2019

Domain Process Model Overcome Limitations of Engineering Models for Developing Artificial Intelligent Systems

Gary O. Langford
Portland State University, gary.langford@pdx.edu

John Green
Naval Postgraduate School

Daniel P. Burns
Home Port Solutions

Alexander Keller
United States Air Force Nuclear Command

Dean C. Schmidt
Home Port Solutions

Citation Details

G. O. Langford, J. Green, D. P. Burns, A. Keller and D. C. Schmidt, "Domain Process Model Overcome Limitations of Engineering Models for Developing Artificial Intelligent Systems," 2019 Portland International Conference on Management of Engineering and Technology (PICMET), Portland, OR, USA, 2019, pp. 1-11.

Domain Process Model Overcome Limitations of Engineering Models for Developing Artificial Intelligent Systems

Gary O. Langford[1], John Green[2], Daniel P. Burns[3], Alexander Keller[4], Dean C. Schmidt[3]
[1]Engineering & Technology Management Department, Systems Engineering, Portland State University, Portland, Oregon, USA
[2]Systems Engineering Department, Naval Postgraduate School, Monterey, California, USA
[3]Home Port Solutions, LLC, Monterey, California, USA
[4]United States Air Force Nuclear Command, Control and Communications Center, Barksdale AFB, LA, USA

***Abstract*—The integrated set of prognostic domains (ISPD) of technology presented here provides a normative means to construct a wholly new process model for guiding Technology Management of Artificial Intelligent Systems (AIS). Seventeen domains represent all-inclusive stakeholder perspectives that encapsulate lifecycle analyses, evaluations, feasibilities, and tradeoffs with the domain contexts. Following Systems Model-Based thinking (SMBT), a postulated focal point interaction is the entry condition from which each domain is considered and thereafter traversed. Domains are interactive with each other through concurrent, iterative, recursive, and non-recursive processes. This interactive work continues until the completion milestones of each domain are satisfied. Techniques such as agile, progressive, prescriptive, and spiral are used as appropriate to reconcile functional and process requirements, risk, schedule, and budget. An important factor of ISPD is that multiple concepts of operation, widely diverse architectures, and explicitly differentiable designs are inevitable and highly desirable. The entire gamut of interactions of AIS with other posited systems is exposed and examined by the work carried out in each domain. While the ISPD was inspired by the use of SMBT with traditional process models, the essences of ISPD were extracted from developing systems of systems products and services. SMBT was piloted on a one-year project to integrate informational, social, and behavioral exchanges between humans and intelligent systems. This paper introduces SMBT for designing and building AIS by applying ISPD.**

I. Introduction

Traditional process models navigate milestones through netted-workflows by identifying and completing schedule and unscheduled work in blockchain fashion. As used by the project team, these process models are structured and intended to develop systems. While Artificial Intelligent Systems (AIS) are thought to be systems (as is purposely incorporated into the naming convention for AIS), the immense scope and enormity of interactions that must take place demand more sensitivity to the diversity of inputs from other systems. Advanced AIS must deal with a great deal of information that traditional process models either cannot discover early in the development process or are able to accommodate in design or architecture. The traditional process model is limited to systems rather than systems of systems. The same limitation has been discovered in systems engineering, necessitating a new thinking about process models predicated on system inputs and outputs. The AIS of tomorrow – the challenge for Technology Management of AIS – requires advanced process models that carefully dissect complexity to facilitate better behavioral models, incorporate the amount and diversity of content that must be handled by AIS, and recognize the necessary kinds and quantities of inputs that are needed. For example, AIS for retail applications must view the systems exemplified by the individual, the social, the cultural, the buyer, and the seller.

Each system brings with it interactions and a plethora of use cases that drive behavior, content, and inputs. Healthcare, banking, and entertainment each have potential for AIS to have impact in excess of hundreds of billion dollars. A systems of systems' perspective is paramount.

A fundamental aspect of coordinating AIS through the interactions of systems in a system of systems is the distribution of work products with a common update protocol and time stamp. Blockchains are a recent advancement that may evolve to improve concurrency and reduce latency in systems of systems.

A. Blockchains

A blockchain model is essentially a distribution of work products, such that every location that has a set of work products from a particular Domain or Model has the entire record of work products available from that domain. In that manner, the work products become part of the entire set of blockchain work products available to all in their instantiated form. Version control is essential for blockchains, so that an update to one is an update to all.

A key factor in architecting an AIS is to incorporate event-driven blockchains into the development process model so that asynchronous distribution of information and data can accommodate the disparate data structures and delivery modes. The event-driven paradigm organizes flow of information to instantiate immutable, verified transactions (i.e., blockchains). Within the process model, data has provenance, latency, security, and trust. Provenance provides history of data and formation of information; latency needs to be incorporated into the uncertainty associated with interpretation; security must encompass real-time applications; and trust relies on provenance, a query or concatenation/verification mechanism at runtime, and an analysis of provenance patterns. The process

978-1-890843-39-7

model phases need to capture interactions and their provenance between all sources of data and information, i.e., throughout the systems of systems involved with the AIS.

B. *Process Model Phases*

Defining phases within the process model differentiates work by project maturity and worker's experience in the maturity cycle of their set of knowledge, skills, and abilities [1]. Phases are usually based on events rather than on time. Events are definitive, as such, defined work processes are considered either complete or incomplete. Event-based planning and execution offer a means to track work process completion and start of next work processes. Process models have two parts – the objective and subjective parts. The objective part of the model deals with the physical objects, i.e., people who carry out tasks. The subjective part of the model deals with the thinking, acts of doing something that accomplishes work, and the cognitive modeling of how that work is and is to be accomplished. The process model is the relations between the objective and subjective parts. As such, the process model corresponds to the perspectives of the stakeholders who control various aspects of the model (such as the customer who pays for and lays out requirements to ensure their completion; the users who express their needs to achieve operational success; the developers who endeavor to deliver improved performance and quality higher than previous products or services; and the project managers who need to deliver on schedule, on budget, on performance). The principle of planning states "Integration planning is predicated on pattern scheduling (lowest impact on budget), network scheduling (determinable impact on budget), and ad hoc scheduling (undetermined impact on budget)" [2]. Applying this principle to theory, brings about process models that are structured to lay out tasks with work packages that encompass all the work for which an agreement to pay is enacted. The key issues for planning involve the type, amount, and availability of required resources, determining the internal and external factors that conspire to delay work progress, and ameliorating the consequential impacts of missing a milestone or delivery.

C. *Integration and Interoperability*

The process model facilitates the integration and interoperability of the ideas, mechanisms, and models of thinking and behavior [2]. Integration can create tension amongst stakeholders to support their own agendas, between priorities, how products and services are managed and resourced, to highlight a few stresses. Interoperability creates loss, yet helps to control emergence. Without interoperability, much might not happen that is expected and much that might happen might not be expected. Integration results in connectivity, i.e., physical interaction that bind things physical, e.g., objects. Interoperability achieves functional performances.

D. *Process Model Selection and Tailoring*

The initial challenge facing AIS developers is how to evaluate and select an appropriate process model for the tasks. The objective is to gain a sufficiency of understanding about the implications and consequences of the model as it impacts schedule, costs, worker's performance, and quality of the final product or service. Moreover, the goal of the model is to allow exploratory thinking early in the process phase(s) to provide a modicum of predictiveness about the unidentified problems that may lurk in current and future work. A primary condition for selecting an appropriate process model is that the scope and depth of the work be within the model boundaries of extensibility and scalability. All these considerations that impact model selection are bifurcated between systems and a system of systems or systems of systems.

The notion of process models for systems has been sought after as a harbinger of projects that build and deliver systems to meet their objectives. Since the mid 1960's, developing software using process models (notably the Waterfall [3]) has garnered great interest in business and government and has found a home in systems engineering. Many process models are in use, each having merits and shortcomings vis-à-vis the preferences of the project's key stakeholders. All process models can be tailored to accommodate various constructs in keeping with the limits of extensibility and scalability.

Tailoring of a process model [2] can mean to break apart work modules by partitioning, regranularizing the functional synthesis, redacting the work breakdown structure, reanalyzing functions by changing inputs and outputs of modular objects or functions, change the boundaries or boundary conditions for objects or aggregations of objects, change interoperability between objects by adjusting connectivity, coupling, and cohesion, or modifying the lifecycle of the product or service.

Modeling processes benefits from contributions in scripting workflows, identifying and solving difficulties with sequential and concurrent processes, mapping activities and context into idealized use cases [4], capturing actual events in reificatory use cases [5], utilizing systems model-based functional analysis to thread sequential and concurrent behavioral patterns [6], applying the speech-act model developed by Winograd et al. [7], formulating task scripts [8], abstracting business processes [9], incorporating event driven workflows [10], and including the notion of persistent workflows [11].

II. Brief History of Process Models

The overall gestalt of task and process is early written by Adam Smith (1827), extrapolated by Elbert Hubbard and the Roycrofters (1895) with enlightened views on worker-controlled processes, and then extended by Gantt (1910) to combine the two activities of planning and scheduling. In the 1930s, scheduling tasks expanded to sequential and parallel tasks; and in the early 1940s a method to identify the critical path was introduced by Morgan Walker of DuPont and James Kelley of Remington Rand. Soon after in 1957, Booz Allen Hamilton and the U.S. Navy laid out processes using a network diagram approach, referred to as PERT (Program Evaluation and Review Technique). In 1984, Eliyahu Goldratt posed the Theory of Constraints that noted most projects have chokepoints that require due attention to restructure, orchestrate, and implement work expeditiously. The Theory of Constraints, different than all previous organizing methods, deviated from a task-schedule model of processes to represent process that were time-critical to completion into a chain of tasks that have flexible start times and leveled resources

project-wide. Respect for the project, the task-customer-worker triad, the Theory of Constraints, and the Roycrofter passion for improving work that was seen as the means for personal achievement and pride, may inspire the best model of processes to deliver products and services. A process model is structured to track and measure what work is accomplished and what tasks need to be done to satisfy a customer and user. Adopting the process approach subsumes the importance of the customer, the user, the worker, and the project's organizational policies and rules.

Adding 200 years of thinking and applying process models, has evolved to a process model as a set of activities, structured by the dictates of key stakeholders (a business or organization, their customers, users of their products or services, regulators). A process model should be set up to reflect the rules of an activity, task, project, or business. Further, a process model should respect its stakeholders' political, economic, and social boundaries [2]. Modeling a work effort in such manner is the intention of modern process models.

III. Organizing Domains

Domains, discussed in detail in published Langford lecture notes [12], organize the process models by arranging stages according to aggregate groups of like-kind structures. These structures advance work products so that tasks that need to be completed can be enhanced with knowledge necessary to design, then build, then test the required product or service. One such process model, "build a little, test a little", suggests a concerted effort to figure out what will work to a high level of precision, i.e., the specifications are at a highly-advanced level of technology that pushes the state-of-the-art to an extreme. The domains are structures of processes to demonstrate snippets of subfunctionality could be built, integrated and then concatenated into threads of functionality that flowed throughout the subsystems of the product or service. Such was the case for the Hubble Telescope. For AIS, the discovery process for requirements was beyond the state-of-the-art from a technological perspective hampered by a systems of systems development.

Domain design means to translate functions into SoS uses and capabilities. Boundaries and boundary conditions of the problem must be known, stakeholder needs should be at least partially determined, the purpose for the product or service should be determined, the goal of the work effort should be spelled out clearly and coordinated with key stakeholders, the objectives of each task should be written and discussed with the development team, and the principles related to each of the process areas must be carefully implement to ensure efficient and effective execution of development work. The team should be well-versed on recent advances in technology, and appreciate the disruptive innovations that might ensure during the development and delivery phase of the product or service. And, the current and proposed systems need to be identified and described sufficiently so that the interfaces, exchanges of EMMI, and the temporal sequences for integration and interoperability can be planned and laid out for testing and verification.

The various domains that span the issues with systems of systems are shown in Figure 1 [12] on the next page. Each of the domain models and schemas are meant to help determine the systems of systems capabilities, users' needs and requirements. Note: the movement of work products from one domain to another is not a distribution, but rather a blockchain.

A. Problem Domain

The problem domain encapsulates the simultaneous considerations of problem and relation with stakeholder's need to solve the problem. A problem is something (objects or processes) that causes or will cause great harm, mayhem, disruption of civility, or loss of a system's ability to fend off its destruction or degradation to the point of being precipitously close to near destruction. A massive meteorite is a problem for the continuing existence of human lifeforms. If one refuses to believe that a problem exists, then that problem either does not exist for them, or exists but is deemed immaterial to their well-being, or is not yet relevant to their interests. Therefore, a problem is recognized as intense state of difficulty leading to irreparable harm that needs to be solved. The problem continues to exist when either you do not have a way of satisfying the stakeholders' requirements or that satisfying their requirements requires more than you are willing or able to invest in EMMI. If you determine you have a problem, then stating that problem determines how you think about the problem and its context. The problem statement is a brief discussion of the observations that lead you to believe there is a state of difficulty that is required to be resolved; a discussion about the physical, functional, and behavioral boundaries that demark the problem domain; consideration of emergence by an investigation of the boundary conditions by which boundaries are changed or permit input/output transactions of EMMI; and the salient issues that characterize the problem through measures, independent and dependent variables, and metrics. A problem is thus given characteristics that determine its merit or ranking amount problems that need to be solved. The problem domain is the only domain that has its drivers from outside all other domains. Without a statement of the problem there are no boundaries to delimit the problem, all objects and processes are likely candidates for inclusion, the solution is unbounded, all products and services necessarily need to be included, all mechanisms and all interactions necessarily are operative, and everyone is a user of whatever is discussed. The problem domain needs to be at least partially defined so that through a highly iterative process, the other domains may be enlightened enough to contribute to the problem statement. The Domain Problem Model provides inputs to the Domain Boundary Model, the Domain Object Model and the Domain Process Model (since each of these three receiving models can best benefit at the top-most portion of systems thinking, i.e., boundary, objects and processes).

Domain design (within the problem domain) captures the solution and scope of the development work. Limiting the scope of the solution, in effect, limits the amount of the problem that will be address.

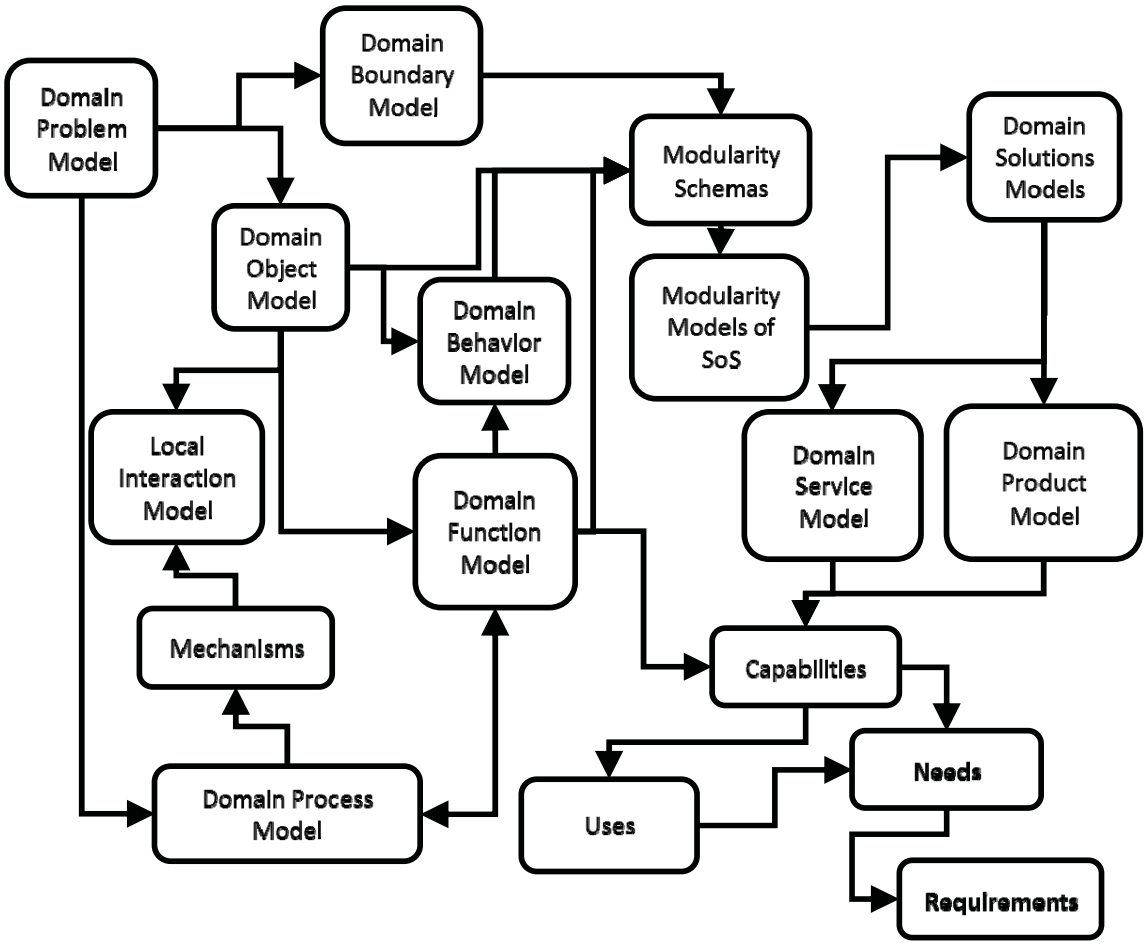

Fig. 1. Domain Models and Schemas Meant to Determine SoS Capabilities, Needs and Requirements

B. Domain Boundary Model

The Domain Boundary Model reflects the boundaries of physical objects. There are three types of boundaries: physical, functional, and behavioral [2]. Physical boundaries are the corporeal surfaces, functional boundaries are determined by the interaction between two physical objects that results in something that is used, and behavioral boundaries exist when objects either exist or do not exist (and are expected) and when functions exist or do not exist (and are expected). The Principle of Partitioning states, "Partitioning of objects can create tractable problems to solve if and only if boundary contiguity is achieved" [2]. The condition of contiguity ensures all that was conceptualized at the highest level of abstraction was indeed included in all subordinate levels of abstraction, nothing more (i.e., no overlapping boundaries) and nothing less (no underlapping boundaries). Successful integration depends on partitioning in accordance with the Principle of Partitioning. The Domain Boundary Model receives input on the problem from the Domain Problem Model; and provides its output to the Modularity Schemas.

C. Modularity Schemas

The assumptions for design of products and services are expressed through various characteristics including that of modularity. The Modularity Schemas embodies its design environment by implementing an indivisible function or subfunctions, has only one input and one output, is Markov process-like, i.e., not affected by input's source or output's destination or module history. Important to the schemas, characteristics of modularity are represented by a single requirement statement, with all requirements within the domain restricting the impacts locally. The process model would need to be accommodate functional equivalency through modularity. Domain analysis determines operations of unit modularity of data and associated processing (data). The Modularity Schemas reflects the overall structure of modules and is therefore a form of partitioning from the bottom-up (as is erroneously often espoused from the top-down). The Modularity Schemas receive input from the Domain Boundary Model, the Domain Object Model, the Domain Function Model, and the Domain Behavior Model. The Modularity Schemas are outputs to the Modularity Models of Systems of Systems.

D. Domain Object Model

The Domain Object Model provides a representation of the requirements described within the domain through the objects and their intellectual representations [2]. The Domain Object Model describes the structure and flow of data due to the objects, functionality due to interactions of objects, limitations due to object boundaries, constraints due to object boundary conditions, and controls (i.e., mechanisms) within the domain. The object model provides the representation of the rule set as enacted by processes within the Domain Object Model. The Domain Object Model receives input from the Domain Problem Model; and outputs to the Modularity Schemas, the Domain Behavior Model, the Local Interaction Model, and the Domain Function Model; and outputs to the Local Interaction Model, the Domain Function Model, the Modularity Schemas, and to the Domain Behavior Model.

E. Domain Behavior Model

The Domain Behavior Model depicts the behaviors due to objects (note software instantiations are objects [2]) and interactions between physical objects. From the rule set (a part of the domain object model), an object's behavior can be premised on the existence of an object and its expected behavior or the non-existence of an object and its lack of behavior. Further, the rule set may prescribe the same logic for functions as for objects – implying an object's behavior can be premised on the existence of interacting objects and it's the expected functional performance or the non-existence of an object and its lack of functional performance. The Domain Behavior Model receives input from the Domain Object Model and the Domain Behavior Model; and outputs to the Modularity Schemas.

F. Local Interaction Model

The Local Interaction Model describes the emergence from interacting objects and the lack of emergence that may be expected. Notably, product and service designers are focused on the emergence that is useful, i.e., functions [13]. However worthy, AIS design means to detect unwanted emergence, such as excess waste in EMMI or time, unwanted and spurious implementations of expected behaviors at unexpected times, or unintended mechanisms inserted by an object or enabled because of inadvertent or unplanned interactions or lack of interactions. A careful examination of the what inputs are needed when and by whom should be included in the Local Interaction Model. The Local Interaction Model receives inputs from domain object model and the mechanisms; and has no outputs to other domain models. The Local Interaction Model captures emergence that is not useful. Most importantly, unwanted or disruptive emergence is exactly what can be gleaned from objects and Mechanisms or lack thereof.

G. Domain Process Model

The Domain Process Model configures four structures – requirements, behaviors, architecture, and verification/ validation to capture the customer view (requirements), the mechanisms (behaviors), the priorities of operations (architecture), and the verification to requirements and the fitness for use. The Domain Process Model typifies the dependencies of objects (e.g., requirements, views, behaviors, priorities, architecture, fitness) and processes (e.g., mechanisms, operations, verification, requirements, use) through the interpretive integrative framework [2], [4], [14], [15]. That framework is formed by the nexus of the object ontology and the process ontology. The purpose of the Domain Process Model is to relate the acts, activities, and processes to the cogitating, mechanizing, and representing of enablers of force that are causal to actions from influences on physical objects. The Domain Process Model interacts with the Domain Function Model exchanging inputs and receiving outputs. The Domain Process Model outputs mechanisms.

H. Domain Function Model

The Domain Function Model translates the interactions between physical objects into uses and capabilities. For example, the domain design methodology transforms and combines the model of designing a product or service with the model of a design task and the model of the design process into the function of 'to design'. The Domain Function Model is operative at both the product and service level (during the lifecycles) as well as the implementations for development. Functions are measurable and as such readily lend themselves to the machines that people use. The boundary of the object's functional domain is reached when the use that originated with the user through interactions with a physical object has ended when the user ceases interaction with the physical object. If the user continues to look at the physical object, form an image of that physical object, or behaves as if the physical object must be avoided (even if the physical object is not visible), then the function of 'to see', 'to image', 'to behave', or 'to avoid', respectively, are considered functions within the domain function model. The Domain Function Model takes inputs from the Domain Process Model and the Domain Object Model; and outputs to the Domain Process Model, the Domain Behavior Model, the Modularity Schemas, and the Capabilities.

I. Mechanisms

Mechanisms are broadly defined to be that which influence objects by means of forces. Mechanisms are made up of the impact of EMMI on objects. A simple mechanism can be a single input of EMMI received by an object by which that object releases EMMI – the mechanism of to move object by applying a force. Complex mechanisms result from the combining of many simple mechanisms that can offer mechanical advantage, electrical transformation, or magnetic influence. Mechanisms derive from the Domain Process Model and influence the Local Interaction Model. The inputs for Mechanisms are the Domain Process Model, i.e., the elements of influence [2] and force [16].

J. Modularity Models of Sysems of Systems

The Modularity Models of SoS depend on the inputs from the Modularity Schemas. These inputs are driven by the problem, the domain objects and processes, the behaviors, the interactions between objects, the boundaries, and Modularity Schemas. Typically, when a lack of appropriate work products is discovered within the Local Interaction Model, the process

model evaluation and use flounder and attempts to use the traditional stages, phases, and milestones to fill in what is found to be missing will cause inordinate delays. Disruption of work products that are either not needed for a particular milestone or work products that have not been started or proceeded to maturity will cause a milestone to be missed. Indeed, these ill-fated attempts to improvise with provisional work products assumes that the process model is salvageable and that progress can continue until an acceptable replacement can be made. For the particular problem of process models that are geared for notasystem or systems, the challenge of a system of systems or systems of systems is beyond the validity of these traditional process models. When such issues of missing work products arise, the key management and technical people need to consider non-traditional process models. When significant issues are discovered, a new exemplar for managing the development process must be sought and put into action.

K. Domain Solutions Models

For systems of systems, the Domain Solutions Models prescribe a manner of regaining control over the origination and flow of work products. A process model for developing complex systems of systems such as AIS is heterarchical in structure (posing each attribute of development on equal footing) and focused on achieving requisite systems and system of systems behaviors and functionalities. During product or service development, the domains of each constituent system are interactive with the domain of the system of systems in a reflexive, self-referent manner. Therefore, each work product used one domain and will be used similarly in all other domains. A change in the work product is a change in all work like-kind work products. To maintain objective control over the work package, the system of systems development needs to be implemented in blockchain fashion. The inputs of the Modularity Models of Systems of Systems into the Domain Solutions Models have all blockchains of work products from all the Domains and Models. The output work products from the Domain Solutions Models is the input to the Domain Service Model and the Domain Product Model. All products have complements from the Domain Service Model and the Domain Product Model; same as all services have complements from the rom the Domain Service Model and the Domain Product Model.

L. Domain Service Model

The Domain Service Model recognizes that service is done by an intelligent system for an individual. The risk of the individual who takes part in receiving the service (offered or provided by AIS) is identified, evaluated, assessed, and determined to be acceptable or not acceptable. As a key factor in design, this decision based on risk may impugn an inherent portion of the AIS business model before, during, or after development and being put into operations. From a social perspective, the Domain Service Model should incorporate the emotional, social, ethical, cultural, equality, and level of conviviality. There needs to be several forms of feedback that indicate how the interactions between AIS and humans measure up in terms of stakeholders' standards for success – to imply that the standards and measures for success is an integral part of the Domain Service Model. Overall, the top-level functions and processes for the Domain Service Model could be parsed as "to request (service)", "to provide (service)", and "to status activities (service)". Moreover, the apprehension and realization of loss of EMMI is a useful consideration to assist with work products in the Domain Service Model.

A general loss function quantifies the loss as a metric to be used in tradeoff studies [2]. Determining losses as part of the key interactions within the Domain Service Model serves to quantify more versus less for all domain tradeoffs and development of work packages. With regards to effectiveness of the Domain Service Model the output of the Domain Service Model is to Capabilities.

M. Domain Product Model

The Domain Product Model incorporates the physical objects, processes, and mechanisms embodied in the thing with certain capabilities that is produced or manufactured. For those familiar with systems engineering of products, the Domain Product Model is the representation of the work flows and work products of systems engineering. Conceptual design, detailed design, architecture, prototypes, verification and testing, and validation testing are incorporated into the Domain Product Model. As with the Domain Service Model, the concept of risk is incorporated, not from the perspective of the individual, but rather from the views of the AIS through its stakeholders in their business models. The risk associated with the AIS that provides the service (received by the individual interacting with the AIS) is identified, evaluated, assessed, and determined to be acceptable or not acceptable. The risks associated with the product design and operations is an integral part of the Domain Product Model. Of primary concern and focus is the development of the product's capability, to include product effectiveness. With regards to product effectiveness, the output of the Domain Product Model is to Capabilities.

N. Capabilities

The inputs to the Capabilities are work products from the Domain Service Model, the Domain Product Model, and the Domain Function Model. Capabilities are determinable as a set of measures of effectiveness (derived from functional performances) [2], [13]–[15] along with measures of the degree of dependability and measures of the availability of the product or service. Combined, Capabilities, dependability, and availability form a tuple known as effectiveness. For a system, that tuple is sufficient to help determine effectiveness. However, for systems of systems, the overall effectiveness of the whole of systems may or may not be impacted if a system is degraded and another system is enhanced – suggesting the net effects of efficiency may be applicable. Further, we distinguish between operational effectiveness when the product is first delivered (i.e., for initial capabilities) [17] and after maturation in the product's or service's operational environment (i.e., for dynamic capabilities). Operational effectiveness for systems of systems planned for development and those delivered for initial use have the benefit of planning for initial use. Operational effectiveness for systems of systems that are mature in their use face environments that may have changed from their design intent and initial usage. A key distinction between initial capabilities and matured capabilities

is the impacts of updates and improvements to technology, maturation of systems and systems of systems management, improvements in operational skills and abilities, and overall coming to a high-level of competencies with the full development of product or service intricacies and complexities. The notion of dynamic capabilities suggests that products and services can be changed after first operations through adaptations, upgrades and improvements to remain responsive to a changing environment and variabilities in context. Consequently, Capabilities encompasses the acquisition process, development, concept of operations, operations sustainment, and end-of-lifecycle utility. As a part of systems of systems effectiveness, we view Capabilities as the penultimate consideration for work-products fitness for use, and satisfying needs to solve stakeholders' problems.

O. Uses

The uses of the service and product Capabilities interact with work products from the service and product components as well as the functional performances of physical objects by users. The broad term user is meant as the aggregate of all users of the product or service. Consequently, the net actions of the users take its inputs from Capabilities. This distinction between an individual user of product or service and the aggregation of users (as the organization), averages the policy, training, availability, dependability and capability of individual users. The users have two primary considerations –net results from interactions with the interfaces with service/product components and the users' interactions with service/product as measured by speed and accuracy. Users have needs and if functional performances are involved, users have requirements. As such, users are part of the system of systems architecture. The perspective of the designer (developer), the customer (who pays), the user (who use), and other key stakeholders, initially will have different needs and different requirements. Those differences reflect their different boundaries of the issues they perceive, i.e., the various patterns of the objects and processes that present as capabilities. Consequently, the views of those boundaries. "Different observers may perceive boundaries differently and in conflict with other observers" [2]. Users have the greatest sensitivities to boundaries. For AIS interactions with humans, the users should be considered the most important stakeholder, who's actions, opinions, propensities, and habits wholly determine the outcome of effectiveness for the proposed and actualized Capabilities.

Uses are elementary forms of accomplishing or achieving something for a practical purpose, for changing value or worth, or for offering benefit or convenience. Use derives from Capabilities as having meaning to the user. System and system of systems behaviors reveal their intended uses through functional performances and quality.

P. Needs

A need is a condition requiring relief dye to problem(s). The output of Uses and the output of Capabilities are the inputs for Needs. Needs can be managed and resolved. Needs derive from stakeholders who are willing to recognize their problems. A stakeholder is anyone or anything that can have agent representation for the concerns of a thing and who significantly affects or is affected by the influences of a product or service. In a broader sense, the stakeholder is someone at risk due to the product or service; and particularly associated with the user and uses of the product or service. The Need encircles the problem, stakeholder, and product and service models.

Needs are perceived by the stakeholders, expressed by those stakeholders who realize there is a problem, and retained by those stakeholders who are resolute in finding solution(s) to the problem. There can be described a hierarchy of needs that delineate stages of progress in determining a requirement. Figure 2 expresses the hierarchy of Needs to Requirements. The output of Needs is the input of Requirements.

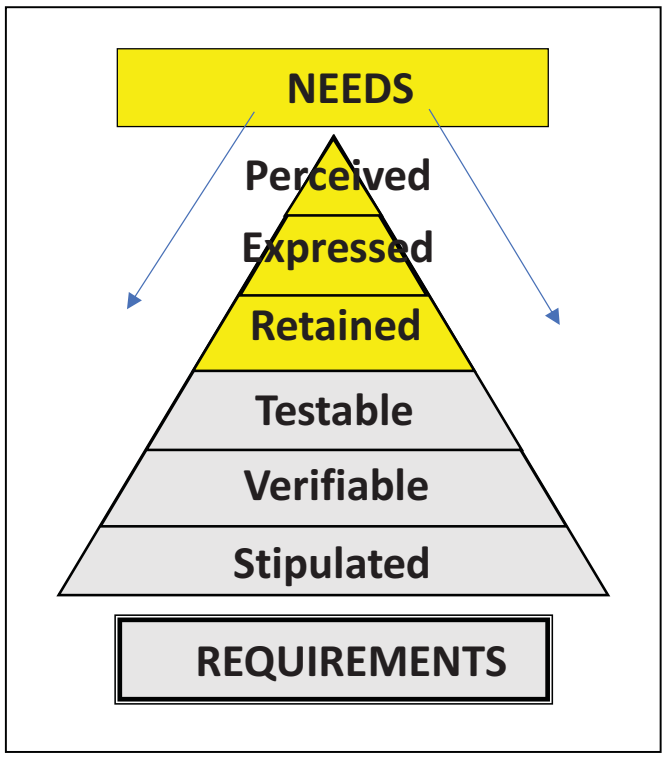

Fig. 2. Needs to Requirements

Q. Requirements

The interactions between AIS and Humans result in functional requirements with various performance(s). From stakeholders needs there may be products or services that must be acquired, tailored, or developed. IF, the stakeholders deem it necessary to accept these products or services with compulsory testing to determine if certain specifications are met, then those products or services are determined to be systems requirement. It is the sustained interactions that result in useful emergence, i.e., functions that have testable performance(s). The method of Needs to Requirements indicated in Figure 2 is used for both system requirements as well as system of systems requirements. The structure and stages in Figure 2 are reminiscent of Maslow's hierarchical model of instinctual needs [19].

IV. Basics for Phases in a Systems of Systems Process Model

Every system and system of systems has multiple concurrent interactions both internally and externally enacted. The arrangement of objects and the execution of processes (in the form of architecture) can be structured in part as hierarchical inferences and rule sets that both mediate and resource access to capacity and movement of capacity with distributed priorities. The contraposition of priority and control in such architectures leads to latency and variability to processing and throughput; ambiguation of input and output anteriority for processing and caching; and inefficient partitioning in processing pipelines. These challenging tensions that moderate flow of EMMI in resource constrained hierarchical architectures reduce effectiveness from that modeled using standard tools. Complete analyses on key threads of concatenated functions that are step-wise continuous is essential to discovering missing functions, identifying redundancy of functionality, detecting and tracing contention for control of an object, and determining the effectiveness of poorly designed modules that enwrap functions and subfunctions. Analyses of functional threads is a convenient method to check the synthesis of the functional design as well as to begin the allocation of functional requirements to the software and hardware work packages.

The premises of the authors are that modeling of processes must be based on theory that engenders rules; be formed from principles that guide integration of processes or structuring of models of processing; be dependent on using a systems approach to ferret out the complexities of interactions; and be classified by strict definitionals that are consistent and compatible with the mereology of objects and processes, such that there is no ambiguity or epistemological problems. Without theory, there is no predictive capability; without principles that apply to the processes of integrating processes there are only ad hoc methods to help surpass the limits of praxis reached with best practice; without a systems approach there will be missed opportunities, ambiguities in outcomes, and ineffective understanding of causal mechanisms that result in negative emergence; without strict definitionals and well-defined categories and clear classifications, instructions and specifications will be misinterpreted, nuances and subtleties will create doubt which will delay progress, and the requisite formalization and clarity needed for project success will be wanting.

In blunt contrast to the failed attempts to use traditional process models for system of systems development, the effects of concurrency and their emergent traits must be properly recognized as the primary spoiler of predictable project behavior. A complete revamping of the notion of how work products from one stage contribute to the needs of another stage of pending work products is warranted. Therefore, a new process model paradigm is needed to span both the temporal needs of each phase of process model work as well as accommodating the inputs of multiple sources of work products event-based needs for inputs as the project progresses. Figure 3 displays a multi input-dependent labyrinth of work-product flows from tasks to tasks, some in concurrent phases of work and some in sequential phases. These work-product flows provide for a well-behaved process model that can properly handle the needs and issues of the AIS lifecycle as a system of systems or as systems of systems. In particular, the interweaving of process derived work-flows in the temporal phases are juxtaposed with the concurrent needs of the process phases. On the next page, Figure 3 depicts these various phases.

Analytical modeling and simulations do not and cannot capture all things necessary to prevent degradation. The phasing without degradation begin with each constituent system beginning with phase of determining feasibility and exploring the design. Feasibility analyses are necessary for new systems (those to be built) and for existing systems to determine if the problem solved by each system is in keeping with the overall problem that is planned to be solved by the system of systems.

V. Domain Structures to Process Models

The domains form the essential elements from which to layout the relationships between work products within the process model. Accordingly, there are multiple stages in a process model that form to generate work products that result in milestone reviews. For the simple case of starting with a problem, defining the necessary functions, laying out those functions according to design principles, formulating an architecture that moves EMMI appropriately to support the four conditions of systemsness in keeping with the requisite priorities, then test, verify, and validate to ensure requirements are satisfied and needs are met. The simple case is not sufficient for a system of systems. Therefore, the process model must be able to handle multiple differences in systems, competing priorities amongst the needs of the constituent systems, optimizing the system of systems performance across all operations. All the while, the criteria for systemsness must be maintained. The four conditions for systemsness to expedite the metastable operations for key meta-functionalities (at the system of systems-level) and provide for each system's requirement for their respective metastable operations; continually promote and enable the agile movement of internal EMMI to satisfy the system-level and the system of systems-level needs; remain responsive to external conditions so as to smoothly adapt without precipitous demands on internal operations; and ensure non-reciprocal / irreversible processes are carried out when decisions must be determined, made, and enforced. These four conditions for systemsness are inviolable [2], [4], [12]–[15], [20].

A. SMBT Step One

To construct the set of processes necessary to build a system of systems product or service, a starting point must first be selected from which all else follows. Applying SMBT to the task, the first issue is to identify the action center of the basic interaction, i.e., the problem to solve, the question to answer, the conflict to resolve [2], [20]. The action center for a system of systems is the primary aim that drives all stakeholders. The aim of AIS is to first serve a customer (one who pays) to solve a problem, to answer a question (to solve or avoid having a

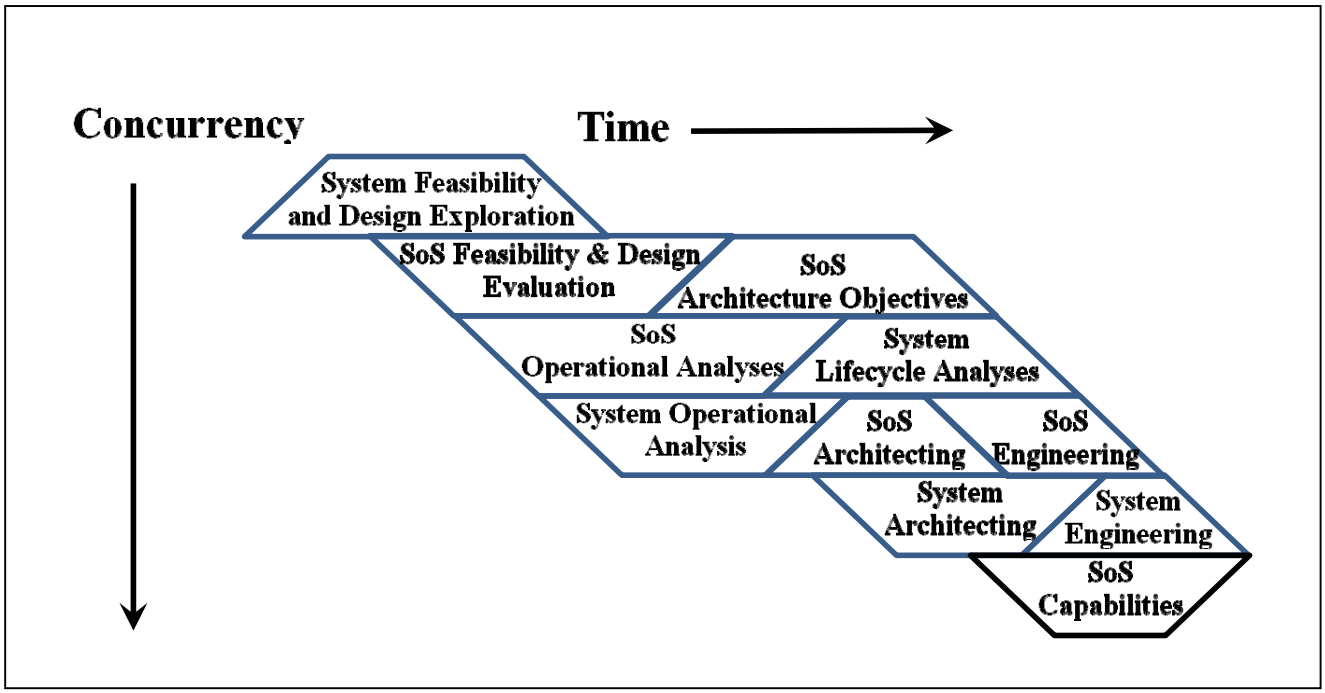

Fig. 3. System of Systems Process Model – a New Paradigm

problem), or to resolve a conflict between two parties before there is a problem. In this regard, AIS is intended to have all the novelty of another person, with better competency, greater knowledge, omni-present availability, and reliable, accurate, and precise fixes to address the requester's needs. Note: the user is the one who accomplishes something of value [2].

B. SMBT Step Two

The second step is to determine whether the primary domain of activity is at the system level or the system of systems level. If the product or service is at the systems level, then the standard systems model (i.e., that used in systems engineering) should be tailored to be adequate). If the product or service is at the systems of systems level, then dealing with a constituent system must be handled before concatenating activities into meta-functions that work across systems, i.e., at the system of systems level. Few users will interact at the system of systems level of activities, the vast majority will engage with a system that has been enhanced or supplemented with meta-functionalities.

Thus, in Figure 3, the first action is with each individual system to determine the feasibility of each and then explore the design space of each. Feasibility studies are design to facilitate a decision. For example, should an investment of EMMI be limited in terms of time and skills implies a study focused on the practicality of the project and a determination of the likelihood the project achieves its expected outcomes. Feasibility studies encompass technical, financial, managerial, operational, social, and legal issues. The limitations or time and

EMMI. Feasibility implies strategy is time of use and EMMI. In other words, feasible is determinable within the limits of maximum commitment and capability.

C. SMBT Step Three

Somewhat concurrent in time as the analysis for feasibility and design for each of the constituent systems in the system of systems, the system of systems undergoes the same evaluation for feasibility and design. Note the vertical overlap in Figure 3 for the Systems Feasibility and Design Exploration with the System of Systems (SoS) Feasibility and Design Evaluation with the SoS Operational Analyses and the Systems Operational Analysis. These four stages in the system of systems process model provide interactive work products that culminate in event-space with exchange of those work products to assist with the respective tasks. Horizontally, the tasks proceed through milestone reviews, each review marking the documentation that is exchange or provided to other tasks within the model

D. SMBT Step Four

As the milestones progress and work products mature by highly iterative, incremental improvements, the build-up of system capability moves from System Feasibility and Design Exploration to System Operational Analysis to System Lifecycle Analyses, to System Architecting, and then to Systems Engineering. At that stage, an existing system can be fitted with appropriate requirements and the system of systems can be defined. If the systems are to be newly built, then those system requirements will be assured of satisfying the system of systems operational needs.

E. SMBT Step Five

A key objective is to provide those system of systems meta-functions with full performances and quality along with operational, yet degraded functional performances when the

architecture is unable to sustain full functional performances. Testing and verification, then validation are typical activities carried out in SMBT Step Five. Early validation is possible with a top-down approach to determining system of systems capability. For out purposes, a system of systems is an integrated, interoperable set of systems that achieve a set of meta-system functionalities in which all constituent systems participate to varying degrees.

VI. SMBT Domain Activities

Work products are developed by using such methods as agile work groups that cross domain boundaries and work simultaneous in both domains; progressive work products that have a "boss" that leads tasks in multiple domains as the work package is moved from domain to domain; prescriptive work packages that produce products according to a master plan (involving multiple domains); and spiral process work flows that move sequentially from one domain to the next, then returning to an earlier domain to update and mature the work package. These groups that coordinate within and cross domains are the interfaces for the work flows who plan and schedule depending on the detail and level of content. Agile, progressive, prescriptive, and spiral methods are used as appropriate to reconcile functional and process requirements, risk, schedule, and budget. In addition, various common techniques are used to organize and develop work packages, including the work breakdown structure. Work breakdown structure can be based on physical, functional, process, behavioral – with physical and functional being the primary categories. Hybrid mixes of physical and functional as well as interacting objects (function) and mechanisms (process) have been used on military and commercial products and services (Intel Corp, U.S. Navy, U.S. Army, Department of Homeland Security, and NASA.

An important factor of ISPD is that multiple concepts of operation are necessarily developed during the early development of work packages through the paths of work flow. These concepts of operation, though with many elements in common, are typically without enough overlap. Part of the determination of capability (Capability) is to resolve discrepancies. There will also result a widely diverse set of architectures. With some developments, the concept of operations will emerge before the architecture has completed its maturation through the ISPD processes. The prognosis for a better architecture is increased if the concept of operations proceeds at a faster pace than the architecture work products. In that manner, the concept of operations can enlighten both the design and the architecture. However, the prognosis for a more effective integration plan and execution results with the architecture precedes the concept of operations. In this latter instance, the design suffers by not having a better-developed concept of operations. This tradeoff highlights the type of process flows that are active when developing work products.

Rather than deep development of a particular aspect of a domain, the work products append multiple layers of detail to update a previous incarnation of the work product. For example, an interface might be suggested by particular functionality that would benefit from modularization. An interface is suggested due to partitioning. However, the partitioning task has not yet begun, so this initial modularization is only a starting point. As the iterations incorporate additional functionalities that also appear to benefit from modularization (albeit with incomplete data), a pattern begins to emerge as to the preliminary list of interfaces. With an increasing number of domain perspective adding to the work package a typical outcome is to have a few competing ideas begin to develop with different partitioning. Those competing ideas are then subject to a tradeoff study that will cultivate higher-level concepts and meta-functionalities. Instead of leaving tradeoffs and stakeholder posturing for late in design or architecting, significant issues materialize and become factors to investigate and discuss, instead of taking sides. For a system of systems, the SMBT has shown to save significant time, achieve consensus quicker, and in the long-run to work through significantly more robust and achievable requirements.

VII. Conclusions

A new paradigm in process models helps to build an AIS that promotes deep investigation into literally thousands of trade-spaces. Existing process models meant for systems development are prone to fail in delivering on cost, on schedule, and on performance products and services. Historically, most development efforts neither instigate such number of tradeoff nor spend the time during the formative stages of design to identify the ramification of such a large number of tradeoffs. Usually there are a few "big" mistakes made when stakeholders cannot agree or when there is a major investment of EMMI at stake. That situation is not the case with SMBT work through the newly introduced system of systems process model. SMBT embraces tradeoffs as a natural consequence of its structure and ISPD supports the interaction between the work products for constituent systems for the system of systems.

The procedural steps for SMBT illustrate the entry point and conditions to begin thinking systems – the key interaction that couples the AIS with the Human. If another starting point is selected (such as by a system within the system of systems), the SMBT work will show the relative importance during the evaluation of loss functions applied in the Domain Service Model. Use of the new systems of systems process model over the past 9 years support both the approach using SMBT as well as the methods of investigating each of the domains in a step-wise fashion to account for trade spaces, entry points, conditions for systemsness, and effectiveness.

Comparing the systems domains with the systems of systems domains through inputs, outputs, interfaces, and tradeoffs promotes a thorough investigation of requirements.

References

[1] R. A. de Souza, "Maturity Curve of Systems Engineering," Master's Thesis, Systems Engineering Department, Naval Postgraduate School, Dec. 2008.

[2] G. O. Langford, Engineering Systems Integration: Theory, Metrics and Methods. Boca Raton, Florida, CRC Press/Taylor & Francis. 2012.

[3] W. Royce, "Managing the development of large software systems," Proceedings of IEEE WESCON 26, August. 1-9, The Institute of

Electrical and Electronics Engineering, Inc (originally published by TRW) 1970.
[4] A. Cockburn, "Structuring Use cases with goals" – JOOP/ROAD 10(5), Sep 1997 and 10 (7) Nov 1997.
[5] M. Jackson, "Problems, Methods and Specialisation," in Special Issue of Systems Engineering Journal on Software Engineering in the Year 2001.
[6] G. O. Langford. "System of Systems Process Model," Chapter 7, Engineering Emergence: A Modeling and Simulation Approach, L. Rainey and M. Jamshidi, Eds. Boca Raton, Florida: CRC Press/Taylor & Francis, 2018.
[7] R. Medina-Mora, T. Winograd, R. Flores, F. Flores, "The Action Workflow Approach to Workflow Management Technology," CSCW 92 Proceedings, November 1992.
[8] B. Henderson-Sellers, D. G. Firesmith, and I. Graham, "OML Metamodel: Relationships and state modeling," JOOP 10,1, March/April 1997.
[9] T. Gale and J. Eldred, "The Abstract Business Process," Object Magazine, 6(11) 21-37, Jan 1997.
[10] P. Fingar, J. Clarke, and J. Stikeleather, "The business of distributed object computing," Object Magazine - 7(2) Apr 28-33, 1997.
[11] M. A, Beedle "A 'light' distributed OO Workflow Management System for the creation of OO Enterprise System Architectures in BPR environments," OOPSLA-97 conference, 1997.
[12] G. O. Langford, Engineerng Systems Integration Lecture Notes, Published Dec 2016 – Feb 2017.
[13] G. O. Langford, Verification of requirements: system of systems theory, framework, formalisms, validity. 27th Annual INCOSE International Symposium (IS 2017), International Council on Systems Engineering, Adelaide, Australia, July 15-20, 2017.
[14] G. O. Langford. "Phenomenological and ontological models for predicting emergence," Chapter 6, Engineering Emergence: A Modeling and Simulation Approach, L. Rainey and M. Jamshidi, Eds. Boca Raton, Florida: CRC Press/Taylor & Francis, 2018.
[15] G. O. Langford and T.S-Y. Langford, "The making of a system of systems: Ontology reveals the true nature of emergence," in Conf. Proc. 2017 12th Annual System of Systems Engineering Conference IEEE Int.
[16] R. Feynman, The Feynman Lectures on Physics – 3 Volume Set, Addison-Wesley Longman. 1970.
[17] G. O. Langford and T.S-Y Langford, "The Changing Moral Mirror of Society," Paper 19R0256, Portland International Center for Management of Engineering and Technology (PICMET), Technology Management in the World of Intelligent Systems, 25- 29 August 2019.
[18] D .J. Teece, G. Pisano and A. Shuen, Dynamic Capabilities and Strategic Management. Strategic Management Journal 18(7): 509-533, 1997.
[19] A. H. Maslow, Eupsychian Management: A Journal. Homewood, IL: The Dorsey Press, 1965.
[20] G. O. Langford and T. S-Y Langford, "Building A Sustainable Business Model Through Technology Entrepreneurship: An Analysis Of Business Models From A System And A Systems Of Systems Perspective," Paper 19R0256, Portland International Center for Management of Engineering and Technology (PICMET), Technology Management in the World of Intelligent Systems, 25- 29 August 2019.