Development Cycle Modeling: Transdisciplinary Implications

Samuel E. Denard

Empirical Products and Services, Houston, Texas 77254, USA.
proprietor@empiricalproducts.com

Received 11 June, 2021; Revised 14 July, 2021; Accepted 15 July 2021
Available online 15 July, 2021 at www.atlas-journal.org, doi: 10.22545/2021/00159

This paper argues that development (product, system, software, etc.) is an inherently transdisciplinary activity. Development is defined as the conversion of ideas into their manifestations. This conversion is often characterized by development phases, e.g., concept, requirements, design, implementation, and evaluation/testing (CRDIE). Iterative sequences of these phases form development cycles. Development cycles drive new product creation as well as product quality and cost and utility. Consequently, understanding development cycles is important. Models can provide insight; however, end-to-end quantitative development cycle models are, at best, rare. This paper outlines such a model, the Statistical Agent-based Model of Development and Evaluation (SAbMDE). For purposes of this paper, transdisciplinarity is defined as a developer's holistic view of reality as filtered by that developer's sensory input and perception of that reality. The model builds its mathematical and logical structures on a foundational concept that includes and describes this sensory and perceptual integration. Because the proposed model has this transdisciplinary characteristic, the model's use and results will have transdisciplinary implications. One implication: Ideas are discovered, not created. Another: A developer must first adjust their perception to see the development path that leads to a desired end product before they can traverse that path. A third: The ordering of information in a development space must be maintained.. This paper defines a minimal SAbMDE model that logically and mathematically reveals these and other SAbMDE transdisciplinarity implications.

Keywords: design theory and methodology (DTM), transdisciplinarity, system engineering, decision theory

1 Introduction

A development cycle model represents the phases of system and/or product development. There are many such models currently in use. For example, Microsoft offers their Security Development Lifecycle [1, 2]. The National Institute of Standards and Technology (NIST) and other government organizations specify standards and instructions [3, 4] that are necessarily incorporated by industry [5, 6]. Researchers such as [7], have often enumerated and compared various methodologies. However, all these models and methodologies

represent a developer's effort to convert an idea into an end product. The models describe the intermediate conceptual, requirement, design, and implementation (CRDIE) phases as well as the testing phase that maintains the integrity of the conversion process. Figure 1 illustrates this representation in broad terms.

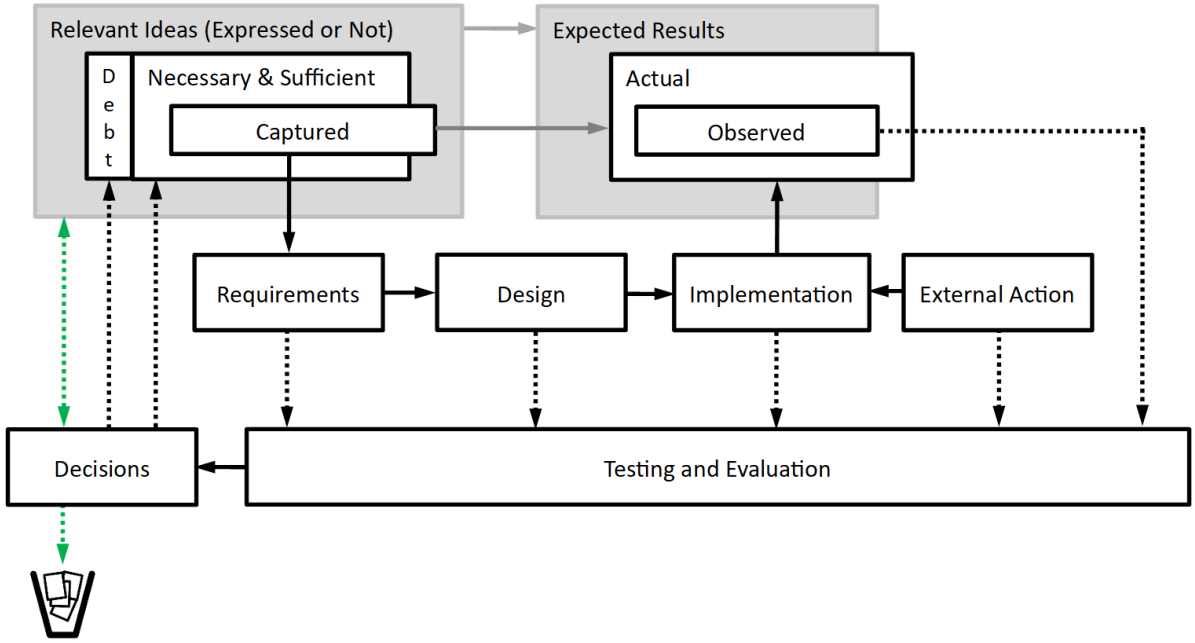

Figure 1: *Development cycle with conceptual, requirement, design, and implementation phases.*

The conventional models are usually expressed as best practices, guidance, and policies [8, 9]. For that reason, the models' effectiveness depends strongly on the skill and will of the developers who interpret and execute the models [10]. The models are largely qualitative; it is difficult to apply them objectively and consistently. Many quantitative function- and/or phase-specific sub-models exist: software reliability growth models (SRGM) [11], technical debt [12], run-time behavior extraction [13, 14], and more. However, the sub-models typically measure output from an existing system. Consequently, they analyze history rather than identify possibilities. In addition, the sub-models represent development phases differently, so differently that the idea of an end-to-end development cycle model has seemed out of reach. Even so, there is some consensus in the design theory and methodology (DTM) community that a phase-independent underlying model exists. Literature reviews [15, 17] state this consensus; and [18] describes the evolution of that consensus. There is also evidence of this thinking outside of the DTM community. For example, while formulating a theoretical foundation for building construction, Antunes and Gonzalez [19] state, "*In this research, construction is not restricted to civil engineering and architecture, but comprehends a broader understanding of building, putting up, setting up, establishing and assembling. Construction is the materialization of a concept through design, taking into account functional requirements and technical specifications for a project product utilizing specialized labour.*" Antunes and SAbMDE both envision underlying domain-independent models. The Defense Advanced Research Projects Agency (DARPA) is one of the organizations that fuels renewed interest in the underlying model by soliciting the research results [20–23] that such a model might promise. The National Science Foundation (NSF) is another [24]. The increasing complexity of modern systems [25, 26] is a major reason for the renewal. Previous work by Gericke and Blessing [15] and more recent work by Hazelrigg and Saari [16] confirm that an end-to-end development cycle model is still needed.

2 The Model

The model outlined herein addresses this need. The outline is presented in three parts: the model's evolutionary structure, its root concept, and its operations. The structure is derived from intuitive product assembly concepts; and it enables the model's quantitative capabilities. The structure's evolution is driven by human decisions. The decisions are represented by the model's interpretation of the mind-body-environment paradigm, i.e., the model's root concept. At a micro-scale, the root concept executes a set of operations that govern the interaction of the mind, body, and environment. The structure's evolution is accomplished with a set of macro-scale operations that emulate the root concept operations.

2.1 Structure

SAbMDE was conceived by re-thinking the process by which an end product is asembled from parts, pieces, and sub-assemblies. The process seemed to be self-similar whether the parts were the words of a requirements statement, the flowchart or CAD components in a design document, or the subroutine, door panel or feedstock in a manufacturing process. To give substance to this self-similarity concept, SAbMDE uses process algebra ideas, such as Wang's [27, 28] and Reich's [29], to represent each development phase so that analytical techniques can be uniformly applied across the entire development cycle. Wang has shown [30] that a desired end product (DEP) can be developed by sequentially composing intermediate products (IP) from sets of fundamental composable elements (CE): processes and relations. SAbMDE introduces an agent who decides which elements to compose. A correct decision set produces a sequence of compositions that become an end product (EP), hopefully, the DEP. SAbMDE recognizes (a) that each decision is one of a set of alternatives, (b) that the hierarchical super-set of alternatives forms a development space (DSpace), and (c) that the correct DEP decision set is the best of many development paths (DPaths) through DSpace. The model's foundation reflects itself in this representation via three inter-related sub-models: Agent, Development, and Evaluation. Model D houses the algorithms and structures that quantitatively define DSpace as well as the tools for DSpace navigation and traversal. Model E contains the testing and evaluation mechanism that informs the decisions that compel DSpace compositions. Model A emulates a development agent's perception, experience, vocabulary, and other human factors needed to create the Model E tests and to evaluate their results. Because each DSpace composition is connected by a decision to an evaluation of test results, there is an ESpace that mirrors DSpace. Because each ESpace test and/or evaluation maps to an agent's perception and vocabulary, there is an ASpace that mirrors ESpace. These spaces, known generically and collectively as XSpace, have similar forms and share a mathematical description. When executed together, Models A, D, and E represent a development cycle quantitatively and flexibly. This capability allows SAbMDE to hypothesize that DSpace characteristics constrain and guide an agent's decision-making in ways that conventional development cycle models can not. SAbMDE constructs DSpace from sets of composable elements: vocabulary items, V, and relations, R. For example, equations (1) and (2) are the basis of the simple DSpace fragment in Figure 2. Composable elements are supplied directly by an agent or extracted from documents by simple parsing or more sophisticated techniques such as those applied by Park and Kim [31].

$$V = \{v_0, v_1, v_2, v_3\} \tag{1}$$

$$R = \{r_0\} \tag{2}$$

DSpace construction begins at composition index 0 ($l_c = 0$) with an empty set. Construction continues by enumerating the composable elements at $l_c = 1$, and then by using the set-product operator to compose all the combinations of composable elements for each $l_c > 1$. As a result, at every DSpace node (DNode) for $l_c > 1$, an agent decision-maker has exactly $|V||R|$ options from which to choose. Note that at $l_c = 1$, only the vocabulary items are listed because composition is the same as vocabulary item selection at this composition index; relations have been enumerated but not applied. DSpace is characterized by the choice

of composable elements, the numbers of types of composable elements, ($|V|$ and $|R|$), and the number of compositions, L_c, required to produce the DEP.

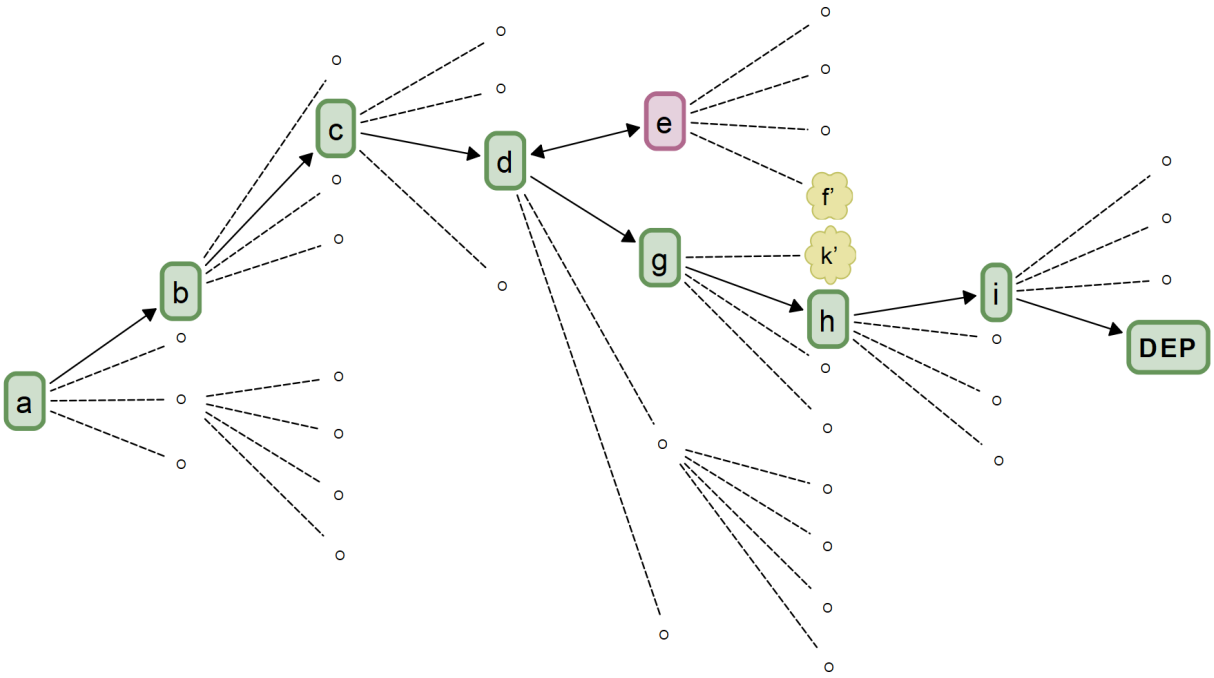

Figure 2: *A simple DSpace excerpt with highlighted DPath.*

A development agent navigates or traverses DSpace by deciding which DNode to compose next. In so doing, the agent leaves a trail of composed DNodes, a DPath, through the DSpace. Figure 2 shows a DPath: a-b-c-d-e-d-g-h-i-DEP. Note that the DPath is not a direct one. The decision to traverse to DNode **e** was a mistake that had to be corrected. An evaluation of DNodes **f** and **k** confirmed the error. Thus, the actual DPath is contorted. At each DNode, the structure of DSpace offers a probability floor for the agent's likelihood of making a successful decision, i.e., one that leads to the DEP. Those floor values are defined by (4), (5), and (6), and they correspond to random choices. In real situations, agents act with some level of skill. To recognize this skill, the DNode probabilities are scaled with a 0-10 (random-perfect) index, f_s, as in (7).

$$Q = |V||R| \tag{3}$$

$$u = \frac{1}{|V|} \tag{4}$$

$$q = \frac{1}{Q} \tag{5}$$

$$p(l_c) = \begin{cases} 0 & , l_c = 0 \\ u & , l_c = 1 \\ q & , l_c > 1 \end{cases} \tag{6}$$

$$p = x + \frac{f_s}{10}(1 - x), \quad f_s = 0, 1, ...10 \quad \text{and} \quad x = u \text{ or } q \tag{7}$$

The f_s scale factor is not arbitrarily selected; it is calculated from a set of agent-generated tests. SAbMDE prescribes tests in the following way.

$$t_i = \text{test definition} \tag{8}$$

$$\tau_i = \text{test result} = \begin{cases} \text{-1} & \text{not performed} \\ 0|1 & \text{failure|success} \\ [0,1],r & \text{degree of success} \end{cases} \tag{9}$$

$$r_i = \text{test result resolution} \tag{10}$$

$$w_i = \text{weight value for } t_i \tag{11}$$

$$T = \text{test set, e.g., } \{ \{t_0,\tau_0,w_0\}, ...\{t_i,\tau_i,w_i\},...\{t_{n-1},\tau_{n-1},w_{n-1}\} \} \tag{12}$$

Each test, t_i, is conceived, constructed, and added to a test set, T. When a next-DNode decision is to be made, the test set members are performed on each DNode candidate. For each DNode candidate, the test results are combined into a non-dimensional quantity according to (13).

$$\nu(l,k) = \frac{1}{|T|} \sum_{i=0}^{|T|-1} w_i \tau_i \quad \text{where } \sum w_i = |T| \tag{13}$$

To the degree that the test set is complete, sufficient, and executed properly, (13) provides a scaled value that directly represents a DNode's fitness for inclusion in a DEP DPath. This value supplies the additional probability of DNode selection success by replacing (7) with (14). Note that (14) should be applied according to the rules stated by (15).

$$p = x + \nu(l,k)(1-x) \quad \text{where } x = u \text{ or } q \tag{14}$$

$$p = \begin{cases} x & \text{when no tests are available} \\ p & \text{when any test results are available} \\ 0 & \text{when all test results fail} \end{cases} \tag{15}$$

A testing technique that employs these definitions is detailed in [40].

2.2 The Root Concept

SAbMDE joins with those [32] who accept development as an inherently human process and builds on a neuroscience foundation [33–36] of agent mind, body, and environment interaction. SAbMDE interprets that interaction in the following way.

SAbMDE considers a brain to be a device that constantly accepts sensory input from the environment. The brain processes and interprets that input. The brain then issues commands to the brain's host organism to respond to changes in the input [33]. Those responses are considered successful if subsequent sensory input confirms that the host organism has survived. The brain records and categorizes its (pre- and post-command) inputs and correlates the inputs with issued commands. In that way, the brain learns [37] and can, therefore, improve the quality of its subsequent responses to sensory input. Presumably, brains that produce better interpretations (and correspondingly better responses) are favored by natural selection. In essence, winning organisms learn to effectively test their environment and accurately match their perceptions to environmental reality.

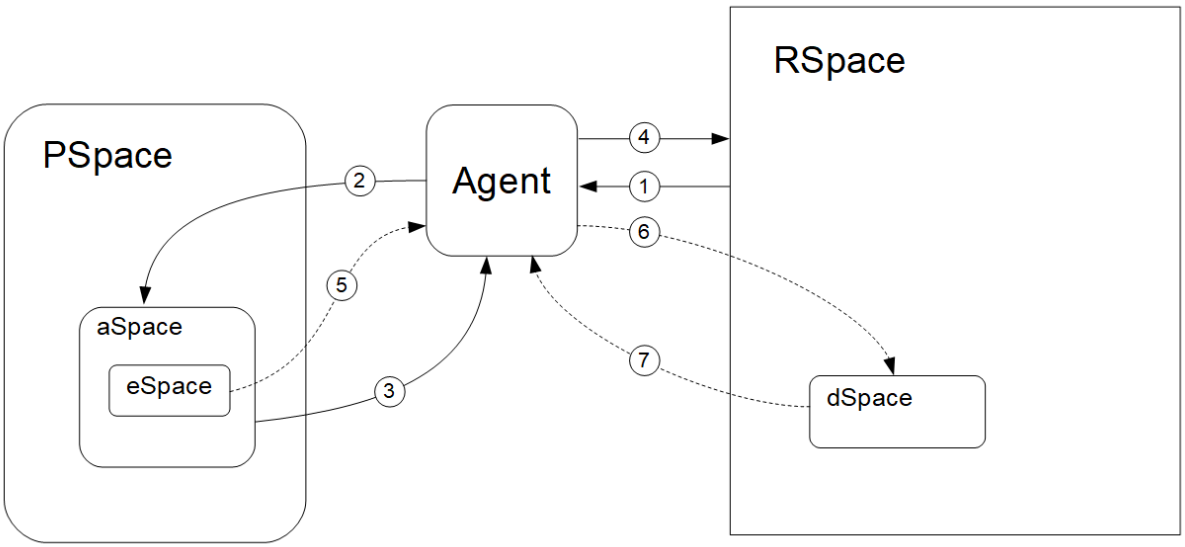

Figure 3: *The root concept.*

SAbMDE interprets the brain's sense-interpret-command sequence as illustrated in Figure 3. In that figure, RSpace is reality, the hard environment that is and can be. A dSpace is a particular subset of RSpace that an agent wishes to manipulate. PSpace is the set of all perceptions, i.e., interpreted sensory input, that can be mapped from RSpace. An aSpace is an agent's personal PSpace subset. An eSpace (where "e" refers to evaluation and testing) is the aSpace subset associated with a dSpace.

The titles of the various spaces begin with uppercase or lowercase letters. To clarify, the former designates a generic version of a space; the latter refers to an instance of the generic version. One example: a particular agent has an aSpace stamped from an ASpace template. The template defines all the characteristics and properties to be particularized for a given instance. Another example: PSpace is the union of all aSpaces. As noted during the structure discussion, all of the spaces are interdependent; a set of interdependent spaces will be referred to as XSpace.

This root concept assumes that the brain is hard-wired to receive sensory input {1}. The brain then senses, processes, and interprets this input to form a perception of the input {2}. The brain formulates responses {3} and commands the host organism {4} accordingly. This {1-2-3-4-1} loop executes autonomically. However, the brain may choose to cause a specific change in the environment. In that case, in addition to its normal activity, the brain purposefully constructs tests specific to the desired change {5}, issues commands to execute those tests and/or effect the change {6}, and receives input specific to those tests {7}. Depending on the inputs {7}, the brain issues additional and/or adjusted commands. The {7-2-5-6-7} loop is performed, with adjusted tests and perceptions, until the desired change is achieved. In the context of SAbMDE, the "brain" is an "agent"; the {7-2-5-6-7} loop is a development cycle; and the sought-after environmental change is a desired end product.

An agent's perceptions are formed from a range of input: the simplest raw data to the most esoteric philosophical concepts and everything in between. And, all of the input passes through a single sensing, processing and interpretation mechanism {2}. It follows that each {7-2-5-6-7} loop iteration can be influenced by any (or many) previous perception(s). SAbMDE assumes that the {7-2-5-6-7} loop is atomic and immutable. That is, the loop is the smallest unit of development; it is the building block for higher level development activity. And higher level development activity cannot violate the loop's operational integrity. The {7-2-5-6-7} loop is an atomic unit of development (AUD).

Detouring slightly from the main line of thought, note that an a agent need not be singular. SAbMDE assumes that contemporary and co-located agents sense the same reality. Because, as discussed above,

different agents will have different sensory input and perception histories, SAbMDE further assumes that individual agents exposed to the same reality will process and perceive that reality differently. Additionally, no single agent is likely to have sensed and perceived all of the available input. Figure 4, a notional Venn diagram, shows that the union of multiple agents' sensed input and perceptions are a better representation of the reality than that of any single agent. Therefore, a composite agent, i.e., a team, is likely the better choice to develop a project; although this depends on the team's diversity and the individual agents' ability and willingness to share their inputs and perceptions.

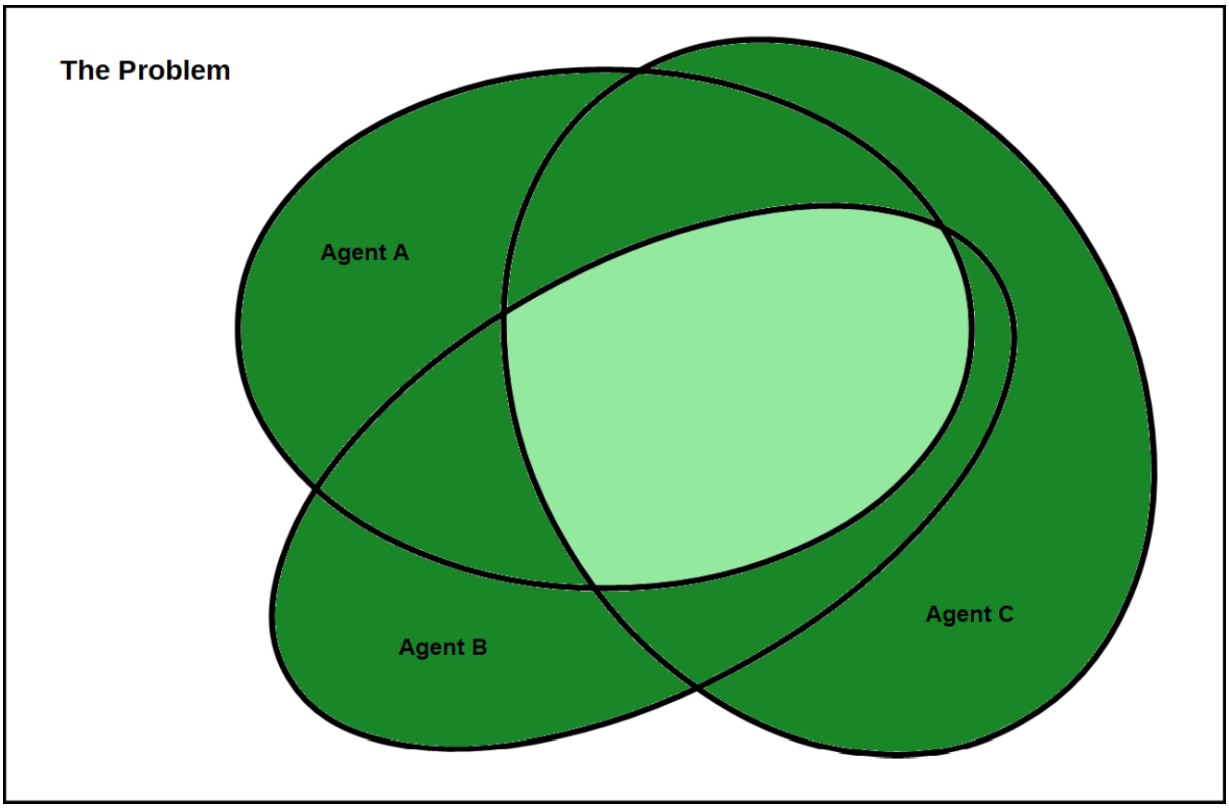

Figure 4: *Composable element coverage of a problem by composite versus single agent.*

Returning to the main line of thought, it is clear that an AUD is unique to each agent whether that agent is singular or composite. So, how does an AUD enter the model?

2.3 Operations

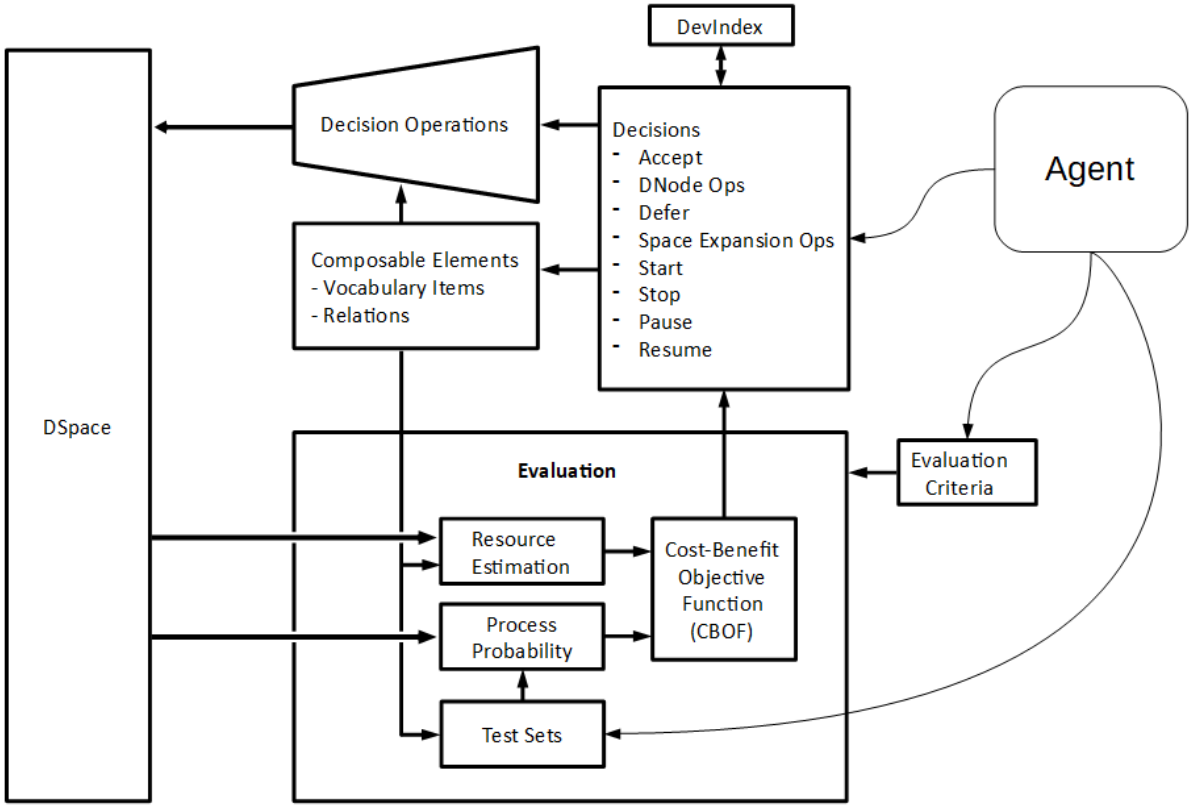

Figure 5: *Basic Unit of Development, an implementation of the root concept's Atomic Unit of Development.*

Figure 5 illustrates an XSpace traversal mechanism that explains how AUD decisions enter the model. Because, the DSpace described above is structured and uniform, the structure's traversal mechanism can also be uniform; it is applicable to every DNode. This mechanism implements the AUD at the macro-scale and can be thought of as a Basic Unit of Development (BUD). (Note: A BUD is similar to the well-known OODA loop [38].)

A BUD performs the various decision operations that manipulate and explore XSpace. The state of XSpace informs an agent's decisions via an evaluation process. This iterative activity is motivated, modulated, and managed by an agent.

Agent decisions are made based on the XSpace state, the set of evaluation tests created by the agent, and the evaluation criteria set by the agent. As has been shown, XSpace state is converted into a cost (actual and estimated resource utilization) and a benefit (DNode and DPath likelihood of leading to the DEP) which, in turn, feed a cost-benefit function (CBOF) that guides an agent's next decision. The guidance is modulated by evaluation criteria established by the agent or other development stakeholder.

The structure of XSpace can be fully managed with a fixed set of operations that fall into three categories: DNode, Space Expansion, and Control. These operations are itemized in Tables 1, 2, and 3. An explicit agent decision causes each operation to be performed.

Table 1: *All BUD operations.*

Name	Action
Accept	Designate current product as the end product.
DNodeOps	Perform appropriate DNode operation, see Table 2.
Defer	Assign a required DNodeOp to a composition index other than the current one.
SEO	Perform appropriate space expansion operation(s), see Table 3.
Start	SEO.Begin; initiate development operations.
Stop	Terminate all development operations.
Pause	Temporarily halt development by halting decision clock.
Resume	Continue development by restarting decision clock.

Table 2: *DNode Operations.*

Name	Action
Compose	Apply next-DNode relation to next-DNode vocabulary item to realize next DNode.
Decompose	Undo composition of current DNode; Navigate.
Navigate	Designate current DNode, i.e., set composition index, l_c and absolute node index, k_a.

Table 3: *Space Expansion Operations (SEO).*

Name	Action
(B)egin	Perform a first space expansion operation.
(E)numerate	Create, add, or remove composable elements. This may be done at any composition level.
(C)ombine	Given a current DNode, use the set-product set operation to create a next-level DNode set whose members are all possible combinations of the current DNode and the available composable elements.
(M)ultiplex	For a current composition index in which all DNodes are independent of each other, use the powerset set operation to create a next-level DNode set whose members are the unique combinations of the current index's DNodes.
(W)arp	For an existing XSpace and after an enumeration, especially an enumeration that must be applied at composition levels prior to the current level, re-combine and re-multiplex as needed. After the warp, (re)locate the current DNode and re-evaluate the DEP DPath.

This modeling approach's potential was demonstrated by Denard et al [39, 40]: SAbMDE can estimate the effort required to traverse a DPath; and it an compute a DPath's likelihood of reaching a DEP.

3 Implications

If transdisciplinarity is defined as a developer's holistic view of reality as filtered by that developer's sensory input and perception of that reality, then the AUD ensures that agents are naturally transdisciplinary. The BUD translation of an agent's AUD ensures that SAbMDE development modeling inherits this natural transdisciplinarity. Some implications of this inheritance are described below in the context of a simple SAbMDE example.

Figure 6: *Kenney's Lego portrait of Oliver.*

Consider Figure 6, a Lego portrait by Sean Kenney of a boy named Oliver. This portrait is composed from two types of Legos (1), black and white, sequentially positioned (2) to form the portrait. That sequence can be represented by a set of 4-tuples where the tuple members are a sequence index, a Lego block color, a row position, and a column position. Equation (16) is such a tuple; and it represents a single DSpace composition. Equation (17) is the beginning of one sequence of L compositions.

$$C = i, b | w, r, c \tag{16}$$

$$S_i = 1, b, 0, 1,\ 2, w, 0, 2,\ 3, w, 0, 3,\ 4, b, 0, 4,\ 5, b, 0, 5, ..., L, b, r_{max}, c_{max} \tag{17}$$

$$M = S_0, S_1, S_2, ... \tag{18}$$

There is a particular composition sequence, S_i that completely and (perhaps) uniquely defines Oliver's portrait. It follows from the discussion in The Model section that changing even a single composition creates a different sequence and, thus, a different version of the portrait. The human eye might not be able to distinguish such small differences; but the differences could be easily distinguished algorithmically. There are many composition changes that would leave the portrait recognizable as Oliver. There are also many sequences that are images of other people and things. But, in the same way that a geographical map allows a driver to navigate to a destination, the set of all sequences, M, forms a map (18) that can direct an artist from a blank canvas to a particular portrait; see [40]. Given a sequence, precisely reproducing an image is a straightforward, albeit tedious, task.

3.1 Ideas Are Discovered

According to the portrait artist, Kenney, at least 1000 Legos were used to create the Figure 6 portrait. If the artist normally builds the portrait from left to right, row on row, then the sequence indices and row-column positions do not change from sequence to sequence. Therefore, the number of different sequences is a function of the number of colors as shown in (19).

$$|M| = 2^L = 2^{1000} = 1.07150860718626732094842504906e^{301} \tag{19}$$

This number is very large but finite. This number of sequences likely includes all the recognizable portraits of Oliver. Actually, this number of sequences is large enough to map all the recognizable portraits of everyone who has ever lived. And, there is still room for images of many other things. The map includes all of the possible images that can be produced with L compositions of only black and white Legos. Conversely, the map contains only those possibilities.

Every composition sequence, S_i, can be enumerated. Every composition sequence could have been enumerated long ago. Had Picasso or Michaelangelo or a caveman worked with Legos (or black and white stones), each artist could have duplicated Kenney's sequence. Apparently, every composition sequence is independent of its potential composer(s). Similarly, each sequence is largely independent of time and physical location. It follows that all the composition sequences in every composition map are omnipresent; they are each just waiting to be (re-)discovered and (re-)interpreted.

Kenney did not create the Oliver portrait composition sequence, he discovered and interpreted it.

3.2 Level of Detail Matters

The criterion for a portrait's quality/meaningfulness is the ability for a human observer to see the subject in the art. The physical resolution of the medium limits this ability. This limitation can be reduced by increasing the size of the frame and using more Legos to compose the image, i.e., increasing L. These characteristics are compositional adjustments that strongly influence the image's level of detail (LOD), in this case, its resolution. This compositional resolution could be lowered to obscure the image or raised to provide photographic quality.

Additionally, the portrait was composed with "sequentially positioned" Legos. Precise or high resolution positioning was assumed but need not be the case. As noted above, sequential positioning is the SAbMDE relation (2) for this example; and that relation has two implicit arguments that represent the precision of the horizontal and vertical positioning. High argument resolution leaves Legos well-aligned and immediately

adjacent to one another. Low argument resolution creates misalignment and larger, random inter-Lego spacing. Argument resolution could be lowered or raised to reduce or increase the portrait's quality.

Increased compositional resolution, positional resolution, and other LOD components improve portrait quality; but the improvement comes with a cost: more Legos, increased build time, greater effort, etc. Presumably, there is a balance to be struck between quality and cost. There are minimum LOD values below which an end product is unacceptable, i.e., it cannot be a DEP. For various reasons, an agent may set or adjust those LOD values during development, e.g., when speaking with a non-artist. However, a development project (a portrait in this case) cannot be complete until L compositions have been completed at or above every LOD component's minimum acceptable value.

When compositional and positional resolution increase, there are more Legos for the agent to position; and precise positioning requires more careful measurement. In other words, developing the portrait becomes more complex; in fact, development complexity is proportional to LOD. XSpace structure provides the quantitative basis for specifying LOD, and, in so doing, provides a quantitative measure of development complexity.

LOD selection is an essential development consideration.

3.3 Where Is The Meaning?

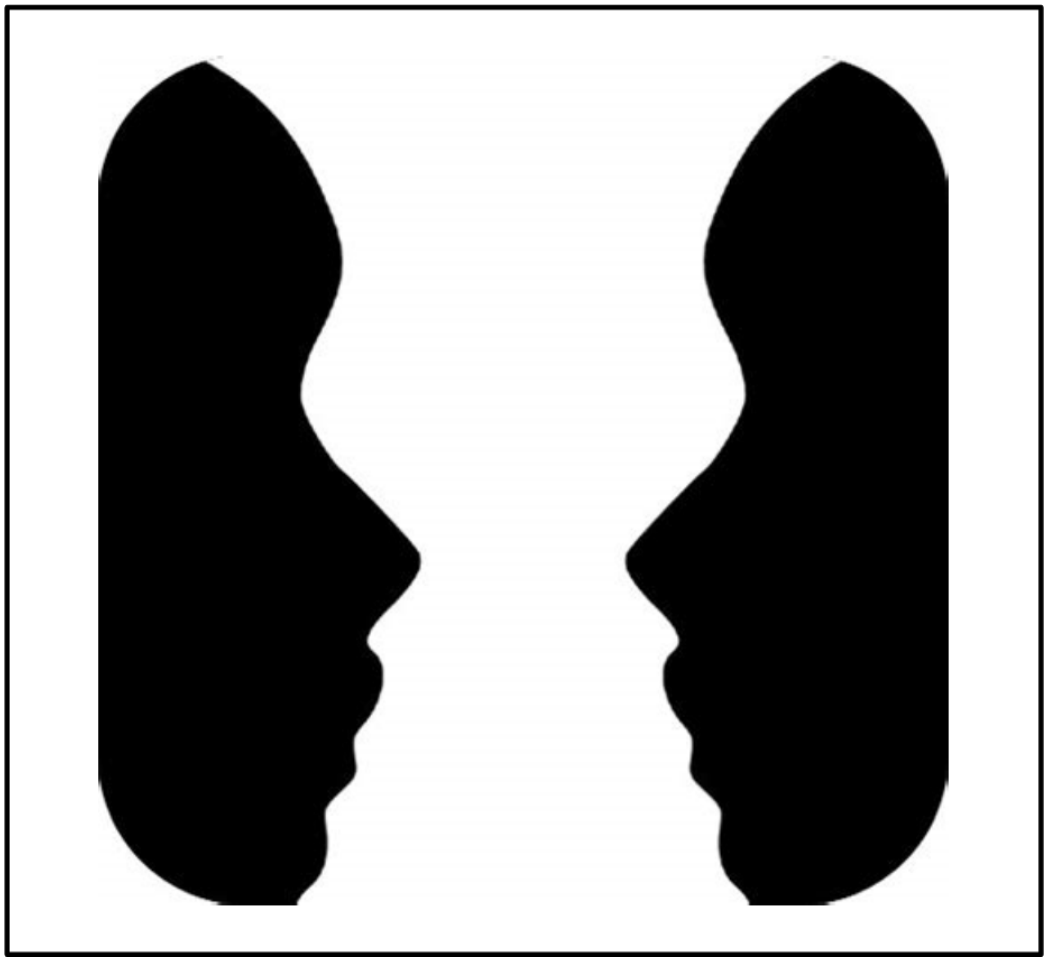

Figure 7: *An optical illusion, i.e., a single Lego DPath with multiple meanings.*

Figure 7 is an image of a familiar optical illusion. Constructed with Legos, this image, like the Figure 6 portrait, is represented by exactly one DPath. However, unlike the Figure 6 portrait, Figure 7 has multiple interpretations: a double silhouette, a vase, an urn, a chess piece, and/or a complex hydraulic damping piston. And, each interpretation might be some agent's DEP.

The Figure 6 and 7 DPaths were each defined by single corresponding sets of DSpace decisions. The decisions were based on corresponding sets of ESpace test results. It is not clear, especially for Figure 7, that the sets of test results were generated by single sets of tests. It is even less clear that there is a 1-to-1 mapping of test sets to an agent's ASpace. Stated another way, an agent can perceive a unique DEP in their ASpace, translate that perception into specific ESpace tests, and then make the appropriate decisions that drive DSpace compositions to the perceived DEP. But, this procedure is not uniquely reversible: there may be many ASpace DEPs that can be assigned to a given DSpace EP. Apparently, a DSPace EP can harbor a superposition of ASpace DEPs. Presumably, this superposition appears incrementally as a DPath is composed. For example, when only half of the bottom-most row of Legos has been composed, the illusion is definitely absent. When that row is fully composed, the first full increment of the illusion appears because the agent can then ask, "Am I building something white or something black?" Of course, the first hint of an illusion's existence was the placement of the first black Lego on the right. The likelihood of the illusion's possible existence increased towards certainty with every successive black Lego's placement.

As an illusion indicator, these particular observations are weak in part because they are entangled with assumptions about the image's symmetry. But, even with the indicator's best results for a single (or a few) rows, there is insufficient information for an agent to assign (intentionally or mistakenly) any of the aforementioned interpretations to the current EP. It is the continued composition of successive rows that allows the illusion to incrementally take its full effect.

At the current stage of SAbMDE development, mechanisms for detecting EP illusion onset or existence are immature; however, the structure of XSpace and the interaction of its sub-spaces offer a path forward.

Nevertheless, this discussion sheds light on a critical question: "Where is the semantic content of a development effort stored?" An EP encapsulates an agent's decisions that were made based on the results of a specific test set. But, the selection of tests for inclusion in the test set and the interpretation of their results depends on the content and structure of ASpace. In summary, DSpace defines what an EP is; ASpace, as expressed via ESpace, defines what an EP means. The semantic content of an EP is distributed throughout XSpace.

3.4 This Is Like That

When composing with Legos, the elements of the composition were clear: selected Legos and a rule for positioning them. The composition could have proceeded just as well with bricks or cinder blocks; perhaps with a positioning rule to produce masonry patterns required by building codes. Similarly, using carbon and hydrogen atoms, an agent could have composed families of hydrocarbon products. An agent could have made a more colorful portrait with four Lego colors. Or, with four particular chemicals and the same rule, an agent could sequence DNA. The underlying concept is the same: a composition process is a path through a map of possibilities. The map's structure enumerates those possibilities even though the map is independent of the product being composed.

These examples illustrate that the composition process is domain-independent. Similar CE sets apply to different domains. Like algebraic variables, only when CE are assigned a meaning do they become domain-specific; the rules for their manipulation are not.

SAbMDE manipulates CE with rules of composition, i.e., BUDs. A BUD is domain-independent; so the DSpaces formed by operations for different problems are likely to be similar. In the examples above, lessons learned from the interplay of white and black Legos might provide insight to an architect designing with light and dark tiles. The artistic rules for creating eye-catching patterns of four Lego colors might help a geneticist interpret DNA sequences. This is possible because the underlying development structure, i.e., DSpaces, are similar. The DSpace's mathematical representation facilitates algorithmic discovery of the similarities. Consequently, agents can learn, and can be assisted in their learning, not only from the

personal experience of DPath traversal but also, by simile and analogy, from the paths traversed by diverse others.

3.5 XSpace Must Evolve.

A color-blind artist might have chosen Legos with contrasting, but not black and white, colors to reproduce Kenny's portrait. As the work progressed, a color-capable observer might have pointed out that the developing portrait was not black and white, i.e., it was not developing towards the expected DEP. At that point, the artist would have realized that the required portrait could not be produced with the selected Legos. The current DSpace contains *a* DEP, but not *the* DEP. The alternatives would be either to accept the sub-optimal work or to scrap the current work, select the correct Legos, and re-build the portrait. The latter case is the artist's choice to change to an updated DSpace; or, in the language of the model, the artist warped the XSpace to include the more clearly perceived DEP.

The Lego example above was contrived; however, the need to warp XSpaces is real and valid. An agent will most likely start a project with an incomplete and imperfect CE set; at that point, the agent does not yet know enough about the EP to be developed. By definition, the XSpace that expands from an imperfect CE set cannot contain the agent's ideal DEP. Over the course of a project, as the agent's understanding grows, the CE set evolves to better describe a DEP. Therefore, the XSpace derived from the CE must also grow and evolve.

An agent's initial task is to find the XSpace in which a DEP exists. An agent does this by traversing a current XSpace, discovering it's inadequacy, transforming the XSpace into something more suitable, and then repeating the procedure. This procedure is a practical example of the adage, "A problem must be defined before it can be solved." Only when the agent discovers an adequate XSpace can the agent traverse the DPath to a DEP in that XSpace. As explained in [40], XSpace structure and characteristics can facilitate XSpace discovery.

3.6 Order Begets Order

The manner in which CE are enumerated matters. Table 3 implements the Combine space expansion operator with the set-product operator. Given the sets A and B defined by (20) and (21), respectively, equation 22 defines set C as the result of the set-product operator applied to sets A and B.

$$A = \{\, a_0, a_1 \,\} \tag{20}$$
$$B = \{\, b_0, b_1, b_2 \,\} \tag{21}$$
$$C = setproduct(A, B) = \{\, a_0b_0, a_0b_1, a_0b_2, a_1b_0, a_1b_1, a_1b_2 \,\} \tag{22}$$

Values can be assigned to the members of sets A and B. Equation 23 is one such assignment that happens to satisfy the condition specified by (24).

$$val(a_0) = 1,\ val(a_1) = 2\ ;\ val(b_0) = 7,\ val(b_1) = 8,\ val(b_2) = 9 \tag{23}$$

$$val(a_0) < val(a_1)\ ;\ val(b_0) < val(b_1) < val(b_2) \tag{24}$$

Equations 25 and 26 show the C set member values derived from the (23) value assignment using two valuation operations: addition and multiplication. Both valuation operations show that the set-product operator preserves the (24) ordering. However, the additive valuation exhibits overlap which may be less desirable.

$$val(C, +) = \{\, 1+7, 1+8, 1+9, 2+7, 2+8, 2+9 \,\} = \{\, 8, 9, 10, 9, 10, 11 \,\} \tag{25}$$
$$val(C, *) = \{\, 1*7,\ 1*8,\ \ 1*9,\ \ 2*7,\ \ 2*8,\ \ 2*9 \,\} = \{\, 7, 8, 9, 14, 16, 18 \,\} \tag{26}$$

As a counter-example, revise the B value assignment so that it violates (24). Equation (27) is one such revision.

$$val(a_0) = 1,\ val(a_1) = 2\ ;\ val(b_0) = 9,\ val(b_1) = 7,\ val(b_2) = 8 \tag{27}$$

Then recalculate the value of C to yield (28) and (29). In this case, the B set value disorder is reflected in the C set member values.

$$val(C,+) = \{\, 1+9, 1+7, 1+8, 2+9, 2+7, 2+8 \,\} = \{\, 10, 8, 9, 11, 9, 10 \,\} \tag{28}$$

$$val(C,*) = \{\, 1*9,\ 1*7,\ 1*8,\ 2*9,\ 2*7,\ 2*8 \,\} = \{\, 9, 7, 8, 18, 14, 16 \,\} \tag{29}$$

For these small examples, order preservation is not so important because the C set is small; and its members can be easily evaluated by inspection. However, had the C set magnitude been 100 or 1000 rather than 6, the C set evaluation process would have been much more difficult. The evaluation would have required either a linear search or a pre-search sort. Either option is time-consuming. Both options can be avoided or mitigated by ordering the A and B sets.

In the SAbMDE context, the A and B sets are the CE, V and R; and the C set is the DNode set at each next-composition level. The C set member values are a function of their cost and their likelihood of being on the DPath to the DEP. An agent will search for the minimum cost and the highest likelihood. A disordered CE enumeration, therefore, multiplies agent effort at each composition level, of which there could easily be hundreds or thousands. More importantly, a disordered DSpace is at the surface of a disordered XSpace. The DSpace is linked by decisions and evaluation tests to agent perceptions that are the root of the disorder. The ASpace is like a garden that must be tended if its plants are to grow and flower properly. Teachers are particularly aware of this. They provide students with meticulously ordered and indexed packets of knowledge; and then they give students tools to nurture that knowledge as it grows.

The transdisciplinary implication of these observations is that CE ordering, particularly in ASpace, is of paramount importance.

4 Conclusion

This paper outlined the SAbMDE development cycle model, applied the model to a simple development project, and then logically demonstrated the transdisciplinary implications that inevitably flow from the model. The model is transdisciplinary at its core. The implications derived from the model show that the core transdisciplinarity operating at the micro scale expresses itself at the macro scale in the model's products and operations. The demonstrated implications are listed below.

1. Ideas are discovered, not created.
2. LOD is a quantitative, multi-dimensional characteristic of the development process; and it is an essential consideration for any development project. LOD is proportional to and a measure of development cycle complexity.
3. Although an EP is a singular object, it may harbor a superposition of semantic content. That content is distributed throughout XSpace.
4. Many development projects have XSpaces with similar structures. Consequently, agents will be able to transfer their experience with an XSpace in one domain to a similar XSpace in a different domain.
5. Composable elements are likely to evolve over the course of a development project; therefore XSpaces must also evolve. Because ASpace is the source of composable elements, it is the agent's perceptions that must evolve in order to drive a project to its DEP.
6. An agent must maintain the order of its ASpace. Otherwise, the DNode options for each composition decision may appear to be random; and related development processes will stall.

On one hand, because the selected example project is simple, the implication list is not exhaustive. On the other hand, even the simple Lego portrait project revealed interesting implications. SAbMDE, applied to more involved projects, offers the hope that more implications will be revealed, ideally unexpected and counter-intuitive ones.

Authors' Contribution: The author is responsible for the concepts and research presented in this paper.

Conflicts of Interest: The author declares that there are no conflicts of interest.

Funding Statement: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Supplementary Materials: There are no supplementary materials.

References

[1] M. Howard and S. Lipner, *The Security Development Lifecycle.* Secure Software Deelopment, Redmond, Washington: Microsoft Press, 2006.

[2] Microsoft. (2010, 5/3/2020). *Simplified Implementation of the Microsoft SDL.* Available: https://www.microsoft.com/en-us/securityengineering/sdl/.

[3] N. J. T. Force, "NIST Special Publication 800-37 Revision 2. Risk management Framework for Information Systems and Organizations," Report NIST Special Publication 800-37 Revision 2, Department of Commerce, 2018.

[4] Department of Homeland Security, "Systems Engineering Life Cycle." Instruction Number: 102-01-103, 2015, Available: https://www.dhs.gov/sites/default/files/publications/Systems%20Engineering%20Life%20Cycle.pdf.

[5] D. Seal, D. Farr, J. Hatakeyama, and S. Haase, "The System Engineering Vee - Is It Still Relevant in the Digital Age?," presented at the NIST Model Based Enterprise Summit 2018, 4/4/2018, 2018. Available: https://www.nist.gov/system/files/documents/2018/04/10/4se7_seal_sysengvee_final.pdf.

[6] FHWA. (2007, 5/3/2020). *Systems Engineering for ITS Handbook - Section 3 What is Systems Engineering?* Available: https://ops.fhwa.dot.gov/publications/seitsguide/section3.htm

[7] H. S. Modi, N. K. Singh, and H. P. Chauhan, "Comprehensive analysis of software development life cycle models," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 6, p. 5, 2017.

[8] Sedmak, A. DoD Systems Engineering Policy, Guidance and Standardization In Proceedings of the 19th Annual NDIA Systems Engineering Conference, Springfield, VA, USA, USA, 26 October 2016; p. 21. Available online: Https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2016/systems/18925-AileenSedmak.pdf (accessed on 1 June 2020).

[9] Systems Engineering Plan Preparation Guide. Department of Defense. 2008. p 96. Available online: Http://www.acqnotes.com/Attachments/Systems%20Engineering%20Plan%20Preparation%20Guide.pdf (accessed on 1 June 2020).

[10] Jolly, S. Systems Engineering: Roles and Responsibilities. In Proceedings of the NASA PI-Forum, Annapolis, MD, USA, 27 July 2011; p. 21. Available online: Https://www.nasa.gov/pdf/580677main_02_Steve_Jolly_Systems_Engineering.pdf (accessed on 1 June 2020).

[11] D. Kaur and M. Sharma, *Classification Scheme for Software Reliability Models*, pp. 723–733. New Delhi, India: Springer India, 2016.

[12] P. Kruchten, R. L. Nord, and I. Ozkaya, *Managing Technical Debt: Reducing Friction in Software Development.* SEI Series in Software Engineering, Boston, MA: Addison-Wesley Professional, 1st ed., 2019.

[13] V. K. Palepu and J. A. Jones, "Visualizing constituent behaviors within executions," in *2013 First IEEE Working Conference on Software Visualization (VISSOFT)*, pp. 1–4, IEEE: Los Alamitos, CA, USA.

[14] V. K. Palepu and J. A. Jones, "Revealing runtime features and constituent behaviors within software," in *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT)*, pp. 86–95, IEEE.

[15] K. Gericke and L. Blessing, "Comparisons of design methodologies and process models across disciplines: A literature review," in *International Conference On Engineering Design, ICED11*, Technical University Of Denmark, 2011.

[16] G. A. Hazelrigg and D. G. Saari, "Towards a theory of systems engineering," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC/CIE 2020*, ASME.

[17] R. Thakurta, B. Mueller, F. Ahlemann, and D. Hoffmann, "The state of design – a comprehensive literature review to chart the design science research discourse," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, Hawaii, 2017, pp. 4685–4694.

[18] J. Forlizzi, E. Stolterman, and J. Zimmerman, "From design research to theory: Evidence of a maturing field," in *Korean Society of Design Science*, 2009, pp. 2889–2898.

[19] Antunes, R.; Gonzalez, V. A Production Model for Construction: A Theoretical Framework. *Buildings* **2015**, *5*, 209–228, doi:10.3390/buildings5010209.

[20] J. Vandenbrande. (2018, 5/3/2020). *Transformative Design (TRADES).* Available: https://www.darpa.mil/program/transformative-design.

[21] J. Vandenbrande. (2018, 5/3/2020). Enabling Quantification of Uncertainty in Physical Systems (EQUiPS) Available: https://www.darpa.mil/program/equips.

[22] J. Vandenbrande. (2018, 5/3/2020). Fundamental Design (FUN Design). Available: https://www.darpa.mil/program/fundamental-design.

[23] J. Vandenbrande. (2019, 5/3/2020). Evolving Computers from Tools to Partners in Cyber-Physical System Design. Available: https://www.darpa.mil/news-events/2019-08-02.

[24] K. Jablokow. (2020, 8/23/2020). Engineering Design and System Engineering (EDSE). Available: https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=505478&org=CMMI&from=home

[25] Ertas, A. *Transdisciplinary Engineering Design Process*; John Wiley & Sons, Inc: Hoboken, NJ, USA, 2018; p. 818.

[26] Suh, N.P. *Complexity Theory and Applications*; Oxford University Press: Oxford, UK, 2005.

[27] Y. Wang, "On contemporary denotational mathematics for computational intelligence," *Transactions on Computer Science II, LNCS 5150*, pp. 6–29, 2008.

[28] Y. Wang, "Using process algebra to describe human and software behaviors," *Brain and Mind*, vol. 4, pp. 199–213, 2003.

[29] Y. Reich, "A critical review of general design theory," *Research in Engineering Design*, vol. 7, no. 1, pp. 1–18, 1995.

[30] Y. Wang, X. Tan, and C. F. Ngolah, "Design and Implementation of an Autonomic Code Generator Based on RTPA," *International Journal of Software Science and Computational Intelligence*, vol. 2, no. 2, pp. 44–65, 2010.

[31] B. K. Park and R. Y. C. Kim, "Effort estimation approach through extracting use cases via informal requirement specifications," *Applied Sciences*, vol. 10, no. 3044, p. 15, 2020.

[32] K. Friedman, "Theory Construction in Design Research. Criteria, Approaches, and Methods," presented at the 2002 Design Research Society International Conference, London, United Kingdom, 2002. Available: http://hdl.handle.net/1959.3/41967.

[33] D. Eagleman, *Incognito.* New York, NY: Vintage Books, 2011.

[34] J. Saunier, C. Carrascosa, S. Galland, and S. K. Patrick, *Agent Bodies: An Interface Between Agent and Environment*, vol. 9068, pp. 25–40. Springer, Cham, 2015.

[35] F. Heylighen and C. Vidal, "Getting things done: The science behind stress-free productivity," *Long Range Planning*, vol. 41, no. 6, pp. 585–605, 2008.

[36] Z. Dienes and J. Perner, "A theory of implicit and explicit knowledge," BEHAVIORAL AND BRAIN SCIENCES, vol. 22, pp. 735-808, 1999.

[37] D. Eagleman, *The Brain*. New York, NY, USA: Pantheon Books, 2015.

[38] C. Richards, "Boyd's OODA Loop," *Necesse*, vol. 5, no. 1, 2020.

[39] S. Denard, A. Ertas, S. Mengel, and S. Ekwaro-Osire, "Development Cycle Modeling: Resource Estimation," *Applied Sciences*, vol. 10, no. 14, p. 5013, 2020.

[40] S. Denard, A. Ertas, S. Mengel, and S. Ekwaro-Osire, "Development Cycle Modeling: Process Risk," *Applied Sciences*, vol. 10, no. 15, p. 5082, 2020.

About the Author

Dr. Samuel E. Denard is a mechanical engineer trained at MIT, Stanford, and Texas Tech University. He has spent much of his professional career engineering software. For 25 years, he was an independent consultant with Empirical Products and Services. In addition, he has been employed in consumer, nuclear energy, petroleum, and space exploration industries. Currently, he is a Senior Security Engineer with Fortify, a Micro Focus company.