2015 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
SYSTEMS ENGINEERING (SE) TECHNICAL SESSION
AUGUST 4-6, 2015 - NOVI, MICHIGAN

Re-Uniting Decision Analysis with Systems Engineering: Explicating System Value through First Principles

Troy Peterson
Technical Fellow
Chief Engineer
Booz Allen Hamilton
Troy, MI 48084

Bill Schindel
President
ICTT System Sciences
Terre Haute, IN 47803

ABSTRACT

System complexity continues to grow, creating many new challenges for engineers and decision makers. To maximize value delivery, amidst this complexity, "both" Systems Engineering and Decision Analysis capabilities are essential. For well over a decade the systems engineering profession has had a significant focus on improving systems engineering processes. While process plays an important role, the focus on process was often at the expense of foundational engineering axioms and their contribution to system value. As a consequence, Systems Engineers were viewed as process shepherds which diluted their technical influence on programs. With the recent shift toward Model Based Systems Engineering (MBSE) the Systems Engineering discipline is "getting back to basics," focusing on value delivery via foundational engineering axioms built upon first principles, using established laws of engineering and science. This paper will share how Pattern Based Systems Engineering (PBSE), as outlined within INCOSE's Model Based Systems Engineering (MBSE) initiative, is a methodology which explicates system value through an understanding and explicit modeling of first principles, better re-uniting Systems Engineering and Decision Analysis capabilities.

INTRODUCTION

Complexity in Systems and Decisions

System complexity is growing at an unsettling rate. In attempt to describe these new levels of complexity, labels such as System of Systems, Collaborating Systems, Cyber Physical Systems and others are often used. Today, the scope of engineering efforts often rapidly expands to include more and more external interactions. Additionally within a defined system boundary, systems are becoming significantly more interconnected. Collectively this is accelerating the number of interactions engineers need to understand and manage. The increase in system complexity and associated challenges show no sign of abatement as shown in Figure 1 which depicts the explosion of the Internet of Things (IoT). IoT is a significant contributor to the increase in connectedness and system complexity, and we are still only in the formative stages of this exponential growth. Furthermore, this interconnected phenomena is ubiquitous, occurring across domains and with systems we use every day.

In addition to the increased density of interactions, the pace of contextual change is also increasing. The contextual dynamics have the effect of continually altering a systems fitness and value. This further complicates matters, adding the challenge to design into systems the necessary flexibility and agility, giving rise to a more stochastic view of design rather than what once may have been a more steady state, deterministic perspective.

This context obviously brings about many challenges for engineers and decision makers, which extend beyond the technical domain. Given the complexity and web of interactions, a decision that may appear simple at first could have significant strategic, social, political and economic impact. Where an engineer or manager's intuition may have been sufficient decades ago – today, when trying to consider of 2nd, 3rd, and 4th order impacts, the complexity can quickly overwhelm any one person or even a highly capable team very quickly.

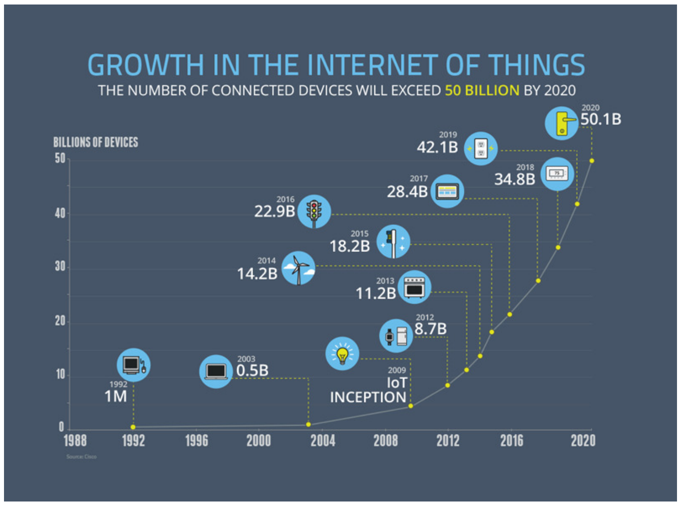

Figure 1: Explosive Growth in the Internet of Things

In his book *Notes on the Synthesis of Form*[1], Christopher Alexander articulated this context eloquently over 50 years ago. The bullets below are excerpts from his book:

- *Today more and more design problems are reaching insoluble levels of complexity*
- *At the same time that problems increase in quantity, complexity and difficulty, they also change faster than before*
- *Trial-and-error design is an admirable method. But it is just real world trial and error which we are trying to replace by a symbolic method. Because trial and error is too expensive and too slow*

These bullets are likely more applicable today than they were 50 years ago and they will likely be even more applicable 50 years from now. Consequently, approaches which leverage symbolic method, speed iterations, build in agility and ensure a holistic view when making decisions are essential on complex engineering programs. One important aspect of ensuring our methods emphasize such results is to better couple the decision making and innovation processes and related models.

One might at first assume that this sets up a rivalry between symbolic model-based analysis and simulation versus waiting for post-development market judgment. However, the Agile System Life Cycle Pattern reminds us of the limits of symbolic models and provides a "middle way": Using "the market" throughout the development cycle, moving "who makes the decisions" of development-time Decision Analysis, to include the ultimate decision-maker—the stakeholder.

RE-UNITING DECISION ANALYSIS WITH SYSTEMS ENGINEERING

Many frameworks group, categorize or connect Decision Analysis with Systems Engineering. This is true within the overview of System Engineering provided by the Defense Acquisition University shown in Figure 2, and with the INCOSE Handbook as shown in Figure 3.

The Defense Acquisition University states that the decision analysis process transforms a broadly stated decision opportunity into a traceable, defendable, and actionable plan. Furthermore, that it is performed at all systems levels and across the life cycle. The DAU outlines Decision Analysis integration specifically with the process areas of Technical Planning, Assessment, Stakeholder Requirements, Requirements Analysis and Architecture Design all shown in Figure 2. INCOSE also notes the Decision Management Process, which includes Decision Analysis, integrates with all other SE processes in its System Life-Cycle Process N^2 Chart found in the Appendix A of the Systems Engineering Handbook V3.2.2. Figure 3 provides a view of the system life cycle processes aligned with ISO 15288 and INCOSE's Systems Engineering Handbook.

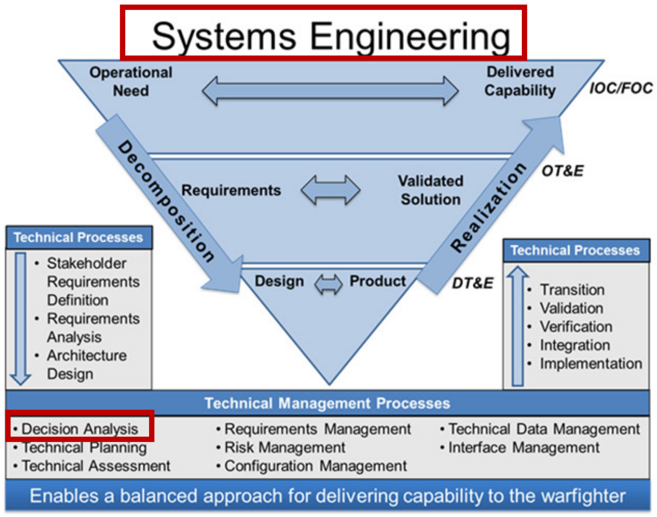

Figure 2: DAU Systems Engineering Diagram

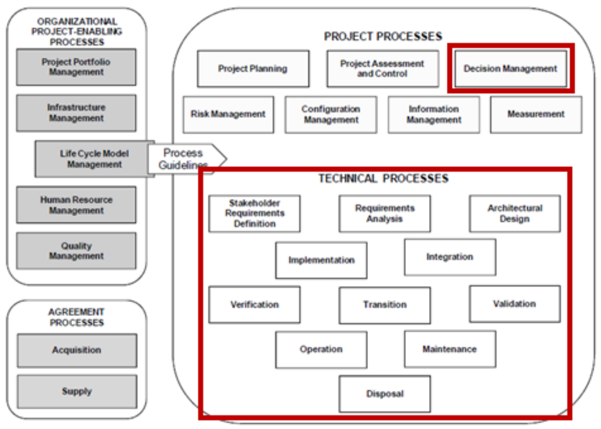

Figure 1-1 System Life-cycle Processes Overview per ISO/IEC 15288:2008

Figure 3: INCOSE System Life-Cycle Processes Overview per ISO 15288[2].

While these views and their associated processes play an essential role in engineering complex systems, the focus on the Systems Engineering process was often at the expense of some foundational engineering axioms. As a consequence, Systems Engineers were often viewed as overhead creating processes and ensuring compliance to these processes. This diluted their technical influence on programs and brought the value systems engineering into question.

For well over a decade the systems engineering profession has had a significant focus on improving systems engineering processes – as illustrated by CMMI and ISO 15288. Connections between the SE and DA exist at a high level as shown in Figures 2 and 3 as well as within many more detailed process architectures. These connections are important and help program teams manage the complex system of innovation. However, there is a deeper need in connecting the disciplines, both more deeply and in a more explicit way to ensure value delivery.

Models of Process vs Models of Systems

Process integration is helpful, but alone it is not sufficient to manage the complexity in systems today--in fact it can become nearly impossible to avoid unintended consequences without detailed models of the system of interest. Much of the integration effort of SE and DA has been focused on process – the infrastructure for information about the system of interest. It has not been, however, as focused on the information that passes through the process about the system of interest.

With the recent shift toward Model Based Systems Engineering (MBSE), the Systems Engineering discipline is "getting back to basics" and potentially back to the foundational engineering axioms built upon first principles and established laws of science and engineering. This focus is more aligned with the genesis of classical mechanics, beginning with Newtonian interactions and their emergent properties, so that the whole is greater than the sum of the parts.

Using models to explicate the translations of first principles to stakeholder value is accomplished in the Agile System Life Cycle Pattern, which expresses stakeholder value as demonstrated by selection interactions. . To obtain this benefit, our models must represent the expanded view of stakeholders and external value domains and ensure that technical and non-technical objectives co-exist to ensure a holistic view of system value.

History and a call for a new view

The history of Systems Engineering has strong ties to fundamental engineering disciplines, the sciences and to mathematical modeling and managerial decision support (management sciences) - often referred to as decision analysis, industrial engineering or operations research. So in many ways a discussion of how to integrate these disciplines is a return to the early foundation of systems engineering.

To help address the complexity outlined in the introduction and to better re-integrate Systems Engineering (SE) and Decision Analysis (DA) many efforts are underway within industry, the government and non-profits. For example, a working group within the International Council on Systems Engineering (INCOSE) focuses on Decision Analysis with the purpose of advancing the state of the practices, education and theory of Decision Analysis and its relationship to other systems engineering disciplines. The Council of Engineering Systems Universities (CESUN) is another example that collaborates with both INCOSE and the Institute for Operations Research and the Management Sciences (INFORMS), which was formed to address the great challenges posed by large-scale, interconnected, and therefore highly complex and dynamic, socio-technical systems. The excerpt from the CESUN website which follows articulates the contributions of SE and DA to Engineering Systems.

As many engineers began to delve deeper and deeper into science, some others stressed the design perspective and explored how to solve the problems arising from greater technical complexity. Operations research, systems and decision analysis, industrial engineering, systems engineering—these all contributed to the expansion of engineering—but at a certain point there was a recognition that some of the greatest challenges were precisely where the technical systems had their interfaces with people, policies, regulations, culture, and behavior. [3]

This excerpt also calls out the expanded and new view at the "…interfaces with people, policies, regulations, culture and behavior." This perspective brings with it a diverse set of stakeholders and an expanded view of value. To achieve value delivery in this new view we must have an improved coupling of Systems Engineering and Decision Analysis. The disciplines are absolutely complementary with Systems Engineering providing an overall approach to systematically innovate and Decision Analysis providing a systematic approach to think about, experiment with, and analyze complex problems or opportunities throughout the innovation process. To fully integrate these disciplines the third bullet from Alexander noted above makes an important observation about the use of "…symbolic method. Because trial and error is too expensive and slow." This brings us first to the use of models and model based systems engineering (the symbolic part) and then to the Agile Systems Engineering Life Cycle Pattern (the sped-up "trial and error" part).

MODEL-BASED SYSTEMS ENGINEERING (MBSE)

INCOSE defines Model-Based Systems Engineering (MBSE) as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases…[4]" The Object Management Group's MBSE wiki notes that "Modeling has always been an important part of systems engineering to support functional, performance, and other types of engineering analysis.[5]"

The application of MBSE has increased dramatically in recent years and is becoming a standard practice to help manage the complexity seen in systems today. MBSE has been enabled by the continued maturity of modeling languages such as SysML and significant advancements made by tools vendors. These advancements are improving communications and providing a foundation to integrate diverse models. MBSE is often discussed as being composed of three fundamental elements – tool, language and method. The third element, method, however has not always been given proper consideration. Because the language and tool are relatively method independent, it is methodology which further differentiates the effectiveness of any MBSE approach and its ability to help manage the complex and interrelated functionality of today's highly interconnected systems. For the approach discussed in this paper, the "methodology" includes not only process as discussed in the previous section in accordance with ISO 15288, INCOSE, DAU or others, but more significantly the very concept of the underlying system information those processes produce and consume, independent of modeling language and tools.

PATTERN-BASED SYSTEMS ENGINEERING

Pattern Based Systems Engineering (PBSE) as outlined within INCOSE's Model Based Systems Engineering (MBSE) initiative[6], is a methodology which formalizes historical pattern efforts using explicit, re-usable, configurable S*Models (S*Patterns). Moreover, it explicates system value through an understanding of system interactions. Pattern-Based Systems Engineering (PBSE) can address 10:1 more complex systems with 10:1 reduction in modeling effort, using people from a 10:1 larger community than the "systems expert" group, producing more consistent and complete models sooner. These dramatic gains are possible because projects using PBSE get a "learning curve jumpstart" from an existing pattern and its previous users, rapidly gaining the advantages of its content, and improving the pattern with what is learned, for future users. The major aspects of PBSE have been defined and practiced for many years across a number of enterprises and domains. To increase awareness of the PBSE approach, two years ago INCOSE started a Patterns Challenge Team (now the Patterns Working Group) within the INCOSE MBSE Initiative[7].

The term "pattern" appears repeatedly in the history of design, such as civil architecture[8], software design[9], and systems engineering[10]. These are all similar in the abstract, in that they refer to regularities that repeat, modulo some variable aspects, across different instances in space or time. However, the PBSE methodology referred to by this paper is distinguished from those cases by certain important differences:

1. S*Patterns are Model-Based: We are referring here to patterns represented by formal system models, and specifically those which are re-usable, configurable models based on the underlying S*Metamodel. (By contrast, not all the historical "patterns" noted above are described by MBSE models.)

2. Scope of S*Patterns: We are referring here to patterns which will usually cover entire systems, not just smaller-scale element design patterns within them. For this reason, the typical scope of an S*Pattern applications may be thought of as re-usable, configurable models of whole domains or platform systems—whether formal platform management is already recognized or not. (By contrast, most of the historical "patterns" noted above describe smaller, reusable subsystem or component patterns.)

S*Patterns are similar to architectural frameworks, although they contain more information.

Fundamental to Pattern-Based Systems Engineering is the use of the S*Metamodel (summarized by Figure 4), a relational / object information model used in the Systematica™ Methodology to describe requirements, designs, and other information in S*models such as verification, failure analysis, etc.[11.12.13.14.15.16.17]. A metamodel is a model of other models—a framework or plan governing the models that it describes. These may be represented in SysML™, database tables, or other languages. As an MBSE enabled approach PBSE can be implemented across multiple third party COTS tools and languages (i.e. PLM systems, modeling tools, architecture tools, databases, SysML, IDEF0, et al.)

Specifically, an S*Pattern is a re-usable, configurable S*Model of a family of systems (product line, set, ensemble etc.) as shown in Figure 5 below.

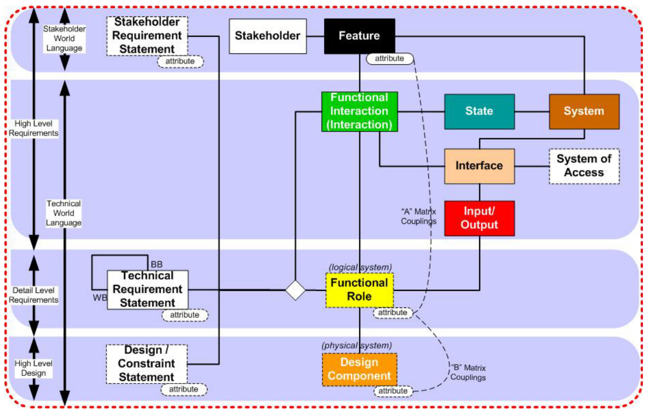

Figure 4: A summary view of the S*Metamodel

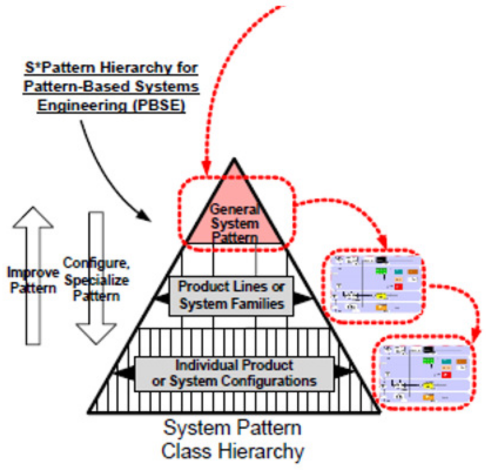
S*Pattern Hierarchy for Pattern-Based Systems Engineering (PBSE)
General System Pattern
Improve Pattern
Configure, Specialize Pattern
Product Lines or System Families
Individual Product or System Configurations
System Pattern Class Hierarchy

Figure 5: Pattern Hierarchy for PBSE

Over several decades, Pattern-Based Systems Engineering has been developed and practiced across a range of domains, including carrier grade telecommunications, engines and power systems, automotive and off road heavy equipment, telecommunications, military and aerospace, medical devices, pharmaceutical manufacturing, consumer products, and advanced manufacturing systems[18.19.20].

Engineers in these and many other domains spend resources developing or supporting systems that virtually always include major content from repeating system paradigms at the heart of their business (e.g., core ideas about airplanes, engines, switching systems, etc.). In spite of this, the main paradigm apparent in most enterprises to leverage "what we know" is to build and maintain a staff of experienced technologists, designers, application engineers, managers or other human repositories of knowledge.

The physical sciences are based upon the discovery of regularities (patterns), which we say express laws of both nature and systems value markets. Although re-usable content has some history in systems engineering, there is less recognition of a set of "Maxwell's Equations" or "Newton's Laws" expressing the nature of the physical world, as the basis of those systems patterns. If Electrical Engineering and Mechanical Engineering disciplines have physical law at their foundation, why cannot Systems Engineering do the same?

By contrast, the S*Metamodel is focused on the very physical interactions that are the basis of the physical sciences, and which we assert are at the heart of the definition of System (in this methodology) as a collection of interacting components[21]. The S*Patterns that arise from the explicit representation of physical Interactions re-form the foundation of system representations to align more explicitly with the physical sciences.

At its very foundation, the ASELCM Pattern of PBSE links decision analysis and systems engineering ensuring system configurations are directly traceable and driven by stakeholder values. PBSE explicates system value via a formal model of interactions, whether force, mass flow, energy or information exchanges which are foundational to science and to the first principles of system design and market responses.

SYSTEM VALUE – Stakeholder Features

System value is measured by the selection interactions of stakeholders or their representatives; in the S* metamodel these values are expressed explicitly as Features. In the ASELCM Pattern, these selections are as explicit as the (other) interactions of the system of interest. Features and their associated attributes contain the value space for a system of interest codified as formalized stakeholder needs/values. The connection between Stakeholders and Features is clear within the S* metamodel shown in Figure 4. Features are shown at the top of the figure using a black box. Figure 6 reformats and displays just a portion of the S* metamodel clearly annotating the classes from which we derive system value. Features are parameterized by Feature attributes which provide a measure of value – including all stakeholder measures of effectiveness (MOEs)[22]. Within Figures 4 and 6 these Feature attributes are represented by a white elongated oval adjacent to the black Feature box.

As outlined in the introduction, just as the system boundary has broadened, the set of stakeholders and their respective values must also be broadened. It is important to note that stakeholders include all classes of stakeholders and not just those who may purchase or use a product or system of interest. Stakeholders include shareholders, manufacturers, society, et al. Every trade off or decision which sets the direction of a system design is a value judgment (selection interaction) from the perspective of one or more stakeholders. Given this view it is absolutely necessary to have a holistic view and identify the full complement of stakeholders. In fact, it is the omission of stakeholders early in a systems program that often leads to costly rework, redesign, failures in system validation and sometimes program cancelation. When Feature space is mature and expansive it can significantly reduce technical and programmatic risk. While ensuring the set of stakeholders is comprehensive it should not be assumed however that all stakeholders and their associated values are equal.

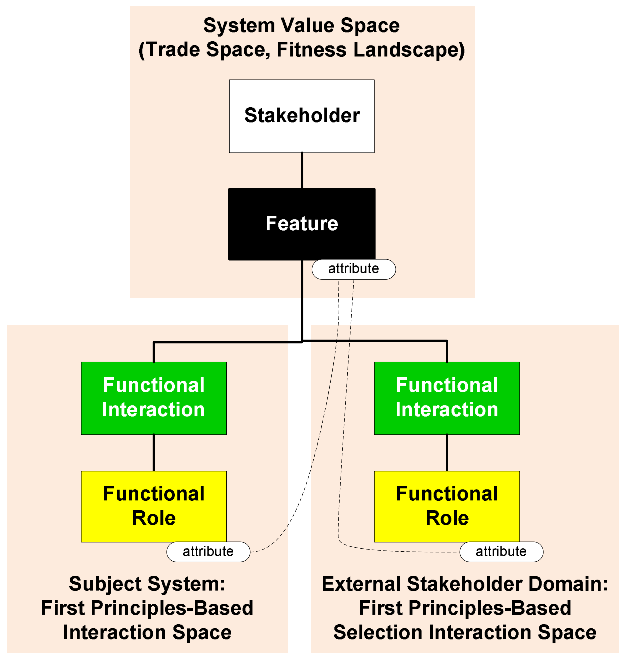
System Value Space
(Trade Space, Fitness Landscape)
Stakeholder
Feature
attribute
Functional Interaction
Functional Role
attribute
Functional Interaction
Functional Role
attribute
Subject System: First Principles-Based Interaction Space
External Stakeholder Domain: First Principles-Based Selection Interaction Space

Figure 6: S*Agile Systems Engineering Life Cycle Pattern extract, highlighting System Value which is generated via interactions - the first principles of engineering and science

Since feature space contains the full complement of stakeholder values (the fitness landscape) it contains the entire trade space for the design and development of systems. This includes the full breadth and hierarchical depth of value including objectives and measures, weights and rationale prescribed in my texts focused on Decision Analysis (Keeney[23], Clemen-Reilly, the Defense Acquisition Guidebook 4.3.3 et al.). With Stakeholders and their Features well understood the Features are used to configure systems that conform to the selections and the dialing in of their associated attributes.

Feature space is where selection-based decision analysis occurs, it's used as the basis of analysis and defense of all decision-making including optimization and trade-offs[22]. This gives rise to the next class of information in Figures 4 and 6 which delivers system value - Functional Interactions.

FIRST PRINCIPLES - Functional Interactions

Functional Interactions are what define a system (a group of interacting, elements forming a complex whole) and through which the system delivers value. Functional interactions involve the exchange of forces, mass, energy or information. When we think of these fundamental exchanges, it brings to mind the work one would become intimately familiar in physics, chemistry, mechanics and many other engineering, science, or mathematics courses. These exchanges return us to the first principles of these disciplines and how they apply to the systems we design and develop. Additionally, as our understanding grows within a particular domain or with a specific type of system we often begin to learn the first principles of these systems which are also expressed as interactions. These interactions can be at the component, subsystem or system level, and especially with the external environment of a system of interest.

Alternatives as outlined within Decision Analysis are the options to evaluate against decision criteria or the objectives and measures. Given that Functional Roles, displayed as the yellow box in Figures 4 and 6, are solution neutral logical roles which participate in an interaction. An identified Functional Interaction may be implemented by various combinations of functional roles. This gives rise to many alternatives when making traditional functional allocations.

In Figure 6 the Feature and Functional Role attributes are coupled as shown by the dotted line. This particular coupling qualifies the fitness or trade space. The Feature attribute defines the measure of effectiveness and the Functional Role attribute provides a means and measure of value delivery (level of performance) depending upon the selection of design components filling the Functional Role.

Parameterization and Configuration

S*Models are intended to establish modeled Feature sets for all Stakeholders. This (Features) portion of an S*Pattern is then used to configure the pattern for individual applications, product configurations, or other instances. It turns out that the variation of configuration across a product line is always for reasons of one stakeholder value or another, so Feature selection becomes a proxy for configuring the rest of an S*Pattern into a specifically configured instance model.

Because S*Features and their Feature Attributes (parameters) characterize the value space of system stakeholders, the resulting S*Feature Configuration Space becomes the formal expression of the trade space for the system. It is therefore used as the basis of analysis and defense of all decision-making, including optimizations and trade-offs. The S*Feature Space also becomes the basis of top-level dashboard model views that can be used to track the technical status of a project or product. All "gaps" and "overshoots" in detailed technical requirements or technologies are projected into the S*Feature Space to understand their relative impact.

As illustrated by the "down stroke" in Figure 5, a generic S*Pattern of a family of systems is specialized or "configured" to produce an S*Model of a more specific system, or at least a narrower family of systems. Since the S*Pattern is itself already built out of S*Metamodel components, for a mature pattern the process of producing a "configured model" is limited to two transformation operations:

1. Populate: Individual classes, relationships, and attributes found in the S*Pattern are populated (instantiated) in the configured S*Model. This can include instances of Features, Interactions, Requirements, Design Components, or any other elements of the S*Pattern. These elements are selectively populated, as not all necessarily apply. In many cases, more than one instance of a given element may be populated (e.g., four different seats in a vehicle, five different types of safety hazard, etc.). Population of the S*Model is driven by what is found in the S*Pattern, and what Features are selected from the S*Pattern, based on Stakeholder needs and configuration rules of the pattern, built into that pattern.

2. Adjust Values of Attributes: The values of populated Attributes of Features, Functional Roles/Technical Requirements, and Physical Components are established or adjusted.

This brings into sharp focus what are the fixed and variable aspects of S*Patterns (sometimes also referred to as "hard points and soft points" of platforms). The variable data is called "configuration data". It is typically small in comparison to the fixed S*Pattern data. Since users of a given S*Pattern become more familiar over time with its fixed ("hard points") content (e.g., definitions, prose requirements, etc.), this larger part is typically consulted less and less by veterans, who tend to do most of their work in the configuration data (soft points). That data is usually dominated by tables of attribute values, containing the key variables of a configuration. Since this is smaller than the fixed part of the pattern, in effect the users of the pattern experience a "data compression" benefit that can be very significant, allowing them to concentrate on what is or may be changing. (Schindel, 2011).

Just as feature attributes parameterize stakeholder values, functional role attributes parameterize technical behavior. The coupling of these attributes shown in Figures 4 and 6 provides a model based approach to coupling the first principles of engineering and science with stakeholder value. It's through this coupling that Pattern Based Systems Engineering explicates system value through first principles.

The Agile System Life Cycle Pattern

INCOSE is currently executing the 2015-16 Agile Systems Engineering Life Cycle Model (ASELCM) Project. Working across a series of North American and European enterprises and industries, this discovery project is articulating and validating the ASELCM Pattern mentioned in this paper, in the form of a formal S*Pattern.

The ASELCM Pattern explicates the points summarized in this paper, including:

1. The deeper re-integration of DA and SE, with the decisions shared between "internal" decision-makers and agile-measured "external" stakeholder representatives, whose selection behaviors are studied as a faster and surer path to good decisions.
2. The use of explicit MBSE Models to express life cycle system requirements, design, generated from MBSE Patterns by configuration and reconfiguration, as the environment changes in non-deterministic ways, and as a point of accumulation of learning.

CONCLUSIONS

System complexity and interconnectedness continues to rapidly increase making systems development extremely challenging. Additionally the context in which developed systems operate is continually changing altering the fitness and value delivered systems provide. For over a decade the Systems Engineering discipline has made many great improvements through process definition and integration. While these improvement have enabled and structured innovation they are not sufficient to overcome the outlined challenges which are likely to only increase over time. Our traditional development activities must be revisited and enhanced to manage significant complexity, nth order impacts, highly dynamic contexts, complicated decisions and significant ambiguity.

An important aspect to an improved approach is to better integrate the Decision Analysis and Systems Engineering and to leverage “symbolic method” (to the extent that symbolic analysis and simulation are sufficient) while also improving ability to capture stakeholder and market judgments without undue delay (to the extent that empirical experiment is also required). This leads us to modeling methods and the promise provided by Model Based Systems Engineering. As a particular MBSE methodology PBSE is particularly well suited to model complex systems. With interactions at the core of is S* metamodel PBSE focuses the engineering effort on how systems fundamentally provide value. It couples system value, expressed by Stakeholders as Features, with the first principles of engineering and science, expressed as Functional Interactions, making for a strengthened Systems Engineering approach. This approach also shifts the focus from the innovation process to the information passing through the process which describes the system of interest which ultimately determines the level of value provided to stakeholders. The tight coupling within the modeling approach permits rapid iteration, configuration, assessment and analysis.

PBSE provides a data model and framework that is both holistic and compact. It addresses the core system science or first principles of systems required to design complex systems by making interactions more visible and directly relating these to how they deliver value described by stakeholders, noted as features in the S* metamodel. Additional benefits of the PBSE approach include:

- Strong expression of fitness landscapes as the basis for selection, trades, improvements, decisions, innovations, configuration, and understanding of risk and failure.
- Explication of the System Phenomenon as a real world-based science and math foundation for systems engineering, amenable to systems science, connected to historical math/science models of other engineering disciplines, and encouraging discovery and expression
- A detailed MBSE approach to Platform Management for system families and product lines.
- Compatibility with contemporary modeling language standards.
- Direct mapping into contemporary modeling tools, PLM information systems, and other COTS tools and enterprise systems, increasing the value of existing information technologies.
- Deeper support for federated data across differing information systems, for integration with emerging open systems life cycle standard technologies.

Pattern Based Systems Engineering (PBSE) is a methodology which explicates system value through an understanding and explicit modeling of first principles better uniting the Systems Engineering and Decision Analysis capabilities.

REFERENCES

[1] Christopher Alexander, “Notes on the Synthesis of Form” Harvard University Press, Cambridge Massachusetts, 1964
[2] https://acc.dau.mil/CommunityBrowser.aspx?id=638297
[3] http://cesun.mit.edu/about/purpose
[4] INCOSE SE Vision 2020
[5] http://www.omgwiki.org/MBSE/doku.php?id=start
[6] http://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns
[7] http://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns
[8] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. A Pattern Language. Oxford University Press, New York, 1977.
[9] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company, Reading, MA, 1995.
[10] Robert Cloutier. Applicability of Patterns to Architecting Complex Systems: Making Implicit Knowledge Explicit. VDM Verlag Dr. Müller. 2008.
[11] Bill Schindel, Troy Peterson, “Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques”, in Proc. of INCOSE 2013 Great Lakes Regional Conference on Systems Engineering, Tutorial, October, 2013.
[12] W. Schindel, “System Interactions: Making The Heart of Systems More Visible”, in Proc. of INCOSE Great Lakes 2013 Regional Conference on Systems Engineering, October, 2013.
[13] Abbreviated Systematica Glossary, Ordered by Concept, V 4.2.2, ICTT System Sciences, 2013.
[14] W. Schindel, “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.
[15] W. Schindel, “Failure Analysis: Insights from Model-Based Systems Engineering”, in Proceedings of INCOSE 2010 Symposium, July 2010.
[16] W. Schindel, “Pattern-Based Systems Engineering: An Extension of Model-Based SE”, INCOSE IS2005 Tutorial TIES 4, (2005).
[17] W. Schindel, “Requirements Statements Are Transfer Functions: An Insight from Model-Based Systems Engineering”, in Proc. of INCOSE 2005 International Symposium, (2005).
[18] W. Schindel, and V. Smith, “Results of Applying a Families-of-Systems Approach to Systems Engineering of Product Line Families”, SAE International, Technical Report 2002-01-3086 (2002).
[19] J. Bradley, M. Hughes, and W. Schindel, “Optimizing Delivery of Global Pharmaceutical Packaging Solutions, Using Systems Engineering Patterns”, in Proc. of the INCOSE 2010 International Symposium (2010).
[20] W. Schindel, “Integrating Materials, Process & Product Portfolios: Lessons from Pattern-Based Systems Engineering”, in Proc. of 2012 Conference of Society for the Advancement of Material and Process Engineering, 2012.
[21] W. Schindel, “What Is the Smallest Model of a System?”, in Proc. of the INCOSE 2011 International Symposium, International Council on Systems Engineering (2011).
[22] W. Schindel, “Pattern Based System Engineering Methodology” MBSE Initiative, Methodology Summary for INCOSE June 2015.
[23] R.L. Keeney. Value-Focused Thinking — A Path to Creative Decision Making. Harvard University Press, Cambridge, MA, 1992.