Formulation and Development of the Wasson System Engineering Process Model

Charles S. Wasson[1]

ABSTRACT - During the past several decades, numerous attempts have been made to capture and communicate the highly iterative and recursive characteristics of the System Engineering process via graphical models. Examples include: the US Army FM 770-78, USAF AFSCM 375-5, MIL-STD-499, IEEE 1220, et al. Models to date: 1) are often considered abstract and 2) fail to include the essential elements that serve as the foundation for multi-disciplinary design collaboration and decision-making. To overcome and correct these deficiencies, this paper presents a proven System Engineering Process model that is explicit, easy to understand and communicate, and applicable to multi-discipline engineering design. The model is based on a robust, highly iterative and recursive, problem-solving / solution-development methodology. The paper concludes with a summary of benefits that will significantly strengthen the System Engineering element of engineering education including capstone projects, produce engineering graduates who are better prepared to enter industry and perform multi-disciplinary System Engineering (SE), and enhance the university's reputation for graduates who are highly recognized and recruited by industry, government, and academia.

Keywords: Systems Engineering, system engineering process, system development paradigm, problem solving / solution development methodology, multi-discipline engineering design, and capstone projects.

INTRODUCTION

If you ask most engineers to define an engineering problem-solving / solution-development method, many will respond with the Scientific Method or some form of scientific inquiry process. The responses: 1) reflect how we are educated, beginning in middle school general science classes through high school chemistry and physics, to think in terms of scientific inquiry, investigation, and experimentation and 2) highlight the Systems Engineering void in engineering education addressed by Wasson [1].

Problem-solving / solution-development concepts beyond those of scientific inquiry, investigation, and experimentation are seldom presented as a part of many engineering curricula. In fact, few engineering programs and curricula offer a course in Systems Engineering Fundamentals. Unfortunately, when some courses labeled as System Engineering are taught, they only address fragments of what is required to successfully perform and accomplish Systems Engineering as a problem-solving / solution-development methodology-based discipline.

In general, most engineering course problems focus on discipline solutions that require application of mathematical and scientific concepts, principles, and methods for solving boundary condition problems for a single entity. As a result, engineering graduates enter industry or government to engineer complex, multi-level, systems lacking an efficient and effective problem solving / solution development methodology for transforming abstract user requirements into the physical realization of multi-level systems, products, or services. Capstone project courses attempt to solve this challenge as illustrated in Figure 1. Capstone project plans reflect all of the tenets of SE activities; however, what is being accomplished is SE Management, not Systems Engineering.

To solve this problem, methodology-based models or paradigms for "engineering systems", namely Systems Engineering (SE), have evolved since World War II. Analysis of these models reveals that the developers often intermixed project management system development model paradigms with the SE Process model. For example, Figure 2 is commonly referred to as a SE Process Model. However, the model depicted is actually the V-Model for system development [2] used by technical project management as a strategic roadmap for translating user

[1] ASEE, INCOSE, and PMI; www.wassonstrategics.com, Email: wslse@cpws.net

requirements into the deliverable system; planning work activities; and evaluating and reporting technical program performance – i.e., status, progress, and risk. The V-Model is one example of several system development models and approaches such as spiral development, incremental and evolutionary development, et al that can be employed.

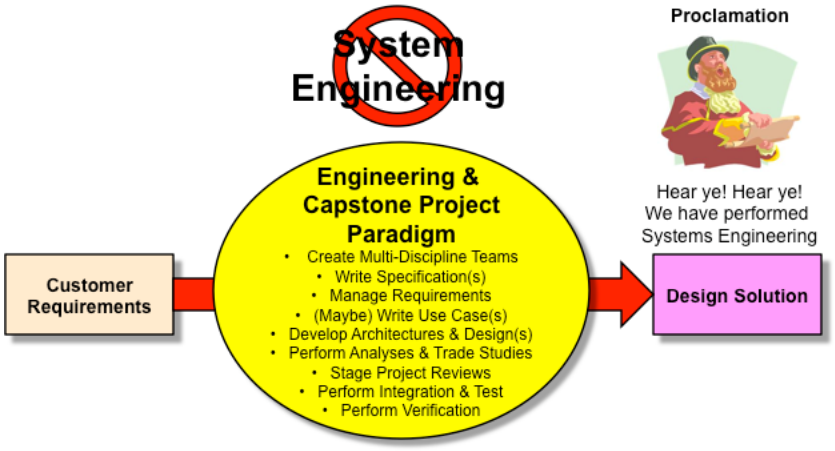
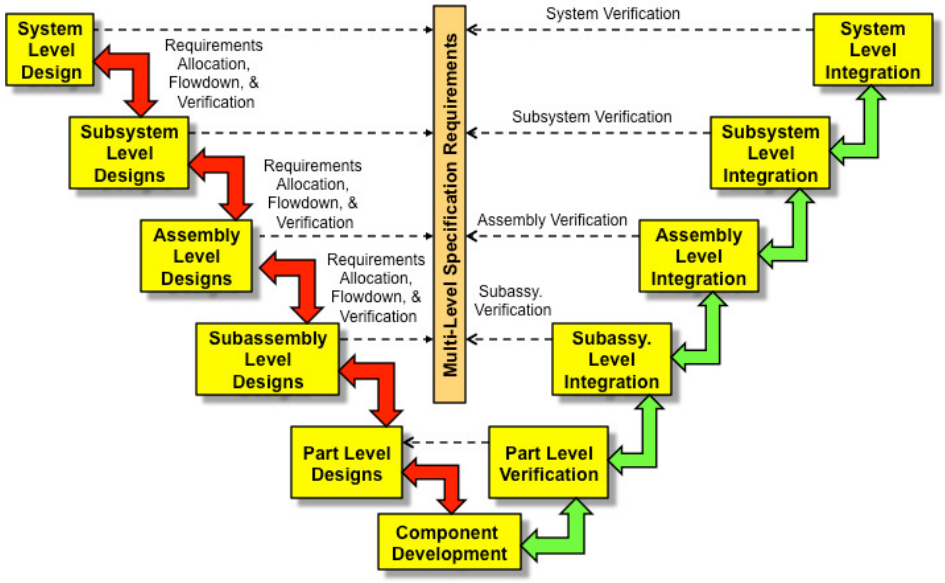


Figure 1: SE Management Paradigm incorrectly labeled as Systems Engineering.

An SE Management paradigm does not provide the problem-solving / solution-development methodology required to enable teams of multi-discipline engineers to: 1) transform abstract contract or task requirements of a multi-level system through its levels of abstraction; 2) derive, allocate, and flow down capability requirements at each level; 3) formulate, select, and mature system / entity architectures and designs; 4) select components; and 5) verify and validate entities and work products at each level, into a fully integrated end item deliverable for customer acceptance.

Figure 2: System Development V-Model Paradigm [2]



Historically, SE process models, each a variant of its predecessors, emerged over several decades. These include:

- 1966 - USAF AFSCM 375-5 [3]
- 1969 - US Army FM 770-78 [4]
- 1969 - USAF MIL-STD-499 [5]
- 1974 - USAF MIL-STD-499A [6]
- 1994 - USAF MIL-STD-499B DRAFT [7]
- 1999 - ANSI / EIA 632-1998 [8]
- 1994 - ANSI / IEEE Std. 1220-1994 [9]

The infrastructures of these SE Process paradigm models focus on activities such as requirements analysis, functional analysis, synthesis (of design solutions), and system analysis and control. Today, advancements in multi-discipline SE – i.e., electrical, mechanical, software engineering, et al – require a paradigm shift to essential aspects of system development. Examples include: 1) understanding the problem, issue, or concern to be solved, 2) identification of users and stakeholders concerning how they intend to employ the system – e.g., use cases and scenarios - to fulfill organizational missions, 3) recognition of deficiencies in functional versus capability analysis, 4) the need to model system behavioral outcomes and performance – e.g., Model-Based Systems Engineering (MBSE), et al – as an integral activity to multi-level specification requirements derivation, analysis, allocation, and flow down. Whereas engineers are often educated and trained that functional analysis is a core activity of SE, it is a *necessary* but *insufficient* activity to fulfill the contexts of SE as a discipline and as a methodology for multi-disciplinary application to the "engineering of systems."

Current SE Process paradigm models have evolved and matured over time. However, model activities such as functional analysis and synthesis are often viewed as abstract, difficult to understand, and viewed as unnecessary due to a lack of emphasis in engineering courses, namely the fundamentals of Systems Engineering. Given this educational deficiency, projects in some industry organizations evolve into *ad hoc*, *inefficient* and *ineffective*, implementations that impact project technical compliance, budgets, schedules, and risk performance. Several factors such as technical leadership, plans, resources, tools, et al contribute to these results. Technically, the inefficiencies and ineffectiveness of ad hoc engineering methods contribute to the undeserved stigma applied to engineers that exceed minimum requirements by "gold plating" designs, "never knowing when to quit", "always late on schedules", "always over budget", etc.

STATEMENT OF THE PROBLEM

Current System Engineering problem-solving / solution-development model paradigms need to be shifted to a new paradigm that: 1) leverages advancements in the state of systems engineering practice, 2) avoids abstract semantics that are difficult to understand and communicate, 3) emphasizes the need to understand user and stakeholder operational needs, 4) provides a common framework for introduction in a fundamentals of SE course and application to capstone projects, and 5) corrects short-cuts in ad hoc, inefficient, and ineffective engineering decision-making paradigms that result in system deficiencies at customer delivery and acceptance.

MODEL SUCCESS CRITERIA

Our mission in formulating and developing a new SE problem-solving / solution development methodology requires more than simply creating another model. We establish success criteria that will address deficiencies in current SE model paradigms and incorporate state of SE practice methods. The success criteria for a new SE decision making paradigm are to:

1. Correct engineering decision making quantum leaps from abstract requirements to physical solutions.
2. Employ a proven, problem-solving / solution development methodology as the model's infrastructure.
3. Provide flexibility, adaptability, and scalability for application to projects that range from small to large.
4. Be independent of but applicable to any engineering discipline or business domain.
5. Be teachable and easily understood by a diverse audience – e.g., engineering students to professionals.

MODEL STRUCTURE DRIVERS

One of the shortcomings of engineering decision-making is the premature leap from customer requirements to a physical solution without due consideration of: 1) WHO the users and stakeholders are; 2) HOW they need to deploy, operate and support, and dispose of the system, 3) WHAT behaviors, outcomes, and levels of performance users expect of the system, 4) WHEN behavioral responses are required and at what levels, and 5) an analysis of alternatives (AoA) of and selection of an optimal solution from a set of viable candidate options.

When engineering students and professionals lack formal education and training in systems engineering [1], engineers instinctively make a "quantum leap" from requirements to a physical solution referred to as a "point solution." As a result, the physical solution may or may not be *verifiable* in meeting system requirements, be deficient in capabilities or performance due to missing or conflicting requirements, or may not satisfy the operational needs of the user – e.g., system *validation* – in conducting their missions. How do we overcome this

condition? We do this by establishing a series of "decisions." These decisions serve as a strategy to develop an optimal system / entity design solution at each level of abstraction selected from a set of viable candidates subject to procurement and system life cycle constraints – i.e., technical, technology, cost, schedule, and acceptable risk. The key decisions are:

Decision #1 – Who are the System's Stakeholders? - Logically, engineering decision-making begins with a clear understanding of who the system / entity stakeholders – e.g., owners, users, end users, et al - and their operational needs and life cycle constraints such as technical, technology, cost, schedule, risk, and success criteria.

Decision #2 – What is the Problem(s), Issue(s), or Concern(s) Stakeholders Need to Solve? - Once the stakeholders are identified, the system developer identifies, bounds, and characterize each problem space. Each problem space is decomposed into one or more solution spaces. Each solution space is then bounded by a set of project requirements and constraints such as operational and technical requirements, technology, development and life cycle operating costs, development schedule, and acceptable levels of risk.

Decision #3 – How Does the User Intend to Employ the System as an Organizational Asset? - As operational and technical requirements and life cycle constraints are identified, engineers investigate how the users intend to deploy, operate and support, and dispose of the system, product, or service to fulfill their organizational missions.

Decision #4 – What Are the System's Required Behavioral Performance Outcomes? - Accomplishment of mission operations requires the time-dependent integration of the user and system capabilities to produce a set of performance-based behaviors to achieve specific outcomes in response to operating environment stimuli or cues.

Decision #5 – What is the Optimal Physical Solution that Satisfies the User's Needs? - Finally, if we understand what behavioral capabilities are required to provide a pre-defined set of system performance-based responses, then what arrangement or configuration of individual or sets of *physical* components and human interactions is required to produce the required performance-based behavioral responses and outcomes?

Synthesizing these decision-making gates into a logical decision-making workflow, the System Engineering problem-solving / solution development model should be responsive to the following precepts:

- Precept #1 – Stakeholder needs provide the basis for deriving a Requirements Domain Solution that bounds and specifies the system operational outcomes, environmental operating conditions, and constraints.
- Precept #2 – The Requirements Domain Solution provides the basis for deriving an Operations Domain Solution that defines how the system is envisioned to be deployed, operated, sustained, and disposed.
- Precept #3 – The Operations Domain Solution provides the basis to derive the Behavioral Domain Solution that characterizes how the system responds to operating environment stimuli, excitations, or cues.
- Precept #4 – The Behavioral Domain Solution provides the basis to derive the Physical Domain Solution that includes selection of physical components and configurations to implement behavioral capabilities.

Summarizing these precepts, a system design solution consists of the assimilation of Four Domain Solutions – i.e., Requirements, Operational, Behavioral, and Physical [10], each consistent with the other and traceable to source or originating requirements. These workflow precepts serve as the basis for the Wasson SE Process Model.

THE WASSON SYSTEM ENGINEERING PROCESS MODEL

Objective 1 Identify and Understand the Problem(s) / Issue(s) / Concern(s) to be Resolved

One of the most important aspects of problem solving and solution development is simply understanding and articulating the problem, issue, or concern the user is attempting to solved. Based on this objective, we establish steps required to identify who the users, end users, and stakeholders are in resolving the problem; understand their perspectives, views, and viewpoints of the problem, and begin to formulate how the problem might be partitioned or decomposed into one or more manageable, acceptable risk, solution spaces that can be solved with existing components, commercial-off-the-shelf (COTS) products, or new development.

Step 1. - Identify and Understand the System / Entity Stakeholders
The first step requires an understanding the answers to three fundamental questions:

a. Who are the system / entity stakeholders – e.g., owners, users, end users, et al?
b. What is their mission: operational needs, objectives, performance-based outcomes, and success criteria?
c. What are the technical, cost, schedule, and risk constraints imposed by the system's users and stakeholders?

Step 2. - Bound the Problem / Issue / Concern to be Solved
Based on the results of Step 1:

a. What is the central problem, issue, or concern to be solved?
b. Is this the root problem / issue / concern or a symptom of a larger problem?
c. Is there more than one problem / issue / concern to be solved?
d. What are its boundaries, relevant interfaces, assumptions, and environmental conditions?

Objective 2 Partition, Bound, and Specify the System / Entity Requirements Domain Solution
Potential solution space options are partitioned, modeled, and analyzed define operational and technical boundaries.

Step 3. - Partition Each System / Entity Problem / Issue Space into Solution Space(s)
For each problem space, partition it into one or more potential solution spaces, preferably the least number.

Note: *Initially, the Solution Space boundaries may be notional, fuzzy, and lack well-defined boundaries. Several iterations may be required until the Solution Space boundaries mature and stabilize.*

Step 4. Specify and Bound System / Entity Solution Space Outcome(s), Capabilities, and Performance
For each Solution Space, bound and specify its initial state and conditions, external interfaces, and operating environment conditions.

Objective 3 Formulate, Select, and Develop the System / Entity Operations Domain Solution
Once each solution space is bounded in terms of performance-based capability requirements, the system / entity developer needs to understand how the user intends to accomplish mission objectives as an organizational asset via system deployment, operation, sustainment, and disposal. This requires interviewing users, understanding their perspectives and paradigms, their operating environments and conditions, and collaborating to formulate and select the optimal Operations Domain Solution from a set of viable candidates.

The Operations Domain Solution is typically documented in a Concept of Operations (ConOps) document. The ConOps developer, in collaboration with the stakeholders and development team personnel, expresses the operational approach concerning how the system, product, or service is to be deployed, operated, sustained, and disposed. Key topics include but are not limited to:

a. Identification of system, product, or service users, end users, and stakeholders
b. Definition of user / stakeholder mission(s) and supporting system objectives, use cases, and scenarios
c. Selected operational architecture from a set of viable candidates
d. System interfaces and interactions with its operating environment
e. Phases, modes, and states of operation
f. Mission event timeline (MET)
g. Expected performance-based responses and outcomes
h. System deployment, operations, sustainment, and disposal concepts.

Step 5. Formulate, Select, and Mature the System / Entity Operational Architecture
As the solution space boundaries are established and mature, the next step is to formulate a set of viable candidate operational architectures, perform an Analysis of Alternatives (AoA) of those candidates, and select the operational architecture that best meets the solution space requirements and life cycle constraints. Decision factors are refined into decision criteria and weight allocations, preferably by the stakeholders, for the AoA selection process.

Step 6. Allocate Entity Solution Space Requirements to System / Entity Operational Architecture Elements
Once the operational architecture is selected, solution space specification requirements and constraints are allocated and flowed down to the elements of the operational architecture.

Step 7. Verify the System / Entity Operations Domain Requirements Traceability
When completed, the solution space requirement allocations to operational architecture elements are verified to ensure traceability to the system's source or originating requirements.

Objective 4 Formulate, Select, and Develop the System / Entity Behavioral Domain Solution
As each entity's Operations Domain Solution evolves and matures, the next objective is to define how the entity will respond to stimuli, excitations, cues, and conditions originating from external systems in its operating environment. This requires formulation, selection, and development of the entity's Behavioral Domain Solution that expresses the time-based, sequential and concurrent, control and control (C2) data flow actions or task dependences to be performed and their expected outcomes... as a function of phase, mode, and state of operation. The sequences of actions or tasks involve more than simply algorithmic processing and decision-making. Each action requires the establishment of the initial set of operating conditions and assumptions, performance of the assigned task or action, and post-processing – i.e., housekeeping closure actions and preparations for the next execution of the capability.

Step 8. Formulate, Select, and Mature the System / Entity Behavioral Architecture
As the initial step for Objective #3, we formulate and select the behavioral architecture from a viable set of candidates via AoA. The behavioral architecture, which expresses the configuration of the set of functional capabilities, depicts how the entity is required to respond to external operating environment stimuli, excitations, cues, or conditions originating from the system / entity operating environment. This step serves the value-added transfer function required to produce behaviors and performance-based outcomes for a given set of inputs.

Note: *Observe that our decision-making began with how the user will employ the system to perform organizational missions and progressed from there to how the system will behaviorally respond to stimuli, excitation, or cues from external systems. Key point: we have not convoluted our thought process with how we will select and configure physical elements – e.g., Physical Domain Solution components – to implement the Behavioral Domain Solution which comes later.*

Step 9. Link the System / Entity Operational and Behavioral Architectures
Once the behavioral architecture is selected, we employ mapping techniques to link, reconcile, and harmonize the operational architecture to the behavioral architecture for consistency and completeness.

Step 10. Allocate System / Entity Solution Space Requirements to its Behavioral Capabilities Architecture
As part of the system /entity integrated design solution, the Behavioral Domain Solution is required to be fully compliant with the system / entity solution space requirements. Therefore, we allocate solution space specification requirements directly to individual behavioral architecture capabilities.

Note: *It is important to note here that the solution space requirements are derived within each specification until contributory performance-based capability requirements statements can be individually and directly allocated to a single architectural element at the next lower level of abstraction. However, please note that the problem solving / solution development process progresses to lower levels of abstraction over time. The realities are the total set of Four Domain Solutions are accomplished top-down / bottom-up through various levels of abstraction and left-right / right-left within each level of abstraction.*

Step 11. Characterize the System /Entity Behavioral Architecture Capability Interactions
When entity solution space requirements have been allocated to the Behavioral (Capability) Architecture elements, we describe how the system / entity interacts with external systems in its operating environment. Using an MBSE approach, model the input / output (I/O) processing thread through various configurations of integrated capabilities required to produce specific performance-based behaviors and outcomes in response to stimuli, excitations, or cues from each external system. Example tools include: Unified Modeling Language (UML™) / Systems Modeling Language (SysML™) use cases, sequence diagrams, activity diagrams, Mission Event Timeline (MET), et al.

Objective 5 Formulate, Select, and Develop the System / Entity Physical Domain Solution
As the system / entity Behavioral Domain Solution evolves and matures, the developer initiates activities to formulate, select, and develop the Physical Domain Solution for the system, product, or service. When Systems Engineering education and training are lacking, the Physical Domain Solution becomes the primary focal point of most ad hoc engineering teams. Given the exclusive focus on the Physical Domain Solution, ad hoc engineering

teams ignore the preceding Operations and Behavioral Domains Solutions as key decision-making dependencies. In fact, most ad hoc engineering teams treat the Behavioral Domain Solution as a "design implementation detail" after the Physical Domain Solution has been selected rather than as a "driver" to leads to a specific Physical Solution.

Note: *What is especially important here is to recognize and understand WHAT has to be accomplished operationally and behaviorally BEFORE you commit to a physical architecture for the system, product, or service. This is a deficiency common to many ad hoc engineering decisions.*

Step 12. Formulate, Select, and Mature the System / Entity Physical Architecture
Given the evolving and maturing Operations and Behavioral Domain Solutions, the developer begins formulating a set of viable physical architecture candidates. This is accomplished in collaboration with the user to identify decision criteria and relative weights for the AoA of the candidate architectures. Decision criteria include weighted factors such as technical performance requirements, technology, development and life cycle costs, development schedule, and risk and their underlying subcriteria. On completion of the AoA, an optimal Physical Architecture is selected.

Step 13. Allocate the System / Entity Solution Space Requirements to its Physical Architecture Elements
Given the selected Physical Architecture, the next step is to allocate the system / entity solution space performance-based capability requirements – e.g., Requirements Domain Solution – to the Physical Domain Solution via its multi-level subsystems, assemblies, subassemblies, or part levels [3] - architectures.

Step 14. Link the System / Entity Behavioral Capabilities Architecture to its Physical Architecture Elements
When the Physical Architecture is selected, the next step is to link, synchronize, and harmonize the Behavioral Domain Solution with the Physical Architecture elements. The purpose of this step is to allocate Behavioral Domain Solution capabilities to physical architecture elements that will physically implement command and control (C2) capabilities to produce system / entity responses.

Step 15. Verify Entity Physical Architecture Requirements Traceability to the Requirements Domain Solution
When fielded, the physical system, product, or service implementation and multi-level, Physical Architecture elements are accountable for providing the capabilities and performance-based outcomes specified by the Requirements Domain Solution specification or task requirements. Therefore, verify that specification requirements allocated to each Physical Architecture element are traceable to their source or originating requirements.

Step 16. Develop the System / Entity Technical Description
Given the entity's Physical Architecture and Behavioral Domain capability requirements, the development team proceeds with developing and maturing the system description. In general, the system description includes the following work products as "artifacts" of the technical decision making process: mechanical documentation – e.g. assembly drawings, interface control documents (ICDs), 3-D models, weight allocations, parts lists, etc.; electrical documentation – e.g., schematics, ICDs, electrical power allocations, board component layouts, wiring lists, timing diagrams, parts lists, etc., software documentation – e.g. software design descriptions (SDDs), software interface design descriptions (IDDs), database design descriptions (DBSSs), operating systems, languages, coding, et al.

Objective 6 Evaluate and Optimize the System / Entity Four Domain Solutions
The I / O thought process thread and transfer function from Requirements to Operations to Behavior to Physical Domain Solutions requires multiple passes to achieve convergence to an overall solution that balances technical, technology, development and life cycle cost, schedule, and risk factors. Reconciliation of imbalances in these factors occurs along each step and following the first pass. As a universal problem solving / solution development methodology that is applicable to any entity within the multiple levels of abstraction of a system, imbalances that are discovered in entity's at lower levels of abstraction require highly iterative reconciliation of capability or performance requirements at higher levels. Objective 5 focuses on ensuring that the overall, multi-level, set of solutions is optimal within the set of project and life cycle constraints.

Step 17. Verify and Validate the System / Entity Four Domain Solutions
Requirements allocated and flowed down to the system / entity Operations, Behavioral, and Physical Domain Solutions should be verified for traceability back though various levels of abstraction to their higher-level source or originating requirements. Each solution should be complete and consistent with each of the other solutions in terms of semantics, compatibility, and interoperability.

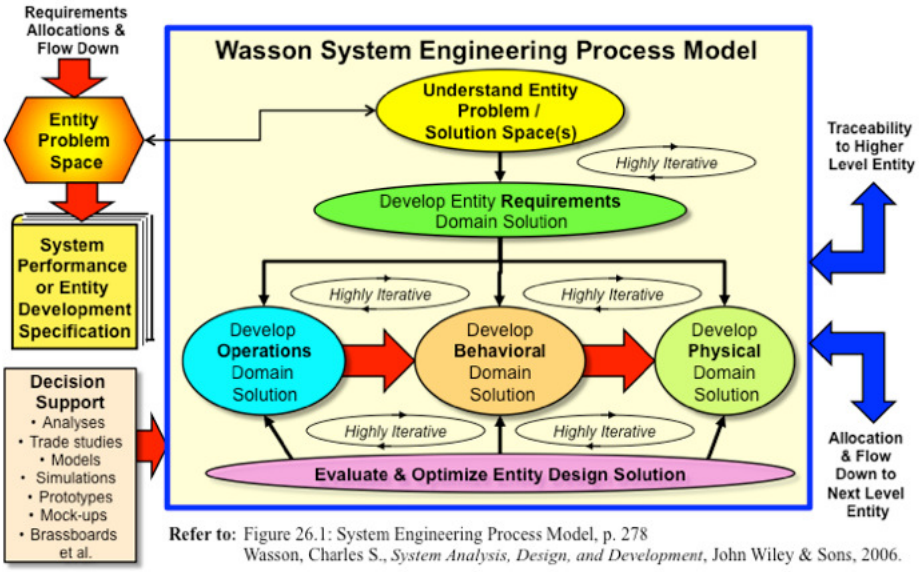


Figure 3: Wasson System Engineering Process Model [11] Representation

Step 18. Optimize Overall System Performance

On completion of Step 17, the overall system design solution should be reviewed, evaluated, and optimized to meet essential mission outcomes. This includes consideration of deployment, operations and sustainment, and disposal of the fielded system / entity as well as minimization of total lifecycle operating costs.

TRANSFORMING THE SE PROCESS METHODOLOGY INTO A STRUCTURAL MODEL

As the preceding methodology indicates, the approach for evolving and maturing the development solution for a system, product, or service that transforms an abstract user need into the physical realization of the system requires a progressive set of decisions. The steps presented above support the sequence of objectives that enable us to traverse the void from abstract user needs to delivery of the physical solution that satisfies those needs. Each step represents a collection of SE preferred practices and knowledge-base required to accomplish each objective. Thus, we can ascertain that the objectives stated above constitute a higher-level framework of linkages – e.g., entity relationships (ERs) – that enable us to transform a user's operational need into the physical system.

Based on the dependencies identified for each objective, we identify eight ER linkages:

1. User / Stakeholder Operational Need(s) to Understand Problem / Solution Space(s)
2. Problem Space / Solution Space(s) to the Requirement Domain Solution
3. Requirement Domain Solution to Operations Domain Solution
4. Operations Domain Solution to Behavioral Domain Solution
5. Requirement Domain Solution to Behavior Domain Solution
6. Behavioral Domain Solution to Physical Domain Solution
7. Requirement Domain Solution to Physical Domain Solution
8. Evaluate and Optimize System Design Solution to Operations, Behavioral, and Physical Domain Solutions

Using these linkages, we construct the framework for the Wasson SE Process Model [11] using the representation shown in Figure 3. The Wasson SE Process Model serves as a simplified, problem solving / solution development methodology paradigm for transforming a user's operational need(s) into an integrated solution that:

1. Responds to source or originating requirements.
2. Provides a logical progression of technical decision making steps that overcome the deficiencies of traditional ad hoc engineering.

3. Provides a framework for multi-discipline (specialty) engineering integration.
4. Provides linkages to ensure that solution space boundaries – i.e., technical, technology, development and lifecycle costs, development schedule, and risk are in balance.

APPLICATION OF THE WASSON SE PROCESS MODEL

A problem solving / solution development model should exhibit universality, utility, and consistency of application regardless of the size or complexity of the system, product, or service being developed. Given that systems, products, or services have structural frameworks consisting of stratified levels of abstraction, each with two or more interacting entities, the model should be applicable to any entity at any level of abstraction. The Wasson SE Process Model satisfies these criteria via *iterative* and *recursive* characteristics.

Iterative Characteristic

For a given entity within any level of abstraction, the SE Process Model activities are *highly iterative* as illustrated in Figure 3. As the system / entity developer applies the methodology, *highly iterative* feedback loops provide mechanisms to reconcile prior decision making deficiencies to accommodate downstream obstacles related to balancing technical, budgetary, cost, or risk performance. Viewing the model as an organizational "system" of interacting activities, the model enables us to repeatedly and predictably transform user needs – i.e., specification or task requirements - into the Physical Domain Solution. The Four Domain Solutions should be consistent, complete, balanced, reconciled, and harmonized with full traceability to the source or originating requirements – i.e., technical cost, schedule, and risk - for the entity.

Recursive Characteristic

One of the challenges in developing a problem-solving / solution-development model is its flexibility, adaptability, and scalability to accommodate a wide range of small to large, complex systems; any level of abstraction within a system; or entity within any level of abstraction. When a problem-solving / solution-development methodology can be applied to any level of abstraction or entity of a system, product, or service within each level of abstraction regardless of size or complexity, we refer to that characteristic as *recursive*. Figure 4 represents *recursive* application [12] of the SE Process Model to each entity within each level of abstraction.

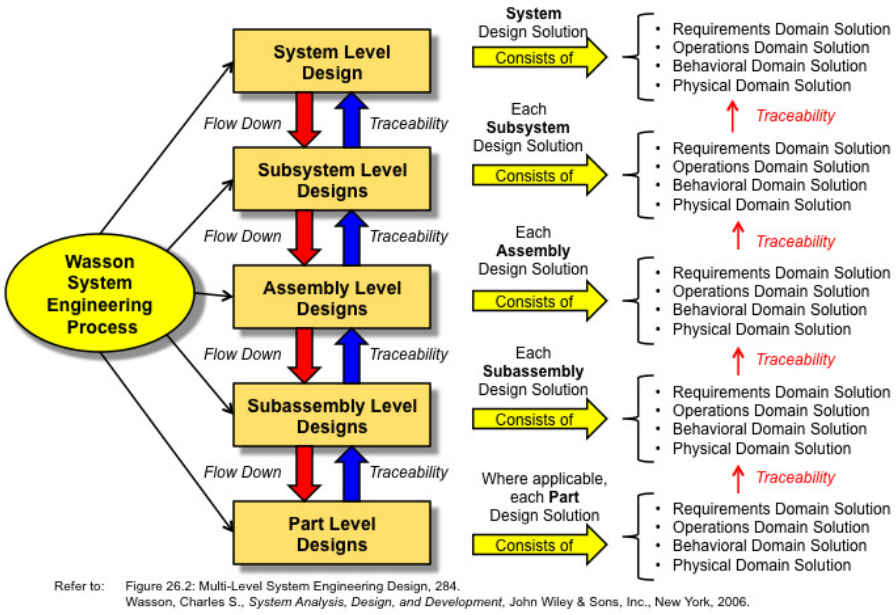


Figure 4: Recursive application of the Wasson SE Process Model to the System Development Process and entities within each level of abstraction [12].

Multi-discipline teams accountable for entities within each level of abstraction *iteratively* and *recursively* apply the SE Process Model to develop and mature the Four Domain Solutions for their assigned entity. This is accomplished under the purview of higher level teams that serve as "internal" customers and integrate lower level work products.

APPLICATION TO CAPSTONE PROJECT COURSES

Over the past 25 years, research studies have been conducted and papers written concerning capstone project courses. Given the need for capstone projects to provide the opportunity for students to apply education, knowledge, and skills acquired through previous engineering, math, and science courses in multi-disciplined team environments, research by Schmidt, et al [13] and Nemes, et al 14] have shown that SE fundamentals should be introduced prior to capstone project courses. Corns and Dagli [15] observed difficulties of introducing students to "key system characteristic attributes" during a capstone project. Based on this research, coupled with the author's Systems Engineering training programs and consultancy, a Fundamentals of SE course should be pre-requisite for students prior to a capstone project course. This postures the capstone project for success and avoids the distraction of introducing new concepts in a course intended for application of accumulated education, knowledge, and skills.

SUMMARY

In summary, we have introduced the Wasson SE Process model that serves as a new problem-solving / solution development paradigm for introduction in engineering courses. The model solves the Statement of the Problem and Success Criteria introduced earlier. The power of this model, which is discipline-independent, enables engineering students and professionals to:

- Correct the problem of taking a "quantum leap" from requirements to a "point solution."
- Employ a method that is easy to understand and communicate.
- Apply a flexible, adaptable, and scalable methodology to small to large, complex projects.
- Provide an approach that alleviates ad hoc engineering methods that are inefficient and ineffective.

REFERENCES

[1] Wasson, Charles S., *System Engineering Competency: The Missing Element in Engineering Education*, International Council on Systems Engineering (INCOSE) 2010 International Symposium (IS2010), Chicago, IL, July, 2010.

[2] Wasson, Charles S., *System Analysis, Design, and Development: Concepts, Principles, and Practices*, John Wiley & Sons, Inc. (New York), Figure 25.5, pp. 272.

[3] AFSCM 375-5, *Systems Engineering Management Procedures*, Air Force Systems Command, U.S. Air Force, 10 March 1966.

[4] FM-770-78 Field Manual: *System Engineering*, Headquarters, Department of the Army, April 1979.

[5] MIL-STD-499, *Systems Engineering Management*, USAF, Department of Defense (DoD), 17 July 1969.

[6] MIL-STD-499A, *Engineering Management*, USAF, Department of Defense (DoD), 1 May 1974.

[7] MIL-STD-499B DRAFT, *Systems Engineering*, Joint OSD/Services/Industry Working Group, 6 May 1994.

[8] ANSI / EIA-632-1998, *Processes for Engineering a System*, American National Standards Institute (ANSI) / Electronic Industries Alliance (EIA), January 7, 1999.

[9] IEEE 1220-1994, *IEEE Trial-Use Standard for Application and Management of the Systems Engineering Process*, February 28, 1995.

[10] Wasson, ibid, Chapter 23, pp. 243 – 245; Chapters 37 – 4-0, pp. 430 – 479.

[11] Wasson, ibid, Figure 26.1, p. 278.

[12] Wasson, ibid, Figure 26.2, p. 284.

[13] Schmidt, P., Zalewski, J., Murphy, G., Morris, T., Carmen, C., van Susante, P., *Case Studies in Application of System Engineering Practices to Capstone Projects*, American Society for Engineering Education (ASEE) Conference and Exposition. June 26-29, 2011. Vancouver, Canada.

[14] Nemes, J., Hochstedt, K., Brannon, M.,Kisenwether, E., and Bilen, S., *SE Capstone – Introduction of Systems Engineering into an Undergraduate Multidisciplinary Capstone Course*, American Society for Engineering Education (ASEE) *2011 Annual Conference Proceedings*, June 26 – 29, 2011, Vancouver, Canada.

[15] Corns, Steven, and Dagli, Cihan H., *SE Capstone: Integrating Systems Engineering Fundamentals to Engineering Capstone Projects: Experiential and Active*, American Society for Engineering Education (ASEE) *2011 Annual Conference Proceedings*, June 26 – 29, 2011, Vancouver, Canada.

BIO

CHARLES WASSON, is a member of the International Council on System Engineering (INCOSE), the American Society for Engineering Education (ASEE), and the Project Management Institute (PM). He holds BSEE and MBA degrees from Mississippi State University and a certificate in Systems Engineering from Stevens Institute of Technology.

In Dec. 2005, Charles released a new system engineering text entitled *System Analysis, Design, and Development* as part of the highly acclaimed John Wiley & Sons System Engineering and Management Series. The textbook, which was written for executives, program/project managers, project engineers, engineers, and business analysts, provides the WHATs, WHYs, and HOW TOs of SE *concepts*, *principles*, and *practices*. One of the key aspects of the text is the translation of SE theory into scalable practices that enable application across multiple business and engineering product domains such as aerospace and defense, medical, transportation, financial, et al.

In October, 2006, the International Academy of Astronautics (IAA) headquartered in Paris, France awarded the text its most prestigious book award, the *2006 Engineering Sciences Book of the Year Award*, at their annual Congress in Valencia, Spain. Although the IAA has traditionally selected space-flight centric publications for their Engineering Sciences Book of the Year Award, Charles text was written for general application to any domain such as medical, financial, transportation, telecommunications, aerospace and defense, et al.

As a member of the System Engineering Senior Staff for Lockheed Martin Corporation, Charles served as a leader for a number of multi-divisional projects. Corporate assignments included: membership on the System Engineering Council (SEC) and chairperson of the Education and Professional Development Working Group. In March 2007, Lockheed Martin Maritime Systems and Sensors (MS2) awarded him their prestigious Author of the Year Award.

Charles is President of Wasson Strategics, LLC (www.wassonstrategics.com), a professional training services firm specializing in systems engineering, project management, organizational development, and team development. His professional career experience includes over 39 years of proven leadership in program/project management; system, hardware, and software engineering design, development, integration, and test; as well as enterprise and organizational development with Fortune 500 companies. He works with private clients to formulate and implement practical, cost effective approaches, plans, and system solutions for achieving organizational performance success. As an internationally recognized author and instructor in system engineering and its organizational application, he is an invited guest speaker and panelist at professional meetings and symposia, and commencement speaker.

In support of advancing the practice of System Engineering and Project Management, Charles champions the need for a more robust approach to strengthening undergraduate engineering curriculums to include a System Engineering fundamentals course. Based on the "gap" between Engineering and Project Management (PM) integration and how systems engineers cope with the disconnect, he champions and teaches system engineering and development courses for both Engineers and PMs.