The Philosophy of Systems Engineering

Derek Hitchins

Abstract

Systems engineering was engendered, partly, to manage rising complexity in man-made systems, which was supposedly threatening to overwhelm engineering managers in the second half of the 20th Century; it was also inspired by a need to "get the best" out of, i.e. to optimize the performance of, complex military systems. The, then, new discipline appears retrospectively to have been founded in biology, rather than the physics of engineering, perhaps since biologists and anatomists had previously found an effective method of managing complexity in the human body. This method could, in principle, be applied in the new discipline of systems engineering to manage complexity, using holism, synthesis and organicism as guiding principles. The paper considers how systems thinking, systems practice and systems engineering were inspired by this systems philosophy then, and now.

Systems engineering emerged as a problem-solving discipline, initially adopting a simple systems engineering problem-solving paradigm that allowed designers to manage complexity, optimize solutions and avoid becoming embroiled in confusing detail at an early stage of systems design. An alternative form of systems engineering developed in Japan, operating at industry level rather than the West's project/enterprise level. The Japanese also famously employed *kaizen*, the philosophy of continuous improvement, as opposed to the West's so-called Big Bang approach, (get it) Right First Time!

Following the biological metaphor, systems engineering could be seen to synthesize wholes from subsystems, themselves wholes. Wholes were made from interacting parts where the whole could be greater than the sum of its parts. The subsystems could be made from engineered artifacts, or people, or teams, etc., so long as they were wholes. Some 'internal systems' could also be seen as organizing and managing other systems within any whole. Ways were devised of conceiving and designing systems such that the complexity did not overwhelm the designers.

A review of systems engineering as it had evolved by 1986 showed that it was limited in its ability to address complex issues and problems: a systems methodology from 2007, based on the original system theoretic/biological metaphor, showed the difference between the two. Even more marked, however, was the differences between both of these and today's version of systems engineering, which describes itself as multidisciplinary engineering (i.e. engineering), no longer vaunts any claim to the management of complexity, and appears quite unlike its progenitor.

Introduction

Systems engineering was founded, *inter alia*, in two concepts: first, that it was sensible to view some sets, or collections, of interacting 'things,' elements, artifacts, people, etc., that could be consistently described by their behavior, as singular wholes, holons or 'systems'; and second, that it was possible to synthesize more complex systems from parts and systems of lesser complexity. Both concepts transcended the characteristics of discrete parts; in other words, the same principles seemed to apply regardless of the nature of the elements involved.

Systems scientists in the second half of the 20th Century, notably Ludwig von Bertalanffy, expounded General Systems Theory (GST) (Bertalanffy, 1950). Systems theory and systems science were intended, *inter alia,* to overcome the increasing degree of engineering specialization prevalent at the time, leading inexorably to increasing complication and perceived complexity, which was supposedly overwhelming engineering managers. GST was heavily mathematical in style, and did not catch on; however, the models and methods of GST were simpler to grasp and are still much valued.

Although the term would not be coined until around 1940 in the US, 'systems engineering' as a concept and philosophy may be traced back to ancient Egypt and the Pyramids, if not earlier. In the modern era, it was manifest in the progressive optimization of the UK Air Defence System over the period 1936 to 1940, at which point it played a crucial role in the Battle of Britain. It rematerialized in the Cold War of the 1950s as the Total Weapon System (TWS) Concept, a grand design which, in the UK, saw air defense ground radars, airborne early warning aircraft, and squadrons of Hunter and Javelin interceptors integrated into one system with the single aim of defending the UK against a perceived Soviet air threat.

The first real flowering of the TWS Concept came in the late 50s and early 60s, with the advent of Linesman Mediator and the iconic Lightning interceptor, reputedly capable of Mach 2+, with climb capability of 0 to 90,000ft in three minutes…but with radar and missile capabilities limited by the need to sustain such phenomenal performance. The ambitious scheme, Linesman Mediator, saw the coupling of civil and military air traffic to afford nationwide surveillance and control. Linesman, the military part, was to form a focal point from which all of the Lighting aircraft were to be remotely controlled, simultaneously to engage the threat from Soviet standoff weapons, many-on-many (Hitchins, 2007.) The situation eerily presaged the Soviet MIRV threat facing the US two decades on, which instigated President Reagan's Strategic Defense Initiative.

Successive UK air defense systems were more capable and more complex. Systems engineering philosophy and capability evolved, becoming preeminent in managing complexity, and with ensuring TWS performance, which was stretching contemporary technological boundaries in defense. In the 1950s, 60s, 70s and 80s, systems engineering became a byword in the West for innovation, creativity and the management of complexity.

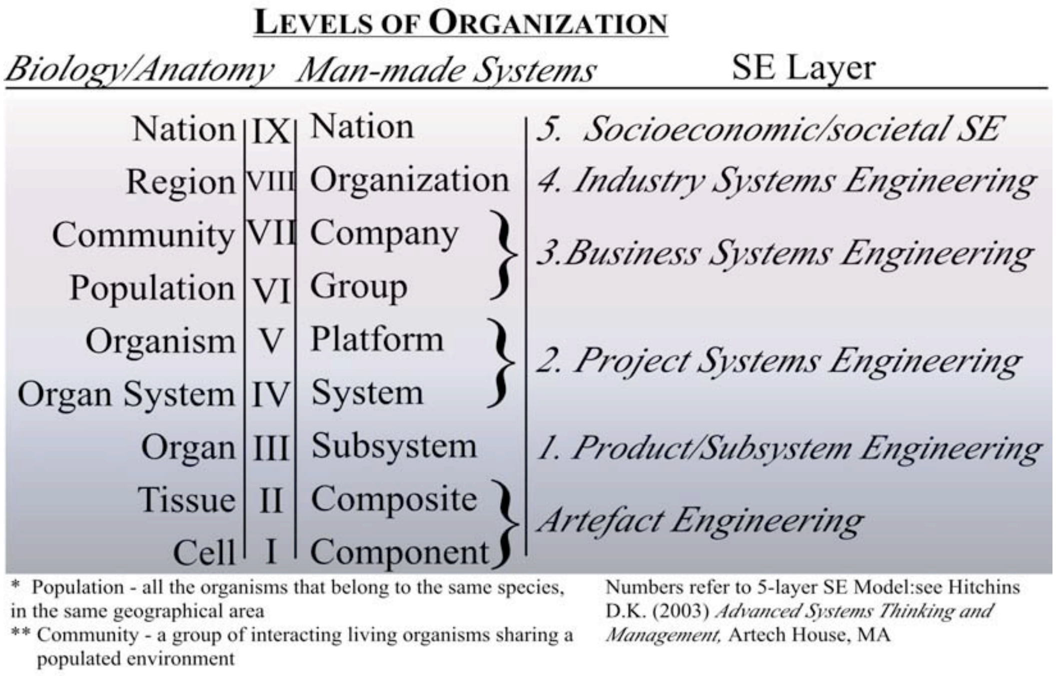
The philosophy[1] of systems science and of systems engineering theorizes how these disciplines were intended to manage innovation, creativity and, particularly, complexity. The paper seeks to identify the foundations of that philosophy and how, if at all, it has motivated and sustained systems thinking, systems practice and systems engineering…

Levels of Organization

Part of the "systems approach" to managing complexity appears, retrospectively, to have found its origins in biology's 'Levels of Organization,' with which today's schoolchildren may be familiar – see the left hand column of Figure 1, which shows the well-known holarchy/hierarchy: cell, tissue, organ, organ system, organism, etc. So, a tissue is more complex than a cell, an organ than a tissue, an organ system than an organ, and so on.

[1] *Study of the theoretical basis of a particular branch of knowledge or experience*. Oxford American dictionary

Plus, each level of organization is formed from the *emergent* properties of the level below, rather than from its *intrinsic* properties. The center column of Figure 1 relates this biological/anatomical holarchy/hierarchy to that of manmade systems.


LEVELS OF ORGANIZATION

Biology/Anatomy		*Man-made Systems*	SE Layer
Nation	IX	Nation	*5. Socioeconomic/societal SE*
Region	VIII	Organization	*4. Industry Systems Engineering*
Community	VII	Company	*3.Business Systems Engineering*
Population	VI	Group	
Organism	V	Platform	*2. Project Systems Engineering*
Organ System	IV	System	
Organ	III	Subsystem	*1. Product/Subsystem Engineering*
Tissue	II	Composite	*Artefact Engineering*
Cell	I	Component	

* Population - all the organisms that belong to the same species, in the same geographical area
** Community - a group of interacting living organisms sharing a populated environment

Numbers refer to 5-layer SE Model:see Hitchins D.K. (2003) *Advanced Systems Thinking and Management,* Artech House, MA

Figure 1. Levels of Organization in Biology and Man-made Systems

The figure suggests that humans, as organisms *('wholes of interdependent parts')*, correspond to platforms, which are typically vehicles, ships, planes, tanks, etc.: this works, in the sense that we humans effectively carry with us, as part of us, our means of locomotion, our sensors, and our intellects, much as does, say, an automobile or a ship.

The right-hand column of Figure 1 presents the corresponding layers (rather than levels) from the 5-layer systems engineering model (Hitchins, 2003), with the basic Layer 1, Product/Subsystem Engineering, corresponding to manmade Subsystem level: this suggests that the synthesis of subsystems is, relatively, the least complex of SE tasks. The 5-layer systems engineering model, then, concerns itself with managing the synthesis of successive 'degrees' of complexity, of which societal/socioeconomic systems engineering, as practiced, typically, by governments and politicians, is the most complex.

Below Layer 1 in Figure 1 is artifact[1] engineering, concerned with the making of parts and composites of parts. This is consistent with the acknowledgment that systems engineering does not make anything (sic), at least not in the sense of manufacture i.e., making by hand. Making artifacts is engineering. Managing the making of artifacts is engineering management. It seems eminently reasonable, therefore, that software – a handwritten artifact – is engineering too…

[1] *Dict*: Artifact; any object made by human beings, especially with a view to subsequent use.

On the other hand, systems engineering may synthesize manmade systems from artifacts previously made by engineers. Systems engineering may also synthesize manmade systems *without* using artifacts, e.g. in creating an organization (arrangement of people systems), in creating a strategy (a plan of action or policy designed to achieve a major or overall aim), in revising a systems architecture (reconfiguration) with differing emergent properties, etc., etc. As with the Battle of Britain Air Defence System in 1940, systems engineering may operate *in vivo,* as well as *in vitro.* So:

- A systems engineer may devise, plan and implement new tactics for a squadron of fighters to undertake interceptions, to evade enemy defenses, etc.
    - S/he might be called a 'Weapon Systems & Tactics Officer.'
- A systems engineer may devise alternative architectures (3:4:3; 4:4:2) and tactics for a soccer team comprised of defenders, midfielders and strikers (subsystems).
    - S/he might be called a soccer team coach or manager, and may be unhappy to be called a systems engineer…

In vivo systems engineers may go by alternative titles, happily unaware that they are *de facto* practicing systems engineers of a kind…

Systems engineering's concern with emergence and levels of organization so closely parallels that of biology and anatomy that it seems not unlikely that early system scientists employed a biological/anatomical metaphor in formulating complex systems engineering with its powerful approach to managing complexity…

Managing Complexity in Large-scale Complex Systems

Using the reductionist hierarchy of Figure 1, it is possible to 'divide' the complexity of complex organizations, technological devices, socio-economies, etc., into a succession of hierarchy levels, such that an individual addresses only three levels: the one of immediate interest; the one above, to which an individual may report or refer; and the one below, which 'contains' subordinates.

This three-layer concept for managing complexity in both systems management and systems engineering may be elaborated to accommodate large-scale, complex system, Figure 2. The figure might be best viewed as looking down upon a cone, at the peak of which is a central Systems Engineering Management team, Level N.

Immediately surrounding the center are archetypal systems engineering functions: problem-solving; solution conception; concept of operations (CONOPS); purposing; solution space and threat assessment; functional design; functional/physical architecture; specification of requirements; and so on. Each and every SEM team at Levels N-1 and N-2 will perform these same systems engineering functions for their own (organic) subsystems, although at different levels of complexity. Where these SE tasks result in the need for technology engineering and/or acquisition, then the appropriate SEM team will task engineering management/project engineering to design and manufacture to their specification.

In this way, the whole is partitioned into interconnected parts with carefully defined interfaces, such that no individuals or teams find themselves overwhelmed by complexity, each operating within their own 3-layer 'scenario:' the most complex of challenges may be met and managed in this manner. In practice, such organizations are supported by management and design committees, such that meetings at Level N include delegates from Level N-1, meetings at Level N-1 receive delegates from N-2, and so on. Policy decisions from Level N may be passed down through the levels, and problems from the lower levels may be passed up the hierarchy for attention at higher level.

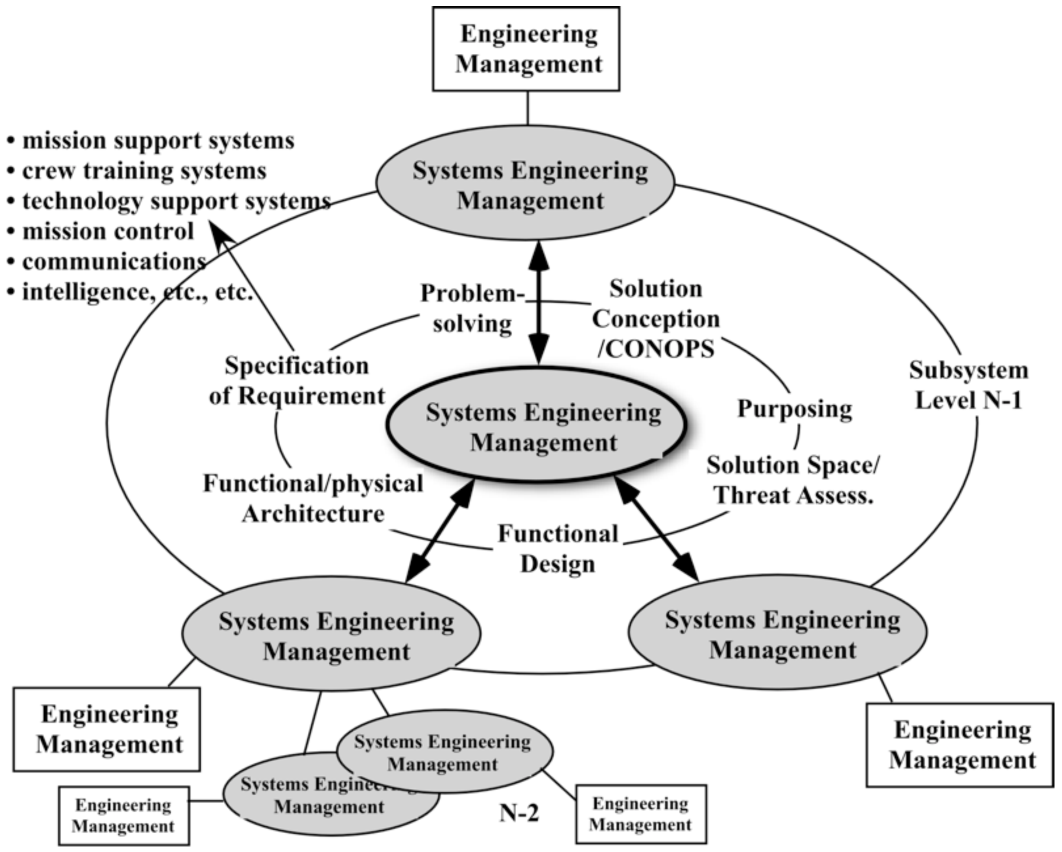

Engineering Management
• mission support systems
• crew training systems
• technology support systems
• mission control
• communications
• intelligence, etc., etc.
Systems Engineering Management
Problem-solving
Solution Conception /CONOPS
Specification of Requirement
Systems Engineering Management
Purposing
Subsystem Level N-1
Solution Space/ Threat Assess.
Functional/physical Architecture
Functional Design
Systems Engineering Management
Systems Engineering Management
Engineering Management
Systems Engineering Management
Engineering Management
Engineering Management
Systems Engineering Management
N-2
Engineering Management

Figure 2. Self-similar Systems Engineering Management Schema

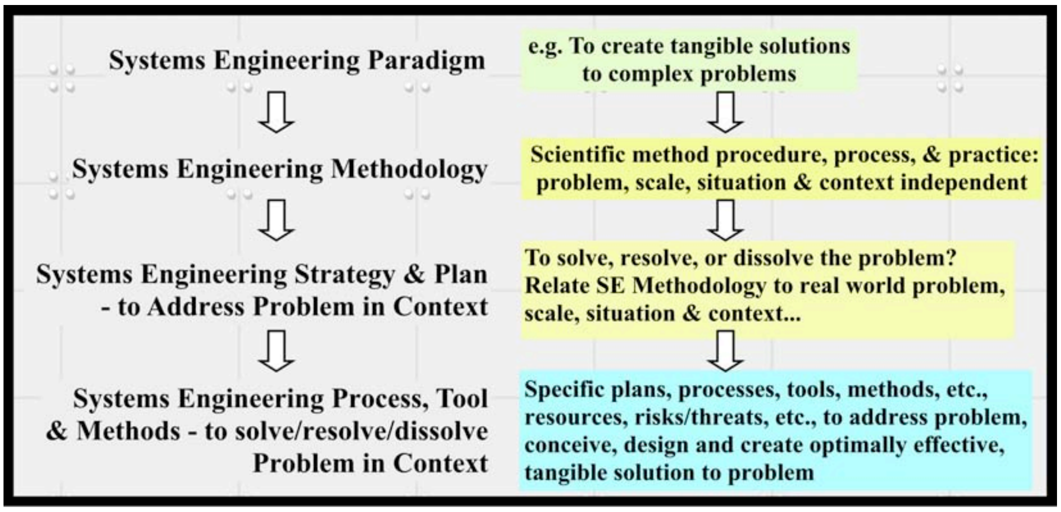
For large scale, complex systems and their programs, then, it appears to be the case that the management of complexity employs the holistic and organismic practices of considering the whole as comprising interacting parts, themselves wholes (holons), and so sensibly evading the detail of what might be 'inside' the parts; and also employs corresponding systems management structures and organization.

Systems Engineering Paradigms

Systems architecting and systems engineering may be concerned, then, with managing complexity, often in large-scale systems and multi-level programs. Managing complexity is neither purpose nor objective, however: systems architecting and systems engineering need purpose in managing complexity. To this end, systems engineering may be described as paradigmatic, and there may be more than one paradigm.

Problem-solving paradigm

Systems engineering in the West has been considered, generally, to follow a problem-solving paradigm. See Figure 3, which elaborates the problem-solving paradigm, first to a systems engineering methodology[1], then to a systems engineering strategy and plans to address a problem in context, and finally to systems engineering process, tools and methods to solve/resolve/dissolve the problem in context.



Systems Engineering Paradigm
e.g. To create tangible solutions to complex problems
Systems Engineering Methodology
Scientific method procedure, process, & practice: problem, scale, situation & context independent
Systems Engineering Strategy & Plan - to Address Problem in Context
To solve, resolve, or dissolve the problem? Relate SE Methodology to real world problem, scale, situation & context...
Systems Engineering Process, Tool & Methods - to solve/resolve/dissolve Problem in Context
Specific plans, processes, tools, methods, etc., resources, risks/threats, etc., to address problem, conceive, design and create optimally effective, tangible solution to problem

Figure 3. Elaborating from the Systems Engineering Problem-solving Paradigm

It follows from the figure, that – while conceivably there may be one archetypal systems engineering methodology – systems engineering strategies and plans may vary according to the problem in context, and in particular upon whether to solve the problem (i.e. find a “correct,” or “best,” answer), resolve the problem (i.e. find a solution that satisfices, is “good enough,”) or dissolve the problem (i.e. change the situation such that the problem no longer arises, or becomes insignificant). (Ackoff & Emery, 1972.)

Japanese Continuous Circular Flow Paradigm

Toyota in Japan has developed a substantially different way of going about systems engineering on the grand, global scale, one that might be said to follow a “continuous circular flow paradigm.” See Figure 4, showing a clockwise “market pull” concept, where nothing is made unless it is already sold, complemented by an anticlockwise flow of money. (Womack, 1990.) A lead company assembles parts from a ‘fan-out’ of supply companies. The lead company supplies to the market. Unlike mass production, however, the flow of materials, parts, subassemblies, etc., occurs only on demand. The aim is to reduce work-in-progress to a minimum, ideally to zero, and at the same time to have a steady flow of products being sold to the market. The market is encouraged to ‘demand’

[1] Methodology: a system of methods used in a particular area of study or activity. *Oxford American Dictionary*

by continual innovation, making each new product so attractive that it becomes a "must-have" to the customer.

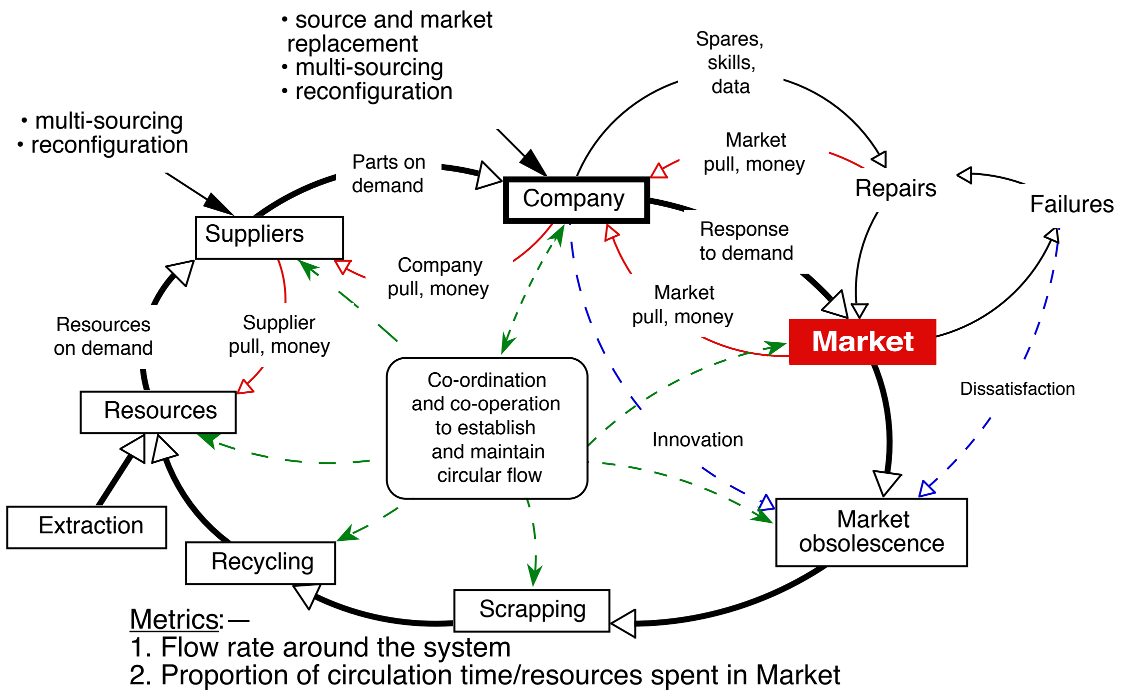


Figure 4. Lean Volume Supply Circular Concept (Hitchins, 2003)

Recycling obsolescent vehicles completes the circle, reminiscent of the Ouroboros, the ancient symbol of the snake or dragon eating its own tail, representing the perpetual cyclic renewal of life, the cycle of life, death and return, leading to immortality. Certainly, the circle is seen as strong, with no point of entry or exit, with no beginning or end, and – since the elements within it are being continually adapted and renewed – the lean volume supply circle has no perceivable lifecycle…

Optimization within this paradigm is through *kaizen*, such that elements in the circle are continually being adjusted and adapted to improve overall 'circular performance,' reminiscent of the earlier progressive optimization of the UK Air Defence System.

Comparisons There is, however, a significant difference between the continuous (virtuous?) circle of Japanese industrial systems engineering and the West's problem solving systems engineering: in the West, the cycle is usually seen as a continual arising and solving of problems, with each solution comprising some new or modified system to be delivered to a customer.

In Japan, the industrial circle *is* the system solution, i.e., the systems architects and systems engineers exist and operate continuously within, and as part of, their solution system, which steadily rotates material – innovative products – around the loop. So, their systems engineering is of the operational variety, in that they are continuously improving, enhancing and rebuilding the system in which they exist, perform and operate – in this respect, they are their own customers.

This practice is consistent with *kaizen*, the Japanese philosophy of continuous improvement, which contrasts markedly with the West's 'Big Bang' philosophy of 'Right First Time.' Comparing the Japanese approach with that of US mass production at the end of the 20th century, a distinct difference could be seen. See Table 1 (Hitchins, 2003). As the table suggests, the philosophical differences between US Mass Production, Henry Ford style, and Toyota's Lean Volume Production, are legion. (Womack et al, 1990)

Table 1. Comparing 20th Century US Mass Production with Japanese Lean Volume Supply

Mass Production	*Comparison*	Lean Volume Supply
Profit	***Objective***	Survival
Free	***Competition***	Between circles
Free Market	***Regulation***	Indiginization
Production Push	***Assembly***	Market Pull
Cost Plus	***Pricing***	Market Minus
Adversarial	***Contract***	Synergistic
Specialist	***Defense***	Homogeneous
Hire and Fire	***Labour***	Jobs for Life
Specialization	***Skills***	Multi-skilled
Lowest Bid Wins	***Suppliers***	Vital source—protect
Supplier stocks	***Inventory***	Nobody stocks

Complexity and Systems Engineering Principles

Systems engineering seeks to solve complex problems by creating one or more system solutions comprised of interacting subsystems: the subsystems and their interactions are designed and configured to cooperate and coordinate their various functions and actions synergistically to create a unified whole, which will perform with optimum effectiveness in its operational environment and context. Systems engineers, in effect, seek to predict a future in which the yet-to-be-created solution system solves the complex, now-and-future problem. To achieve this, systems engineering draws upon *a priori* knowledge of how systems and subsystems behave in context, coupled with simulation and testing of proposed solution systems in their anticipated context and environment.

In seeking a solution to any complex problematic situation, systems engineering may observe three guiding systems principles, which require justification:

- *Holism*: the theory that parts of a whole are in intimate interconnection, such that they cannot exist independently of the whole, or cannot be understood without reference to the whole, which is thus regarded as greater than the sum of its parts.
    - Holism requires that the systems architect/systems engineer focus and practice on the whole system; whole problem, whole solution, in context:
        - Providing less than the whole solution fails to solve the whole problem

        - Addressing only part of the problem permits the unaddressed problem-parts to aggravate the problematic situation, so potentially counteracting any part solution…(Lewin, 1949)
        - Context is vital to anticipate emergence, performance and effectiveness
    - Holism also invokes the subtle notion of concinnity, such that the various subsystems balance function, form, behavior and configuration in comprising the whole.
- *Synthesis*: the combination of parts – subsystems – to make a unified whole. It is the antithesis of reductionism, decomposition and analysis:
    - As Figure 1 showed, each Level of Organization comprises the emergent properties of the level below. It follows logically that to create some whole, the system parts must be brought together *in interaction,* to engender the emergent properties of the interacting system parts. Since this applies at each and every level, synthesis becomes the *sine qua non* for creation of complex solutions…
- *Organicism*: Organicism emphasizes the organization, rather than the composition, of systems:
    - Organicism emphasizes the way in which the system parts are brought together, i.e. their functional/physical architecture
    - Organicism may be seen as an aspect of synthesis

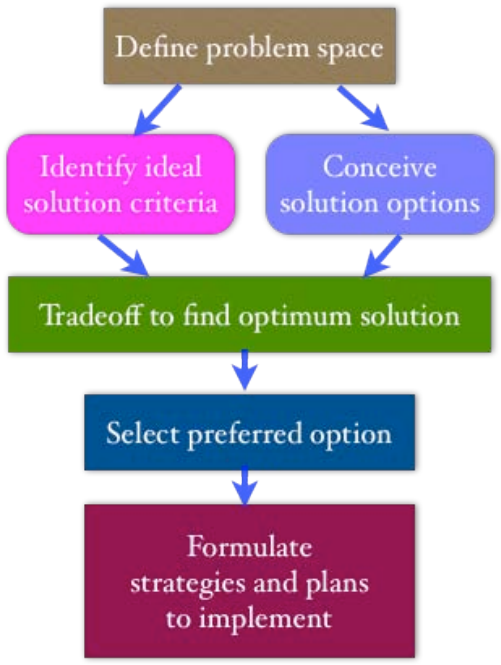
The three guiding principles infuse the proper practice of systems engineering. Holism, for example, pervades systems design *optimization*, which seeks the best operational solution-in-context. Such optimization of the whole may be effected by:

- Altering/rebalancing the emergent properties of the various interacting subsystems (synthesis)
- Amending the interactions between the subsystems – making new connections, changing connections, altering the *degree* of interactions (binding and coupling)
- Re-organizing/reconfiguring subsystems to form different functional/physical architectures with different emergent properties (organicism).

Systems Engineering and Innovation

The systems engineering problem-solving paradigm (SEPSP), Figure 5, has been used to solve problems for over sixty years, not only in systems engineering, but also in industry, commerce, management, economics and politics. The idea is disarmingly simple. Having defined a problem space, conceive a number/variety of optional solutions and – independently – identify criteria by which to judge a good solution. These might include feasibility, performance and effectiveness across a range of foreseeable environments and contexts, availability, survivability, affordability, risk, and so on…The solution options are then traded against the various criteria to find the preferred option (best fit), which would be deemed optimum, or "best in context and circumstances."

By introducing a variety of optional solutions, the door is opened to innovation – for, perhaps, a different way of doing things, or using different technology, or introducing automation, or having a self-healing solution, or going for the cheapest option that might do the job, or choosing the most effective regardless of cost…



Define problem space
Identify ideal solution criteria
Conceive solution options
Tradeoff to find optimum solution
Select preferred option
Formulate strategies and plans to implement

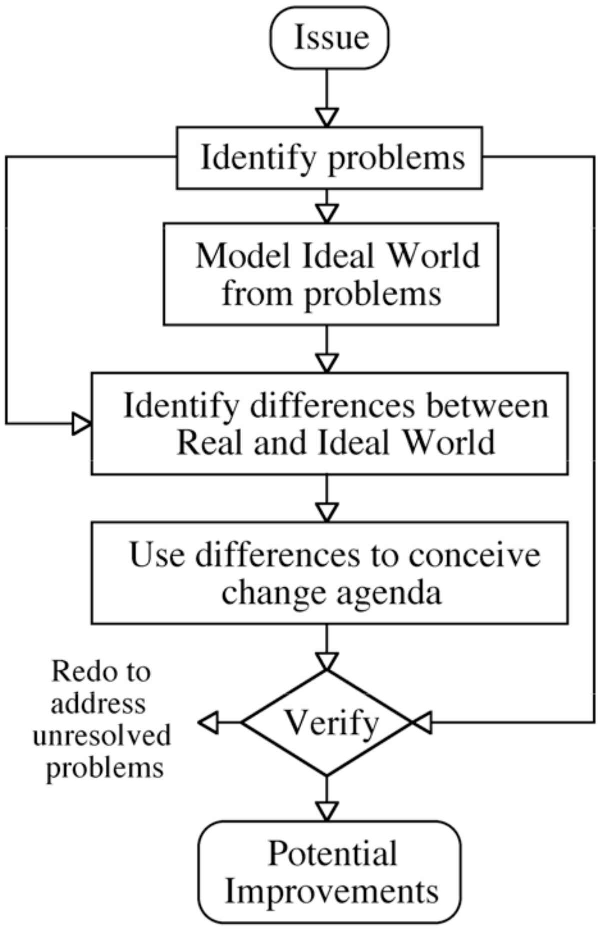
Figure 5. The Systems Engineering Problem-solving Paradigm

The systems engineering problem-solving paradigm can be used extensively throughout a project. Some organizations have employed the SEPSP as an in-company mantra, employing it on a team-by-team basis, sometimes several times a day, whenever a decision had to be made. The results were impressive, perhaps because it necessarily included many of the systems engineers in decision-making, which not only improved decision-making, but also created a sense of inclusion in these so-called Systems Thinking organizations

The General Problem-solving paradigm

While the Systems Engineering Problem-solving Paradigm is valuable in decision-making, it is not particularly helpful when it comes to exploring and solving complex problems. Problem investigation and solving is crucial to complex systems engineering. One approach is to employ the General Problem-solving Paradigm (GPSP) shown diagrammatically as a procedure in Figure 6.

The procedure is straightforward: first, nominate the issue (problematic situation) to be addressed; then, following the procedure, identify problem components, group them into problem themes, and model these themes to create an ideal world, i.e., one in which the problems do not appear; then compare this ideal world with the real world and all its problems, and use the differences as an agenda for change, resulting – hopefully – in improvements to the problematic issue. Towards the end of the procedure, as indicated by the decision diamond, verify that the change agenda, if implemented, would eliminate the original set of symptoms, else repeat the procedure since some problems may have been overlooked…



Issue
Identify problems
Model Ideal World from problems
Identify differences between Real and Ideal World
Use differences to conceive change agenda
Redo to address unresolved problems
Verify
Potential Improvements

Figure 6. General Problem-solving Paradigm (GPSP)
(Hitchins, 2007)

Creativity and innovation in systems conception and design.

One aspect of problem solving not adequately addressed by the GPSP is that of creativity and innovation within the "change agenda" – which essentially comprises solution-systems conception, CONOPS, and systems design.

Innovation does not result from prescriptive process. It is the product of human ingenuity and creativity: this can be promoted by bringing together a group of people of contrasting backgrounds, skills and experience who may interact in a creative, supportive, non-pejorative environment; systems engineering management creates that environment.

The early stages of systems engineering may be characterized by series of brain-storming sessions, some including customers, stakeholders[1], future user-operators, domain experts,

[1] Stakeholders: those who stand to gain, or particularly to *lose*, from the successful completion of a project or enterprise.

experienced and inexperienced systems engineers, etc. It is the mix of bright, open-minded people and the uninhibited environment that generate fresh, new ideas. So, inevitably, no smart people – no innovation: authoritarian control – no innovation: prescriptive process – no innovation.

Pursuing the CONOPS can create a significant variety of strategies and consequent Prime Mission Functions (PMFs) to effect those strategies.[1] Why PMFs? To distinguish such 'externally visible' functions from the 'internal' functions that must take place within the system in the management of the PMFs: see Figure 7. In any mission, the many PMFs may not be active all at once. Some may not be active at all, but exist as safeguards, or precautions. So there arises the need to "orchestrate" the cooperation between, and the coordination of, the many PMFs. In complex systems, too, the PMFs may provide overlapping and inconsistent data, which then has to be merged according to source error characteristics.

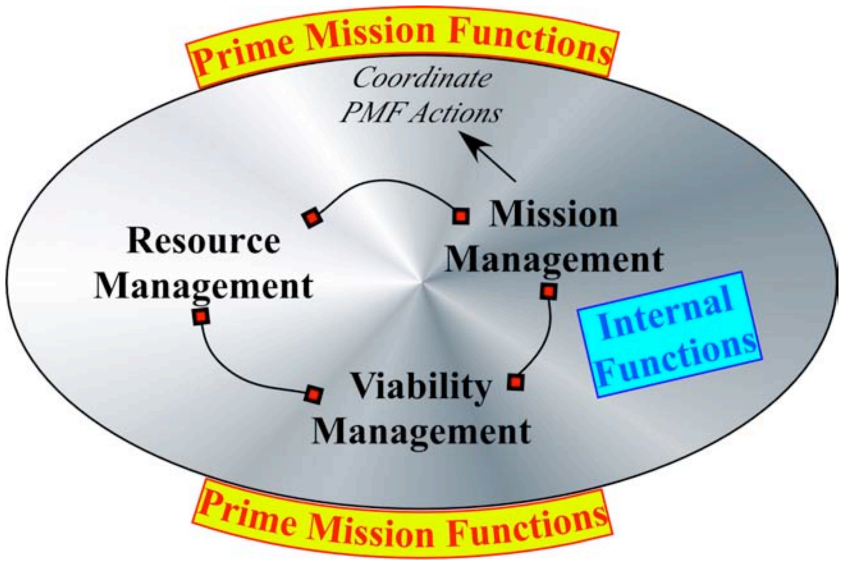


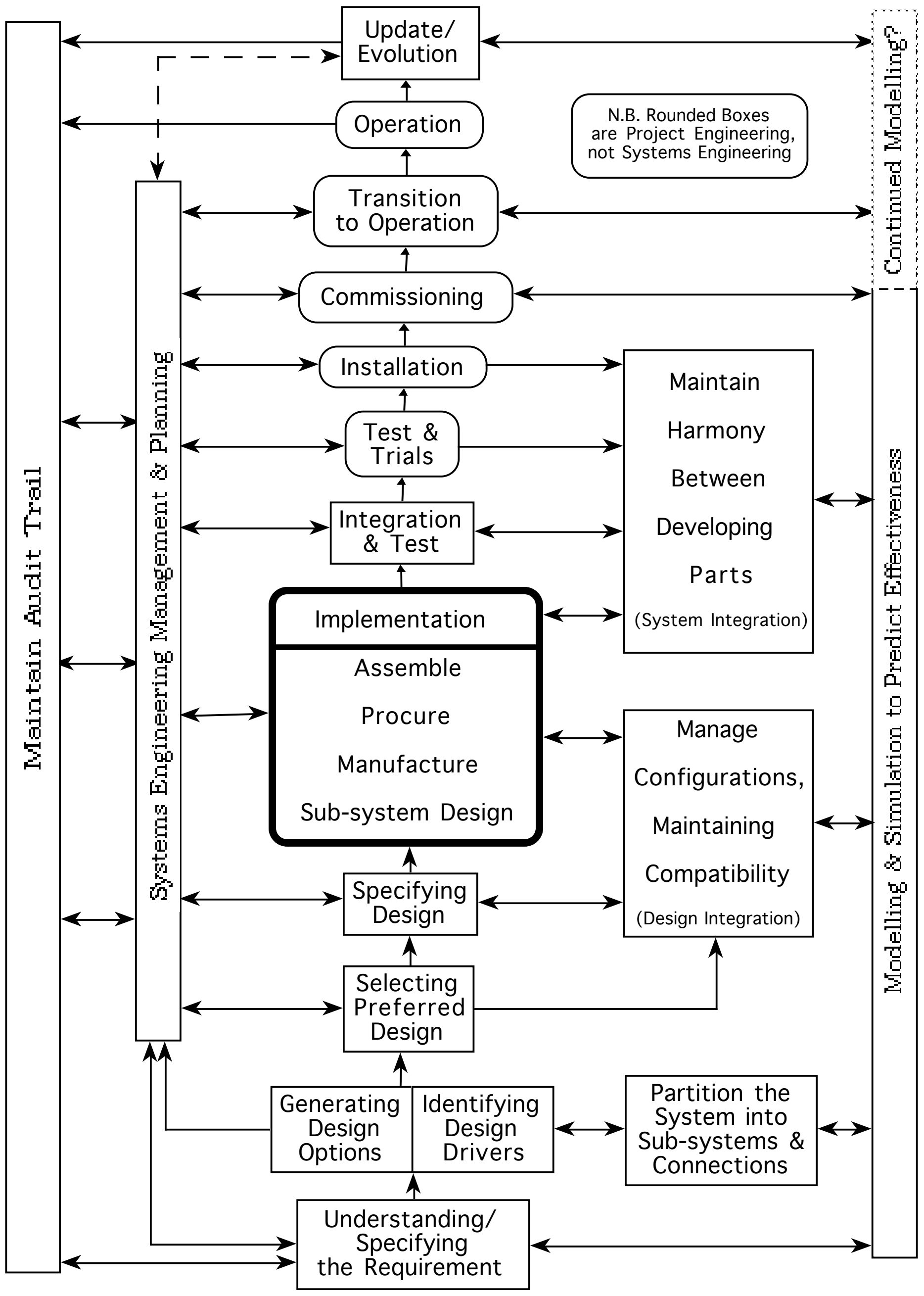
Figure 7. Relationship between Prime Mission Functions and Internal Management Functions.
Prime Mission Functions are those evident to an external observer. Internal functions are those that may be deduced to exist 'within' any open system as it exchanges energy, substance and information with its environment and as it 'orchestrates' PMF activities

It is helpful to examine these Internal Management Functions under three headings, as shown in the figure:

- **Mission management systems**. These are systems that manage, control, co-ordinate and deploy mission-specific functions. A weapon, or a strategy, might be mission specific, so the facilities associated with arming/delivering the weapon or coordinating the strategy would constitute mission management systems.

[1] The conception of PMFs and internal functions is but one of many ways to formulate systems design and architecture. Systems architects may have idiosyncratic approaches.

- **Viability (or platform) management systems**. These are systems that maintain the platform in a capable state, ready for anything, but are not mission-specific. A navigation system or a commercial department would be part of viability management suite, since these would be continually active, regardless of particular mission…
- **Resource management systems**. These are systems that acquire, distribute, supply, convert and dispose of resources. Resources include fuel and energy, consumables, payloads, finance, personnel, etc., etc. according to system type.



Update/ Evolution
Operation
N.B. Rounded Boxes are Project Engineering, not Systems Engineering
Continued Modelling?
Transition to Operation
Commissioning
Installation
Test & Trials
Integration & Test
Maintain Harmony Between Developing Parts (System Integration)
Implementation
Assemble
Procure
Manufacture
Sub-system Design
Manage Configurations, Maintaining Compatibility (Design Integration)
Specifying Design
Selecting Preferred Design
Generating Design Options
Identifying Design Drivers
Partition the System into Sub-systems & Connections
Understanding/ Specifying the Requirement
Maintain Audit Trail
Systems Engineering Management & Planning
Modelling & Simulation to Predict Effectiveness

Figure 8. Archetypal Systems Engineering Plan. (Hitchins, 1992)

Round-cornered boxes are project engineering, not systems engineering

Models of Systems Engineering

Classic Systems Engineering, mid '80s

Models of systems engineering have existed for many years; Figure 8 shows a typical systems engineering plan from 1986. Figure 8 starts at the bottom with the problem (Understanding and Specifying the Requirement) and works to the top with update or evolution – it was *not* assumed that the system will be replaced; instead, it may evolve and adapt in line with the changing situation and need for improved performance and effectiveness.

Returning to the start, the systems engineering problem-solving paradigm is evident in: Generating Design Options, Identifying Design Drivers and Selecting Preferred Design: note, too, Partitioning System into Subsystems and Connections, which effectively formed architecture (organicism).

The plan was consistent with the understanding that systems engineering made and manufactured nothing, while engineering did indeed make, manufacture, install and commission, operate and update... The plan was archetypal in the sense that, for any real project, the plan would be adapted/converted to serve real world situation, domain and context.

Note the continual reliance on simulation modeling to predict effectiveness, throughout; the maintenance of an audit trail, essential to accommodate customer's frequent changes of mind (often later 'conveniently forgotten;') and systems engineering management & planning, which was continually revising in line with problems and progress.

Classic Systems Engineering Organization & Method

Systems Engineering Management followed a standard plan, of which Figure 9 would be typical. The figure indicates the skills that would be required under each of the major headings. As with Figure 8, the headings marked Equipment Engineering and Software Engineering were not systems engineering, *per se,* but engineering, to be effectively *managed* by systems engineering which provided the requirement specifications for, and supervised, engineering work to guard against requirement 'creep,' and changes to emergent properties.

Activity moved from left to right in the figure, starting with:

- *Operations Analysis*. The principal output from Operations Analysis was the User Requirement Specification (URS), showing what the user needed (wanted?) in the way of facilities and capabilities to perform in his rôle.
- *Requirements Analysis*, which used the User Requirement Specification (URS) as a principal input, and which gave, as its principal output, the System Requirement Specification (SRS).
- *Systems Design*, which took as its principal input the SRS, and gave as its principal output a matched set of Performance and Design Requirement Specifications (PDRs)

- These PDRs then served as the inputs to *Equipment Engineering*, and *Software Engineering*. In this classic systems engineering scheme, equipment[1] and software were subcontracted. PDRs would be used as part of competitive tendering to select preferred subcontractors.
- Once equipments and software were engineered – or acquired – the finished products were brought together for *Test and Integration*
- Finally, *Acceptance Trials* followed *Installation and Commissioning…*

Note the absence of project management from Figure 9, which was in effect performed by systems engineering management. There is, however, a section for Project Support, which addressed programme, financial, data, quality and configuration management. As the work was so complex and unpredictable, systems engineering was generally contracted on a cost plus basis.

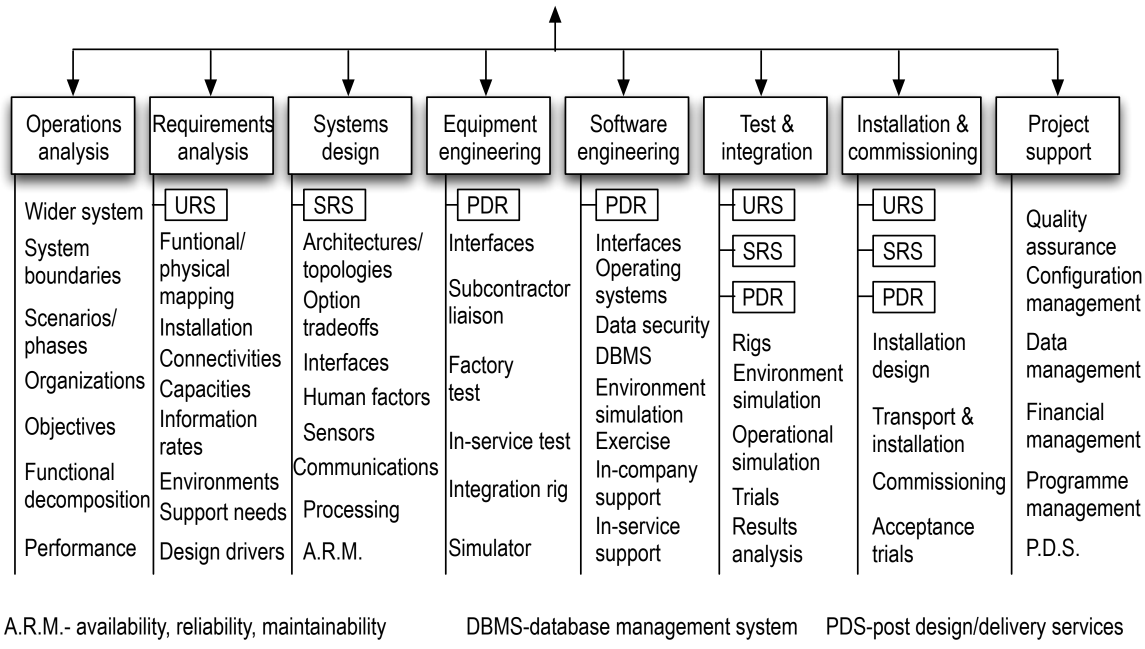


Figure 9. Typical Systems Engineering and Project Engineering Skills circa 1985 (Hitchins, 1986)

Updating Systems Engineering

Since 1986, times and situations have changed. The archetypal plan of Figure 8 would no longer suffice, in particular because situations, issues and problems have become evermore complex. Major steps forward have been made in:

- complex problem-solving,
- dynamic simulation,
- dynamic architecture and its impact on system performance,
- genetic algorithmic systems design methods

[1] Note the contemporary use of 'equipments' to denote technological components, which today are confusingly (erroneously?) called 'systems.'

- networking,
- psychology and human behavior,
- robotics,
- understanding of deterministic chaos and self-organized criticality in systems
- and many, many more.

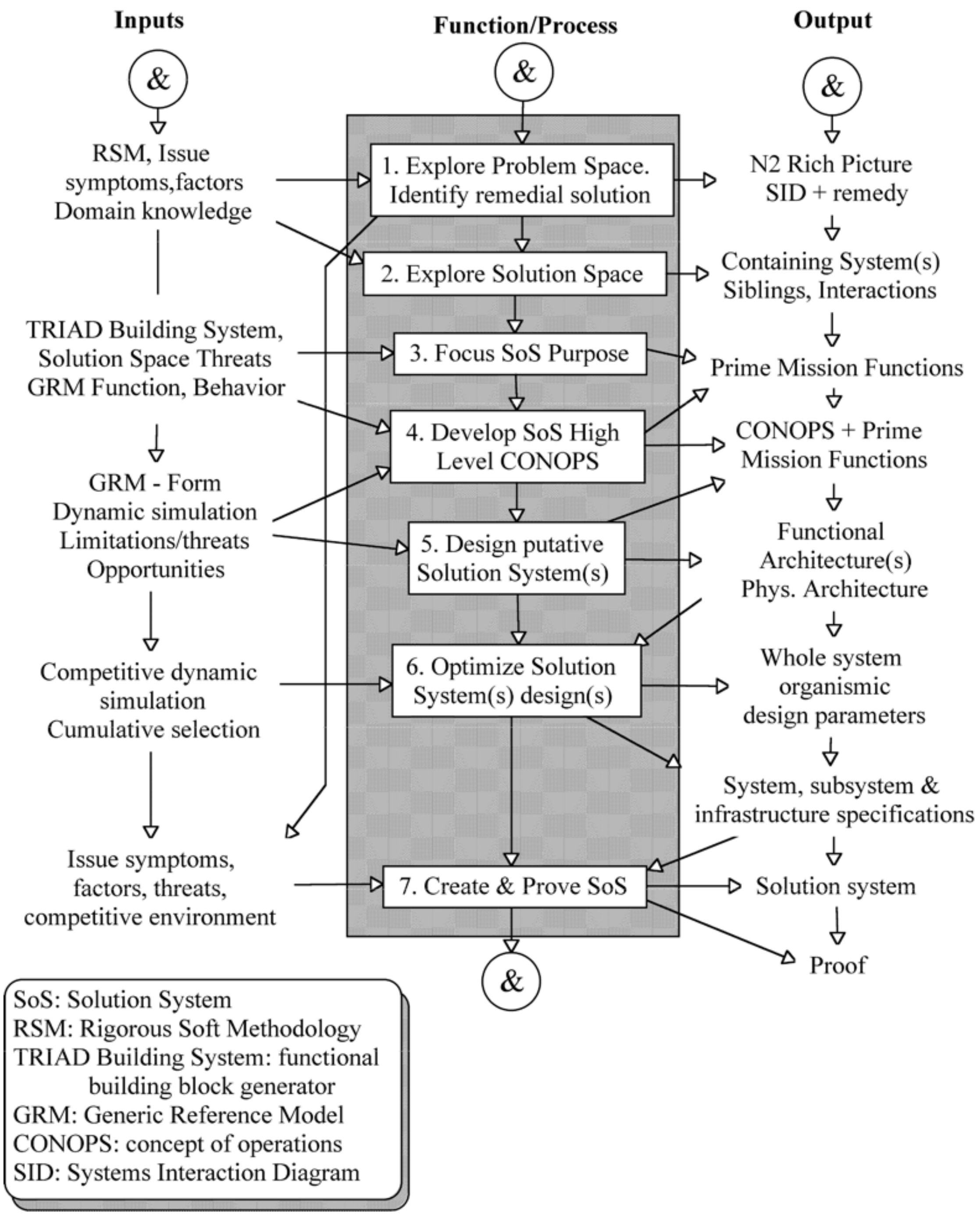
The same fundamental concepts and principles still define systems engineering, however: systems engineering seeks to create an optimum (best) solution to a complex problem, employing the same principles of holism, synthesis and organicism as guiding lights. And the understanding persists, too, that complex systems exhibit emergent properties, where the whole is greater than the sum of its parts, such that emergence may offer significant advantage in performance, capability, effectiveness and affordability.

The challenges presented by complex problems were researched in the 1980s and subsequently. Systems-theoretic 'soft' methods were developed, specifically aimed at "messy organizational problems,' so, dealing with groups of people as subsystems, functioning and interacting within some social system. Problem-solving methods formed around the GPSP of Figure 6, a leading proponent being Checkland's Soft Systems Methodology (SSM)(Checkland, 1981).

The GPSP also inspired a more general problem-solving methodology – the Rigorous Soft Methodology (RSM) (Hitchins, 2007), which is systems-tool supported, but which has yet to prove as popular as SSM. Combining the RSM with other systems tools and methods results in a Systems Methodology shown at high level in the Behavior Diagram of Figure 10. There could be other systems methodologies, according to the particular SE paradigm considered. However, the Systems Methodology of Figure 10 is sufficiently high level to be applicable, for example, to the continuous circular flow paradigm (Toyota style) and to *in vivo* systems engineering, such as that prosecuted by armed forces on land, at sea, and in the air, often in the middle of operations, and sometimes as a result of inflicted enemy damage. (The ill-fated Apollo 13 mission also gave rise to some remarkable *in vivo* systems engineering, which resulted in the safe return of the crew after their in-space catastrophe…)

Even at high level, the Systems Methodology reveals many aspects of systems thinking, architecting and design which were not evident in the SE Plan of Figure 8, including Problem and Solution Spaces, Prime Mission Functions, containing and sibling systems (for environment and context), concept of operations (CONOPS), functional and functional/physical architectures.

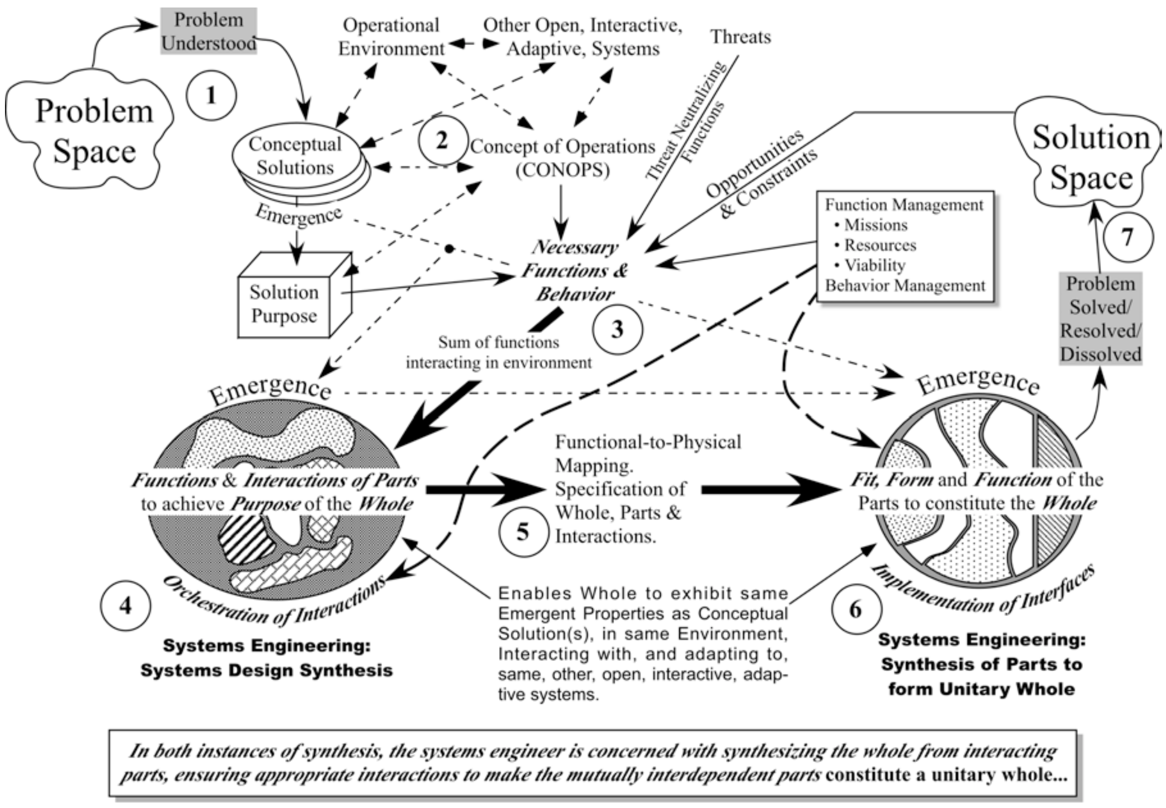
The Behavior Diagram of Figure 10 comprises three columns: Input, Function/Process, and Output columns. The center column shows a logical succession of functions/processes, 1 to 7, from 'Exploring the Problem Space' to 'Creating and Proving the Solution System;' these have been encountered already in Figure 2. The right hand column shows the outputs, or deliverables from each process: since the processes form a logical sequence, so too do their outputs. At left are inputs, including proprietary systems methods, as listed in the inset box. (Systems methods are, by definition, methods that are problem, scale, situation and solution independent, and are therefore not inappropriate in a high-level systems engineering methodology schematic.)



Inputs
Function/Process
Output
&
RSM, Issue symptoms,factors Domain knowledge
TRIAD Building System, Solution Space Threats GRM Function, Behavior
GRM - Form Dynamic simulation Limitations/threats Opportunities
Competitive dynamic simulation Cumulative selection
Issue symptoms, factors, threats, competitive environment
1. Explore Problem Space. Identify remedial solution
2. Explore Solution Space
3. Focus SoS Purpose
4. Develop SoS High Level CONOPS
5. Design putative Solution System(s)
6. Optimize Solution System(s) design(s)
7. Create & Prove SoS
N2 Rich Picture SID + remedy
Containing System(s) Siblings, Interactions
Prime Mission Functions
CONOPS + Prime Mission Functions
Functional Architecture(s) Phys. Architecture
Whole system organismic design parameters
System, subsystem & infrastructure specifications
Solution system
Proof
SoS: Solution System
RSM: Rigorous Soft Methodology
TRIAD Building System: functional building block generator
GRM: Generic Reference Model
CONOPS: concept of operations
SID: Systems Interaction Diagram

Figure 10. Systems Engineering Methodology – Behavior Diagram (Hitchins, 2007)

Comparing the 1986 SE Plan of Figure 8 with the 2007 SE Methodology of Figure 10, and remembering that they are at different levels *vis-à-vis* Figure 3 (Elaborating the SE Paradigm), it is evident that the Systems Methodology concerns itself much more with problem solving, conception and functional design, context and environment, whereas the SE Plan emphasized the management "nuts and bolts" of bringing the various parts together, making them work and proving that they perform according to design and customer expectations. In this, the SE Plan assumed that the solution system was socio-technical or technological, which would have been reasonably expected at the time.



Problem Space
Problem Understood
1
Operational Environment
Other Open, Interactive, Adaptive, Systems
Threats
Conceptual Solutions
2
Concept of Operations (CONOPS)
Threat Neutralizing Functions
Opportunities & Constraints
Solution Space
Emergence
Solution Purpose
Necessary Functions & Behavior
Function Management
• Missions
• Resources
• Viability
Behavior Management
7
Problem Solved/ Resolved/ Dissolved
3
Sum of functions interacting in environment
Emergence
Emergence
Functions & Interactions of Parts to achieve Purpose of the Whole
Functional-to-Physical Mapping. Specification of Whole, Parts & Interactions.
5
Fit, Form and Function of the Parts to constitute the Whole
Orchestration of Interactions
4
Enables Whole to exhibit same Emergent Properties as Conceptual Solution(s), in same Environment, Interacting with, and adapting to, same, other, open, interactive, adaptive systems.
Implementation of Interfaces
6
Systems Engineering: Systems Design Synthesis
Systems Engineering: Synthesis of Parts to form Unitary Whole

In both instances of synthesis, the systems engineer is concerned with synthesizing the whole from interacting parts, ensuring appropriate interactions to make the mutually interdependent parts **constitute a unitary whole...**

Figure 11. Systems Engineering – an Ontology

A Contemporary SE Ontology

Figure 10 presents the overall process of going from Problem Space to Solution Space as a Behavior diagram. An alternative representation of the same journey is presented in Figure 11 in the seven marked steps:

1. Understand the problem and develop conceptual solutions
2. Develop one or more concepts of operations (CONOPS) in context.
3. Gather the necessary functions and behavior from the various sources shown, to create the sum of functions which would be needed, interacting in the future environment, to provide a purposeful, effective solution to the problem
4. Organize functions and orchestrate their interactions to create a dynamic, interactive functional architecture operating and exhibiting emergent properties in (simulated?) context and environment.
5. Map functional architecture on to one or more physical architectures, specifying the whole, the parts and the interactions in solution-transparent form for acquisition or development or both.
6. Integrate the acquired parts in simulated or real context and environment, to create a whole with the same emergent properties in context as those of Step 4.
7. Install, commission, etc., the whole into the solution space as a new system, a replacement system, a reorganized system

At each and every step, including the last one, check that the developing solution system *solves the problem in context.*

Summary & Conclusions

Systems engineering appears founded, not in physics as one might suppose engineering to be, but in biology and anatomy, which – with their Levels of Organization – had previously developed a cogent approach to the accommodation and management of complexity. A biological metaphor is, then, consistent with systems engineering's: approach to complexity management; purpose in systems; and, in its unique emphasis on emergence and emergent properties.

Systems engineering has firm philosophical and systems theoretic foundations. Systems engineering incorporates fundamental principles, which – while not assuring universal success in every outcome – can afford high-integrity solutions to large-scale, complex and dynamic problematic situations and issues, where fathomable by innovative systems thinkers, architects and engineers. Ignoring such foundations prejudices successful outcome.

The first of these founding principles is holism: the theory that parts of a whole are in intimate interconnection, such that they cannot exist independently of the whole, or cannot be understood without reference to the whole, which is thus regarded as greater than the sum of its parts. Holism applies equally to the complex problem, the complex solution, to complex systems engineering-as-a-system, and is the foundation for optimization, i.e. best performance in context.

The second principle, synthesis, is the opposite of reduction. Reduction provides knowledge, of 'how things work.' But reduction views parts in mutual isolation, such that the effects of cooperation, coordination, synergy and context are overlooked; it is these that conspire to create emergent properties, which may be disregarded by engineers employing reductionist methods, but are fundamental to the synthesis of complex systems.

(Traditional systems engineering methods such as functional decomposition, the waterfall method and the Vee-method are reductionist, not holistic. They stem from software methods; it may be reasonable to decompose a software function into separate sub-functions, where – within the software at least – there will be no subsequent interaction between the sub-functions that may affect their respective behavior. Systems decomposition into separate subsystems is not valid, however, since the various subsystems *will* affect each other's respective functioning and behavior, as do the organs within the human body.

The third principle, organicism, emphasizes organization and architecture. Functional architecture is the pattern formed by mutually bound clusters of functions on one hand, and coupling between such clusters on the other (*functional binding* and *coupling*.) Architecture emerges *ideally* from the problem as a configuration to enhance performance, availability, survivability and security, to optimize emergent behavior/performance-in-context.

These three guiding principles are observable in 'natural systems engineering,' which has evolved over the last 520 million years: complex living organisms serve as metaphors for complex manmade social, socio-technical and technological systems. E.g., Nature's organisms are maximally efficient, consistent with survival. (Lotka, 1922).

Paradigmatically, systems engineering can be seen as problem solving, or perhaps as maintaining circular flow. The latter applies not only to industrial systems engineering in Japan, but also to ring-roads around major cities (London's M25, Paris' Périphérique, Washington's Capital Beltway, etc.), to airports (e.g. Heathrow, Stansted and Gatwick around London), to 'ring main' systems for digital communication, electrical and water supplies, and to many more complex transport, logistical and infrastructure systems designs that have found the circular paradigm appealing, robust and practical.

Final Note The philosophy of systems engineering set out above appears at odds with some current practice. Yet, the philosophy is rational, logical, justifiable and demonstrable. Compared with some contemporary practices:

- Systems engineering is founded in systems theory and practice, and so is substantially different from multidisciplinary engineering (i.e. engineering);
- Managing complexity is "built in;" so, too are innovation and creativity;
- Orthodox hierarchy displaces the tautologically challenged 'system of systems'
- There is no mention of project management: systems management appears both necessary and sufficient
- If the SE philosophy is to continually evolve and adapt systems to changing circumstances then there is no perceivable 'end,' therefore no anticipatable system lifecycle (although there may be many technological lifecycles.)
- There has been little mention of cost. SE philosophy finds "solving the problem" to be both necessary and sufficient, and many of the finest achievements of systems engineering have not been noticeably cost-limited. Cost is principally for engineering…
    - Premature consideration of cost in systems engineering can impede innovation, restrict creativity and result in ineffective, short-lived solutions that prove more expensive in the long run…
    - Contemporary emphasis on cost, performance and schedule is consistent with multidisciplinary engineering, and with the engineering of systems, but not with 'systems engineering as complex problem solving.'
    - Systems design may be instantiated using many different approaches, including commercial off-the-shelf (COTS) artifacts, phased implementation, manpower in place of machines, etc; so cost derives mainly from the different ways in which systems designs may be engineered… yet each solution may "solve the problem."
    - However, cost can be of concern where, for instance, there is a view to optimize overall systems design for best performance in context, using e.g., cost-effectiveness ('best value of money') as an optimizing parameter…

Finally, the systems theoretic SE philosophy espoused above – although rational and logical – nonetheless may emerge as something of a surprise. The correspondence between emergence and the management of complexity in biology/anatomy on the one hand, and man-made systems and systems engineering on the other, shown in Figure 1, may appear obvious once expressed, but is neither widely appreciated nor taught.

(D K Hitchins) Monday, June 11, 2012

References

Ackoff R.L. and Emery, F.E., (1972) *On Purposeful Systems*, Aldine-Atherton, Chicago IL

Bertalanffy, Ludwig von, (1950) An Outline of General System Theory, British Journal for the Philosophy of Science 1, p. 139-164

Checkland, P.B., (1981) *Systems Thinking, Systems Practice*, John Wiley, Chichester

Hitchins, D.K., (1986) *Managing Systems Creation,* IEE Proceedings, Vol.133, Part A, No 6

Hitchins, D.K., (1992) Putting Systems to Work, Wiley and Sons, Chichester, UK

Hitchins, D.K., (2003) *Advanced Systems Thinking, Engineering & Management*, Artech House, Norwood MA

Hitchins, D. K., (2007) *Systems Engineering: A 21st Century Systems Methodology,* John Wiley, Chichester

Lewin, K., (1949) Frontiers in Group Dynamics: Concept Method and Reality in Social Science: Social Equilibria and Change. *Human Relations,* **1**, (1), 5-41

Lotka, A.J., (1922) Contribution to the Energetics of Evolution. Proc. Natl. Acad. Sci., **8**, 147-155

Womack, James P., Jones, Daniel T., & Roos. Daniel, (1990) *The Machine that Changed the World*, Rawson Associates, NY