applied sciences

MDPI

Article

Development Cycle Modeling: Process Risk

Samuel Denard [1,*], Atila Ertas [2], Susan Mengel [3] and Stephen Ekwaro-Osire [2]

[1] Fortify, Micro Focus International plc, Houston, TX 77027, USA
[2] Department of Mechanical Engineering, Texas Tech University, Lubbock, TX 79409, USA; atila.ertas@ttu.edu (A.E.); stephen.ekwaro-osire@ttu.edu (S.E.-O.)
[3] Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA; susan.mengel@ttu.edu
* Correspondence: sam.denard@ttu.edu

Received: 1 June 2020; Accepted: 20 July 2020; Published: 23 July 2020

Abstract: The first part of this paper outlined the Statistical Agent-based Model of Development and Evaluation (SAbMDE) and demonstrated the model's ability to estimate development cycle resource utilization. This second part of the paper explores the model's ability to compute development cycle information content and process risk. Risk managers focus mostly on outcome risk, i.e., the likelihood that a running system will behave in an undesirable manner. SAbMDE assumes that a subset of outcome risks are not inherent and immutable but are, instead, the result of defects and vulnerabilities introduced during the system's development process. The likelihood of defect and vulnerability introduction is a process risk. SAbMDE further assumes that measuring process risk is a prerequisite for minimizing defects and vulnerabilities and, therefore, outcome risk. The model implements the measurement with Shannon's information–probability relationship similar to its use in Axiomatic Design Theory (ADT). This paper details the SAbMDE's information and risk calculations and demonstrates those calculations with examples. The process risk calculation is consistent with and offers a mechanism for the ADT Information Axiom.

Keywords: Risk Analysis; Design Theory and Methodology; Decision Theory; Axiomatic Design Theory; Information 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 a concept into an end product. The models describe the intermediate requirement, design, and implementation phases as well as the testing phase that maintains the integrity of the conversion process. The top half of Figure 1 illustrates this representation in broad terms.

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 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 predict 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. The literature reviews [15,16] state this consensus; and [17] 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 [18] 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 [19–22] that such a model might promise. The increasing complexity of modern systems [23,24] is a major reason for the renewal.

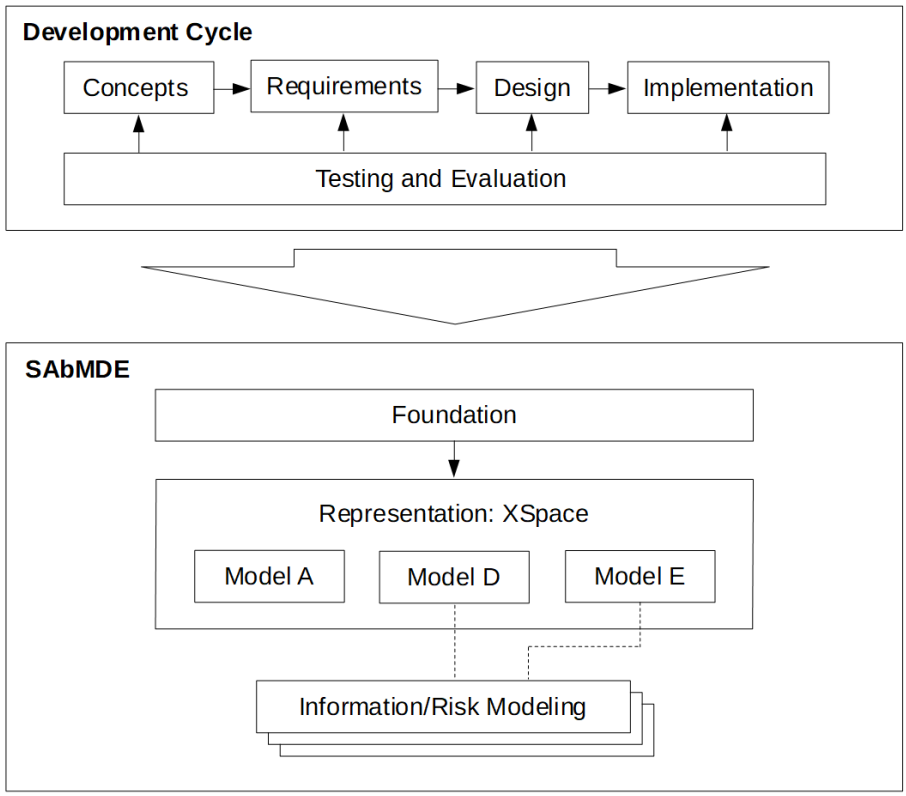

Figure 1. Development modeling framework.

2. Proposed Model

Figure 1 also illustrates an underlying model candidate: the Statistical Agent-based Model of Development and Evaluation (SAbMDE). SAbMDE joins with those [25] who accept development as an inherently human process and builds on a neuroscience foundation [26–28] of agent mind, body, and environment interaction. SAbMDE then uses process algebra ideas, such as Wang's [29,30], to represent each development phase so that analytical techniques can be uniformly applied across the entire development cycle. Wang has shown [31] that a desired end product (DEP) can be developed by sequentially composing intermediate products (IP) from sets of fundamental composable elements: 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, 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, generically XSpaces, have the same form 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, (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 [32].

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

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

The construction begins at composition index 0 ($l = 0$) with an empty set. Construction continues by enumerating the composable elements at $l = 1$, and then by using the cross-product operator to compose all the combinations of composable elements for $l > 1$. As a result, at every DSpace node (DNode) for $l > 1$, an agent decision-maker has exactly $|V||R|$ options from which to choose. Note that at $l = 1$, 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, required to produce the DEP.

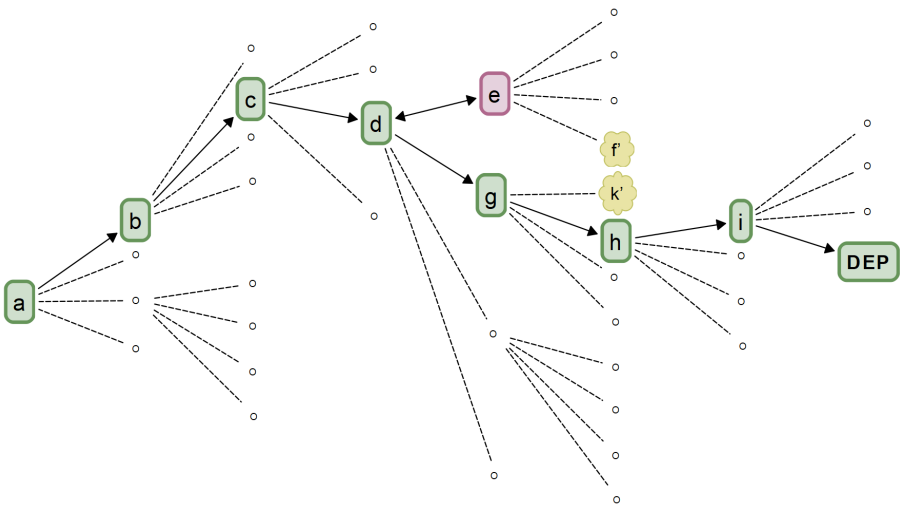

Figure 2. A simple DSpace excerpt with highlighted DPath.

A development agent navigates or traverses DSpace by deciding which DNode to compose next. Figure 2 shows a DPath. 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 (3)–(5), 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 (6).

$$p(l) = \begin{cases} 0 & , l = 0 \\ \frac{1}{|V|} & , l = 1 \\ \frac{1}{|V||R|} & , l > 1 \end{cases} \quad (3)$$

$$u = \frac{1}{|V|} \quad (4)$$

$$q = \frac{1}{|V||R|} = \frac{1}{Q} \quad (5)$$

$$p = x + \frac{f_s}{10}(1 - x), \quad f_s = 0, 1, ...10 \quad \text{and} \quad x = u \quad \text{or} \quad q \quad (6)$$

With this brief outline and with the details to follow, SAbMDE proposes to quantitatively model development processes. This paper supports that proposal by focusing on two closely-related aspects of the SAbMDE framework: development cycle information content and probability of development success. The latter is the complement of development risk.

3. Related Work

There are many researchers and corporate practitioners who investigate development modeling. Their work falls under various headings: Design Theory and Methodology (DTM), Design Research (DR), Modeling and Managing of Engineering Processes (MMEP), or other similar terms.

For example, Quereshi [33] et al. wrote about transdisciplinary commonalities of design processes across various organizations. They concluded that "*An important finding of the study is the higher level commonalities of the design process in terms of the product life phases and its subdivisions, the form of the design process, and the verification of the common design states. Using these commonalities, the authors consider it to be of significant importance to develop further, these elements in a transdisciplinary context so that a coupling of discipline specific processes may be achieved with a minimum loss of context and information.*"

Srinivasan and Chakrabarti [34] wrote about their approach to design analysis and synthesis. Importantly, they included the concept of design causality in their work. They demonstrated their model with several examples, and they concluded, in part, that "*The model in its current form can support only conceptual and early embodiment phases and needs to be extended to support other detailed phases.*"

Reich and Subrahmanian [35] created the PSI Matrix theory of design and applied it to real-world cases with some success. In the end, they stated that "*We are presently conducting multi-case, transdisciplinary, multi-context studies with numerous partners to provide more insight on the utility of the PSI matrix as guidance for design and also to test it as a theory of designing. While they are too early to report, we clearly see the benefit of constantly looking at the situation and its PSI matrix model and deriving guidance for the project. It is clear that many fundamental topics will arise in that process and new research questions be formulated.*"

Gericke et al. [36] wrote about how to discuss design methods so that the methods can be understood and so that industry practitioners could evaluate their utility. One of their conclusions says that "*As a community, we expect that research results such as new methods are carefully evaluated before being published. However, we are still struggling to find consensus about proper research methods for evaluation and qualifiers for uptake of methods in industry.*"

Given this sampling of conclusions, it seems that no definitive model has yet been developed. The work required to build such a model remains to be done.

The state-of-the-art is painted clearly by Gerike and Eckert [37] who summarized the efforts of 39 purposely selected practitioners from academia and 27 companies. The summary was presented as a roadmap of DR goals. Therein, remaining work was identified and categorized. Those authors took the additional step of comparing the roadmap with the actual research topics on which the DR community was then currently working. Gerike and Eckert concluded, in part, "*We need to learn from*

other fields and have a greater dialogue across fields: computing, complexity, operations research, management science, system engineering. As it is a long road to achieve grand visions, we should better go it together."

This paper, while focusing on one aspect of SAbMDE, hopes to help the community take a small step down the road to the Design Research (DR) goals.

4. Problem Description

This paper asks one question: How can the likelihood of a project's success be calculated? Or, conversely, what is the risk of project failure?

There are many types of risk; Spacey [38] lists dozens. However, Kendrick [39] (p. 191) categorizes project risk simply. There are known risks and unknown risks; and, of the known risks, there are those that are controllable or not. This paper is concerned with known controllable risks associated with the construction of a product or other system, i.e., its development. In practice, project development focus tends toward operational risk and related undesirable outcomes. For example, software development weaknesses are cataloged [40] for reference and captured in situ [41]; and their exploitation is publicly reported. Procedures for how to discover and report vulnerabilities are well documented [42]. The undesirable outcomes become the center of attention, and the response to them is usually one of four standard steps [39] (p. 196) and [43]: avoid, transfer, mitigate, or accept.

Unfortunately, outcomes are not directly controllable, but the processes that generate them may be. The root cause of the undesirable outcomes is due, in part, to defects and vulnerabilities that entered the system during its construction. Construction is a controllable process. The likelihood of incorrect construction is the development process risk on which this paper focuses specifically. The software industry invests in the controllability assertion by providing development standards [1,44] and related tools. SAbMDE also invests in process controllability; the first part of this paper hypothesized that SAbMDE can guide an agent towards a DEP. When the guidance is successful, the probability of development success is maximized and the development risk is minimized. This paper focuses on how the guidance is generated. The guiding principle is Suh's Axiomatic Design Theory [45], specifically, the Information Axiom [45] (p. 147). It states that the best end product (EP) of several candidates is the one with the minimum information content. Suh uses domain-specific definitions of information logically combined with the properties of Shannon's information formula (7) to solve real-world engineering problems. Vigo [46], while operationalizing Shannon's formula for human communication, adds a bit of rigor to Suh's logic. Vigo also argues strongly for minimizing human interpretation of probability in (7). Because SAbMDE defines a structure for DSpace, probabilities and information values can be calculated for each development step. In doing so, SAbMDE achieves a similar minimization by extruding human interpretation into those probabilities through a die of formally structured tests. Combining the development step probability values yields a probability for an entire DPath; thus, SAbMDE can satisfy the Information Axiom. Although SAbMDE information and risk is introduced in terms of software development, it should be re-iterated that the model is actually domain-independent. This is not entirely unique. Domain-independence is anticipated in other originally domain-specific models and theories. For example, Antunes and Gonzalez [18] propose a theory of building construction where "*Construction is the materialization of a concept through design...*". They also broaden the conceptual base of their model to put its finer points into perspective. The following sections will describe how these ideas are integrated into SAbMDE. The next section derives and describes the methods and calculations that make the ideas work. After that, the calculations are discussed, and then some conclusions are drawn.

5. Results

The results of this work are presented in four parts. First, building on the Figure 1 framework in the first section of this paper, the value of a development step is defined in terms of its information content. Second, the information source and other details are explained. Third, the DSpace traversal guidance that this technique offers to an agent is demonstrated with two examples. Fourth and finally, the ability to calculate a development process's probability of success is explained and demonstrated.

5.1. Development Step Value

Consider the following conditions of a never-before-attempted project. First, at the very beginning of the project, nothing is known about the end product, so there is a large amount of information that must be gathered and focused on the project. Otherwise, there can be no understanding of how to proceed. Second, at the successful end of the project, everything is known about the DEP; there is no remaining information to be gathered. Third, because the project is developed in a step-wise fashion, each step must somehow use or absorb an increment of the gathered information. A SAbMDE development step is a composition of vocabulary items and relations represented by a development node (DNode). With each DNode absorbing information, Figure 3 shows notionally that the information content of a DSpace, as measured from successive points along a net DPath, decreases from a maximum value at the beginning of a project to zero at its successful conclusion. The original DSpace is densely packed with information for an agent to evaluate. As the agent traverses the DSpace, its information is dissipated by the agent's increasing certainty about the next steps. This decrease is inherent in the structure of DSpace.

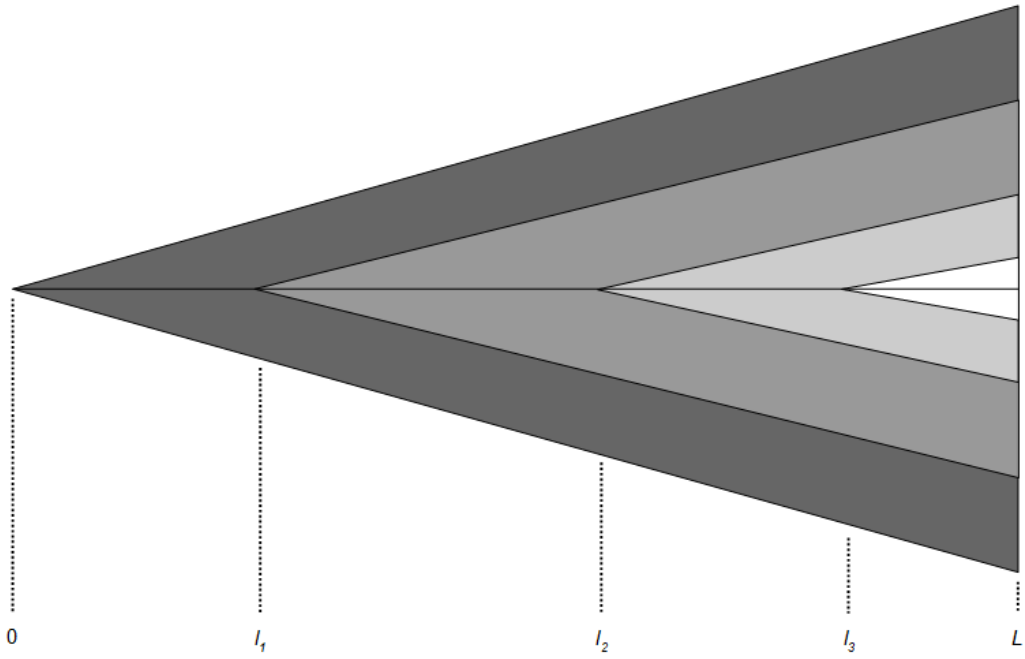

Figure 3. DSpace information content decreases as traversal approaches desired end product (DEP).

This notional concept of information decrease is captured more rigorously beginning with (7), which restates Shannon's basic information definition [47].

$$h = \log_2(\frac{1}{p}) \tag{7}$$

Given this definition, a quantity of information can be assigned to a DNode using the probabilities (4) and (5). A DSpace is a union of DNodes. Equation (8) computes the number of DNodes in a DSpace defined by $|V|$, $|R|$, and L; therefore, (9) defines the total information in that DSpace as it is viewed from the DSpace origin.

$$N_S(V, R, L) = 0 + |V| + \sum_{l=2}^{L} |V|^l |R|^{l-1} = |V| + \frac{1}{|R|} \sum_{l=2}^{L} Q^l \tag{8}$$

$$H_S(V, R, L, skill) = \left[|V| \log_2(\frac{1}{u}) + \frac{\log_2(\frac{1}{q})}{|R|} \sum_{l=2}^{L} Q^l \right]_{skill=0} \tag{9}$$

Consider the view of the DSpace from some later composition level, $l_c > 2$. Equation (10) defines the information in that sub-DSpace.

$$H_S(V, R, L - l_c, skill) = \left[\frac{\log_2(\frac{1}{q})}{|R|} \sum_{l=l_c}^{L} Q^l \right]_{skill=0} \tag{10}$$

Equation (11) is the ratio of the sub-DSpace information to the full DSpace information.

$$\eta_S = \frac{H_S(V, R, L - l_c, skill)}{H_S(V, R, L, skill)} \tag{11}$$

Equations (9)–(11) are symbolic expressions of Figure 3. η_s decreases as l_c increases. If this characteristic is true for the DSpace as a whole, then it must also apply to any sub-section of the DSpace, such as a DPath. Consider the nodes associated with a single DPath. A DPath traversal involves only a portion of the DNodes in the DSpace. The random probabilities, (4) and (5), dictate that at each composition level, l, an agent may examine as many as Q nodes to make a next-DNode decision. Consequently, (12) states the number of DNodes associated with the DPath, and, as was done with (9) for the DSpace as a whole, (13) defines the maximum information in a DPath with respect to the origin of the DPath.

$$N_P(V, R, L) = [|V| + (L - 2)Q]_{skill=0} \tag{12}$$

$$H_P(V, R, L, skill) = \left[|V| \log_2(\frac{1}{u}) + (L - 2)Q \log_2(\frac{1}{q}) \right]_{skill=0} \tag{13}$$

In general, the decision probability p_{skill}, is a function of the composition index, l; therefore, the information in the DPath traversed with p_{skill} is defined by (14). However, if p_{skill} is treated as constant, (4) and (5), p_{skill} dictates the number of DNodes examined at each composition level. The general expression, (14), can then be simplified into (15) where n_l is the number of decision retries associated with p_{skill}.

$$H_P(V, R, L, skill) = \left[\sum_{l=1}^{L} \sum_{k=0}^{n(l)} \log_2(\frac{1}{p_{skill}(l)}) \right]_{0<skill<1} \tag{14}$$

$$H_P(V, R, L, skill) = \begin{cases} (n_{l=1}(skill) + 1) \log_2(\frac{1}{u}) + (L - 1)(n_{l\geq 2}(skill) + 1) \log_2(\frac{1}{q}) & , \quad l_c = 1 \\ (L - l_c)(n_{l\geq 2}(skill) + 1) \log_2(\frac{1}{q}) & , \quad l_c \geq 1 \end{cases} \tag{15}$$

$$\eta_P = \frac{H_P(V, R, L - l_c, skill)}{H_P(V, R, L, skill)} \tag{16}$$

As is the case for the DSpace as a whole, Figure 4 plots (16) and shows that any DPath's information decreases to a minimum as an agent traverses the DPath towards an EP. In accordance with ADT's Information Axiom, SAbMDE asserts that the information reaches its absolute minimum when the DPath is a net traversed DPath to a DEP. On the one hand, an agent can wander in a DSpace creating an arbitrarily long and convoluted DPath that may never terminate at a DEP. On the other hand, some wandering is necessary; the point of an agent's (hopefully) wise explorations of DSpace is to locate the net DEP DPath. A net DEP DPath is the DPath that a perfect agent would choose to reach a DEP. Practically, a net DPath is found, after the fact, to be the DPath that an agent has actually traversed minus any convolutions. For example, in Figure 2, the actual and net DPaths are described, respectively, by the DNode sequences shown in Table 1.

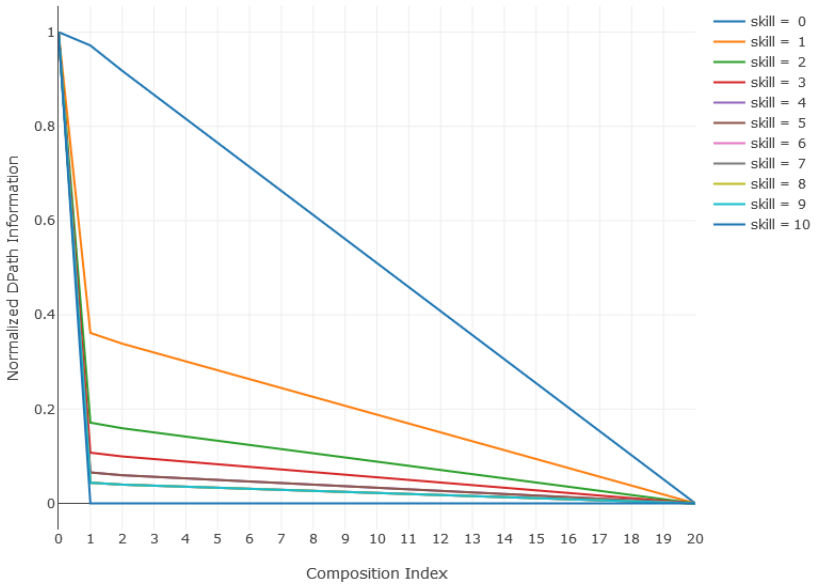

Figure 4. Normalized DPath information vs. composition index, l, by skill index.

Table 1. Comparison of Figure 2 DPaths.

DPath Type	DPath Index (Traversal and Navigation)															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Actual	a	b	c	d	e	f′	e	d	g	k′	g	h′	g	h	i	DEP
Net	a	b	c	d	g	h	i	DEP								

The actual DPath in Figure 2 and Table 1 has several convolutions. For example, a traversal to DNode e turned out to be a misstep that required backtracking; and two next-step contemplations (navigations) to DNodes k and f were discarded. Looking back at the actual traversal history, an agent can see the minimal or net sequence of DNodes that should have been traversed. It is the net DPath's information that decreases most as the traversal approaches a DEP. The net DPaths have the minimum information content of all DPaths in the DSpace as required by the Information Axiom. Figure 4 shows that a DPath conforms to this axiom.

5.2. Information Source

Until this point in the discussion, an agent's ability to traverse DSpace efficiently was represented with a scaled probability of successful next-DNode selection, (6). The structure of DSpace provided a floor value for this probability. However, no justification for assigning particular values above that floor was stated. Equivalently, according to (7), no source of decision-enhancing information was identified. To cure this justification deficiency, assume that any increase in probability must be contributed by an agent. Then, let the agent generate additional probability traditionally: with tests [48]. Such tests allow an agent to acquire and add the necessary information to the DSpace. SAbMDE prescribes tests in the following way.

$$
\begin{aligned}
t_i &= \text{test definition} \\
\tau_i &= \text{test result} = \begin{cases} -1 & \text{not performed} \\ 0 \,|\, 1 & \text{failure} \,|\, \text{success} \\ [0,1] & \text{degree of success} \end{cases} \\
w_i &= \text{weight value for } t_i \\
T &= \text{test set, e.g., } \{\, \{t_0,\tau_0,w_0\}, \ldots \{t_i,\tau_i,w_i\}, \ldots \{t_{n-1},\tau_{n-1},w_{n-1}\} \,\}
\end{aligned}
$$

Each test, t_i, is conceived and constructed by an agent 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 (17).

$$\nu(l,k) = \frac{1}{|T|}\sum_{i=0}^{|T|-1} w_i\tau_i \quad \text{where } \sum w_i = |T| \tag{17}$$

To the degree that the test set is complete, sufficient, and executed properly, (17) 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 (6) with (18). Note that (18) should be applied according to the rules stated by (19).

$$p = x + \nu(l,k)(1-x) \quad \text{where } x = u \text{ or } q \tag{18}$$

$$p = \begin{cases} x & \text{when no tests are available} \\ p & \text{when any test results are available} \\ 0 & \text{when all test results fail} \end{cases} \tag{19}$$

5.3. Examples

The ADT Information Axiom states that a development project's information content is at its maximum value when the project is least well understood. Conversely, the project's information content reaches its minimum value when the project is most precisely defined, i.e., when it achieves the DEP. Figure 4 shows that the SAbMDE DSpace structure inherently handles development information as required by the ADT Information Axiom. The foundation laid in this paper's previous sections supports a mechanism for effective DSpace navigation and traversal. The mechanism can be demonstrated with simple examples. Begin by defining a DSpace for which Figure 2 is an excerpt. Define the composable elements as follows:

$$V = \{-, \text{X}, +, \text{O}\} \tag{20}$$

$$R = \{\text{ “select” }\} \tag{21}$$

Note that SAbMDE is domain-independent, so the names and meanings of the composable elements are flexible; they can be whatever is required for the particular domain under development. In this case, the domain is represented by four arbitrary symbols and the ability to select those symbols in sequence. Figure 5 is a tabular representation of the example DSpace.

Let the DEPs of this DSpace be specified sequences of symbols. An agent can traverse this DSpace with the confidence of reaching the DEPs by applying tests such as (22)–(24) and (26) below, each constructed according to (17).

$$t_0 = \left[node(l_c+1,k) = \text{designated vocabulary item, } v_j = \text{X}\right] \tag{22}$$

$$t_1 = \left[node(l_c+1,k) = node(l_c,k_c) \text{ for } k = 1,2,\ldots,Q\right] \tag{23}$$

$$t_2 = \begin{cases} t_0 & \text{for} \quad l = 0 \\ t_1 & \text{for} \quad 1 \le l \le l_c \end{cases} \tag{24}$$

$$\nu(l,k) = \frac{1}{3}(t_0 + t_1 + t_2) \tag{25}$$

For the first example, define a test set, T_0, with members (22)–(24). T_0 defines a DEP as a contiguous sequence of a specified member of V, in this case, X. Table 2 lays out the traversal mechanism using T_0 and (25) applied to the example DSpace. For this simplified example, the DEP DPath stands out as the darkest highlighted X cells in Figure 5 and the low points in Figure 6. The stark distinction of the single DPath is a consequence of the (22) test's specificity. Equation (22) selects DNodes with 100% certainty, it is known at the first composition, and it applies to every subsequent composition.

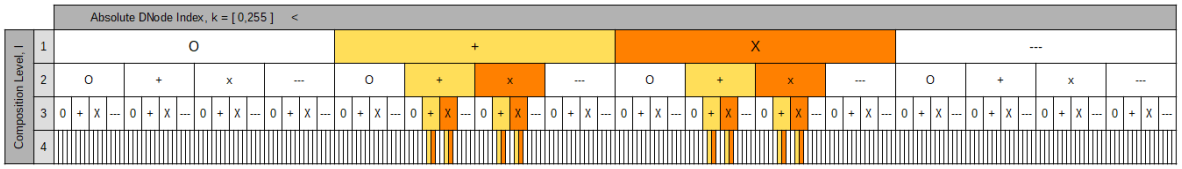

Figure 5. A DSpace in tabular format with highlighted DPaths.

Table 2. A DSpace traversal using Test Set 0.

Current DNode	**Next DNode Options**				
(l,k_a)	k	$v(l+1,k)$	$p(l+1,k)$	$h(l+1,k)$	*Selected*
(0,~)	0	0.00	0.2500	2.0000	Y
	1	1.00	1.0000	0.0000	
	2	0.00	0.2500	2.0000	
	3	0.00	0.2500	2.0000	
(1,1)	0	0.00	0.2500	2.0000	Y
	1	1.00	1.0000	0.0000	
	2	0.00	0.2500	2.0000	
	3	0.00	0.2500	2.0000	
(2,5)	0	0.00	0.2500	2.0000	Y
	1	1.00	1.0000	0.0000	
	2	0.00	0.2500	2.0000	
	3	0.00	0.2500	2.0000	
(3,21)	0	0.00	0.2500	2.0000	Y
	1	1.00	1.0000	0.0000	
	2	0.00	0.2500	2.0000	
	3	0.00	0.2500	2.0000	

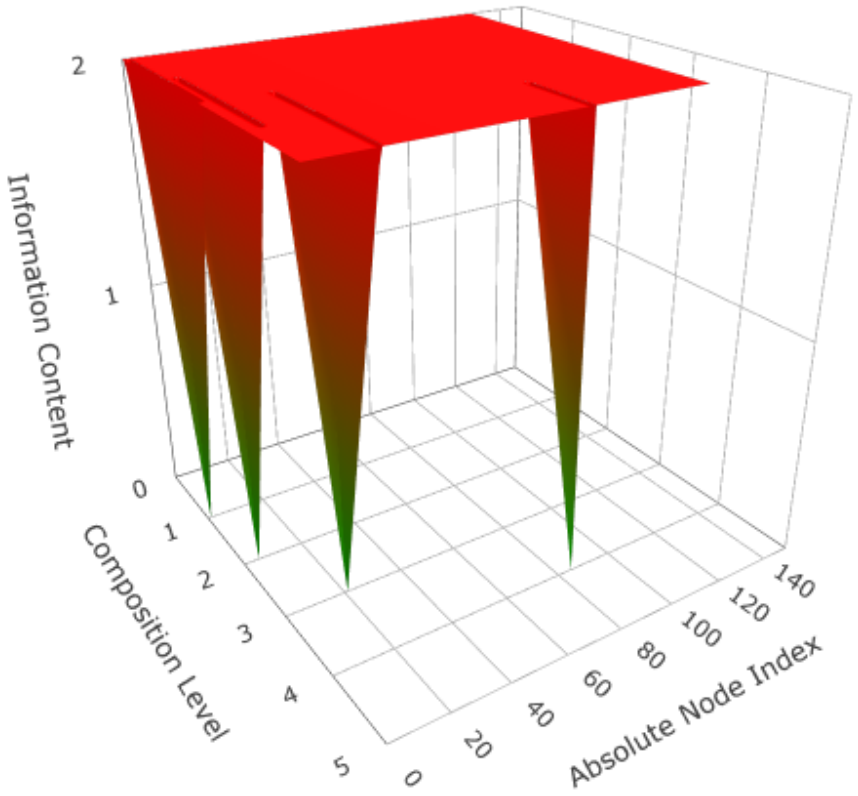

Figure 6. DPath Information for a DSpace traversal using Test Set 0.

For the second example, include the more ambiguous test (26) in the test set T_1 along with (23) and (24). This test set identifies "cross-like" vocabulary items for inclusion in a DEP sequence. SAbMDE hypothesizes that the traversal mechanism can rank alternative DPaths and, thus, guide an agent to the best DEP.

$$t_4 = \left[node(l_c + 1, k) = \text{cross-like}, \tau_4 = \begin{cases} 0.0 & \text{for} & - \\ 1.0 & \text{for} & \text{X} \\ 0.5 & \text{for} & + \\ 0.0 & \text{for} & \text{O} \end{cases} \right] \tag{26}$$

$$v(l, k) = \frac{1}{3}(t_4 + t_1 + t_2) \tag{27}$$

Application of test set T_1 and (27) identifies the 16 DPaths highlighted in Figure 5. Table 3 shows the states of the traversal mechanism, at each composition index, as it drives those DPaths toward their corresponding DEPs. Table 4 also shows the test results, the DNode probabilities, and the information for each DPath. Figure 7 is a graphical representation of Table 4. As hypothesized, the relative information values distinguish the DPaths from one another. At any composition level, an agent can compute the likelihood that the current DPath is the best one. Or, to the degree that the agent projects the DPaths forward (as has been done here), the agent can get a sense of the best DPath options.

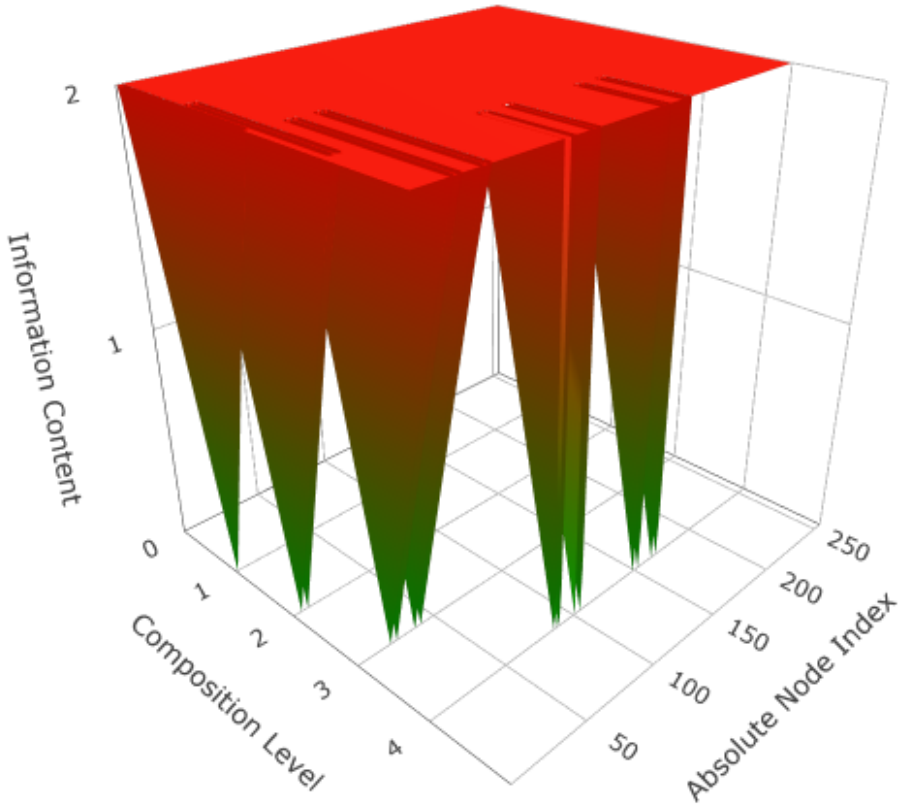

Figure 7. DPath Information for a DSpace traversal using Test Set 1.

Table 3. Traversal mechanism parameters for DSpace traversals using Test Sets 0 and 1.

Test Set	DPath Index	Composition Index (l)																		
		0				1				2				3				4		
		k	$v(l,k)$	$p(l,k)$	$h(l,k)$	k	$v(l,k)$	$p(l,k)$	$h(l,k)$	k	$v(l,k)$	$p(l,k)$	$h(l,k)$	k	$v(l,k)$	$p(l,k)$	$h(l,k)$	k	$v(l,k)$	$p(l,k)$	$h(l,k)$
0	0	~	~	0.25	2.00	1	1.00	1.00	0.00	5	1.00	1.00	0.00	21	1.00	1.00	0.00	85	1.00	1.00	0.00
	Others	~	~	0.25	2.00	—	0.00	0.25	2.00	—	0.00	0.25	2.00	—	0.00	0.25	2.00	—	0.00	0.25	2.00
1	0	~	~	0.25	2.00	1	1.00	1.00	0.00	5	1.00	1.00	0.00	21	1.00	1.00	0.00	85	1.00	1.00	0.00
	1	~	~	0.25	2.00	1	1.00	1.00	0.00	5	1.00	1.00	0.00	21	1.00	1.00	0.00	86	0.83	0.87	0.19
	2	~	~	0.25	2.00	1	1.00	1.00	0.00	5	1.00	1.00	0.00	22	0.83	0.87	0.20	89	1.00	1.00	0.00
	3	~	~	0.25	2.00	1	1.00	1.00	0.00	5	1.00	1.00	0.00	22	0.83	0.87	0.20	90	0.83	0.87	0.20
	4	~	~	0.25	2.00	1	1.00	1.00	0.00	6	0.83	0.87	0.20	25	1.00	1.00	0.00	101	1.00	1.00	0.00
	5	~	~	0.25	2.00	1	1.00	1.00	0.00	6	0.83	0.87	0.20	25	1.00	1.00	0.00	102	0.83	0.87	0.20
	6	~	~	0.25	2.00	1	1.00	1.00	0.00	6	0.83	0.87	0.20	26	0.83	0.87	0.20	105	1.00	1.00	0.00
	7	~	~	0.25	2.00	1	1.00	1.00	0.00	6	0.83	0.87	0.20	26	0.83	0.87	0.20	106	0.83	0.87	0.20
	8	~	~	0.25	2.00	2	0.83	0.87	0.20	9	1.00	1.00	0.00	37	1.00	1.00	0.00	149	1.00	1.00	0.00
	9	~	~	0.25	2.00	2	0.83	0.87	0.20	9	1.00	1.00	0.00	37	1.00	1.00	0.00	150	0.83	0.87	0.20
	10	~	~	0.25	2.00	2	0.83	0.87	0.20	9	1.00	1.00	0.00	38	0.83	0.87	0.20	153	1.00	1.00	0.00
	11	~	~	0.25	2.00	2	0.83	0.87	0.20	9	1.00	1.00	0.00	38	0.83	0.87	0.20	154	0.83	0.87	0.20
	12	~	~	0.25	2.00	2	0.83	0.87	0.20	10	0.83	0.87	0.20	41	1.00	1.00	0.00	165	1.00	1.00	0.00
	13	~	~	0.25	2.00	2	0.83	0.87	0.20	10	0.83	0.87	0.20	41	1.00	1.00	0.00	166	0.83	0.87	0.20
	14	~	~	0.25	2.00	2	0.83	0.87	0.20	10	0.83	0.87	0.20	42	0.83	0.87	0.20	169	1.00	1.00	0.00
	15	~	~	0.25	2.00	2	0.83	0.87	0.20	10	0.83	0.87	0.20	42	0.83	0.87	0.20	170	0.83	0.87	0.20
	Others	~	~	0.25	2.00	—	0.00	0.25	2.00	—	0.00	0.25	2.00	—	0.00	0.25	2.00	—	0.00	0.25	2.00

Table 4. DPath success and risk probabilities for DSpace traversals using Test Sets 0 and 1.

Test Set	DPath Index	Information		Probability	
		DPath	Minimum	Success	Risk
0	0	0.00	0.00	1.00	0.00
	Others	0.00		1.00	0.00
1	0	0.00	0.00	1.00	0.00
	1	0.19		0.87	0.13
	2	0.20		0.87	0.13
	3	0.39		0.76	0.24
	4	0.20		0.87	0.13
	5	0.39		0.76	0.24
	6	0.39		0.76	0.24
	7	0.59		0.66	0.34
	8	0.20		0.87	0.13
	9	0.39		0.76	0.24
	10	0.39		0.76	0.24
	11	0.59		0.66	0.34
	12	0.39		0.76	0.24
	13	0.59		0.66	0.34
	14	0.59		0.66	0.34
	15	0.79		0.58	0.42
	Others	8.00		0.00	1.00

5.4. Probability of Success

It appears that SAbMDE does show promise for quantifying development processes and guiding an agent's decision-making. However, these results can be extended a step further by considering the information captured in the DEP DPath in the context of the Shannon information formula (7) when restated as in (28). The DPath information is the sum of the information in each of the DPath's DNodes.

$$H_P(V, R, L, skill) = \log_2(\frac{1}{p}) \tag{28}$$

The probability associated with the DEP DPath information can be interpreted as the probability of successful traversal (29).

$$p_{success} = \frac{1}{2^{H_P(V,R,L,skill)}} \tag{29}$$

It follows that the probability of project failure, i.e., the risk, is given by (30).

$$risk = p_{failure} = 1 - p_{success} = 1 - \frac{1}{2^{H_P(V,R,L,skill)}} \tag{30}$$

Table 4 is an extension of Table 3 that shows the probabilities of success and failure for each DPath revealed by the T_1 test set as calculated by (29) and (30), respectively. These examples are necessarily small ones; they must fit within the confines of this paper. However, it should be stated that DPaths can be quite long, perhaps hundreds or thousands of compositions long. Because each DNode's worst-case information value corresponds to the DSpace's structural probability at a given composition level, each DNode information value can be large. Consequently, the sum of the DNodes' information, a DPath's information, can grow quite large. In Table 3, consider the DPaths designated as "Others". Those paths are untested. Their DNode information values are an order of magnitude greater than the values for tested DNodes. This differential carries over to Table 4 where (29) imposes a range limit

on the DPath information value. Because the $p_{success}$ range is [0,1], the non-linear $H_p()$ range is [∞,0]. In practice, a probability lower limit value is chosen to fix an information upper limit, h_{ul} and this limit can be used to scale raw calculated DPath information values to the range needed by (29), see (31). For example, a 0.1 probability lower limit value means a 10% chance of DPath success. That probability value corresponds to an information upper limit value of 3.3219281 according to (29).

$$h_{scaled} = \frac{h_{ul}}{H_P(V, R, L, skill)} \tag{31}$$

Table 5 is an example where information values, randomly selected from a [0, 10,000] range, were scaled with (31). Table 5 shows a set of information values that distinguish potentially successful DPaths from the remainder.

Table 5. An example of large DPath information values scaled for probability calculation.

DPath	Information		Probability	
	Actual	Scaled	Success	Risk
0	3021.351	1.07	0.48	0.52
1	8824.484	3.12	0.11	0.89
2	5104.543	1.81	0.29	0.71
3	8669.417	3.07	0.12	0.88
4	5855.308	2.07	0.24	0.76
5	7499.249	2.65	0.16	0.84
6	498.490	1.45	0.37	0.63
7	644.061	0.23	0.85	0.15
8	6843.825	2.42	0.19	0.81
9	1621.288	0.57	0.67	0.33
10	5426.940	1.92	0.26	0.74
11	6042.189	2.14	0.23	0.77
12	7838.449	2.77	0.15	0.85
13	8574.182	3.03	0.12	0.88
14	5791.182	2.05	0.24	0.76
15	9390.285	3.32	0.10	0.90
16	315.768	0.11	0.93	0.07

The scaling technique works because an agent needs only to rank DPath alternatives by their merit, and for this purpose, only the relative information values are needed, not their absolute values. The scaled information allows the probability of success or the risk of failure to be calculated for any DPath.

6. Discussion

First, a DSpace is a set of points (DNodes), and its minimal visualization is a three-dimensional scatter graph. Figures 6 and 7 show this paper's example DSpace as a three-dimensional surface graph. The surfaces, especially, the flattened downward cones, are misleading; they imply that there is something between the DNodes. In fact, at the given level of detail, DSpace is empty except for the defined DNodes. However, the surface graph does express the idea that an agent wanders on a high plain of uncertainty, periodically finds a comfortable burrow, and then re-launches the search for the next safe haven, if not their final destination. The surface graph also points out that the search always begins on uncertain ground, perhaps without even the vocabulary to describe the destination. This initial uncertainty begs the question, "Why would an agent begin a journey toward the unknown?" The answer could be that agents are not blank slates; they have some shiny, half-formed, maybe misguided,

even outright wrong concept of what is beyond where they are. Perhaps any goal is enough motivation to begin. Alternatively, perhaps the prospect of indefinite uncertainty makes it intolerable to stay. Tables 3 and 4 offer some comment on agent motivation. There the DPath information values are summed from composition levels 1 through 4. When composition level 0 is included, it is clear that the success probability of every DPath is reduced. Only by leaving level 0 is it possible, though not certain, for an agent to improve its situation. This departure gives new meaning to the old aphorism, "*Well begun is half done.*"

Second, a slight correction to the first point of discussion: the space between DNodes is likely not empty. One could speculate that the space between DNodes is filled with other DNodes that exist at increased levels of detail.

Third, a caveat: the methods described herein are not a substitute for agent skill and expertise. The agent must still add value via tests if a project is to succeed; the alternative is expensive [49,50]. SAbMDE can offer guidance on completeness and options, it can offer accounting services for the work that an agent has done, and it can enhance communication among interested parties. SAbMDE cannot force an agent to do the right thing, but it can help an agent do the thing right. For example, Table 5 information values have a wide range, and this range made DPaths distinguishable. The information values could just as easily have been too similar to offer any distinction. The information is derived from tests, so uniform information values may indicate poorly designed or otherwise insufficient tests. SAbMDE can alert a developer to this situation but cannot correct it.

Fourth, reconsider Figure 4, the graph of normalized DPath information versus composition index, l, parameterized by skill index. Figure 8 is the same data graphed as information versus skill index and parameterized by composition level. Figure 8 shows that agent information contribution plateaus at certain skill index values. For example, at $l = 15$, about 91% of the necessary information is added by an agent with a 4 or 5 skill index. Adding 6% or 7% more information requires an agent with a skill index of 6 or above. But above 6, there is no information benefit to be had until skill index 9 is exceeded. Of course, the higher-skilled agents use fewer resources to get the job done. This plateaued information contribution was unexpected and warrants further consideration.

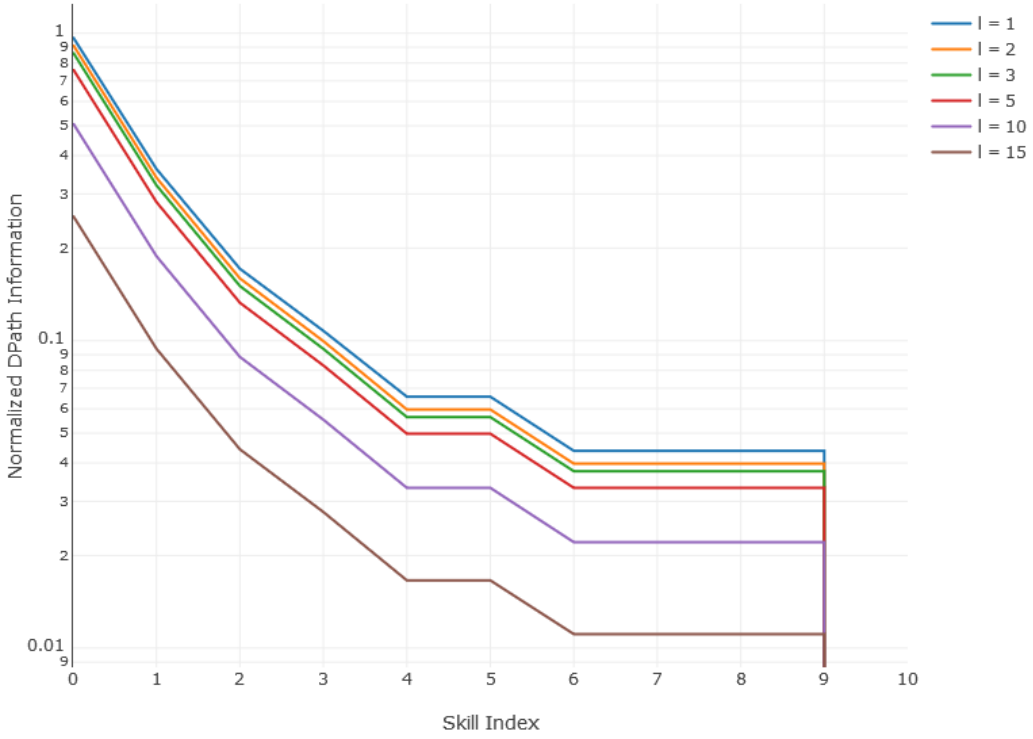

Figure 8. Normalized DPath Information vs. Skill Index by composition index, l.

Finally, SAbMDE is a work in progress. Further mathematical and software development is necessary. Although the model reproduces COCOMO results quite well and has been shown to match certain accepted characteristics of ADT and other design theories, additional validation is necessary and underway. Because the modeling concept is new and because user interface requirements are challenging, practical deployment of the model could be problematic; however, these issues are being given due consideration.

7. Conclusions

First, there is a practical motivation for understanding development process risk and a pressing need to implement techniques for doing so. Second, SAbMDE, briefly outlined herein, is a model of development processes. The model offered an explanation of Suh's ADT Information Axiom. Two simple development examples showed how the model uses development process information to guide a developer towards a desired end product. Third, it is possible to calculate a justifiable, actionable, and traceable probability of project success and/or failure. Fourth and finally, this paper promotes the idea that improved development processes will reduce defects and vulnerabilities that contribute to outcome risk.

Author Contributions: This paper was written from S.D.'s Ph.D. dissertation. A.E., the advisor of S.D., helped draft the preparation of the article. S.M. and S.E.-O., S.D.'s committee members, helped with reviewing and editing of the article. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following nomenclature is used in this manuscript:

Symbols	Definitions
f_s	skill index value
h	Shannon information
h_{scaled}	raw DPath information value proportionately reduced so that h_{scaled} falls within the probability range [0, p_{ll}]
h_{ul}	DPath information value that corresponds to p_{ll}
H_P	total information in a DPath
H_S	total information in a DSpace
i, j	dummy index variables
$k, absolute$	sequential index of all DNodes in a specified composition level
$k, relative$	sequential index of all DNodes reachable from the current DNode
l	composition level or index
l_c	current composition level, the reference level for next-DNode decisions
L	number of composition levels needed to compose a DEP
n	number of retries needed to select the correct DNode
n_l	number of retries associated with a composition index
η_S	normalized DSpace information
η_P	normalized DPath information
N_S	number of DNodes associated with a DSpace
N_P	number of DNodes associated with a DPath
p	general probability variable
$p_{failure}$	probability that a DPath does not lead to a DEP, i.e., risk of project failure
p_{ll}	DPath success probability lower limit
p_{skill}	probability associated with a skill index value
$p_{success}$	probability that a DPath leads to a DEP
q	probability associated with DNode selection
Q	product of V and R

R	set of relations
r	member of the set of relations
$skill, skillindex$	index with range [0,10] that ranks an agent's skill level, see f_s
t	test, a member of test set T
τ	test result corresponding to test t
T	set of tests
u	probability associated with vocabulary item selection
v	member of the set of vocabulary items
V	set of vocabulary items
ν	test set result computed as the weighted sum of results of the test set's members.
w	weight applied to a test result, τ, when calculating a test set result
x	placeholder variable

References

1. Howard, M.; Lipner, S. *The Security Development Lifecycle*; Secure Software Deelopment, Microsoft Press: Redmond, WA, USA, 2006; p. 320.
2. Microsoft. Simplified Implementation of the Microsoft SDL. Available online: https://www.microsoft.com/en-us/securityengineering/sdl/ (accessed on 3 May 2020).
3. NIST Special Publication 800-37 Revision 2. Risk Management Framework for Information Systems and Organizations. Department of Commerce. p. 183. Available online: https://doi.org/10.6028/NIST.SP.800-37r2 (accessed on 23 May 2020).
4. Systems Engineering Life Cycle Department of Homeland Security. p. 15. Available online: https://www.dhs.gov/sites/default/files/publications/Systems%20Engineering%20Life%20Cycle.pdf (accessed on 23 May 2020).
5. Seal, D.; Farr, D.; Hatakeyama, J.; Haase, S. The System Engineering Vee is it Still Relevant in the Digital Age? In Proceedings of the NIST Model Based Enterprise Summit 2018, Gaithersburg, MD, USA, 4 April 2018; p. 10.
6. FHWA. Systems Engineering for ITS Handbook—Section 3 What is Systems Engineering? Available online: https://ops.fhwa.dot.gov/publications/seitsguide/section3.htm (accessed on 3 May 2020).
7. Modi, H.S.; Singh, N.K.; Chauhan, H.P. Comprehensive Analysis of Software Development Life Cycle Models. *Int. Res. J. Eng. Technol.* **2017**, *4*, 117–122.
8. Sedmak, A. DoD Systems Engineering Policy, Guidance and Standardization. In Proceedings of the 19th Annual NDIA Systems Engineering Conference, Springfield, VA, 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. Kaur, D.; Sharma, M. Classification Scheme for Software Reliability Models. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems, Advances in Intelligent Systems and Computing 394*; Dash, S., Ed.; Springer: New Delhi, India, 2016; pp. 723–733. [CrossRef]
12. Kruchten, P.; Nord, R.L.; Ozkaya, I. *Managing Technical Debt: Reducing Friction in Software Development*, 1st ed.; SEI Series in Software Engineering; Addison-Wesley Professional: Boston, MA, USA, 2019.
13. Palepu, V.K.; Jones, J.A. Visualizing Constituent Behaviors within Executions. In Proceedings of the 2013 First IEEE Working Conference on Software Visualization (VISSOFT), Eindhoven, The Netherlands, 27–28 September 2013; pp. 1–4. [CrossRef]
14. Palepu, V.K.; Jones, J.A. Revealing Runtime Features and Constituent Behaviors within Software. In Proceedings of the 2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT), Bremen, Germany, 27–28 September 2015; pp. 86–95. [CrossRef]

15. Gericke, K.; Blessing, L. Comparisons Of Design Methodologies And Process Models Across Disciplines: A Literature Review. In Proceedings of the International Conference On Engineering Design, ICED11, Lyngby/Copenhagen, Denmark, 15–18 August 2011; Technical University of Denmark: Lyngby, Denmark, 2011.
16. Thakurta, R.; Mueller, B.; Ahlemann, F.; Hoffmann, D. 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, Hilton Waikoloa Village, HI, USA, 4–7 January 2017; pp. 4685–4694.
17. Forlizzi, J.; Stolterman, E.; Zimmerman, J. From Design Research to Theory: Evidence of a Maturing Field. *Korean Soc. Des. Sci.* **2009**, *9*, 2889–2898.
18. Antunes, R.; Gonzalez, V. A Production Model for Construction: A Theoretical Framework. *Buildings* **2015**, *5*, 209–228. [CrossRef]
19. Vandenbrande, J. Transformative Design (TRADES). Available online: https://www.darpa.mil/program/transformative-design (accessed on 3 May 2020).
20. Vandenbrande, J. Enabling Quantification of Uncertainty in Physical Systems (EQUiPS). Available online: https://www.darpa.mil/program/equips (accessed on 3 May 2020).
21. Vandenbrande, J. Fundamental Design (FUN Design). Available online: https://www.darpa.mil/program/fundamental-design (accessed on 3 May 2020).
22. Vandenbrande, J. Evolving Computers from Tools to Partners in Cyber-Physical System Design. Available online: https://www.darpa.mil/news-events/2019-08-02 (accessed on 3 May 2020).
23. Ertas, A. *Transdisciplinary Engineering Design Process*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2018; p. 818.
24. Suh, N.P. *Complexity Theory and Applications*; Oxford University Press: Oxford, UK, 2005.
25. Friedman, K. Theory Construction in Design Research. Criteria, Approaches, and Methods. In Proceedings of the 2002 Design Research Society International Conference, London, UK, 5–7 September 2002.
26. Saunier, J.; Carrascosa, C.; Galland, S.; Patrick, S.K. Agent Bodies: An Interface Between Agent and Environment. In *Agent Environments for Multi-Agent Systems IV, Lecture Notes in Computer Science*; Weyns, D., Michel, F., Eds.; Springer: Cham, Switzerland, 2015; Volume 9068, pp. 25–40. [CrossRef]
27. Heylighen, F.; Vidal, C. Getting Things Done: The Science Behind Stress-Free Productivity. *Long Range Plan.* **2008**, *41*, 585–605. [CrossRef]
28. Eagleman, D. *Incognito*; Vintage Books: New York, NY, USA, 2011; p. 290.
29. Wang, Y. On Contemporary Denotational Mathematics for Computational Intelligence. In *Transactions on Computer Science II, LNCS 5150*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 6–29.
30. Wang, Y. Using Process Algebra to Describe Human and Software Behaviors. *Brain Mind* **2003**, *4*, 199–213. [CrossRef]
31. Wang, Y.; Tan, X.; Ngolah, C.F. Design and Implementation of an Autonomic Code Generator Based on RTPA. *Int. J. Softw. Sci. Comput. Intell.* **2010**, *2*, 44–65. [CrossRef]
32. Park, B.K.; Kim, R.Y.C. Effort Estimation Approach through Extracting Use Cases via Informal Requirement Specifications. *Appl. Sci.* **2020**, *10*, 15. [CrossRef]
33. Qureshi, A.J.; Gericke, K.; Blessing, L. Design Process Commonalities in Transdisciplinary Design. In Proceedings of the International Conference on Engineering Design (ICED13), Seoul, Korea, 19–22 August 2013.
34. Srinivasan, V.; Chakrabarti, A. SAPPhIRE—An Approach to Analysis and Synthesis. In Proceedings of the International Conference on Engineering Design (ICED09), Stanford, CA, USA, 24–27 August 2009.
35. Reich, Y.; Subramanian, G.H. The PSI Matrix—A Framework and a Theory of Design. In Proceedings of the 21st International Conference on Engineering Design (ICED17), Vancouver, BC, Canada, 21–25 August 2017.
36. Gericke, K.; Eckert, C.M.; Martin, S. What Do We Need To Say About A Design Method? In Proceedings of the 21st International Conference on Engineering Design (ICED17), Vancouver, BC, Canada, 21–25 August 2017.
37. Gericke, K.; Eckert, C.M. The Long Road to Improvement in Modelling and Managing Engineering Design Processes. In Proceedings of the International Conference on Engineering Design (ICED15), Milan, Italy, 27–30 July 2015.
38. Spacey, J. Risk Guide. Available online: https://simplicable.com/new/risk (accessed on 17 May 2020).
39. Kendrick, T. *Identifying and Managing Project Risk*, 3rd ed.; American Management Association: New York, NY, USA, 2015.
40. MITRE. Common Weakness Scoring System (CWSS^{TM}). Available online: https://cwe.mitre.org/cwss/cwss_v1.0.1.html (accessed on 3 May 2020).
41. WhiteHat. *Application Security Statistics Report*; WhiteHat Security: San Jose, CA, USA, 2019; p. 34.

42. Open Web Application Security Project. OWASP Risk Rating Methodology. Available online: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology (accessed on 21 May 2020).
43. Becker, G.M. A Practical Risk Management Approach. In *PMI Global Congress 2004—North America, Anaheim, CA*; Project Management Institute: Newtown Square, PA, USA, 2004. Available online: https://www.pmi.org/learning/library/practical-risk-management-approach-8248 (accessed on 3 May 2020).
44. Rashid, A.; Chivers, H.; Danezis, G.; Lupu, E.; Martin, A. *The Cyber Security Body of Knowledge*; Centre, T.N.C.S., Ed.; University of Bristol: Bristol, UK, 2019; p. 834.
45. Suh, N.P. *Principles of Design*; Oxford University Press: New York, NY, USA, 1990; p. 401.
46. Vigo, R. Complexity over Uncertainty in Generalized Representational Information Theory (GRIT): A Structure-Sensitive General Theory of Information. *Inf. Syst. Des. Intell.* **2013**, *4*, 1. [CrossRef]
47. Shannon, C.E. A Mathematical Theory of Communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
48. Naik, K.; Tripathy, P. *Software Testing and Quality Assurance*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2008; p. 616.
49. RTI. *The Economic Impacts of Inadequate Infrastructure for Software Testing*; RTI: Gaithersburg, MD, USA, 2002.
50. McConnell, S. *Code Complete*; Microsoft Press: Redmond, WA, USA, 1993.