Delta x_i / \Delta x_k)_{\text{desired}} - (\Delta x_i / \Delta x_k)_{\text{actual}}]] \\ \text{for } \Delta x_i > 0, \Delta x_k > 0$$

The effect of the formula is to place a self-balancing constraint on two (or more) objectives.

Of course, real design problems will be non-linear, with non-linear objectives and constraints. Optimizing multi-objective solutions will involve Pareto fronts and the decision maker's choice among efficient solutions – essentially referring back to some degree of subjectivity.

The utility of mathematical formulation is often debated because of the difficulties of 1) settling on institutional agreement for the definitions of global attributes, objectives and capabilities – and even of design parameters, 2) the large size of the non-linear problem, and 3) the un-definability of some attributes, objectives and capabilities – considering that new designs are being examined, and novelty will always play a role in design.

6, Conclusion

Several hierarchically structured arrangements that lend insight to relations between global attributes and local components have been described, including decomposition of attributes into capabilities and functional allocation, DoDAF and modeling languages, tradeoff studies, architecting principles & heuristics, simulations, and mathematical linear & non-linear programs. The general design problem -- of creating global attributes from collections of low-level components -- is usually solved by a combination of these methods, and emerging methods.

Customer-driven design often requires that design decisions be made at high levels of abstraction far removed from system component parameters. Customers often want the integrated architectural synthesis to exhibit a limited number of prioritized attributes, or high-level holistic qualities, while also fulfilling all performance requirements. Design work for customers who rightfully insist on focusing on high-level attributes or capabilities must incorporate hierarchical methods for considering all levels of design in a transition continuum: Attributes, capabilities, requirements, functions, measures, and component parameters. The evolution of design methods toward such wide ranging continuum is expected.

References

- A. T. Bahill, and W. J. Karnavas, Risk analysis of a pinewood derby: A case study, *Systems Engineering* 3(3) (2000), 143-155.
- A. T. Bahill, Tradeoff studies: A systems engineering skills course, BAE Systems, San Diego, CA, 2006.
- A. T. Bahill, F. Szidarovszky, R. Botta, and E. D. Smith, Valid models require defined levels, *International Journal of General Systems* 37(5) (2008), 553-571.
- G. A. Blaauw, Computer architecture, *Electronische Rechenanlagen*(4) (1972).
- E. Reichtin, *Systems architecting: Creating and building complex systems*, Prentice Hall, Englewood Cliffs, NJ, 1991.
- E. D. Smith, and A. T. Bahill, Attribute substitution in systems engineering, *Systems Engineering* 13(2) (2010).
- C. S. Wasson, *System analysis, design, and development: Concepts, principles and practices*, Wiley-Interscience, New York, 2006.