An Application to Systems Engineering of a Framework of General Schemas Theory

The Advance of the Systems Engineering Discipline through an extension of Systems Theory

Kent D. Palmer, Ph.D.

P.O. Box 1632
Orange CA 92856 USA
714-633-9508
palmer@exo.com

Copyright 2004 K.D. Palmer.
All Rights Reserved. Not for distribution.
Started 08/22/03; Version 0.04; 06/29/04; gs02a04.doc

Keywords: Systems Engineering, Systems Theory,

Introduction

This paper is concerned with the application of a new framework developed under the rubric of General Schemas Theory to Systems Engineering. General Schemas theory is an extension of Systems Theory. Systems Theory is the logical academic foundation of Systems Engineering Practice. However, in the process of exploring the usability of this foundation it was discovered that Systems Theory needs to be extended into a General Schemas Theory in order to be more useful as a basis for Systems Engineering, and in fact Systems Engineering needs to be thought of as a Schemas Engineering to fit into this new context.

Systems are just one schema among many that Systems Engineers might use to understand the problems they face and to design solutions to those problems. Other schemas are Form, Pattern, Meta-system[1], Domain and World. All of these schemas and others form a hierarchy of templates of understanding which might be useful for Systems Engineers as they design and build ever more complex configurations of elements, which perhaps are not adequately described just by the system schema alone. In fact, in Systems Theory George Klir among others have taken to producing advanced Systems Theories that combine several schemas into a single approach. But as yet no one has surveyed the field of schemas and suggested a discipline analogous to General Systems Theory that would study the relations between all the different possible schemas. This paper comes out of a research project which has exactly that goal. It suggests a set of canonical schemas found throughout the scientific disciplines in various incarnations, it suggests a way of thinking about the relations between these schemas, and most importantly it considers a framework that encompasses the hierarchy of schemas and augments it in order to establish an advanced conceptual framework in which we might reframe Systems Engineering practice. But this advanced conceptual framework needs some explanation because it suggests new ways of conceptualizing systems and associated configurations of elements that are more or less than systems. In this paper we will present schemas within this framework and attempt to discuss how the framework might be applied to systems engineering practice to improve the state

[1] I now call the meta-system schema an “open-scape”. But the usage will not be corrected in this paper.

of the art. Unfortunately this new framework is fairly sophisticated and takes us into unfamiliar territory for most classically trained Systems Engineers. Considering the philosophical and scientific basis of our methods is not a normal activity within the Systems Engineering discipline. This discipline arose from industry and has only recently began to put on academic airs. Many working systems engineers are suspicious of this academizing of their practical discipline. But on the other hand some Systems Engineers are worried that the foundations of their discipline are not clearly established. When we look into those foundations we discover that in order to clarify them we need to remake the framework within which Systems Engineerng seeks its foundations. Strangely we have to stop talking just about systems, because slowly we realize that if we call everything a system then the term becomes meaningless. We must distinguish other schemas if for no other reason to give the term system its own meaning in relaton to other possible schemas. But when we distinguish the other schemas we realize that we need those too and that we cannot do with just the system schema after all. Rather as our systems become more complex they break out of the bounds of the system schema and introduce us to the vagaries of all different manner of schemas which interact in complex ways. It is this complex interaction of the schemas to the target of our problem solving and design activities that we wish to address with General Schemas Theory.

Introducing the hierarchy of Schemas

George Klir in his book Architecture of Systems Problem Solving combines three different schemas to produce an advanced general systems theory. But by rights if you combine various schemas together we should call this General Schemas Theory on the assumption that all Schemas should be treated equally in our consideration. The point is that there is a whole hierarchy of schemas that goes beyond system, form, and pattern used by George Klir in his advanced theory. Instead we suggest the following ontological emergent hierarchy of schemas:

?? Pluriverse
?? Kosmos
?? World
?? Domain
?? Meta-system [2]
?? System
?? Form
?? Pattern
?? Monad
?? Facet

This series of schemas establish the relation of phenomena to the human scale. They are templates of understanding for phenomena that presents itself. They are a first categorization of all phenomena since everything that appears must appear in one of these schemas. They are templates of understanding because they are the first unconscious attempt to prepare the phenomena to be understood by schematizing it into one of these kinds of spacetime envelopes. Once a phenomena has been assigned a template of understanding it is possible to begin to come to terms with it by attempting to discover its essence, or the categories it belongs to, or its unique individual characteristics, or to assign meaning to it. If we reverse this process as we do when

[2] *Open-scape* which is the combination of Meta-system and Infra-system. See "Towards a Possible Approach to Metasystems as Escapements" at http://holonomic.net

we design something new then the schemas becomes the anchor of everything we design. It is the means of embodiment within an envelope of spacetime of each part of the design. Individuation and Categorization of design elements ultimately serve the schema. Because if there is no schematization there can be no embodiment. So schemas are very important to us as Systems Designers who see our work through to implementation. But they are hidden from us in normal practice, because we already know what schemas everything uses in its embodiments implicitly as tacit knowledge. So much is this the case that we never focus on schemas as such. Our schematization process remains unconscious, to the extent that we do not name the schemas that we use every day. We only talk about kinds of things as if essence was the only constraint on things. However, embodiment exterts other constraints on things than merely kindness, the schema something inhabits is one of those very basic characteristics of all things which is least remarked on but which is completely different from its essence. In philosophy we talk about the difference between essence and existence. Schematization lies between these two. Like the essence of the thing a schema is part of Being. It is a part of its Thisness or Thatness as a spacetime envelope which allows the thing to be referred to. Existence per se is different from reference. Existence has to do with whether the thing is found or not. It is different from the aspects of Being which are identity, realty, presence and truth. These will become important in our story later. At this point we will merely define existence as that which is neither any aspect nor any anti-aspect. What is both we will call the quintessence. Essences are some combination of the various aspects and anti-aspects of Being. The quintessence is the anamorphic and paradoxical combination of all aspects and anti-aspects at once. Essences define kinds of things and are constraints on instantiated attributes of particulars. The Essence applies to what inhabits a spacetime envelope. But the establishing of the spacetime envelope is independent of the identification of the essence, or even the individual unique characteristics of the thing that goes beyond the definition of the essence or further beyond the signification of the thing by projected interpretation. Schemas are the building blocks of the embodiment of everything but we hardly notice them so enamored are we with the essences, the unique characteristics, and the significations of things.

A representation of the Schemas hierarchy

We can relate the schemas which are templates of understanding for things to a mathematical object in order to specify its definition further. When we do that we are producing a representation of the schema. In this case our mathematical object will be the Triangle of Pascal. This triangle is built up by adding the numerical results of one row together pairwise to produce the next row. So the triangle is produced by repetition of addition, and it is always an addition of all pairs in sequence of the last row to get the next row. The series of numbers generated is always a palindrome and it has the value of 2^n when all the numbers are added together in the same row. This sequence generates the minimal solid with n points of n-1 dimension embedded in each dimension. It also records the possible permutations of polynomials.

The key point here is that the Triangle of Pascal is a kind of dual with the schemas in as much as each schema occupies two

dimensions and also there are two schemas per dimension. This fact suddenly makes the idea of the schema very concrete because we can test our hierarchy of schemas against this mathematical structure to see whether or not it has this form or some other form. Schemas are not reducible to dimensions, they are templates of understanding, but they are governed by the dimensional hierarchy with respect to their relations with each other and to the things which are embodied by the various schemas. The question can be asked why there is this double duality of the relation of schemas to dimensions. We speculate that it is so there can be communication of representations at each level of the ontological hierarchy and so that there can be a dimensional transformation by each schema across dimensions. Going down the series of schemas toward dimension zero results in representational information loss. Going up requires what Deleuze calls Repetition which is the opposite of representation. Through repetition each schema arises as a sui generic emergent event. The repetition of information at the lower level never quite adds up to the emergent characteristics of the new level of the hierarchy of schemas. Repetition of addition produces an unexpected whole which is equal to the sum of its parts but which at each level has its own unique structure as seen by the number of sources beyond reversibility and substitution that are produced at each level The double duality of schemas in relation to dimensions allows the efficacious communication of representations downward despite having data loss at each level All this indicates that schemas have a very odd structure that has not been noticed before. Adjacent Schemas on either side of a target schema are conjuncted to build the intervening target schema. There is a pairing of schemas so that the most macro and the most micro are in each case complementaries of each other. So there is a special relation between pluriverse=facet, kosmos=monad, world=pattern, domain=form, meta-system=system. The schemas form a ring that connects unexpectedly the facet with the pluriverse. Schemas have some strange characteristics when taken as a set that are difficult to explain by reduction to mathematical structure. Templates of understanding are different from the dimensional structure of objects. Each schema carries with it peculiar characteristics which can be developed into a formalism of its own. These formalisms are usually developed within disciplines and there are not many that are discipline independent. However, some of these formalisms have been explored in a previous series of papers which take each schema as a subject on its own and discusses its interaction with the whole set of other schemas[3].

A fundamental distinction: Logos/Physus

Once the hierarchy of schemas has been identified and its relation to the dimensional structure of the Pascal Triangle has been elucidated then we can turn to attempting to understand the context within which schemas exist. It is possible to create a formal representation of each schema and apply that to some domain of a discipline. But here it is more important to establish the context of the schemas themselves in order to attempt to understand what they are, themselves, by making clear their differences from other related things. Therefore, we start with a fundamental distinction between Physus and Logos. This disctinction is fundamental within the Metaphysical Era of our Western worldview. That is the era

[3] http://holonomic.info

that we are in at the present time which superceded the Mythopoietic Era about the time of Anaximander. Since then a fundamental distinction between things has been made, i.e. between things that are thoughts and speech on the one hand, i.e. logos, and between natural growing things on the other. Both of these terms in the distinctions are dynamic and involved in genetic unfolding. They get flattened out into distinctions like mind/body, or idea/matter, and other like distinctions within our tradition. We go back to the Greek original distinctions because they are more complex and interesting than later similar distinctions. Things that display physus and logos are all finite as opposed to infinite. Certain things such as ourselves display a combination of both physus and logos. When we engage in science we have a logos about the physus and in our experiments we have a physus specifically related to a logos. So science is when we produce theories about observations of experimental results from what Bacon called the torture of nature, including ourselves. Logos can break free of the physus and build all sorts of castles in the air. Physus can break free of logos and not correspond to meaningful speeches at all. Most of human history was a confrontation with this disharmony between physus and logos. But slowly mankind has learned to focus on the correspondence and coherence between physus and logos and so slowly science has taken root especially in the Western Tradition. Now as Systems Engineers this shows up as the need for lots of documentation and the problem of the relation of the documentation to the things that are being built. We do specific audits at the end of our development cycle to make sure that the physical as built design corresponds to the as designed documentation, and we also make sure that the as built system's functionality matches that which has been specified. This possibility of a split between physus and logos haunts every system we build. The development process is dynamic and the process of expressing both requirements and design in language are dynamic. Keeping these two dynamisms in lock step can be a major challenge which is assigned to systems engineers to maintain coordination and coherence between requirements, design and implementation. So this distinction made in ancient history within our tradition is still very important to us today in systems engineering. It is not just an arbitrary philosophical distinction but one which we confront the reality of every day. All systems are also finite, and need to be built with a finite reserve of resources. But there are infinite possible designs, infinite ways to fit things together, infinite ways to implement and test that implementation. So the finitude and infinitude distinction is also important to us which underlies the logos/physus distinction.

The Ontic Hierarchy

The hierarchy of schemas is emergent, i.e. each one has its own unique characteristics that are non reducible to the others. There is a relation of supervenence between the various ontological emergent hierarchical levels of the schemas. But the ontological hierarchy, so called because it is a projection of Being onto things that otherwise would merely exist, is not the same as a different hierarchy called the ontic hierarchy. The ontic hierarchy is what cannot be reduced ultimately by scientific analysis to other things. An example of such a hierarchy might be gaia?, social, organism, organ, cell, molecule, atom, particle, quark, string? or what ever thresholds of phenomena that you subscribe to and designate as real

While the ontic hierarchy is created by the pressure of reduction, the ontological hierarchy is created by the pressure of skepticism. In other words we can be skeptical whether a particular schema really exists or not and attempt to reduce it to some other schemas. So what stands up ultimately to skepticism is the ontological hierarchy of schemas and what stands up ultimately to reduction is the ontic hierarchy of emergent levels of the organization of phenomena. Science discovers the ontic thresholds by projecting the ontological thresholds. As Systems Engineers we are dependent on the thresholds of phenomena and how they are described by science. We do not go out and invent our own thresholds of ontic phenomena. But we should not get the idea that what we do as Systems Engineers is not science. Science operates precisely the same way as Systems Engineering as a discipline and we should consider Systems Engineering as a kind of design science. We should study what philosophy of science tells us about the way that science truly operates, not the myths about it that have been created over the centuries, but the concepts like those of Popper, Lakatos, and Feyerabend about how it actually operates, and we will find that it operates in a very similar way to Systems Engineering and Software Engineering practice. One of the key things that Popper, pointed out was the importance of refutation. Any theory that is not refutable is actually philosophy. As Peter Naur says Designs are essentially theories. So testability is very important for design theories as most systems engineers know. Kuhn taught us about Paradigm changes, and how our theories can be revolutionarily changed by an alteration of their fundamental assumptions. Paradigm changes have a big effect on designs as we know when we attempt to implement object oriented designs rather than the more traditional functional designs. Systems Engineering has not completely transitioned across the paradigm shift which alters all the various aspects of our designs when we attempt to apply the new object oriented paradigm. Lakatos similar to Popper focused on refutations, but saw scientific theories as conjectures which the scientist working as part of a team attempted to prove. The group would develop a research programme which they would pursue based on their own self-definitions of the cutting edge of their discipline. Similarly Systems Engineers at times attempt to push the envelope of technology working in teams that define for themselves the groundrules of their projects. Feyerabend attempted to extend the work of Laktos and draw negative conclusions about the usefulness of methods. His maxim was that one should use anything that works, and so one cannot dismiss out of hand, even crank approaches to problems as they might lead to something that works, which normative science misses. Philosophy of science has many discoveries about the actual practice of science which were not understood until recently by the scientists themselves who worked unselfconsciously on their problems without considering how they reached their results. Systems Engineering needs to use those results in order to frame its own projects within the scientific and technological domains. Systems Engineering does not just draw on Science and its results, but cannot actually be distinguished from science ultimately. Scientists depend on large engineered instruments. Engineers depend on what scientists discover from experimenting with those instruments. There is no master/slave relations between science and engineering. Engineering is just as important to science as science is to

engineering. The practitioners of these disciplines are colleagues who need to respect the contributions of the other. Systems Engineers need to be concerned with methods, just as Software Engineers have been concerned with methods. Software methods may be a subset of Systems Engineering Methods, but the two sets are not equal Systems Engineers cannot just use software methods without revising them for their different purposes.

In some ways we can think of Systems Engineers as the opposite of Scientists in as much as that they are concerned with synthesis of technological artifacts and not reduction of nature. This emphasis on synthesis is a key aspect of Systems Engineering because their work is to produce emergent effects in systems that are wholes greater than the sum of their parts, i.e. gestalt systems, within contexts that are wholes less than the sum of their parts, i.e. porto-gestalt meta-systems[4]. In this way the Systems Engineer is the latest addition to the tradition of craftsman which is far older than that of scientist. With industrialization this tradition turned craftsmen into engineers. However, it is a peculiarity of the Western tradition that we are able to synthesize many different technological systems together. This synthesis of different crafts and types of technology together into even more comprehensive integrated wholes is what made necessary the position of the systems engineer within aerospace and other industries. In this process the Systems Engineer creates his own ontic hierarchy of emergent wholes with different characteristics composed of designed components. But whatever the character of these ontic components they must adhere to the template of schematic projection. Schematic projection is the underlying foundation because everything that is produced has some sort of spacetime envelop which is dimensional These spacetime envelopes have their own schematic properties. The process of projection is itself temporal so schemas exist in both time and space and are in fact expressions of spacetime intervals of the sort talked about by relativity theory. In this sense schemas are not just static envelopes or templates of understanding, but are indeed processes which envelop everything which is projected into Being. Recently Peter Lynds[5] has discussed the fact that there is no determinate position with respect to time, and it is this nature of the interval in spactime that offers a different way of looking at Zeno's paradox. This is just a way of saying that there is a difference between Pure Being and Process Being and that we can never actually capture anything in determinate and continuous Pure Being but that everything only has a Process Being which is probabilistic and indeterminate within a spacetime interval This must effect the nature of the schema as each schema is a projection which we tend to reify but which is actually probabilistic and indeterminate. All the syntheses projected by craftsmen, engineers and now systems engineers end up being expressed as schemas of one kind or another which is to say spacetime intervals. Prior to Lynd William James called this the specious present, and G.H. Mead talked about the fact that it takes time for anything to become itself. So allschemas are basically spacetime intervals of different dimensionality, but these intervals also have an aspect that makes them templates of understanding for they are the grounds for reference to the thing and the basis for

[4] Open-scape

[5] http://philsci-archive.pitt.edu/archive/00001197/

the determination of its kind of essence and eventually its idiosyncratic characteristics and our interpretation of it which is expressed as a gloss in language. The templates of undersanding might be seen as the protosynthesis on which the synthesis of the artifact is based which makes it an emergent whole. Thus it behoves us as Systems Engineers to study these templates of understanding or proto-syntheses in order to make our work of synthesis better grounded in the projections that are the lifeblood of our projection of all the things in our world.

The Nonduals between the Duals

We have now understood that Logos is related to the Ontological hierarchy of Schemas and Physus is related to the Ontic Hierarchy of non-reducible things. Systems Engineers attempt to make emergent wholes that are gestalts, i.e. wholes greater than the sum of their parts, with emergent properties through the interaction of the natural wholes and the schemas. Artificial emergent wholes produced by Systems Engineers are virtual in the sense that they are possibilities that are not realized in nature but which are realized beyond nature in the artificial realm we synthesize using what we have learned from science and by a kind of tinkering that is the hallmark of all engineering practice. But if this tinkering is to be guided then we must as in science recognize the non-dual realm of Order between Logos and Physus. As Einstein said the miracle which is so mysterious is that mathematics can connect theory to the observations of experiments on nature. Math is the secret bridge between theory and practice in both science and engineering. When we speak of math we mean all the mathematical categories. The most basic of these is the Set. However, our mathematical foundation for our scientific and engineering work is lopsided and flawed in a way that is little expected because the complement to the Set category is not represented in our mathematics. We discover from studying other cultures such as that in India and China that they had a different basis for their math and logic which was the Mass. A mass is the complementary opposite of the set. The mass is a large body of identical instances that together produces macro effects through their micro interactions. A set on the other hand is a series of different elements called particulars each of which is a different bundle of properties so each is unique in the set. A set cannot have more than one of the same kind of thing. Sets operate on the differences between kinds. Thus the whole emphasis of the set is on the essences of the different things that make up the set. If you want to have repetitions of elements of the same kind you must have a bag and if it is ordered then that is a list. But the set emphasizes difference of its elements and it the natural complement of the mass which emphasizes identity of its elements. But there is no mathematics of masses. Masses are forgotten in our tradition, even though in our language we have ways of talking about them, for instance when we talk about a blade of grass in a yard. Grass is a mass and the blade is a counter of the instance of a leaf of grass that makes up the mass of the grass in the yard. The yard signifies the boundary of the mass of grass. The key point about a mass is that it has its emergent properties at the level equivalent to the set while the set has no emergent properties, rather particulars have emergent properties and instances in a mass lack them. This shift as to where the emergent properties lies is very important. With respect to Science the mass oriented science is thermodynamics. Particle Physics is Set oriented. Thermodynamics

up until recently when it was discovered that negative entropy was possible in far from equilibrium thermodynamic systems has always been a backwater of science proper. But it is important to see how the mass like properties were segregated even in Physics from the mass like properties of thermodynamics.

If we think of a kind of Mathematics that balances the set-like and mass-like approaches then we can see how that would be applied to Systems Engineering. This is because we always design in a set-like manner, but when the system executes or operates then we are suddenly transferred into a mass like behavior as the various parts of the system are instantiated and begin interacting. Because as Systems Engineers we deal with emergent effects at the macro scale which we try to get to happen from designed micro components we need a language to talk about this transition from design to execution or operation. Many unexpected things happen in the realm of execution and operation and the mass like properties exhibited there are not always what we intended or planned. For instance, we design a car, but when it goes out on the street it enters the mass of traffic. We need to see what the emergent attributes of traffic are and use that as a means to improve the design of cars which we then look at in a set like manner. The point is that while sets have a syllygistic logic related to universals masses have a pervasion logic related to boundaries. We can reason about both sets and masses, but we have to use their natural logics, we cannot use set logic to think about masses and vice versa. In systems engineering we are continually dealing with sets and masses and their combinations. For instance, a combination of masses is a solution. Those are interpenetrated masses. Solutions may have different properties than the masses taken each on their own. Masses are unordered and follow the dictates of thermodynamics for the most part. But how local interactions between instances in a mass produce the emergent properties of the mass as a whole can be very different in various cases. For instance space of geometry is a mass of dimensionless points. Ideally we project coordinate grids on these masses. But from physics we know there is the guage phenomena which does not allow the external projection of coordinates. In space time there are geodesics, i.e. internal coordinates to the world line of the particle moving though space. That is how the particle can appear to be in flat spaces along its route but be actually moving through globally curved space. The gage phenomena is general, in masses there is no external point of view from which to project a coordinate set. Another point is the Bekenstein Entropy Bound and the holographic principle which states that the entropy of something is one quarter of its surface area. That means that whatever is going on in a space can be written on the bound of the space. This is a very profound principle that has many implications for schemas theory, due to the fact that each lower level schema is a surface for the next higher level schema considered in its dimensional framework from the relation to Pascal's triangle. There is information loss as we go toward zero dimension representing higher dimensions. It turns out that the schema differences appear as two dimensional jumps, and that is precisely a quartering of information. One quarter of information that is lost is entropic, i.e. a disordering of information that hides the emergent properties of the next higher schema. Given that within a meta-system there is both system and anti-system, then for each system there is a quarter of the information

that is preserved, and the other quarter of the information disorders, not merely lost and vanished, but destroyed. So Bekenstein's bound specifies precisely the emergent thresholds between the schemas. And that has to do with the loss of information to entropy as we move down to simpler and simpler representations. The opposite of this movement is up ward toward the n-dimensional in which case we have repetition operating which repeats the information that is left, but copying it is not enough to gain back the higher dimensional level Rather a negentropy from a singularity must reorder and reorganize the higher level schema. In systems engineering we are fighting entropy continually. We are continually representing our systems we wish to produce in requirements, design or test documents. We know that the myriad of these documents do not capture completely the emergent whole we are attempting to bring to manifestation within the world. It is human effort that bridges to gap to produce the allopoietic artificial systems we attempt to produce. If it were not our imaginations and our theories that we informed these systems through they would never manifest emergent properties we intend. We are the singularities that move against entropy to produce emergence within our technical artifacts. Heidegger in Being and Time called that kind of singularity which we are *Dasein*.

The Meta Level of Logos and Physus

Both Logos and Physus have meta-levels at which they interpenetrate each other. There is a physus in the logos and a logos in the physus. The physus in the logos is Logic, in other words logic gives the hard core of language and speech and thought which is given as the logos arises from the physus. On the other hand there is the logos in the physus which is the schema. That is because we project the schemas as a fundamental partitioning of things within our experience. This partitioning separates things from each other the way letters, phonemes, words, sentences, paragraphs, chapters, books, series of books, libraries all separate parts of language, speech and thought from each other. We lay down the schemas over the physical things as they express their ontic natural complexes within spacetime. So Logic and Schemas are at the same level within our interpretive experience one applying to language as its essential core order, and the other applying to things as we experience them as their essential core order prior to kinds. Logic is also prior to kinds. It does not care what you are trying to say, but only how you relate the various propositions to each other. Our point is that logic can be syllogistic or pervasion logic. And perhaps there are other logics which draw our different fundamental categories other than set and mass into prominence. Logic is based fundamentally on three operators, and, or, and not, but there are many different kinds of logic. Some of those we find significant are the Para-consistent[6] and Para-complete logics described by G. Priest. Along this vein are the Diamond Logic of Hellerstein developed from G. Spencer Brown's Laws of Form and the Matrics Logic of August Stern. As Systems Engineers we have not begun in earnest to attempt to use formal methods as some of our Software colleagues have done. But in terms of our movement toward methods and even formal methods we need to keep our minds open to the important of exotic logics, i.e. we need to be open to moving from the restricted economy of traditional first order propositional logic to the general economy of many different kinds of logics.

[6] http://plato.stanford.edu/entries/logic-paraconsistent/

Another way we need to keep our minds open is in terms of allowing for more aspects of Being than those recognized by standard logical formalisms. Being has four aspects: presence, truth, reality, and identity. Standard formal systems deal with presence, truth and identity but not reality. Thus there are only three standard properties for a standard formal system which are consistency, clarity (wellformedness) and completeness. However, if we add the aspect of Reality suddenly there are three other properties that are important which are verification, validation and coherence which makes integration possible. Notice that systems engineering differentiates verification and validation (did you build the right thing, and did you build it right). Also notice that as systems engineers we are concerned with coherence, of interfaces particularly, but of the system as a whole because it is coherence that allows the emergent properties to emerge. So as Systems Engineers we cannot just use the standard formal logic but must augment it with the scent of reality. It is interesting that the upshot of model theory is that the addition of reality is what generates semantics or meaning. It is only when the emergent properties of the whole system arise that the system has meaning. Without that arising of emergent properties then the system is just so many pieces laying around uselessly on the ground, as when we take apart a car and it cannot run any longer. The meaning of the car is in its travel down the highway or roadways. When the emergent properties fall away so does the meaning of the thing within our world. Logic by itself is not enough. That is why I have proposed that we need to apply logic not just to truth but to all the aspects of being, i.e. presence, identity, and reality as well This is called Vajra Logic[7]. It is a logic which uses the form of the Toposes (the mathematical form of logics) with its binary characterization of statements to describe all the aspects of Being. If we add to that the capacity to deal with paradox of Diamond Logic, and the ability to express both para-completeness and para-consistency as does Matrix Logic then we have a very strong logic to deal with the contradictions that occur in the process of development of complex systems. In systems individual components need to express sometimes divergent and even contradictory properties for the entire system to achieve synthesis. In the process of building the systems we also run into conundrums and enigmas that need to be comprehended. Normal logic does not fulfill these needs. So we have to be open as Systems Engineers to appreciating and exploiting the characteristics of more complex and exotic logics of both the syllogistic and pervasion types.

The logics express the different types of grammars or rules that can control speech or thought. The grammar of language is different from the grammar of thought or meaning arising in speech. One deals with syntax and the other with sematics. When we turn to schemas we see a similar thing in as much as the schemas are dimensional articulations of the envelopes of spacetime but also templates of understanding for things, i.e. the ontic physus. Both of these approaches to things are neglected in our tradition and that hinders their use by Systems Engineers to guide their thought about design. Systems Engineers must connect the software design to the hardware design, i.e. the dynamical information component to the matter

[7] Vajra Logic and Mathematical Meta-models for Meta-systems Engineering INCOSE 2002

energy component. This connection between the two types of components is our way of projecting the physus/logos distinction into what we build. Software algorithms have a certain structure over and above the structure of logic which are represented by various software patterns and software language constructs. Hardware has its own structure which is electrical, mechanical, etc. Producing systems where all these different kinds of objects can interface properly to allow their emergent properties to appear is very difficult. We do this by applying the structures of logic and software languages and patterns. But also we must allow for the embodiment of the parts of the system as schematic envelopes in spacetime. Those envelopes were very static in the past, but as we become more sophisticated and allow for retooling and self-repair of systems these envelopes become more and more flexible. When we understand the intertwining of these envelopes though the schemas then we bring to bear some very robust resources because each schema has its own sets of formalisms like the kinds of logics which allow us to think about the design problems in new ways as we use the different schematic representations at the same time to get a handle on understanding what is happening within the spacetime interval that is being designed.

Logos and Schemas are at the same meta-level Both deal with the syntax of semantics, not syntax by itself. Both bring meaning to bear to organize thoughts or things of experience. But both still partition language and things. However logic is different from grammar, i.e. pure syntax. This is the same with the difference between the schema and the ontic. The ontic is pure syntax, but the schemas are syntax of semantics, both a breaking up but also a bringing to bear of meaning. This is a key point. There is a sense in which there is a breakup of language and things that is just pure syntax, or division. But there is another sense which we are exploring here in which the meaning must be fused with the syntax and that occurs in logic and it occurs in the schemas. Each schema is a kind of logic of things in as much as it can be expressed as a formalism that governs our understanding of a certain class of dimensional envelopes. The formalism expresses the characteristics of that class of things in a general way prior to the discrimination of kinds of things within the class. In that way it serves as a bridge between the various kinds of things in that class. The relation between the schema and logic is though reference. When we say this or that we are pointing to a particular envelope in spacetime, a partitioning of spacetime within which something exists. We project on it the proto-synthesis of the schema first and then attempt to discover its essence. So we can use logical names to refer to these envelopes. The envelopes fit together like Russian dolls each with its own proto-semantics and its own formalism expressing its properties. Also schemas refer to logics because each formalism for a schema can have its model theory when we treat that schema as if it were a mathematical category. So logics at some level control the templates of understanding from within whereas logics refer to schemas as means of referring to things in spacetime. Those spacetime envelopes can be thought of as a mass or as a particular which is part of a set. Sets are arbitrary, but masses are non-arbitrary because masses have emergent properties from the mass action of all their instances like the waves on the sea. On the other hand sets have no emergent properties and are just a pure collection of the different kinds they encapsulate. So mass approach

through pervasion logic is a more natural way to think of spacetime envelopes than the set approach though syllogistic logic that needs to project universals instead of boundaries and see different essences rather than similarities. So we really need the power of the mass approach to things in order to realize the full power of schemas where we identify a dimensional spacetime envelope and see it as pervaded by the proto-synthesis of the schema prior to its pervasion by its kind (called a Form by Plato because he conflated the schema form with its essence). The patterning of the schema itself is the proto-essence of every schematic partition. On that proto-essence the kindness of the actual essence of the thing is built for each spacetime partition. These spacetime partitions may be collected into an empty set, thus filling it with different kinds of things. Although sets without particulars are empty, a mass is never really empty because it must have its instances to exhibit emergent properties. But we can project a de-emergent null mass without instances and consider infinite null masses to be what Democritus and the Taoists meant by the Void, just as an infinite extent of empty sets[8] can be seen to be what the Buddhists called Emptiness. This compensatory pattern of the difference between sets and masses can also be seen in the isomorphism between the two logics, syllogism and pervasion. Peirce made the point that the three statements of the syllogism can be arranged in three different ways to give induction, deduction, and abduction (hypothesis forming from cases). Similarly there is structure that is similar for pervasion logic that reasons about boundaries rather than universals. The key question for pervasion logics is whether we are inside or outside a boundary and thus within the mass or outside the mass. If we are inside the mass we are pervaded by its properties. It has been noted that Plato's idea of forms was probably mass like originally. That is to say that Beauty is a mass and all beautiful things are seen as instances of beauty. If it is within the bound of the beautiful then it is pervaded by beauty and has the emergent properties of the beautiful which have to do with harmony and proportion. This approach obviates the need for a transcendental realm for the *source forms* of Plato to inhabit. The Mass of the Beautiful are just all the beautiful things. That mass has its own sui generis properties that are beyond those of the beautiful things themselves. For instance, all the beauties of nature have a profound effect on the soul It was Aristotle that broke with his predecessors and established the set like bias of our tradition[9].

It is a similar story with adding the reality aspect to presence, identity and truth. We need that to understand schemas because schemas are about the minimal representation of things prior to determining their kindness. Because they are about things there is some measure of reality involved in the identification of the schemas that goes beyond pure logical formalism. It is real objects of experience that are schematized before we know what they are we know that they are as spacetime envelopes. Those envelopes have their own structure that is different for each envelope type. That structure is intelligible even without knowing the kinds of the things that are taking that spacetime configuration. That intelligibility is a sort of infra-structure that all things of that type share. If we

[8] See http://emptysets.com Kajetan Guz

[9] The Discovery of Things: Aristotle's Categories and Their Context by Wolfgang-Rainer Mann (Princeton UP)

understand those infrastructures and how we fit together we have a better chance of designing artificial sets of spacetime envelopes that fit together well. Reality brings with it the ability to verify, to validate, and to discover coherence that allows system integration. Formal systems by themselves with just presence, identity and truth do not allow for these properties and thus remain disconnected from reality. Thus it is good news for us that Being encompasses not just presence, identity and truth but also reality. Systems Engineers need to deal with reality every day, to reason about reality in relation to the other aspects of Being. It is that realism of the systems engineer that brings him to write "The Unwritten Laws of Systems Engineering.[10]"

So it is clear that it is schemas that bring with them the need to expand the aspects of Being considered by Logic as well as the kinds of logic that are acceptable. We especially need a logic like that of Heilerstein which attacks paradoxes and allows us to frame anamorphs[11] that solve paradoxes. But that logic must be a Vajra Logic which applies to all the aspects of Being equally rather than just truth. Statements are not just true and untrue but they can be real or illusory as well. The systems Engineer must deal with all the aspects of Being equally. He deals with what is present and absent, what is identical and different, what is real and unreal, and what is true and untrue every day attempting to be just and practical at the same time between the competing claims of these aspects of Being.

How Logic relates to Schemas

There are three different terms at the meta-level above logos and physus which are Logic, Schemas and Mathesis. Logic relates to Schemas in terms of the Philosophical Categories, i.e. the highest concepts that connect pure ideas to things. These are concepts like quality/quantity, causality, part/whole, etc that were identified by Arsitotle as the most general statements that can be made about any substance or kind of thing. They were also identified by Kant in his table of categories. Kant goes on to identify the schemas as being related to each dialectical set of categories in his table. For our part we like better the categorical scheme developed by Ingvar Johansson[12] which is built on the work of Husserl. There are myriad categorical schemes available from the history of our philosophical tradition. Schemas are things that assume causal relations in time, that combine quality and quantity, that have part/whole relations, etc. We use our categories to think about things in their most basic forms. It is as if the philosophical categories were the infrastructure needed to create the formalisms that describe each schema. The same categories show up in each formalism in different ways suitable to that schema. How the philosophical categories operate over the schemas and are manipulated by logic would be a study in itself. Here it is only necessary to mention that there is this high level connection between the schemas and logic via the philosophical categories. This connection is what we use to create the naïve view of the world. Durkheim said that the Kantian Categories are social, so if we consider them as socially constructed rather than universals of the mind, then we can consider the cultural determinateness of the categories and thus logics and the

[10] David F. McClinton INCOSE 1994

[11] See Donald Kunze Boundary Logic at http://art3idea.ce.psu.edu/boundaries/mainpage/directory.html

[12] See http://hem.passagen.se/ijohansson/

schemas. It is interesting in that light that in our culture schemas are not well described and the mass like way of boking at things not well developed and reality left out of account by our idealist tradition. It is pragmatism of CS Peirce that comes closest to giving a grounding to Schemas theory in his category system. In that system there are only three categories which are called First, Second and Third. The first is the isolated thing that shows up[13] without relation to anything else. Second are relations. Thirds are continuities. To this we add from B. Fuller Fourths which are Synergies. And we add Zeroths which is the background out of which the firsts appear. What we first notice is that the differences between these categories are the kinds of Being. But beyond that we notice that each schema is in fact an articulation of all five of these categories. Every schema takes the bwer level schemas as firsts. It relates those bwer levelschemas to each other and then produces a continuity which is the emergent characteristics of the schema above the discontinuities of the bwer level schema. But also each schema has its inner coherence which is a synergy at its own level The discontinuities between the various schemas are expressions of the zeroth category. If we say that each schema is an expression of the Peircian categories then it must also be an expression of all the kinds of Being and thus what we call a face of the world which is a synthesis of the various fragments of Being.

How Schemas relate to Mathesis

Schemas relate also to Mathesis, which is the faculty for the production of order (nomos). This relation is through representations. Our representation that uses Pascal's triangle as a way to understand the nesting of the different schemas is a case in point. Representations simplify. So we get representations as we move toward zero dimension down the scale of schemas. Moving up as we said called by Deleuze Repetition. We are used to representation but not repetition. Repetition produces more and more complex items. We are geared instead toward reduction and simplification. However, as we are ecstatically projecting Being there is a dimensional overflowing, and a natural repetition that we are engaged in that we suppress in favor of an emphasis on representation alone. As templates of understanding the schemas serve as sites for both representation and repetition. We can represent the class of envelopes of spacetime entities via the formalism associated with a schema. But every application to a new kind of thing or an instance of that kind is a repetition that reasserts the infrastructure or proto-synthesis that the schema represents, as the foundations of understanding prior to the determination of kindness, or individual peculiarities, or interpretations. When you bok at the world it is full of both representations and repetitions, but we only bok at the representations and suppress the repetitions, like we repress reality and mass approaches to things. Schema Theory breaks that impasse and appeals to all three suppressed elements to ground our understanding of Schemas Theory. We need repetition because otherwise we cannot travel both up and down Pascal's triangle. We have already described why we need reality and mass approaches to things. As systems engineers we are stuck with many representations. And we seem to repeat those representations endlessly. Now we are even asked by CMMI to plan our planning

[13] Like "hyle" (content, matter) in Husserl

and monitor our monitoring and configuration manage our configuation managagment. In other words we are called upon to make meta-level representations and to repeat those. All those repetitions of documents cannot capture the system that is being built in full[14]. That is because there is a dimensional difference between the system and the documentation. The documentation must deal with an essential information loss due to de-emergence. It is operating in an arena in which entropy must be confronted as the enemy of the implementation at every turn. What is strange is that although there is no amount of repetition of representations that will capture the as-built system, it is also true that the system as a singularity arises out of the field of those repeated representations under the right conditions. The whole question becomes how to harness the negative entropy of the humans doing the development to bring about the necessary order that will have to be unfolded for the emergent properties to appear as intended. These humans wander around in the trash heap of the repeated representations and somehow bring the system together in spite of the entropy they are fighting against. This would not be possible if there were not some ultra-efficacies at work. One of those ultra-efficacies are the schemas themselves. They allow the communication between representations at various levels. They allow the intertransformation of information between representations at the same level When you look at it deeply you see that the schemas are the backbone on which the flesh of every system is hung. It is the ultra-efficacy of our joint projection of the schemas that allow us to work together on the same system. In other words the schemas are an intersubjective social invention and construction or projection that we share in common which inhabits our mutual, conversational, team memory. And I would like to venture that this projection is housed in a communal intermediate memory which stands between long term memory and short term memory and is purely social We store our conversation trees[15] in this collective memory and it is those trees that are structured by the schemas, because that is what allows us to mutually refer to the same spacetime envelope as this or that and know what we are both talking about. The schemas are not just individual projections but group projections, they are in fact what allows our conversation trees to interface with the world which is portioned spacetime into things. The conversation occurs as we wander though the world. As we wander we point to this or that and indicate a schema even though we do now yet know what it is that we are pointing at, because it has not been completely designed yet. The theory of the design is the gloss on the conversation tree that has given rise to the mutually held theory. We cannot capture the theory because it exists in a communal memory which we have imperfect access to if we are not in conversation with the others with whom we are doing the design work.

How Logic relates to Mathesis

Logic relates to mathesis via model theory. Model theory has to do with the statements one may make about a mathematical category. We need to extend model theory which posits that the realm of sematics has the same structure as the realm of syntax into a meta-model theory by adding the

[14] Naur, Peter, "Programming as theory building," Microprocessing and Microprogramming, 15: 253-261, 1985, reprinted in Computing: A Human Activity, (NY: Addison-Wesley, ACM Press, 1992), pp. 37-49.

[15] Unfortunately, the reference to conversation trees research is lost.

aspect of realty. But once we have a full meta-model theory then we can see that schemas connect to mathematical categories that then connect to theories via models. This is the arc of science, which is different from the arc of philosophy which directly connects the schemas to logic. This round about connection is more powerful because it brings in the order of Mathematics to guide the intuition of theory which tries to find the type of math that underlies the action of the phenomena. Model theory is very important, but more straight forward in our tradition than representation theory. Logic and Math are well developed in our tradition and to connecting them via models is fairly straight forward. So there is not much to say about model theory except that we must extend it as well to deal with realty rather than just truth, presence and identity. Once we add in realty and allow a vajra logic plus other exotic logics then model theory as it is established will serve us well. For systems engineering this shows up as formal methods that combine mathematical structures with logical structures.

Unwritten Laws Revisited

These laws were presented by David F. McClinton at INCOSE 1994. They are a good test case for my framework of schemas theory. Lets see if we can make sense of them through the framework I have suggested.

Everything interacts with everything else

Interaction of quantities is N^2 but interpenetration of qualities is 2^N. Not only is it the fact that the ultimate field is an Lano N2 diagram but in fact in functional decomposition we are positing a single kind that covers the whole system and that decomposition is based on the Venn diagram which is based on interpenetration not interaction. We posit a single kind in order to obtain a unified system. Thus the kindness of the system is all collapsed into a single universal kind that is used as the basis for the unity of the system itself. Notice the Venn Diagram with its 2^N relations unfolds according to Pascal's triangle, so the dimensionality of the interpenetration determines the appropriate schema. The corollary is that *everything interpenetrates with everything else, too.* This means that the fact of interaction of all things with all things has two horizons. There is the outward horizon by which things interact in the physical world. But there is also the inward interaction through interpenetration of all things with all other things. Interpenetration really means that each thing gets its essential characteristics in relation to the differences between it and everything else that is defined in the system. In a way this is the difference also between Physus and Logos. Physus is the unfolding of outward interactions, while Logos is the unfolding of inward interactions through the distinguishing of differences. We need to take into account both the outward interactions which physically occur in the instantiated executing system, but also we need to take into account the implicit interactions within the design through the definition of differences throughout the design process. It is interesting that the executing system acts like a mass while the design acts like a set. Thus different sorts of logic control the design verses the instantiated executing system. So we can see that by allowing for the distinction between inward and outward interaction of everything with everything else we bring into play not only the physus and logos distinction but also

the mass and set distinction. We recognize that this mutual interaction is both explicit externally as intersufacing and implicit internally as interpenetration. External interaction does not capture the essence of the problems of design, because design deals with possibilties and not just instantiations of actualties. Elements of a design interact through this inward dimension of realzable unrealzed possibilties and propensities as much as through the outward probabilties and determinacy. In fact it is only though this interaction with realzable unrealzed possibilties that emergence can be brought about which is the essence of engineering.

Everything goes somewhere

Decomposition according to a single kind, i.e. the function, exposes interfaces which then need to be managed ruthlessly. But also the corollary is true that *everything comes from someplace* and that means with origin and sink we have the arrival of things into an arena and their interaction until they leave the arena and return to their source. Thus this rule suggests the meta-system. The system is indeed a meta-system to the subsystems within it, and everything that circulates in that arena must be tracked from point of origin to its ultimate destination. One loose end can lead to disaster. Therefore there is a kind of accounting that is necessary, like the accounting for energy by physics that says that all the energy is conserved and thus must go someplace after an interaction of some sort. Everything comes and goes from somewhere indicates the fact that every design space is an meta-system (open-scape) which we call the design landscape and as a meta-system it is made up of a source, origin, arena and boundary. Things arise from a source outside the boundary of the arena at a point of origin. System and Anti-system arise together within the meta-system. The anti-system is perhaps only an unrealzed possibilty for the system being designed. All systems as realzed possibilties arise on the background of all the possible instantiations of the design within the design landscape. So the system within the meta-system arises and then is given resources it needs to survive from the meta-system. All those resources need to come from standing reserves within the meta-system and are funneled to the system as the meta-system sees fit. Everything that the system needs comes from someplace within the meta-system and then everything that the system produces goes to somewhere within the meta-system, unless it is consumed by some other system within the same meta-system. This aphorism is only a slight hint at the necessity of our understanding the meta-system. We need to explore the relation of the system to the meta-system much more thoroughly. At this point we do not have a formalization of the meta-system like we have for the system. But the relation between them is like the relation between the Turing machine and the universal Turing machine (which reads and runs other Turing machines form tape). Meta-systems are like computer operating systems in relation to the application systems that run on them. Saying that everything comes and goes, says not only that the system comes and goes from the meta-system, but that the meta-system as a media facilitates the communication between the systems within it, but also provides resources for those systems, and it also provides sinks for their outputs that are not absorbed by other systems.

There is no such thing as a free lunch

Remember the negative consequences of each trade study. But we know that literally this is not true, there are free lunches, it is just someone else pays. The whole question is who pays the piper. If you push things outside your system you are attempting to get someone else to pay the piper. This aphorism suggests that there is a field phenomena at work rather than a reserve phenomena as the last aphorism suggested. In a field each thing is dependent on every other thing within the field and so this mutual connectedness of things allows you to see the trade off of shifting problems around. They are like bumps under the carpet. Push a problem down one place and it will as if by action at a distance pop up in another form elsewhere.

Notice that these laws are meta-systemic not systems laws. They point up the nature of interpenetration as a means to functional unity, but it is the meta-system that maintains interpenetration of the systems within it. Also we mentioned before the Set and Mass mathematical categories. But here are introduced two more mathematical categories which are the Reserve and the Field which are the duals of Set and Mass and each have their own logics. The logic of the Reserve is accounting, as we must do for everything that goes across an interface in the inner meta-system of the system. The logic of the field is transformation, functions are transformations. Within a field the intensities can all be intertransformed by field operators. But whatever changes you make effects the whole field.

The aphorism suggests the reserve, where like with energy there is always an accounting. But we must realize that there is also field phenomena which give free lunches all the time, because waves can interact in such a way to produce very high fluctuations that are abnormal and could not have been generated by the system alone within the field. Yet if the system takes its energy from the field then it must live off these miracles within the meta-system and avoid blackholes, where there is a reverse addition of waves that create a super deep trough. So we can save that from a reserve accounting point of view there is no free lunch. But from a field point of view we can position the system so that it appears locally as if there are free lunches, like we get from our proximity to the Sun every day, that makes life possible on earth. Globally accounting clears up these apparent local free lunches but still it can be that there are some inefficiencies in the market that will return a high investment for a considerable risk. Some systems operate in the meta-system environment in such a way that they take into account of those opportunistic fluctuations in resources.

Simple Truths:

Never Confuse Change with Progress.

This simple truth points to the difference between positive and negative entropy. Intentional action toward a goal is neg-entropic, but this can only be achieved by the production of greater entropy elsewhere. If you get within the backwash of entropy then you are lost in spite your good intentions. In this essay we have spoken about the importance of entropy and how it is equal to one quarter of the surface area of a system. This means that you want to keep your surface area to a minimum in all cases. The more elegant your solution to problems and the less volume they take up the less overall entropy you have to fight against in the development process. Each elegent

solution is a stepping stone of progress. By elegant I do not mean the baroque of the over designed, but rather the simple but significant articulation of a solution.

In design we must continually fight against entropy. Change may not be neg-entropic so that it can in fact be disorganization when it looks like further organization. And example is when different standards are used and unnecessary variety is produced. This causes a thrashing between various standards which wastes energy of everyone involved. Projecting a standard prior to the work so that the different pieces fit together when they exist is difficult but necessary to achieve the desired goal. Progress must be based on intention and it is difficult to sustain and project and intention in a group. It is hard enough for an individual to do that. But that is why there is an interesting relation between the one kind of the functional decomposition and the intentionality of the person or team producing the system under design. Systems design is a projection, and as such it is a projection based on a schema. That Schema may be the system, but it may also be one of the other schemas we have mentioned like the form, or the pattern, or the meta-system. Progress is counted in terms of our projection of the schemas and then following up on them and remaining consistent with them throughout the design process. Change may be mere flux that causes entropy. Change that maintains the ordering of the schema and thus produces intelligibility in the design is the basis of progress within the design process. Neg-entropy means imposing order in the face of overwhelming disorder. Order is based on some schema. Progress means successfully projecting the schema and then realizing the design within the prior ordering of the schema that confers intelligibility on the designed product.

Never be afraid to start over

As human beings we can change our course, we can create and destroy work. That is the nature of the kind of non-routine work that a Systems Engineer does that he produces work for others, but if a better way to do something comes along he can also destroy work, and produce less work for others the second time around when a more elegant solution occurs to him.

Starting over can sometimes mean that after you realize the inner patterning or implicate ordering of the thing to be designed then you can make a better design after having thought through the design already once. This means disordering the old design and reordering it to a new more efficient or more effective patterning. In many cases this means that you realize better which schema should have been used and how to fit that to the necessities of what needs to be designed. But starting over also means allowing the design to evolve as it unfolds in physus and logos. But physus and logos imply developmental change. When things developmentally appear then they will transform in the process of evolving. The willingness to start over recognizes this necessity of transformation in evolution of the design and does not stifle it, rather by allowing for the necessity of rebase lining and starting over we facilitate the appearance of emergent properties. Not being afraid of starting over is to in fact enjoy the benefits and the adventure of the appearance of emergence within the design process.

Better is the enemy of good

The good is a non-dual which lies at the

level of the distinction between to have and not to have which lies beneath the distinction between finitude and infinity whose non-dualis rightness. Good enough is destroyed by the urge to perfection. But more deeply the good is the origin of variety. What is good for one person is poison to another. So in a way good is what allows us to all live together with our different desires for what will make us happy by obtaining what we think is good for us. But the good is also the origin of kinds because it is the variety of kinds that make possible the differences in what is good. Stafford Beer in The Heart of Enterprise argues that humans are variety producers and that you are never going to understand all this variety or stop its production. The deeper lesson is to accept variety in people and their work products, and only expect conformity at a certain level that leaves room for creativity. The better can be a creative elegant solution that no one has thought of before that helps make everything easier. Never close the door on the Better. But the better appears at the level of right, which also means rta in Sanskrit or cosmic harmony, or arte which means excellence in Greek. By saying that the Better is the enemy of the Good on a superficial level we might be saying that one should allow for the solution that is good enough, and not try to over perfect it and make it better for no reason. But at a deeper level the Better being the enemy of the Good means that there are at least two levels of non-duality operating separately. There is the level of right that operates between finitude and infinity, and there is the level of good that operates between having and not-having. These must be satisfied differently and cannot be mixed without trouble. One must allow for variety production at the level of the Good, and attempt to find a solution, that is good enough for the situation. But on the other hand there is the striving after excellence in design, and the striving after a harmonious design, i.e. the right design which is better, rather than a design that just works and is good enough. The competition for excellence and harmony needs to be weighed against the necessity of costs and deadlines that may make good enough a better solution when we take into account external variables in the development process. Thus this aphorism also points us to the meta-system or environment of the development process beyond the system of the project being performed. These externalities may impact the design decisions and make one accept the system that is good enough rather than striving unnecessarily for a more costly better solution. The fact that these criteria are enemies merely states that there are two levels that are non-dual bases of action and decision and that they are both operating at the same time but remain distinct from each other producing different criteria for the judgment of the product of the design process.

If it is not written down it never happened

Our culture is based on writing. It is both our boon and our curse. We write long boring documents about systems, because if we don't others cannot know what we intended and cannot coordinate with us. The record of what happened is what is concentrated on by the CMMI when evidence is gathered to assess compliance to best practices. Between the process and the product there is the written record. We do not seem to be able to avoid this fate of being tied to written records. Action by itself does not build toward anything. Concerted action takes information sharing that means that something needs to be

written down. But it can still be very concise. Agile processes attempt to make a virtue out of the streamlining of the development process. One way to do that is to create teams that work very close together and are not interrupted. As long as that team can be sustained and it is not uprooted by turnover then it is possible to be much more productive and write less down. Written documents do not contain the theory of the design anyway, in order to get that you have to interact with the designer according to Naur, so in a way writing so much down is a waste of time from one point of view, but if you do not write enough down you lose track of what you are doing and cannot hand off what you have done to others. Somewhere as team size gets better there is a cliff that you can falloff that will make the project collapse under its own weight and if you do not have things written down when you cross that uncharted boundary then the whole project will implode. So there is a fine line as to how much should be written down and how much should be kept in the memory of the team. A point here that is important is that we think normally in terms of long and short term memory because we focus on individuals. But the team also has a memory of its conversations, and it is in this dialogic memory that the design theory is stored, not in the long or short term memory of the individuals. This dialogic memory of the team is very efficient in holding all the conversations that the team has had about the design and we can return to where we left off on most of those conversations when reminded in the midst of dialogue very efficiently, whereas we have difficulty returning there when we are alone. The distinction between CMMI like processes that are geared to large projects, and Agile processes that depend on the dialogic memory of the team has to be made carefully because the dialogic memory of the team is only so big and after a certain size it breaks down and if you do not have things written down at that point then it is impossible to return to it and know where you are in the development process. Some combination of conversation and writing that is efficient and effective needs to be instituted in each project based on its size, the difficulty of the project, the turn over of the team and other factors that allow one to draw the line between agile and CMMI processes for each project case. There is no one size fit all answer for this problem. There is a huge overhead for running a CMMI compliant process. Paying for that overhead in every case may not be wise. But that is why it is possible to tailor the processes within the CMMI framework, so that that right balance can be struck between what is written down and what is stored in the dialogic memory of the team that is distinct from the long term and short term memory of the individuals on the teram. As Peter Naur says that is where the design theory is stored and you cannot write it down, and whatever you do write down cannot contain the design theory, so writing everything down is the wrong answer, it ends up being another version of working to rule. When the railroads in Britain work to rule then everything comes to a halt because they apply every rule in the rule book, but no work actually gets done.

Never be above plagiarism

We are taught in school never to plagiarize. Intellectual property is everything in this culture. But copying is also a major way that our culture is promulgated. The real issue is the control of copying which is a question of repetition of representations. Notice that

this is different from the representation of repetitions. Following processes decreases the time it takes to invent your own way of doing something and your on format of the results. It is unnecessary difference that processes attempt to eliminate. Necessary difference is important and needs to be recognized. But unnecessary difference is wasted energy and time. You hear over and over this refrain about shamelessly plagiarizing which is the inversion of the not-invented-here syndrome. In fact, this whole question is a big quandary for organizations which generally have difficulty finding a happy medium between totalitarianism and laisie a faire approaches. But the real issue is not whether to be afraid to use the prior work of others to achieve efficiencies or to build a better mouse trap which may cost more. The real issue is the balance between human creativity and the efficiency and effectiveness of reusing things which are based on an established norm. Not invented here syndrome usually puts forth products that are not based on any norm but are idiosyncratic. But on the other hand reusing products that are inferior and not being creative and producing superior products that establish a new norm is stupid. There is usually a halfway house between these two tendencies and one should look for that happy medium on a case by case basis. This means that this like so many of these sayings are based on a nihilistic dualism which actually calls for a non-nihilistic distinction being made by the designer. If we assume that not invented here is the norm, then to counteract that we say that we should not be afraid to plagiarize. But when plagiarism runs amuck and there is no intellectual property being created, but only bad norms being promulgated blindly then we will swing to the other extreme. Rather it is important to recognize the importance of the non-dual non-nihilistic distinction between the two nihilistic extremes in each case and try to draw that distinction as best we can. Some of the unwritten laws are merely restatements of lopsided positions as the pendulum swings too much to one side at the time of the writing of the author of these aphorisms. We need to recognize the deeper dynamic beyond the surface advice of the individual aphorisms and then attempt to gain wisdom from seeking non-dual solutions to problems that normally are merely exercising a vicious dialectic.

A thing not worth doing is not worth doing well.

Learning to distinguish what needs to be done, and the best route to get it done, is always a problem. Much of the work set up by someone for someone else is wasted effort. Our society is over worked but much of the work is just a waste of time. Thus we enter into an understanding of the fundamental nihilism of our situation which is expressed by the saying "round and round the ragged rock the ragged rascal ran." In other words if we are merely producing for production sake then we are not getting anywhere fast and merely wasting resources in the process. But notice that there is an invocation of Better again, which points to the non-dual of rightness which is expressed by the word *well*. If it is not worth doing it is not worth doing right. In other words, we should consider if the work should be destroyed before we consider whether we should do it excellently. Non-routine process is about the creation and destruction of work. Routine work assumes that what ever work that is defined just needs to be gotten done. Thus we should do the non-routine work first of creating the work and then we should work

off the created work that is worth doing. It turns out that Systems Engineering is mostly non-routine work, while Software and Hardware Engineering contain cores of Routine work. Systems Engineers must do the non-routine work at the higher level of abstraction which will define the routine and non-routine work at lower levels of system abstraction. Part of that is the creation and destruction of work, and the formation of processes. Systems Engineering should be equally concerned with the work process and the system product. In fact, the process and the system are duals of each other. Systems Engineering *IS* Process Engineering, in the sense that the Systems Engineers should engage in the non-routine work of defining the work that needs to be done on the project. To the extent that Systems Engineers only see themselves as Product Engineers they will miss half of their calling of their profession and a lot of wasted work will be done.

There is no shelf.

This is an excellent point. Off the shelf is basically an illusion. That is because everything is context sensitive. But sometimes it is better to use something that does not completely fit for expediency. But usually that is a mistake in the end. Lots of times COTS products change, or have hidden differences that violate the requirements of the system if not when they are first used then eventually as their companies change them in response to market pressures that may have nothing to do with the product they are included within. However, this does not take into account the reuse within domains or product lines which may be more successful. However, most companies cannot afford to create these efficiencies of reusable components unless they are in some very special market where that makes sense. Thus the declaration that there is no shelf may refer to the fact that reuse and product line development, is practically impossible as well. However it seems that the essential issue here is the fact that all projects are context sensitive and that it is very difficult to reapply and reuse material from outside that context within the new context of the project. This context sensitivity points again to the meta-system of the project and its importance. The deeper point is that all projects are embedded in unique contexts, and that context independence is almost impossible to achieve.

Any interface left to itself will sour.

The major role of systems engineers between requirements and systems design at the front of the project and the testing at the end of the project should be riding herd over interfaces and their necessary changes. If this role is neglected disaster will follow. Interfaces left inert and will not take care of themselves. Systems Engineers have as part of their job making those interfaces active by making sure that the appropriate lower level design engineers talk to each other and share information about interfaces. But this points to higher level assertion of the role of the system within the meta-system. All interfacing is through the meta-system and we need to be aware of it both within the project as within the system being designed. Interfaces must remain active and thus there is a constant role of interfacing that needs to be performed by the systems engineer because he represents the meta-system within the various designed subsystems fit within the boundary of the system as a whole being developed. Systems Engineers in their role of keeping interfacing between lower level

designers occurring addresses not only the meta-system of the project but though that the meta-system of the designed system that is being brought into Being through the production process.

Plan your work and work your plan.

Plans are out of date as soon as they are written. In a sense they are a waste of time and energy, except if you don't do it then you are lost, completely lost. Best to plan then throw the plan out the window and start the next plan. If you follow your plan too slavishly then you get out of touch with reality. If you don't plan then you have lost your vision of where you are going and how you are going to get there. Plans should be written on toilet paper. They are written to be discarded because the situation of the project is constantly changing. But the Plans get rid of all the stuff that would happen if there was no plan and if people on the project did not coordinate under a single vision of where the project is going.

We don't have time to do it right but we have time to do it twice.

Bureaucrats in Aerospace companies are not known for their far sightedness. Expediency is the key guideline. But expediency does not always work, in fact it seldom works. Sometimes it is better to work to get something right and spend a little more time and money on it because in the long run it will save money, but for the most part managers are short sighted and that does not happen. But notice here again there is an appeal to the non-dual of rightness (Rta, Arte) which is a central value in the indo-european tradition. When we just throw darts at the dartboard of the design space and pick a solution without simulation and prototyping then we are likely to get the wrong solution. Putting more thought into the design normally pays off in the end but it is difficult to justify in a culture that wants us to start coding from the first day skipping requirements and design completely which then produces defects in integration, verification and validation. The short sightedness of American Business in general is not something that is going to be solved soon and this aphorism is a recognition of this general structural feature of the business world. But the appeal to the non-dual of right lets us know that the central wisdom of the seasoned engineer revolves around traditional Western values based on non-duals, which say it is better to do something right, rather than just any arbitrary and random thing that eventually results in having to do it again and doing it better the second time.

Nothing is impossible to the man who doesn't have to do it.

It is easy for people who create work for others to get out of touch with reality. Systems Engineering is all about confronting realities. Best to create and destroy your own work rather than having it created and destroyed for you. There is a certain in built idealism and unfounded exuberance in those who do not actually have to do the work and confront the actual problems entailed within the work itself. Conferring the reality of the situation normally only comes with experience of similar situations in the past and the confronting of problems that inevitably arise. Not recognizing impossibilities or difficulties that arise from practical situations within the

development process and not being realistic enough is a fundamental problem. For instance, this problem comes out in the interaction of Marketing and Engineering on a frequent basis. Marketers advertise or promise the moon and Engineers are expected to deliver the moon on time and on budget until the project hits the wall of necessity and impossibility and implodes from the weight of too high an expectation on the ability to work miracles by the developers. Impossibility is a way of talking about the relation between finitude and infinity which again refers to the non-dual of the right. It is better to pick a goal that is just right rather than one that is impossible or too prosaic and mundane to be non-effectual Finding the right balance between risk and opportunity is always difficult but always rewarding if it is found.

Don't keep polishing the cannon ball but do get the caliber right.

It is easy to lose site of what is important. Notice here again there is an appeal to a non-dual The reference says that the part needs to fit into its place, but that over perfection is a waste of time. Thus there is a implicit reference here to two senses of the word right in terms fitness and in terms of excellence. It says that we should strive for right in the sense of RTA, cosmic harmony between the pieces, rather than ARTE or unnecessary excellence. Making distinctions between senses of a non-dual really focuses our attention on the non-dual and the role it plays in the design and development process. The non-duals are sources of values. And values play a key role in everything we do including Engineering work which is touted by some to be value free.

Any analysis will be believed by no one but the analyst who conducted it.

Unless you figure things out for yourself it is difficult to believe the results of the analysis. Thus a lot of times the briefing of the results of the analysis turns into a game of second gussing. But this also points to the fact that there is variety production which is associated with the non-dual of the Good. Thus everyone will come up with different results from different premises and that variety when unnecessary will stifle cooperation and consensus. Thus this is also an appeal to a non-dual of the Good which is the indicator of Human Variety Production which cannot be structurally reduced as the norm in human society. What we need to do is to make the most of the inherent creativity of this variety production while limiting unnecessary variety through agreed on norms and standards.

Any test will be believed by everyone but the one who conducted it.

These are two of the major ways a system is verified, the other possibility is demonstration. An analyst gets lost with his head in the clouds, but someone who does a test, is grounded except only he knows the setup for the test as well as the results. The person who creates something knows all of its flaws. If we do not look into something for ourselves then we will not be aware of the possible pitfalls. Thus consultants are always believed because they are outside the organization and no one knows their flaws. The unknown has a kind of sanctity. We use that sanctity as a means of distancing ourselves from things that we do not want to have to handle. We believe our stock brokers, even though their interests are different from our own. It is better for them to recommend the stock with the biggest commission rather

than the one which will gain the highest value over time, if they could know that which they can't, so they recommend what is best for them. Thus there is a kind of voluntary blindness in relation to those things we want to naively believe in but do not want to spend the time to look into on our own. But on the other hand we give responsibility to others to take care of those things they have responsibility for so we don't have to deal with them. So there is another side of that coin, which is that if you hire a test engineer to run tests you depend on him to make sure that the test results are correct, and you trust him to do his job. So the other side of this aphorism is the necessity of trust within organizations. When trust is vital within an organization then individuals are fulfilling their trusts and responsibilities and all the bases are covered, then you can trust the results of the lonely test engineer who tells you that the system is working properly, and it is not just a cover up of unseen flaws that were not uncovered because of faulty tests that no one wants to repeat themselves to test the tester.

Analysis is not believed and Tests are believed. Tests are considered authoritive because there is a confrontation of reality in a test which does not exist in an analysis. But not everything can be tested and compliance is sometimes shown by Analysis and Demonstration rather than testing. So you cannot get rid of analysis and thus there is some part of the system that no-one will believe works until it is validated in the field. Demonstrations also work only in ideal conditions many times. Thus the necessity for validation that actually asks if you built the right system (validation), not just if you built the system right (verification). Notice that again this is an appeal to a non-dual Right as a criterion against which the system is judged. Eventually we will see that there is a relation between the non-duals and the schemas and that the non-duals carry the criteria against which the things designed using the schemas are judged.

One test is worth a thousand expert opinions.

This is what makes Systems Engineering like science. A hypothesis is set up and then subjected to a possible refutation. Before the test there can be endless opinions but the test will narrow them to a few interpretations if it is a good test. It is though testing that reality enters the picture. Reality is one of the four aspects of Being. The others are Presence, Identity, and Truth. These others are the basis of formalisms. It is only when formalisms such as designs are brought into contact with testing that reality enters the picture. Reality confers meaning. By adding Reality to the other four aspects then we generate semantics and rise above syntax according to Model Theory. Systems Engineering needs to appropriate what I have called a Vajra Logic. The Vajra Logic applies not just truth values but also presence, identity and reality values to variables so as to get a more complete picture of the system than mere formalization would allow. For the word formalization we could substitute the word schematization, because we mean conformance to not just the schema form but to various appropriate schemas. In Systems Engineering there is a place where the rubber meets the road, which we find in integration, verification and especially validation. Thus empty formalisms are not enough and we need an expanded way of looking at things we build that takes into account not just the formal aspects but also the aspect of reality, and does not just rely on the set

approach to things that describes the design but also takes into account the mass approach that describes the instantiation and execution of the system. By adding these various features to our framework for understanding systems engineering as schemas engineering then we make it easier to connect our thoughts about the things we build to the actual world in which they come to be used and the problems that occur in that process of inserting the contextless formally designed system into its context within the meta-system niche it was designed to inhabit.

Never conduct a test unless you cant live with allpossible results.

When you enter a lab to make a test anything can happen. This again makes Systems Engineering like science which is based on experiment. Instead of Experiment we have our verification and validation of the system in situ within the world as the source of the possibilty of dis-confirmation of out theory of design. Results again follow the rule of variety generation because the system is a human product and thus anything can happen when it is created due to the fact that humans are inherently inventive and produce variety of both conscious and unconscious varieties.

After allas said and done, a lot is said but very little is done.

The connection between effective words and effective action is hard to achieve. But here we have an appeal to the difference between Logos and Physus. Our culture leans toward the Idealist in some aspects and toward the Pragmatic in other aspects. Here pragmatism is giving its view of Idealism. However, both Logos and Physus are necessary only in the proper balance which is hard to achieve and is a result when achieved of a non-nihilistic distinction.

Never have more than ten blocks in your block diagram.

Short term memory has a limit. 7+/-2 chunks is the norm. But the greater issue is information overload as we build more and more complex systems. How many block diagram pictures can we bear to look at. Some way we must move beyond block diagrams to attempt to capture the whole system at the right level of abstraction and decompose it. This aphorism is assuming that decomposition is the norm. But there is also the object oriented paradigm which distinguishes different kinds of objects and does not assume functional unity of the system. It has been mentioned that it is possible that as humans we treat organisms differently than artifacts and that this is one of the differences between object oriented approaches and functional approaches[16]. If this is true that the difference between objects and functions appeals to a basic difference in perception built in to our perceptual faculties then the duality between objects and functions needs to be recognized and we need a dual statement about how many objects we can have in a diagram. But whether it is objects or functions with increasing complexity of systems no number will ultimately suffice because what we are building is just too complex to be held in formalisms of this kind, rather we need

[16] "Cognitive Fit applied to Systems Engineering Models" Laurence Doyle and Michael Pennotti (Stevens) CSER 2004 paper 121

other schemas that can handle more complex phenomena, schemas like meta-systems, domains, worlds, and kosmos etc. The fundamental reason we need schemas theory is that the sorts of things we are building has burst the bounds of what forms and systems can contain as schematizations and we need higher level constructs to contain these more complex design regions beyond the system.

Never use a word chart when a picture chart will do.

I make a practice of putting up a graphic chart and then talking about something else. That way you get maximum information quality according to Bateson in Mind and Nature. The point here is that we need to use the full panoply of our cognitive processing abilities and perceptual processing is so much more efficient than the processing of written materials. However, there is a danger in graphics not seen by the supporters of UML which is that diagrams are more impoverished than languages in terms of information density that can be conveyed. Thus I have advocated for a long time that we need to develop extensible design languages that do not depend on graphics but can be transformed into graphic representations and vice versa. Just as Objects and Functions are duals, so are Text and Graphics. They complement each other when used judiciously. So much ouf our design is view chart design. We for the most part do not have formal models of our design at the Systems Level of abstraction. Slowly we are using more of the tools of Software Engineering, but these tools are not fitted for the implicate order of the System level that is different from the Software level Thus we have the development of SysML. But still there is a prejudice toward Graphics representations that are ultimately impoverished rather than textual representations in extensible method oriented design languages that are the counterpart of software languages at the design level, but are different from formal languages that represent constraints or truth conditions. We need to strive for balance between the Graphics and Text approaches just as we need to achieve a balance between speech and writing. Thus our wise Systems Engineer is again advocating one of the dualities rather than attempting to find the non-dual non-nihilistic distinction between the duals which our approach advocates.

Never go in with the first wave.

Never go in with the second wave either.

Early adopters tend to crash and burn. But on the other hand that is where the interesting action in the field always lies. Our wise System Engineer is advocating being a late adopter of technology, methods and processes, because of the danger in early adoption, but if we do not adopt early then we can miss the benefits of adoption when the technology is appropriate. Such conservatism is traditional engineering. But on the other hand innovation is what drives our economy. His idea is to let other people take the risks first which can be wise in the appropriate circumstances. But here again it is best to pick a middle ground and to adopt early if it appears to be the right thing to do but to wait if you are unsure. These cases must be decided individually and in each case there is a non-nihilistic distinction to be made based on the criteria drawn from the non-duals.

Have the heart of a child but keep it in a jar on your desk.

Many managers lack the ability to balance being tough when necessary with being kind always. This is another nihilistic duality between tough mindedness and kindness. Both can be bad and one must navigate on a case by case basis as to what is the best course of action. When do we fight for the right alternative and when do we go with the consensus opinion which is probably wrong. How many times are we wrong and we are fighting for something that is ultimately wrong while the consensus is right. Wrong and right are appeals to the criterion of the non-dual and are in the eye of the beholder.

Deny everything, admit nothing, demand proof, and reject the proof.

Never get tangled up in the Legal system. It is your worse nightmare. But in fact there is a hidden resonance of this demand of proof and the schemas because each schema is related to a kind of intelligibility and those form a hierarchy that ends in proof which has the highest kind of explanatory power. All other kinds of intelligibility conferred by the templates of understanding called the templates are weaker than proof. But by mentioning proof that is related to form we invoke all the others. And it happens that it is with Protagoras that the schemas enters into the western tradition, who was a sophist who taught citizens how to defend themselves in court. So in this final aphorism there is an oblique reference to the legal system which is appropriate to the schemas because in fact the schemas appeared first within our tradition in relation to the court system as modes of rhetoric used in the Athenian court.

What we have noticed as we considered the received wisdom of the Systems Engineering Tradition in detail is that there is a connection between our framework of General Schemas theory and that received wisdom. The connection is oblique because many times the received wisdom is pointing to the problems rather than the solution, and many times the problem is nihilism and the solution is a needed appeal to the non-dual which allows non-nihilistic distinctions to be drawn. But ultimately we must see that aphoristic received wisdom is not enough. That is why we need to develop General Schemas Theory as a basis of General Schemas Engineering, in order to get beyond reliance on received wisdom which may lead us astray if we follow it unthinkingly or do not look into it carefully enough. What we need are written laws of Systems Engineering Practice based on the development of Schemas Theory and Practice.

Provisional Written Laws Of Systems Engineering Practice.

Our framework allows us to finally find a way beyond the unwritten laws which are the received wisdom and that points to the general economy of the meta-system surrounding the development of the system.

Know which schema you are operating with at any given time.

Know that the meaning of what every schema you are projecting comes from its relation to all the other possible schemas.

Know you can change schemas and project another one onto the same ontic

phenomenon if necessary.

Know the relations of the schemas to each other because that is the basis of all design. We see all things as space time envelopes and those envelopes have their own organization. When we design something we project those envelopes and when we implement something designed we fill those envelopes and give them content.

Use logic and math to order ones use of the schemas. Logic and Math are well developed in our tradition, but both are deficient in as much as Logic does not recognize deviant logics and Math does not recognize masses, fields and reserves as part of its providence. We need to apply expanded concepts of Logic and Math in order to understand schemas.

Do not take the short cut of using philosophical categories alone to connect logic and schemas. Rather follow the route of science that interposes mathesis between the logic and the schematization and uses models and representations as a basis of the mediation between physus and logos. However, recognize that Model theory adds reality to get meaning from syntax and that you need to consider all four aspects of Being, i.e. Truth, Reality, Identity and Presence. Also representations have an opposite in repetition that needs to be recognized. Representations are abstractions that collapse the dimensionality of whatever is represented while Repetition builds higher dimensions out of lower ones with the possibility of producing emergent transformations along the way out of singularities.

Recognize that the Physus has a logos and the Logos has a physus. The logos of physus is the schemas and the physus of the logos is logic. Logic and Math are well developed, develop Schemas as well to the same level of understanding in order to achieve balance of understanding.

Recognize the non-duals associated with the various schemas, and use these non-duals as the criteria for making non-nihilistic distinctions

Recognize the Rhetorical modes related to each schema and use them as ways of understanding within logos to give the phenomena their own voice.

Use the schemas as the basis of systems design. And take systems design up to Schemas Design by using all the schemas as a basis for design rather than just system and form. Base Schemas Engineering on Schemas Theory extending the basis of Systems Engineering on Systems Theory.

Allow the schemas to carry the complexity of increasingly complex designed artifacts. Form and System are no longer adequate to carry the complexity of world wide designed artifacts.

Recognize the duality of Mass and Set approaches to things, as well as the duality between Field and Reserve approaches. Recognize that each approach has its appropriate logic. Use the appropriate logic.

Recognize the duality between graphics and text, and the duality between speech and writing and give each its due making the determination as to how much of each is appropriate in each case.

Do the non-routine work of Systems Engineering which creates and destroys work prior to the routine work of working off the tasks of development. Do not work blindly doing what was planned by change the plan when appropriate.

Use both extensible design languages and graphical design languages and recognize their duality.

Use both object oriented and function oriented models of systems and recognize their duality.

In general, steer between the duals, as Odysseus steered between Scylla and Charybdis and attempt to hold to the middle path which is non-dual, i.e. which recognizes an alternative that is *not one nor many* but somehow allows the one and the many to coincide without conflict. Recognize in non-duality the path of least resistance which does not thrash back and forth between the artificial nihilistic opposites that continually arise in myriad ways to plague our development efforts. The received wisdom is not wise if it advocates one of these nihilistic opposites even if it is doing so to counteract the other one which has held sway too long.

True wisdom comes from understanding, and the schemas are the royal road to understanding, because they are the projected templates of understanding which appear in speech as rhetorical modes and in nature as levels of avoidance of harm, and in math as dimensionality. They are projected on all things as their spacetime envelopes prior to the distinguishing of kinds, either as function or object, and prior to the distinction of individual differences or significance within context.

We need a theory of schemas to escape the received wisdom because they can give us a criterion on which we can judge the received wisdom and see into its depths so that we have a more sure guide though the vagaries of system design and development of ever more complex systems that require the schemas to carry their load of complexity at higher levels of schematization that are only possible to recognize if we develop General Schemas Theory into a science more general than Systems Theory or Systems Engineering as they stand today.