24th Annual INCOSE International Symposium (IS2015)
Seattle, WA, July 10 – 16, 2015

Maps or Itineraries? A Systems Engineering Insight from Ancient Navigators

William D. Schindel
ICTT System Sciences
schindel@ictt.com

Abstract. Processes and procedures are the heart of current descriptions of Systems Engineering. The “Vee Diagram”, ISO 15288, the INCOSE SE Handbook, and enterprise-specific business process models focus attention on process and procedure.

However, there is a non-procedural way to view systems engineering. This approach is to describe the configuration space “navigated” by systems engineering, and what is meant by system trajectories in that space, traveled during system life cycles.

This sounds abstract because we have lacked explicit maps necessary to describe this configuration space. We understand concrete steps of a procedure, so we focus there. But where do these steps take us? And, what does “where” mean in this context? Clues are found in recent discoveries about ancient navigation, as well as later development of mathematics and physics.

This paper (Part I of a Case for Stronger MBSE Semantics) focuses on the underlying configuration space inherent to systems.

Introduction

Systems engineering processes. In contemporary discussion of systems engineering, we encounter descriptions of “Vees”, waterfalls, spirals, and other picturesque metaphors for the work process. In industry or enterprise-specific descriptions (ISO 15288 2014; INCOSE SE Handbook, 2014) of such work processes, the amount of ink and attention devoted to describing process, sequence, or activity usually exceeds by orders of magnitude the amount devoted to describing the information flowing through that process. We ask here why this is the case, and whether there is a more optimum future state for the effective practice of systems engineering. This inquiry is separately extended to include the life cycle trajectory of systems in (Schindel, 2015).

Maps versus Itineraries: Concepts of Space

Maps and itineraries of the ancient navigator

In an exhibition at New York University’s Institute for the Study of the Ancient World, scholars (Casagrande-Kim et al, 2013) suggested that ancient Greco-Roman navigators did not possess the “ancient maps” of the sort later attributed to them. Instead, it was asserted that these images were generated later, during the Middle Ages, and attributed to the thinking and artifacts of ancient navigators:

> *“Why do we have virtually no ancient maps of the ancient world?” asked a reviewer of the exhibition (Kaylan, 2013). “After all, sailors, traders and soldiers had to find their way around. The show's curator, Roberta Casagrande-Kim, distinguishes between a map and an itinerary. The latter ‘must have existed aplenty, but being strictly*

> *functional probably deteriorated through overuse,' she says. '<u>A map, however small its focus, suggests a kind of implicit overview, and that is the show's subject</u>.'" (Emphases added)*

In describing how human concepts of space and its representations have evolved, these scholars reported that "Greeks and Romans usually employed what are known as *periploi* ('coastal navigations'), which <u>listed</u> ports and landmarks to facilitate commercial and military sailing, and *itineraria* ('journeys'), <u>lists</u> of locations and distances based on land routes" (Casagrande-Kim et al, 2013) (emphases added).

Figure 1 suggests the conceptual difference between a map and an itinerary. An itinerary is a sequence of steps whose performance is expected to move us from Point A to Point B. By contrast, a map describes the geographic space of interest, identifying points in geographic space and the relationships between those points. A map is a relational model that answers an infinity of questions that may arise in various situations. A map is not a procedure. By contrast, an itinerary is a step-wise procedure intended for a limited purpose.

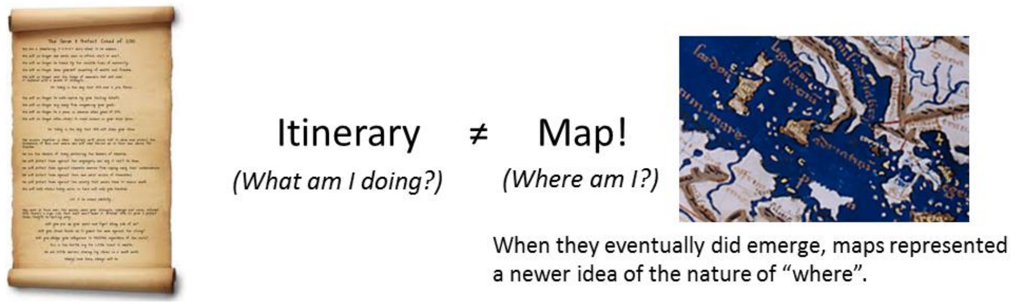

Itinerary ≠ Map!
(What am I doing?)
(Where am I?)
When they eventually did emerge, maps represented a newer idea of the nature of "where".

Figure 1: Map versus Itinerary

A key point examined by scholars is the concept of geographic space held by humans at the time these evolving artifacts were in development (Barkowski, 2002). The important notion here is that a map would not emerge sooner than the related cognitive concepts of the space it describes. To appreciate this, we must imagine a time when concepts of geographic space were not yet as developed as today. For example, recall the development of the Mercator cylindrical projection of a sphere (Figure 2), and consider the practical impacts of conceptual challenges that would have preceded its availability.

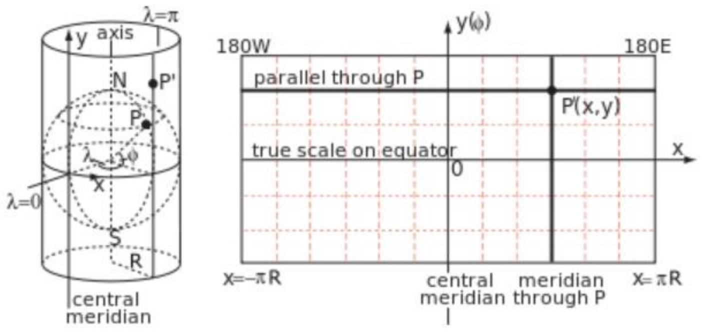


Figure 2: The Mercator Projection of Sphere onto Cylinder

For purposes of this discussion, the important idea is that people can lack a concept of space that is adequate to what they are trying to do in that space. It is difficult to imagine being without an already familiar concept, but important to understanding the current state of systems engineering. We suggest that equally fundamental concepts are not yet in the regular cognitive maps of the current Systems Engineer.

Maps and itineraries of the systems engineer

SE journeys. At least metaphorically speaking, Systems Engineers must “navigate” a type of “journey”, like their ancient navigator counterparts. The “journey” of interest here for the systems engineer is an engineering project:

- More complex and abstract than geographic travel, but …
- it has a starting point and destination,
- with opportunities to become lost or disoriented,
- with risks of not reaching the desired destination.

We will later argue that this is more than just a metaphorical comparison. But first, let us consider the sorts of practical implications at stake for systems engineers.

The limitations of procedural checklists. Experienced practitioners usually admit the following problem situation is a familiar one:

- The junior engineer reports having performed all the required steps.
- All the checklist boxes are checked.
- But, the result is not acceptable.

Why does the junior navigator not recognize, much less avoid, the problem? Often, it is because of deeper knowledge that the senior navigator has internalized through experience, but which is not represented in the official process steps. We will suggest here that what is missing is not just some overlooked steps to record, but relational map knowledge that cannot be represented as process steps alone, because is it about a map of something different than process space.

Are we there yet? Whether the SE journey in a project is based on waterfalls, spirals, or other metaphorical process approaches, certain aspects are inherently iterative, repeating certain activities until a sufficiency is achieved (Figure 3). This is about the underlying nature of design and exploration of spaces, and not about a certain styles of engineering processes versus others.

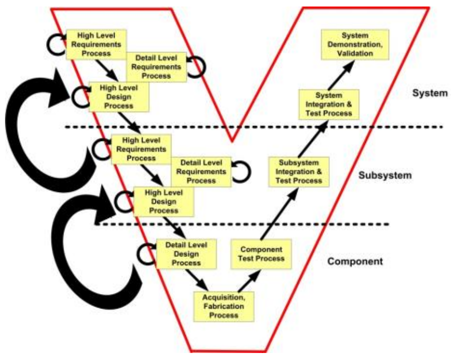

High Level Requirements Process
Detail Level Requirements Process
High Level Design Process
High Level Requirements Process
Detail Level Requirements Process
High Level Design Process
Detail Level Design Process
Acquisition, Fabrication Process
Component Test Process
Subsystem Integration & Test Process
System Integration & Test Process
System Demonstration, Validation
System
Subsystem
Component

Figure 3: Iteration Is Inherent to Systems Engineering:
So When Are We Done?

So, even when individual process steps are clearly defined, a frequently encountered and important question about an SE process is "are we done yet?" This question is answered by different means in different organizations:

- By examining the situation in an underlying information space, or else . . .
- By referring to a checklist of steps that should have been completed, or else . . .
- By referring to schedule or leadership requiring that we be done by now, or …
- By even more arbitrary judgments.

We will argue here that "are we done yet?" should be replaced by "are we there yet?", after we better solidify what "there" and "where" mean.

The above suggest that the practical implications at stake here are significant for the future of systems engineering. The history of science, engineering, and mathematics also offers evidence that improved cognitive maps of spaces have had profound impact in advancing those fields. Two of the most famous cases are the geometrizations offered by Descartes and Hilbert.

The geometrization of algebra. Rene Descartes is credited (Moerdijk, 2012) with moving understanding of symbolic algebra (in particular, algebraic relationships) into a geometric space setting, in which spatial understanding could contribute to understanding of abstract symbolic mathematics, viewed in "Cartesian" coordinates (Figure 4).

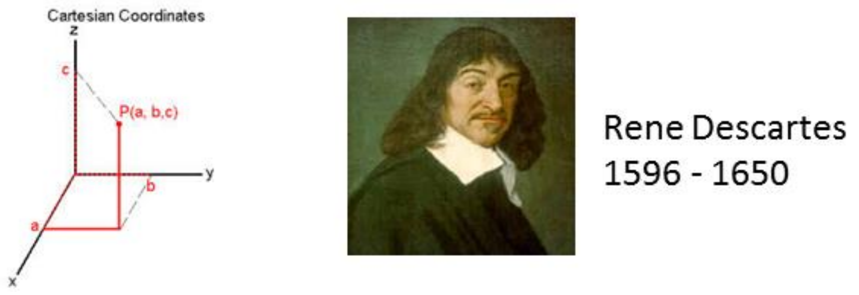


Figure 4: Geometrization of Algebra, by Rene Descartes

The geometrization of mathematical functions. As system models also add modeling of (infinite dimensional) behavior, Hilbert Space (Simmons, 1963) provided the next required generalization, supporting a geometrical view of mathematical function (Figure 5). The tools of the modern controls engineer and communications engineer, among others, have been profoundly impacted by geometry-based intuitive basis for more abstract mathematical operations: distance (metric spaces), projections, inner products (including convolutions and frequency transforms).

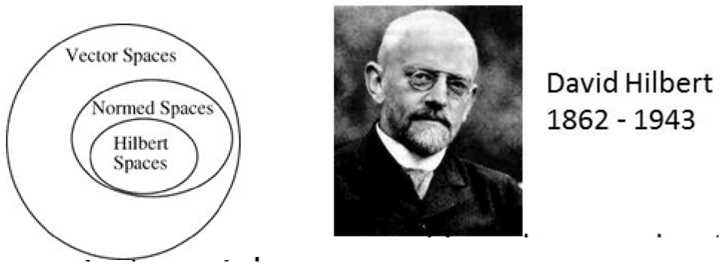


Figure 5: Geometrization of Function Space, by David Hilbert

Clues about a stronger semantic model of system space

It is relatively clear that the description of a sequence of systems engineering process steps (as in ISO/IEC 15288, the INCOSE SE Handbook, etc.) could be thought of as the metaphorical equivalent of the ancient traveler's itinerary. But, in the same vein, what would be the systems engineering equivalent of the geographic map for such a journey? Through what space is the SE traveling? This is not so immediately clear, but we can begin with what it is not.

A map of the space through which the SE travels:

- is not a list of SE tasks
- is not a model of the SE process—ancient mariners were not traveling through "step space", but "geographic space"

A geographic map describes:

- where we want to end up, along with other points in geographic space where we might conceivably be at a given time;
- key relationships between these points, including distance metrics;
- expressed in 1, 2, or 3 dimensions: degrees of freedom in geographic space.

So, what is the conceptual systems space through which the SE is navigating? To help answer this, here are a few things that we also know:

- The work of systems engineering produces, and consumes, information
- The space through which the systems engineer navigates would be a map about that information, not the steps of the travel process
- We assert that the space we are interested in should describe the space of possible places for a system of interest to be, good or not, and how they are related to each other: the configuration space of the system
- We know one kind of map about information: an information model (e.g., an entity-relationship or similar model)
- The hard sciences provide, in the maps for physics, chemistry, thermodynamics, and other domains, representations of underlying relationships (laws).
    - Frequently represented in the form of mathematical equations.
    - These relationships and their impact on systems space are the focus of attention: Imagine instead trying to learn chemistry by studying the process of cooking!
- Can systems science provide maps in the form of underlying systemic relationships?

Semantic models. INCOSE MBSE thought leadership has called for "stronger semantic models" (Long, 2014a, 2014b) to support the future progress of model-based systems engineering. This refers to the notion that, while current and historical modelling language and data exchange standards provide considerable "Metamodel" underpinnings, additional progress is needed.

We strongly agree with the call for stronger underlying MBSE semantics. Before discussing that subject, we recall what is meant here by "semantics".

There is an unfortunate practice in popular culture to use the term "semantics" as a dismissive pejorative, as if that term meant "insignificant detail" or "hair-splitting". To the contrary, "semantics" defines fundamental meaning, whether referring to formal engineering models, databases, cognition, or everyday natural language. Nothing could be more important to the success of human endeavour than shared semantics (meaning) that is sufficient for the activities in which humans engage. For purposes of this paper, we define “semantics” of a conceptual space as the degrees of freedom of that space, and the relationships between them—the “map” of the space.

An example of “semantics” in the technical space of science, engineering, and mathematics is Newton’s Second Law, sometimes expressed in equation form: F=mA. In discovering this natural law, Newton not only arrived at a quantitative relationship, but also a stronger (and inherently circular) definition of the concepts (mass, force, acceleration) that it relates. This was not just a matter of refining dictionary definitions, but a fundamental recasting of the relational cognitive map of the natural world, with profound practical consequences. (The same was true for those who followed Newton, refining that map.)

These three things are inter-related:

- System configuration space—the space described by the degrees of freedom of conceivable systems, in which each point represents one system configuration (Fig. 6)
- Relational models, constraining those same degrees of freedom with respect to each other, often mathematical or other relational models (including various types of information models)
- Semantic “meaning” expressed in the form of relationships

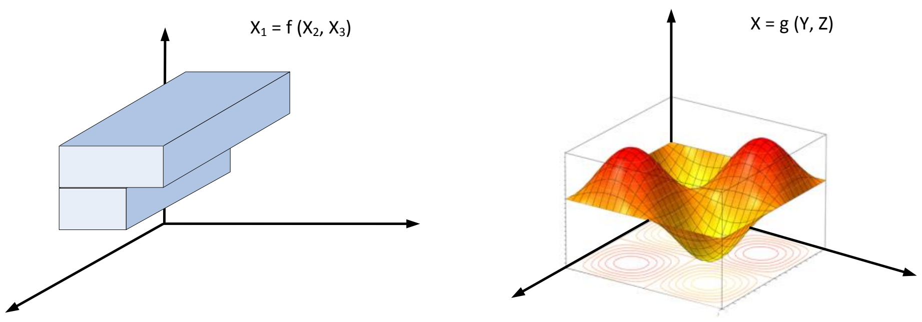


Figure 6: System Configuration (Degrees of Freedom) Space, Constrained by (Discrete and Continuous) Modeled Relationships Representing Semantics, Laws, Designs

Figure 6, representing a subspace of system configuration space, is not the same as the equations, words, or model views (e.g., SysML) that might be used to describe the set of instance points within it. This is an important reminder that a view of a model is not a direct view of the configuration space it describes, but instead a compressed representation of constraints that define such a configuration space—just as Descartes noted that viewing an algebraic equation is not the same as viewing the geometric space it describes—and both have their place.

What we usually refer to as “modelling languages” (e.g., mathematical languages, database modelling languages, systems modelling languages) are not themselves the semantics of the spaces they will be used to describe. The description of English as a language does not itself describe the struggles of Hamlet that Shakespeare encoded using English.

However, we know that architectural patterns, expressed in those modelling languages, can be used to describe the semantics of train systems or manufacturing processes. That is, the semantics of a lower-level language can be used to encode the semantics of a higher level “language”, formalizing the latter (Schindel, 2011b). Semantic models of systems engineering occur at different levels of abstraction. The following example list proceeds from more specific to more abstract cases:

1. Model of a specific automobile instance, configured as sought by its owner.
   - o <u>Example of use</u>: Represents whether Cruise Control option is equipped

2. Model of a product line of automobiles, optimized by designers and planners (ISO26550, 2013)
   - o <u>Example of use</u>: Defines which automobile models allow Cruise Control option

3. Architectural framework model (ISO 42010, 2011) of consumer automobiles, shared across suppliers active in the automotive domain
   - o <u>Example of use</u>: Defines semantics, behavior of “Cruise Control Feature”

4. Metamodel of a specific system modelling language, semantically capable of expressing concepts appropriate to its intended use, along with syntax and views specific to that language.
   - o <u>Example of use</u>: Defines how Stakeholder Features will appear in model views

5. Metamodel of concepts sufficient for the purposes of systems engineering or science, independent of the modelling languages that will express them in specific cases.
   - o <u>Example of use</u>: Defines the semantics of “Stakeholder Feature”

The entire configuration “System DNA” of a given system configuration or series of life cycle configurations can practically be captured by properly configured modelling, PLM, or other tools, as further illustrated in (Schindel, Lewis, Sherey, and Sanyal, 2015).

The dimensionality of this configuration space is high, so we don’t typically <u>view</u> the whole space at one time, preferring instead to view sub-spaces. Figure 7 is a simple example.

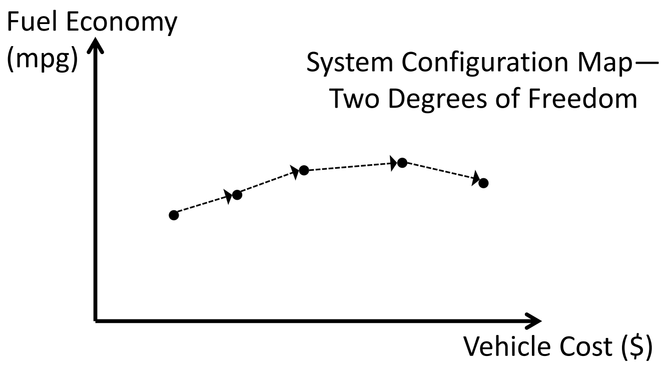


Figure 7: A Simple Sub-space of Configuration Space

The constraints that result in the curve of Figure 7 remind us that a further compression of configuration instance information is provided by modelled relationships:
- Mathematical equations (couplings, dependencies)
- Information models (E-R, SysML, IDEF, etc.)
- Requirements statements, viewed as transfer functions (Schindel, 2005b)

Moreover, Pattern-Based Systems Engineering (PBSE) methods permit even further compression of these views (Schindel, 2011b)—layers of compression are likewise possible.

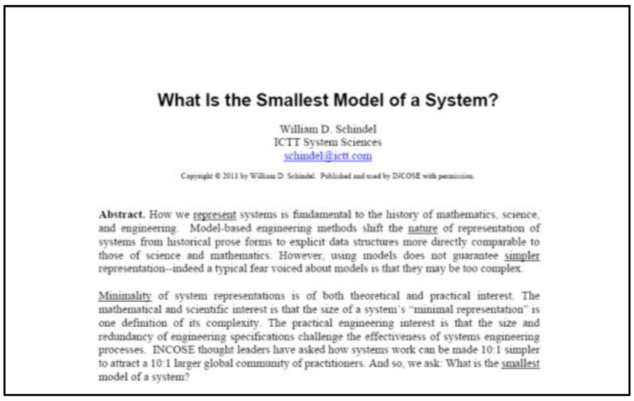
Most of the sub-space relationships are not linear, so certain ideas such as linear combinations and frequency domain transfer functions won't apply in the linear sense. However, other geometric aspects, such as distance norms and projections, do still apply. Of course, we'd likely add many more degrees of freedom (weight, range, etc.)—so system maps will tend to be high dimension, and subject to "slicing" into multiple views. During innovation / development cycles, and some life cycles, the "current configuration" may involve sets of ranges or lists, instead of individual points, so the trajectory becomes an ordered series of envelopes.

Moving to a stronger semantic model of system configuration space

What are the degrees of freedom (relatable variables) needed by System Models to describe system space? Do system modeling languages (SysML, OPM, IDEF, etc.) answer this? Some thought leaders agree (Long 2014a, 2014b) that such languages are more syntactical or view-oriented than about underlying semantics, with none of them currently providing a complete semantic model of the systems they describe. Based on the above arguments, it is perhaps too much to expect that they should, because they are intended to provide views into such an underlying system configuration space. Nevertheless, many of the ideas described in these modeling languages and other frameworks (e.g., OMG, 2012; ISO 10303 U'Ren, 2003) do cover a significant part of the territory. Along with a language description, modeling language specifications typically include an effort to describe the underlying system configuration space (even if entangled a bit in the description of the modeling language), for lack of a pre-existing community agreement on that underlying space.

In the spirit of the physical sciences, we therefore have asked "What is the smallest model of a system?" for effective descriptions in the work of engineering and science, and independent of any specific modeling language. Pursued over a number of years and tests, this work showed that contemporary system models are often both semantically too big (redundant) and too small (missing important information), at the same time (Schindel, 2011b).

In our practice with others across multiple system domains (Schindel and Smith, 2002; Bradley et al, 2010; Schindel, 2012b; Berg, 2014), this led over several decades to a formal model of the semantics of the underlying system space, referred to as the S*Metamodel. Figure 8 illustrates a key subset summary of the longer formal S*Metamodel specification (ICTT 2009, 2013).


What Is the Smallest Model of a System?

William D. Schindel
ICTT System Sciences
schindel@ictt.com

Copyright © 2011 by William D. Schindel. Published and used by INCOSE with permission.

Abstract. How we represent systems is fundamental to the history of mathematics, science, and engineering. Model-based engineering methods shift the nature of representation of systems from historical prose forms to explicit data structures more directly comparable to those of science and mathematics. However, using models does not guarantee simpler representation--indeed a typical fear voiced about models is that they may be too complex.

Minimality of system representations is of both theoretical and practical interest. The mathematical and scientific interest is that the size of a system's "minimal representation" is one definition of its complexity. The practical engineering interest is that the size and redundancy of engineering specifications challenge the effectiveness of systems engineering processes. INCOSE thought leaders have asked how systems work can be made 10:1 simpler to attract a 10:1 larger global community of practitioners. And so, we ask: What is the smallest model of a system?

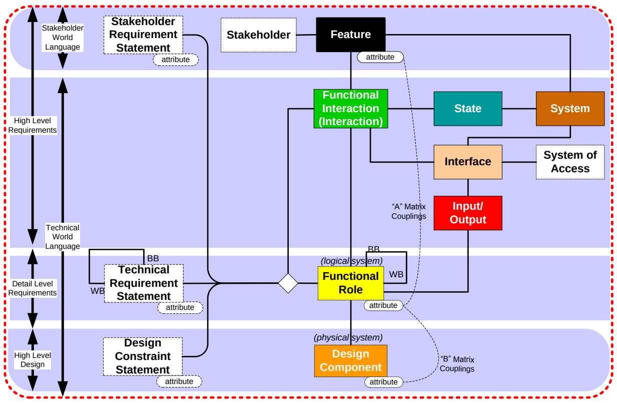


Figure 8: What Is the Smallest Model of a System?

Formal mappings (profiles) of the S*Metamodel have been created for a number of existing third party COTS modeling tool, engineering database, PLM system, and standards-based language offerings. These increase the power of the existing industry assets by strengthening their expressive power and semantic compatibility, in comparison to simple data exchange interfaces (Schindel, Lewis, Sherey, Sanyal, 2015). These systems and their users are enabled to represent and understand systems in S*Space.

Further evidence of the need

Why is such a transition in thought and practice important? An ancient navigator would not have been in a position to articulate the need for a map in the same terms we would use today, so today's systems navigators may face the same kind of barriers to visions of the future.

Further evidence is here offered in three areas:

1. **System Interactions**: One reference is the history of improvement of human life during the last three hundred years, driven by the fruits of science and engineering as they explicated and harvested deeper understanding of nature. A prime connection of systems and that history is the central role of physical interactions as the basis of all scientific laws in the physical sciences, discovered, expressed, and exploited over those three centuries to improve human life. We assert that physical interactions between parts are likewise the foundational perspective of the science and engineering of systems. Interactions accordingly play a central part in the S*Metamodel (Schindel, GLRC 2013a). However, these interactions are not necessarily recognized in the same way by contemporary system modeling languages and tools, or are in other cases merely tolerated by them.

2. **System Failures**: Human engineered systems have purpose, at the risk of failure in that purpose. Analysis of failure modes and effects (FMEA, FMECA, etc.) and other forms of risk analysis are central to systems engineering, and are likewise fundamental to the S*Space described by the S*Metamodel (Schindel, 2010). Purpose is not an add-on, and neither is failure in that purpose.

3. **System Requirements**: Systems engineers know that requirements are important, but they are most frequently conceived as the prose statements used to represent them to humans. Efforts by the suppliers of engineering tools and databases have brought forth databases and later models that incorporate and link to and among these textual structures. However, these text representations are the "prose equations" of the non-linear extension of transfer functions (Schindel, 2005), even if not recognized as such. Imagine an engineering world in which mathematical equations were viewed as being primarily the strings of text that represent them. Accordingly, the related transfer function abstraction is fundamental to the S*Metamodel's integration of Requirements.

Information vs. Process: Re-Integrating SE Maps and Itineraries

Once a stronger semantic model of system space is in hand, its re-integration with systems processes and procedures is possible. We have found this has good positive impact on the traditional procedures with which we re-integrate that systems space, making those processes and procedures more effective while respecting their historical roots and values.

For example, we have created formal models of the ISO15288 Processes, integrating with that well-known framework while giving new insight and power to its implementation. Figure 9 summarizes the notion that this paper began with: the SE Process (summarized at the top of Figure 9 by ISO 15288 Process Areas) consumes and produces information. By using a stronger semantic model of that information, we have strengthened each of the SE Processes that consume and produce that information.

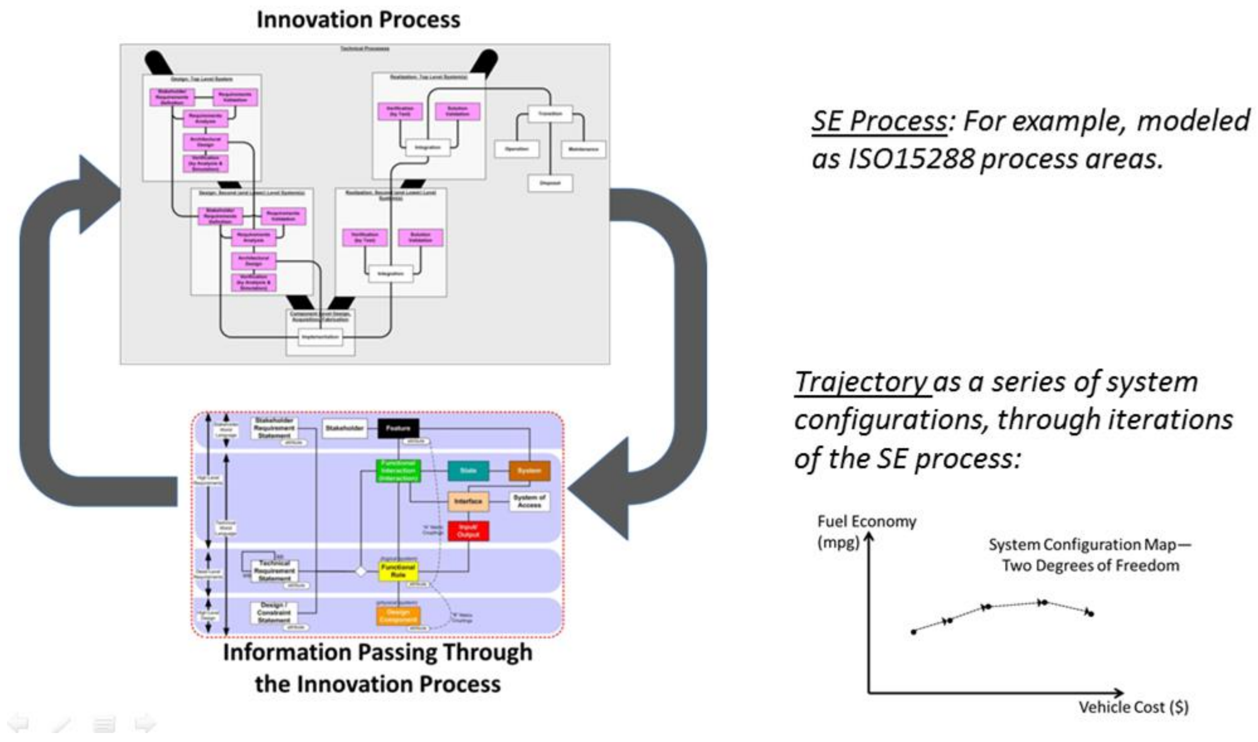

Innovation Process
SE Process: For example, modeled as ISO15288 process areas.
Trajectory as a series of system configurations, through iterations of the SE process:
Fuel Economy (mpg)
System Configuration Map—Two Degrees of Freedom
Vehicle Cost ($)
Information Passing Through the Innovation Process

Figure 9: Process versus Information

A part of that strengthening was to introduce into those SE Process models not only the option for MBSE models of target systems and their views, but the further notion that these models can be constructed from model-based S*Patterns. This is discussed in the next section.

Trajectories, persistent memories patterns: Roads already travelled

System configuration trajectories (Figure 9 lower right) are not just important during development of a single system generation. Across the life cycles of multiple systems, we have the splitting evolution of systems that emerge as responses to their environments. What is the configuration space for these evolving systems across multiple family life cycles (Figure 10)?

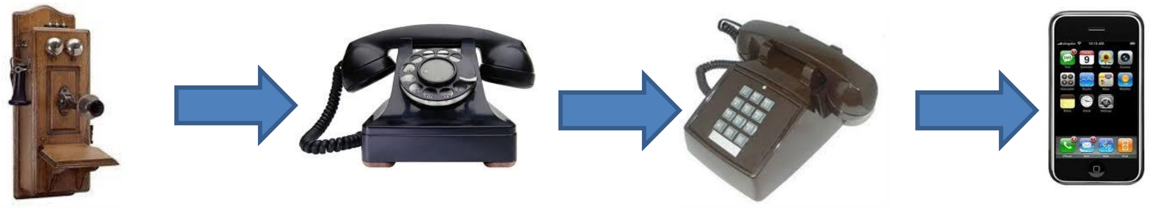


Figure 10: Evolving Systems Over Multiple Life Cycles

The same underlying S*Metamodel, along with the System of Innovation Pattern (Beihoff and Schindel, 2012; Schindel 2013b) supports all these, including more specialized system family, product line, or architectural patterns and frameworks (Fig. 11). In addition to our own firm's work in Pattern-Based Systems Engineering (PBSE) over several decades, PBSE based on these S*Patterns is also being pursued and practiced by the Patterns Challenge Team of the INCOSE/OMG MBSE Initiative (INCOSE Patterns Team, 2014), and the subject of several related IS2015 papers (Cook and Schindel, 2014; Nolan, Pickard, Russell, and Schindel, 2015, Peterson and Schindel, 2015, Schindel, Lewis, Sherey, and Sanyal, 2015).

When persistent memory of configurable re-usable S*Models are pursued as S*Patterns, where we have integrated it into the ISO 15288 process model, emerging themes include:

1. **Centrality of Patterns to Science and Engineering**: Although discovery of patterns may be argued to sit at the heart of the physical sciences, in PBSE they likewise become the heart of engineering and innovation. Indeed, we argue in (Beihoff and Schindel, 2012) that "accumulation of experience" is a key constituent of the System of Innovation, and formalizing it in patterns implements this. Patterns, as the basis for engineered platforms and product lines, become the equivalent of theoretical frameworks and paradigms in science.

2. **Intellectual Assets:** After several decades of investment in computer software, the Financial Accounting Standards Board (FASB) formally recognized accounting for that investment on a capitalized asset basis, joining "bricks and mortar" as a financial asset. Since that time, annual U.S. investment in intangible assets has grown to exceed investment in tangibles. The Model-Based Economy is arriving. S*Patterns satisfy the criteria of being a form of software, eligible for that capitalization of investment in systems IP (Schindel 2007; Sherey 2006)

3. **Process Patterns:** The Systems Engineering Process, or the larger Innovation Process, are themselves systems, and may be modeled as such (Beihoff and Schindel, 2012; Schindel, 2013b, Schindel, Ahmed, Hanson, Peffers, Kline, 2011). Accordingly, there are also S*Pattern representations for these systems, as we have created for ISO 15288. Beginning at the INCOSE IW2015 MBSE Workshop, we will examine the Agile Systems representation in this model-based framework (Dove and Schindel, 2015).

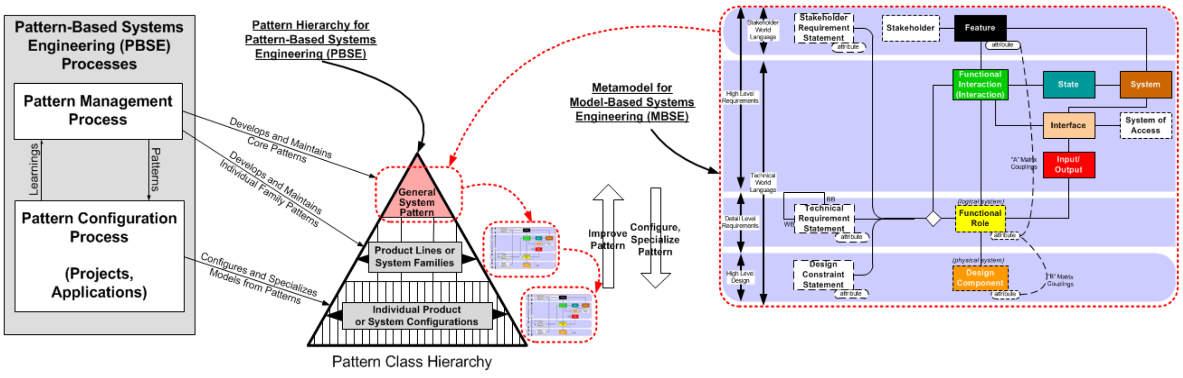

Pattern-Based Systems Engineering (PBSE) Processes
Pattern Management Process
Learnings
Patterns
Pattern Configuration Process
(Projects, Applications)
Pattern Hierarchy for Pattern-Based Systems Engineering (PBSE)
Develops and Maintains Core Patterns
Develops and Maintains Individual Family Patterns
Configures and Specializes Models from Patterns
General System Pattern
Product Lines or System Families
Individual Product or System Configurations
Pattern Class Hierarchy
Metamodel for Model-Based Systems Engineering (MBSE)
Improve Pattern
Configure, Specialize Pattern
Stakeholder World Language
High Level Requirements
Technical World Language
Detail Level Requirements
High Level Design
Stakeholder Requirement Statement
attribute
Stakeholder
Feature
Functional Interaction (Interaction)
State
System
Interface
System of Access
"A" Matrix Couplings
Input/ Output
BB
WB
Technical Requirement Statement
(logical system)
Functional Role
(physical system)
Design Constraint Statement
Design Component
"B" Matrix Couplings

Figure 11: Evolving Families of Systems, Pattern-Based Systems Engineering (PBSE)

Conclusions, implications and future work

1. We assert, and have offered argument and evidence above, that the vision expressed by (INCOSE Vision 2025) will require progress in shared understanding of the underlying semantic model of system space, and that this will be needed independent of specific modelling language/modelling view semantics, even when they are themselves

standards-based. Indeed, these languages and systems can themselves build upon and gain from such a shared underlying semantic model.

2. Current procedure-based systems engineering & innovation processes can be made more effective by increasing the focus on underlying information vs. procedure, without abandoning the value of procedural foundations, and with these impacts:
   - Knowing “where you are, not just what you are doing”
   - Simplification, while speeding and improving outcomes
   - Improved ability to understand, think critically about, represent, and communicate “the current situations” in projects, coupled with more effective risk management (Schindel, 2011c)
   - Increased agility of the overall System of Innovation (Dove, LaBarge, 2014)
   - Availability of an MBSE model of ISO 15288, incorporating PBSE options
   - Improved capabilities for even the currently available generation of automated aids, modelling tools, and PLM systems
   - Realizing more of INCOSE Vision 2025

References

1. (Barkowsky, 2002) Barkowsky, Thomas, Mental Representation and Processing of Geographic Knowledge, Berlin: Springer, 2002
2. (Berg, 2014) Berg, E., “Affordable Systems Engineering: An Application of Model-Based System Patterns To Consumer Packaged Goods Products, Manufacturing, and Distribution”, at INCOSE IW2014 MBSE Workshop, 2014.
3. (Beihoff, Schindel, 2012) Beihoff, B., and Schindel W., “Systems of Innovation I: Models of Their Health and Pathologies”, Proc. of INCOSE International Symposium, 2012.
4. (Bradley, Hughes, Schindel, 2010) Bradley, J., Hughes, M. and Schindel, W., “Optimizing Delivery of Global Pharmaceutical Packaging Solutions, Using Systems Engineering Patterns” Proceedings of the INCOSE 2010 International Symposium (2010).
5. (Casagrande-Kim, 2013) Casagrande-Kim, Roberta, et al, NYU ISAW web site and bibliography on ancient cartography: http://isaw.nyu.edu/exhibitions/space/bibliography.html
6. (Cook, Schindel, 2015) Cook, D., and Schindel, W., “Utilizing MBSE Patterns to Accelerate System Verification”, to appear in *Proc. of the INCOSE 2015 International Symposium*, Seattle, WA, July, 2015.
7. (Dove, LaBarge, 2014) Dove, R., LaBarge, R., “Fundamentals of Agile Systems Engineering—Part 1” and “Part 2”, INCOSE IS2014, July, 2014.
8. (Dove, Schindel, 2015) Dove, R., and Schindel, W., “Agile Modeling and Modeling Agile Systems”, to appear at INCOSE IW2015 MBSE Workshop, Torrance, CA, January 24, 2015.
9. (Estafan, 2008) Estafan, J. 2008. Survey of model-based systems engineering (MBSE) methodologies. INCOSE MBSE Initiative.
10. (ICTT, 2009) “Systematica Metamodel”, Version 7.1, Methodology Release 4.0, May 29, 2009.
11. (ICTT, 2013) “Abbreviated Systematica 4.0 Glossary”, P3125 Ver. 4.2.2, ICTT System Sciences, 2013.

12. (Ieke, 2012) Moerdijk, Ieke, “Descartes and the Geometrization of Algebra”, Descartes-Huygens Lecture, Radboud University, Nijmegen, The Netherlands, April 3, 2012.
13. (INCOSE Handbook, 2014) INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, Version 4, International Council on Systems Engineering (2014).
14. (INCOSE Vision 2025, 2014) “A World in Motion: Systems Engineering Vision 2025”, INCOSE, 2014.
15. (INCOSE Patterns Team, 2014) INCOSE/OMG MBSE Initiative: Patterns Challenge Team 2013-14 Web Site: http://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns
16. (ISO 15288, 2014) ISO/IEC 15288: Systems Engineering—System Life Cycle Processes. International Standards Organization (2014).
17. (ISO 26550, 2013) ISO/IEC 26550 “Software and Systems Engineering—Reference Model for Product Line Engineering and Management”, 2013.
18. (ISO 42010, 2011) ISO/IEC/IEEE 42010 “Systems and Software Engineering—Architecture Description”, 2011.
19. (Long, 2014a) Long, David, “Model-Based Systems Engineering at the Age of Eight”, presentation at NDIA GVSETS Conference, Troy, MI, August, 2014.
20. (Long, 2014b) Long, David, Keynote Address to INCOSE Great Lakes Regional Conference on Systems Engineering, Schaumburg, IL, October 10, 2014.
21. (Melik, 2013) Kaylan, Melik. “A World Without Maps”, The Wall Street Journal, 10.29-30.2013
22. (Mercator, 2014) “The Mercator Projection”, Wikipedia: http://en.wikipedia.org/wiki/Mercator_projection
23. (Nolan, Pickard, Russell, Schindel, 2015) Nolan, A., Pickard, A., Russell, J., Schindel, W., “When two is good company, but more is not a crowd”, to appear in *Proc. of the INCOSE 2015 International Symposium,* Seattle, WA, July, 2015.
24. (OMG, 2012) “OMG Systems Modeling Language, Version 1.3”, Object Management Group, June, 2012.
25. (Peterson, Schindel, 2015) Peterson, T., Schindel, W., “Autonomous Ground Vehicle Platforms and Model-Based System Patterns: An Example”, to appear in *Proc. of the INCOSE 2015 International Symposium,* Seattle, WA, July, 2015.
26. (Schindel, 2005a) Schindel, W., “Pattern-Based Systems Engineering: An Extension of Model-Based SE”, INCOSE IS2005 Tutorial TIES 4, (2005).
27. (Schindel, 2005b) Schindel, W. “Requirements statements are transfer functions: An insight from model-based systems engineering”, Proceedings of INCOSE 2005 International Symposium, (2005).
28. (Schindel, 2007), “Are Patterns Software?”, ICTT System Sciences, January 2007.
29. (Schindel, 2010) Schindel, W., “Failure Analysis: Insights from Model-Based Systems Engineering”, INCOSE International Symposium, Chicago, 2010.
30. (Schindel, 2011b) Schindel, W. “What Is the Smallest Model of a System?”, Proc. of the INCOSE 2011 International Symposium, International Council on Systems Engineering (2011).
31. (Schindel, 2011c) Schindel, W., “The Impact of ‘Dark Patterns’ On Uncertainty: Enhancing Adaptability In The Systems World”, in Proc. of INCOSE Great Lakes 2011 Regional Conference on Systems Engineering, Dearborn, MI, 2011
32. (Schindel, 2012a) Schindel, W. “Introduction to Pattern-Based Systems Engineering (PBSE)”, INCOSE Finger Lakes Chapter Webinar, April 26, 2012.

33. (Schindel, 2012 b) Schindel, W., “Integrating Materials, Process, & Product Portfolios: Lessons from Pattern-Based Systems Engineering”, in *Proc. of Society for Advancement of Materials and Process Engineering* (SAMPE), 2012
34. (Schindel, 2013a) Schindel, W. “Interactions: At the Heart of Systems”, INCOSE Great Lakes Regional Conference on Systems Engineering, W. Lafayette, IN, October, 2013.
35. (Schindel, 2013b) Schindel, W., “Systems of Innovation II: The Emergence of Purpose”, *Proceedings of INCOSE 2013 International Symposium* (2013).
36. (Schindel, 2014) Schindel, W. “The Difference Between Whole-System Patterns and Component Patterns: Managing Platforms and Domain Systems Using PBSE”, INCOSE Great Lakes Regional Conference on Systems Engineering, Schaumburg, IL, October, 2014
37. (Schindel, 2015) Schindel, W., “System Life Cycle Trajectories: Tracking Innovation Paths Using System DNA”, to appear in *Proc. of the INCOSE 2015 International Symposium*, Seattle, WA, July, 2015.
38. (Schindel, Lewis, Sherey, Sanyal, 2015) Schindel, W., Lewis, S., Sherey, J., Sanyal, S., “Accelerating MBSE Impacts Across the Enterprise: Model-Based S*Patterns”, to appear in Proc. of INCOSE 2015 International Symposium, July, 2015.
39. (Schindel, Peffers, Hanson, Ahmed, Kline, 2011) Schindel, W., Peffers, S., Hanson, J., Ahmed, J., Kline, W., “All Innovation is Innovation of Systems : An Integrated 3-D Model of Innovation Competencies ”, *Proc. of ASEE 2011 Conference,* American Association for Engineering Education, (2011).
40. (Schindel, Peterson, 2013) Schindel, W., and Peterson, T. “Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques”, in Proc. of INCOSE 2013 International Symposium, Tutorial, June, 2013.
41. (Schindel, Smith, 2002) Schindel, W., and Smith, V., “Results of applying a families-of-systems approach to systems engineering of product line families”, SAE International, Technical Report 2002-01-3086 (2002).
42. (Sherey, 2006), Sherey, J., “Capitalizing on Systems Engineering”, in Proc. of the INCOSE 2006 International Symposium, July, 2006.
43. (Simmons, 1963) Simmons, George F., *Introduction to Topology and Modern Analysis*, Chapter 10: Hilbert Spaces, McGraw-Hill, 1963.
44. (U’Ren, 2003) U’Ren, J., “An Overview of AP233: STEP’s Systems Engineering Standard”, ISO 10303 AP233 Working Group, October 20, 2003.

Biography

Bill Schindel is president of ICTT System Sciences (www.ictt.com), a systems engineering company. His 40-year engineering career began in mil/aero systems with IBM Federal Systems, Owego, NY, included service as a faculty member of Rose-Hulman Institute of Technology, and founding of three commercial systems-based enterprises. He has led and consulted on improvement of engineering processes within automotive, medical/health care, manufacturing, telecommunications, aerospace, and consumer products businesses. Bill has led the development and practice of Systematica Methodology for Pattern-Based Systems Engineering. He earned the BS and MS in Mathematics. At the 2005 INCOSE International Symposium, he was recognized as the author of the outstanding paper on Modeling and Tools, co-led a 2013 research project on the science of Systems of Innovation within the INCOSE System Science Working Group, and currently co-leads the Patterns Challenge Team of the OMG/INCOSE MBSE Initiative. Bill is an INCOSE CSEP, and president of the Crossroads of America INCOSE chapter.