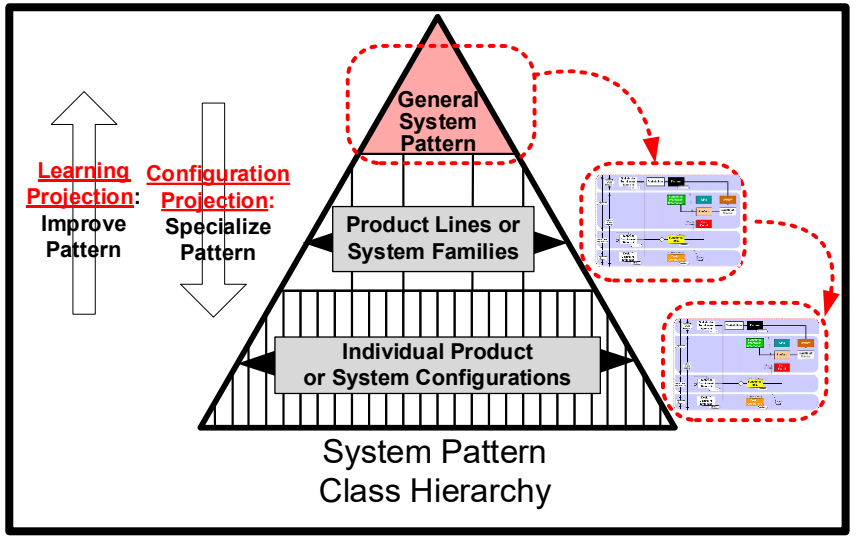
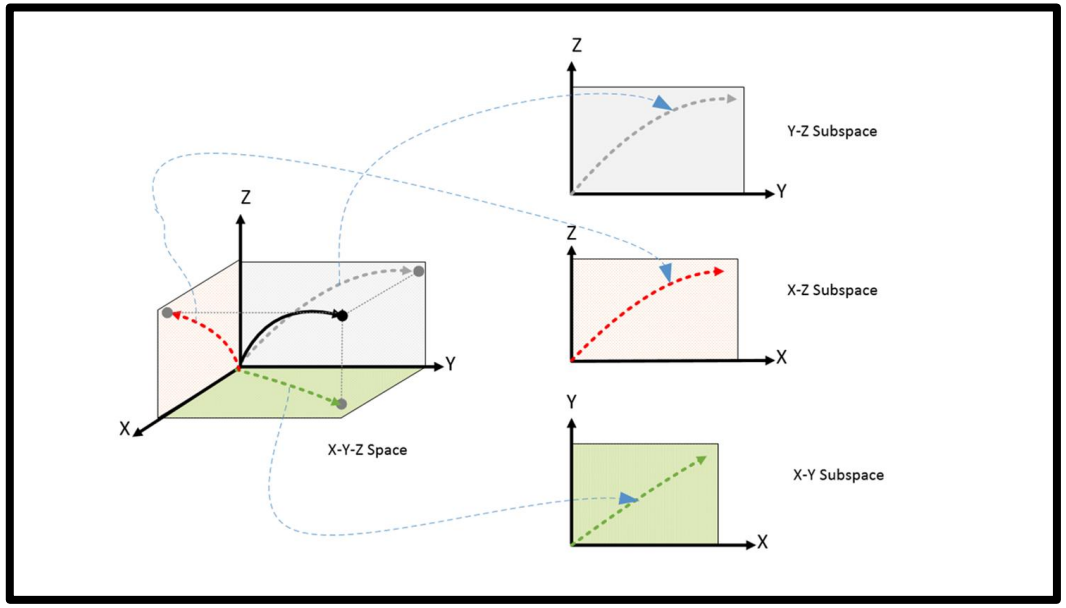
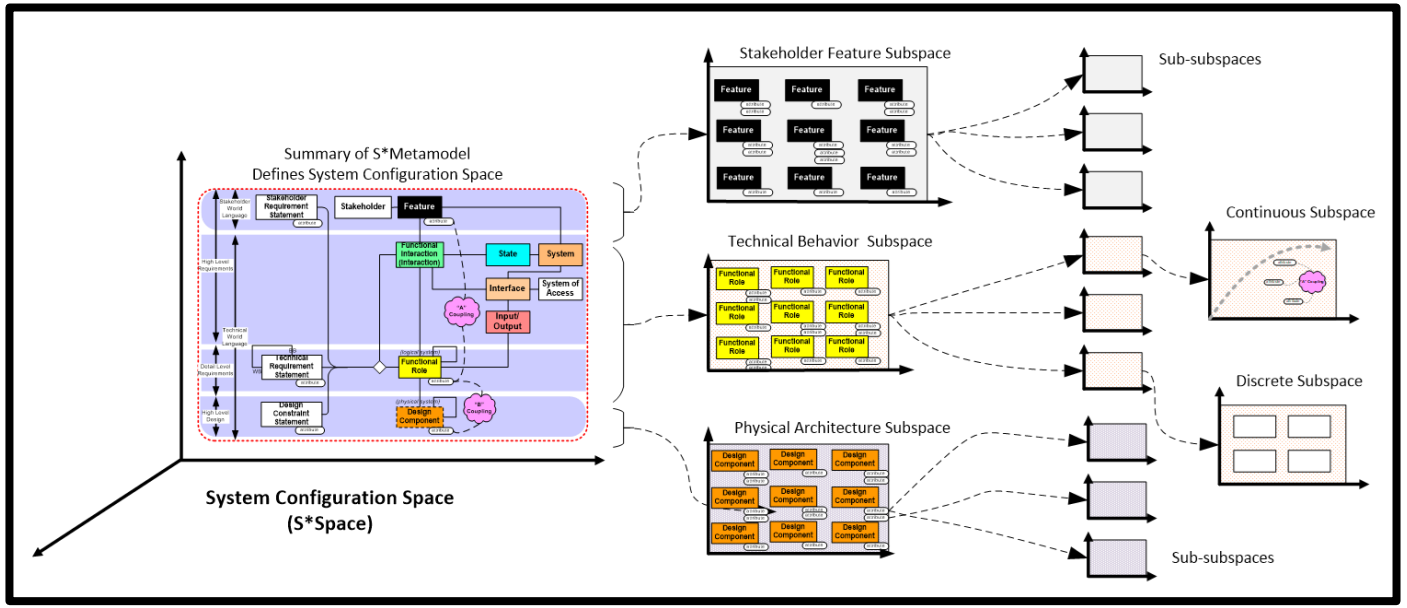
Vector Spaces, Projections, and Patterns: Support for MBSE, Learning, and Trust







Copyright © 2024 by William D. Schindel
schindel@ictt.com V1.2.3

Contents

- Abstract, purpose, and scope
- Patterns are what machines, people, and enterprises learn and (potentially) trust
- Relevance for innovation and the program supply chain: Managing confidence, trust
- Vector spaces in historical STEM
- Projections in vector spaces
- Semantics in MBSE models, LLMs, DSLs, and vector spaces
- More recent applications and contributions from AI
- Physics informed neural networks (PINNs)
- Specialized hardware and database software
- An application: Consistency estimation for information artifacts

- References

Abstract, purpose, and scope

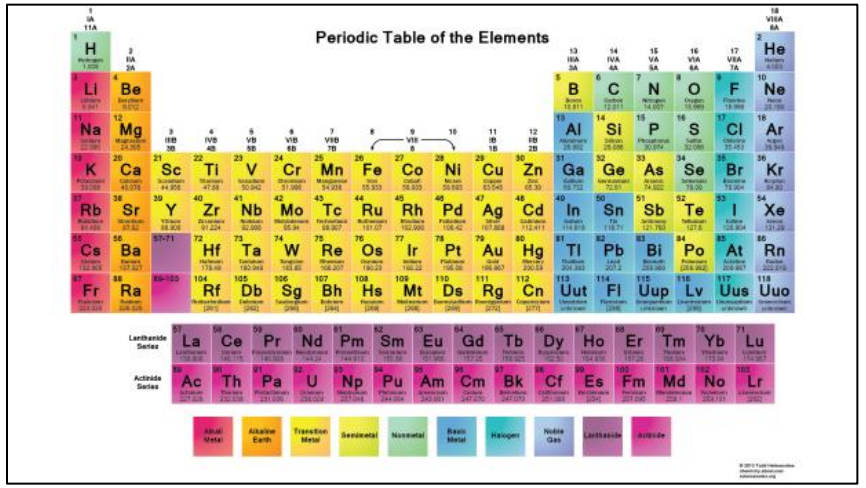
- The following briefly summarizes how the historical mathematical models of STEM have provided powerful vector space representational methods enabling discovery, learning, expression, analysis, and application of recurrent patterns and their models.
- For Model-Based Systems Engineering (MBSE) Patterns in particular, this provides means of understanding degrees of “closeness” (consistency) of MBSE models—not just for quantitative aspects, but also for their discrete, stochastic, qualitative, or semantic structure.
- Among related implications are a unified view of model VVUQ methods for numerical computational models versus MBSE system models and the systems they describe.
- It also provides a basis for understanding aspects of current AI representation of prose (LLM) as well as technical models, and the integration of human, enterprise and AI capabilities.
- This material is limited to a brief, high-level, informal summary, but includes listing of more detailed references.
- Prior knowledge of mathematics of linear algebra of vector spaces is not assumed, but can increase understanding of this material [1-4]. Some familiarity with MBSE, S*Models, S*Patterns, and S*Metamodel is assumed. Additional references for all those subjects are included [5-8].

Patterns are what machines, people, and enterprises learn, and (potentially) trust



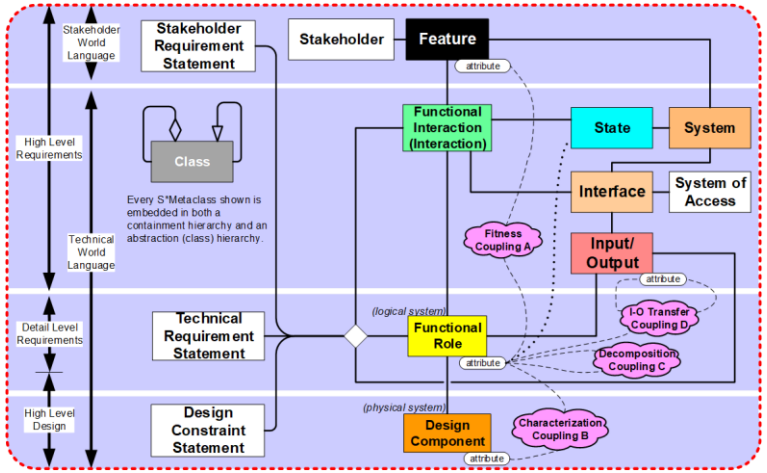
- Patterns are much more than just “templates for reusable models”;
- Recurrences (regularities, invariances), across natural phenomena, time, locations, projects, products, customers, applications, people, companies, or otherwise;
- The basis of all known laws of physical sciences for the last 300 years;
- The basis of theoretical foundations of the engineering disciplines;
- The basis of learning--for individuals, groups, and machines--what AI learns, what people learn, what teams and cultures learn;
- The basis of human cognition, intuitive heuristics, formal reasoning.

Patterns are what machines, people, and enterprises learn, and (potentially) trust



- What we didn't learn when we repeatedly miss the same opportunities or repeat the same mistakes;
- Why we wake up to a mostly recognizable world each day;
- Described by both fixed and variable (parameterized, configurable) aspects;
- Described informally by natural language;
- Described formally by the models of science, engineering, and mathematics;
- Not just about engineered products, but also methods of engineering, life cycle management, and (also) socio-technical systems in general.

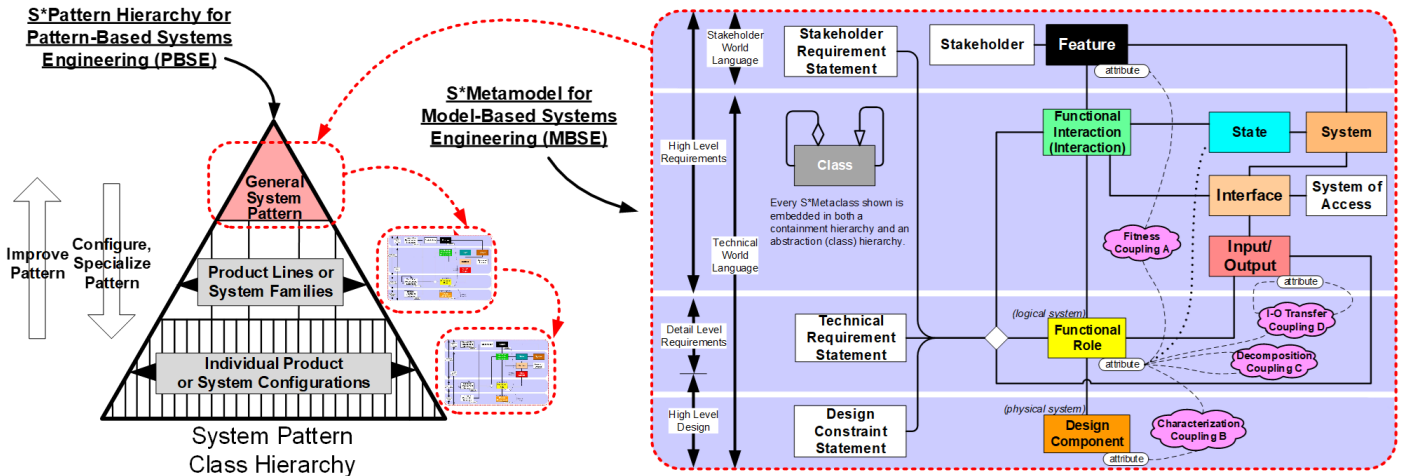
S*Models . . .



- Are models containing a certain minimal set of STEM-based elements. [6]
- Those underlying elements are called the S*Metamodel, which was inspired by the unmatched success of the physical sciences and impact of STEM [8]:
  - S* is short for “Systematica”.
- S*Models using those elements may be expressed in any modeling language and tooling, via formal mapping (e.g., in OMG SysML, or in other languages).
- S*Models can be (have been) created and managed in many different COTS modeling tools using diverse languages. [5]

S*Patterns: Configurable, Re-usable MBSE Models

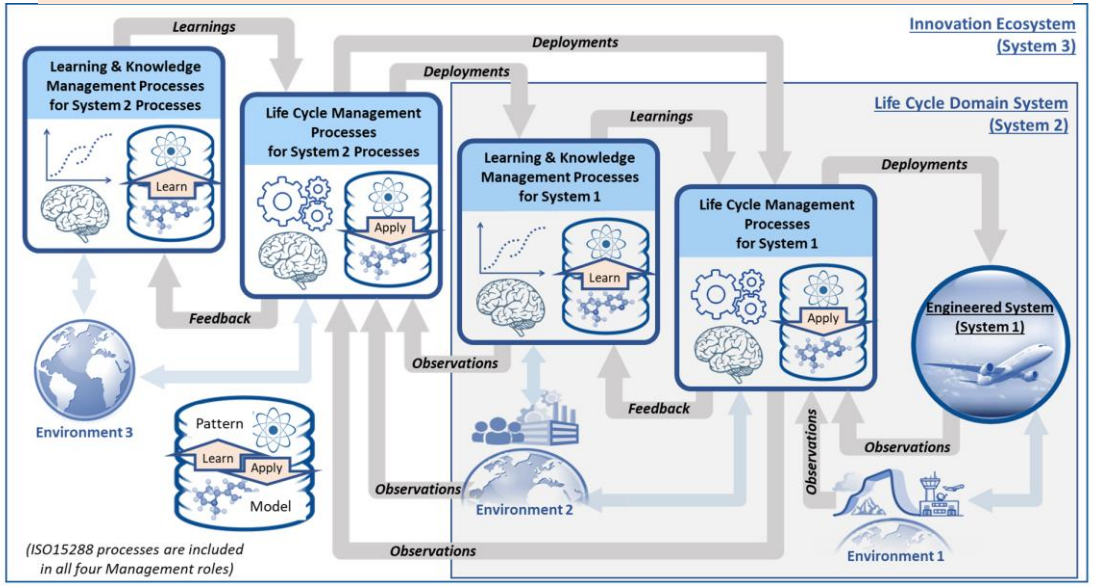
- S*Models made to be re-usable and configurable are called S*Patterns. [7]
- By “Pattern-Based Systems Engineering” (PBSE) we mean MBSE enhanced by these generalized assets to enable fast, semi-automatic model generation from trusted patterns.
- These are typically system-level patterns (models of whole managed platforms), not just smaller-scale component design patterns.
- Including patterns of not just engineered systems, but also patterns for their systems of engineering, production, distribution, and support. [5]

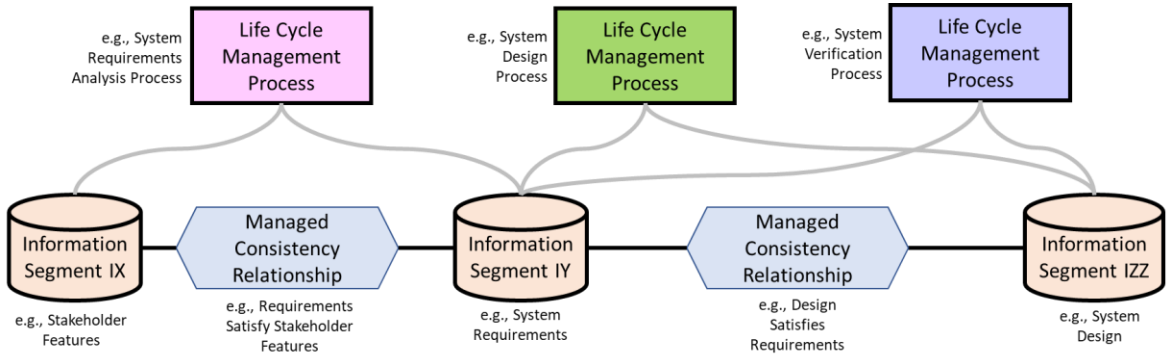


Relevance for innovation and the program supply chain: Managing confidence, trust

- Consistency management is a general term referring to practices that include verification and validation (V&V) of engineered systems as well as the models that describe them or other phenomena--including the quantification of uncertainty about that consistency (UQ). [13, 16]
- The history of models in science and engineering provides some powerful tools for not only evaluating consistency, but also driving it. [15-16]
- This has been further exploited in connectionist AI (neural nets). [12]
- Revisiting the fundamentals of those earlier tools helps illustrate their relevance to the innovation ecosystem supply chain. [22, 14, 23]

Innovation Ecosystem Pattern—Level 1 [10, 11]

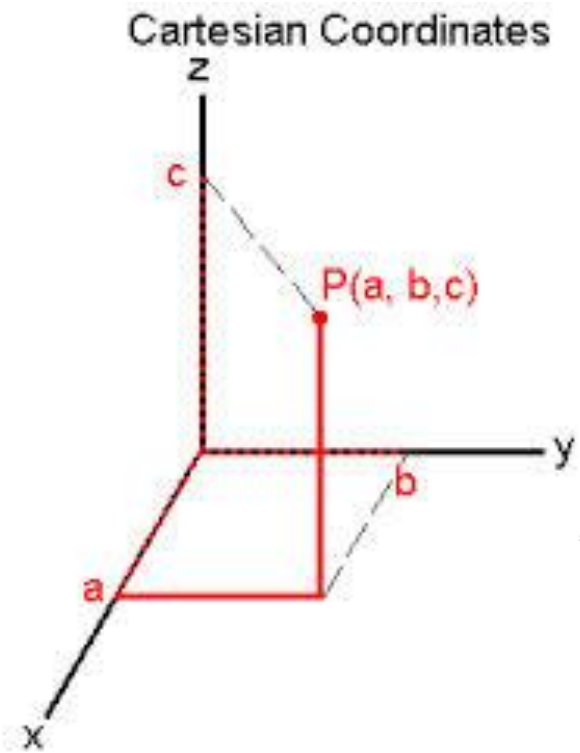




Example Consistency Thread [9, 13]

Vector Spaces in STEM: The Geometrization of Models

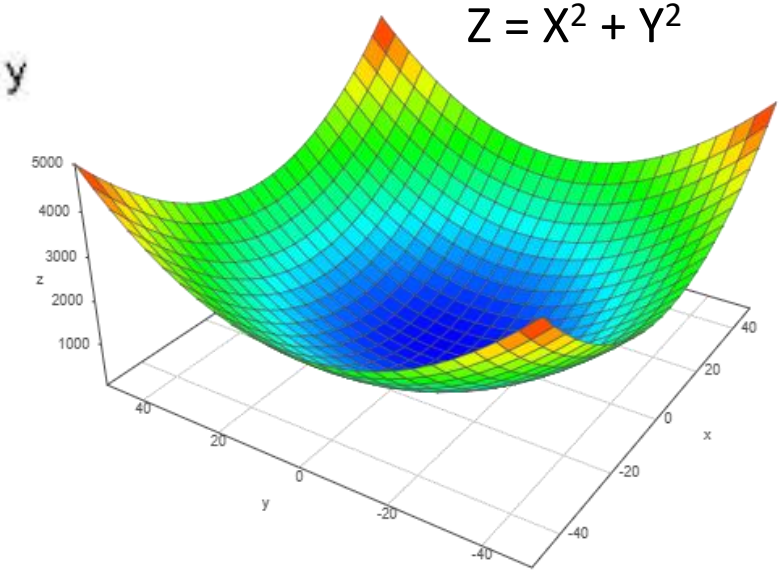
- Geometrization of algebra, by Rene Descartes ("Cartesian" coordinates):



Rene Descartes
1596 – 1650
[17]



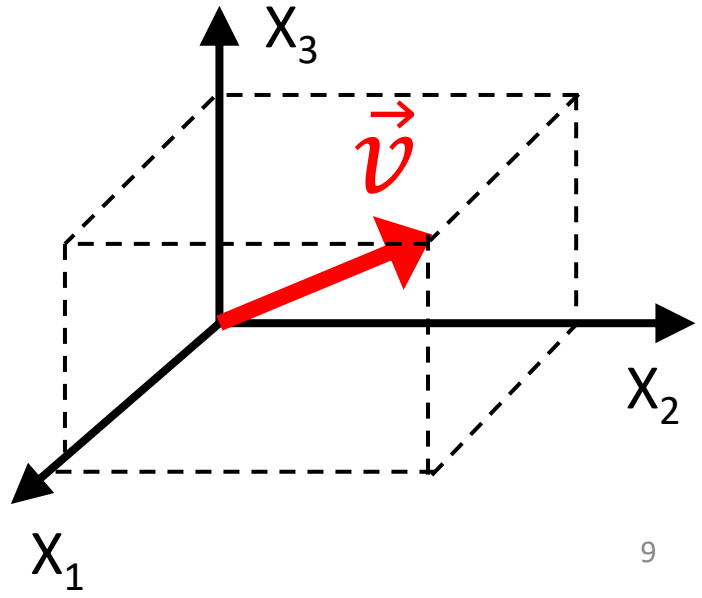
$Z = X^2 + Y^2$



- Vector analysis had subsequent (more complex) history (Hamilton, Heaviside, Gibbs, Grassmann et al), leading to current practice [1]:

Vector Spaces:
Home of Vectors, Linear Algebra, Complex Variables, Vector Calculus, Math Physics of Multi-dimension Systems, Simulations

Vector $\vec{v}$

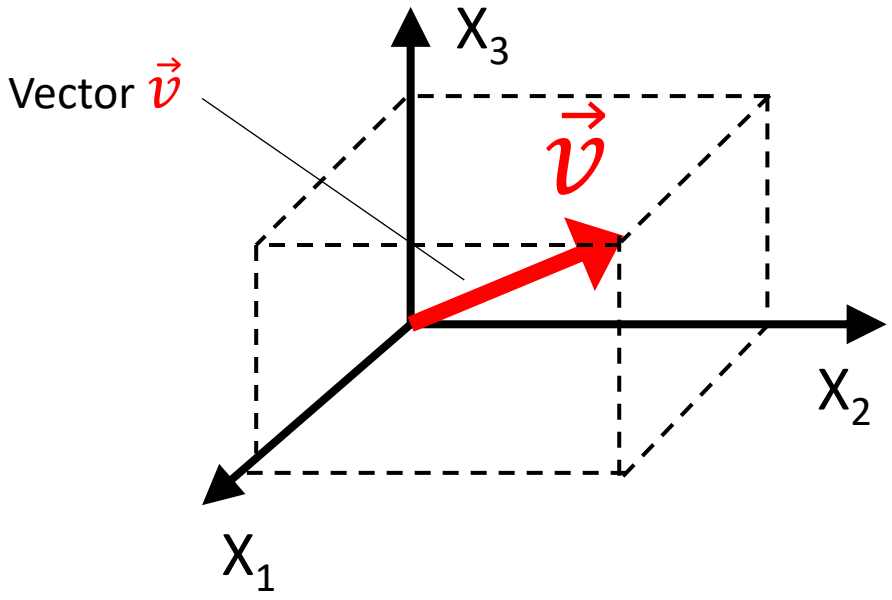


Vector Spaces in STEM: The Geometrization of System Models

- As mathematical models described continuous and non-linear behavior, David Hilbert provided a geometrical view of multi-dimensional Hilbert Space [2]:
  - Including not just finite dimensional spaces;
  - Also infinite dimensional spaces of mathematical functions as embedded vectors



David Hilbert
1862 – 1943



Systems Engineering Evolution: Hilbert Space Enables Mathematical Comparison of MBSE Model "Nearness"

Definitions [2]

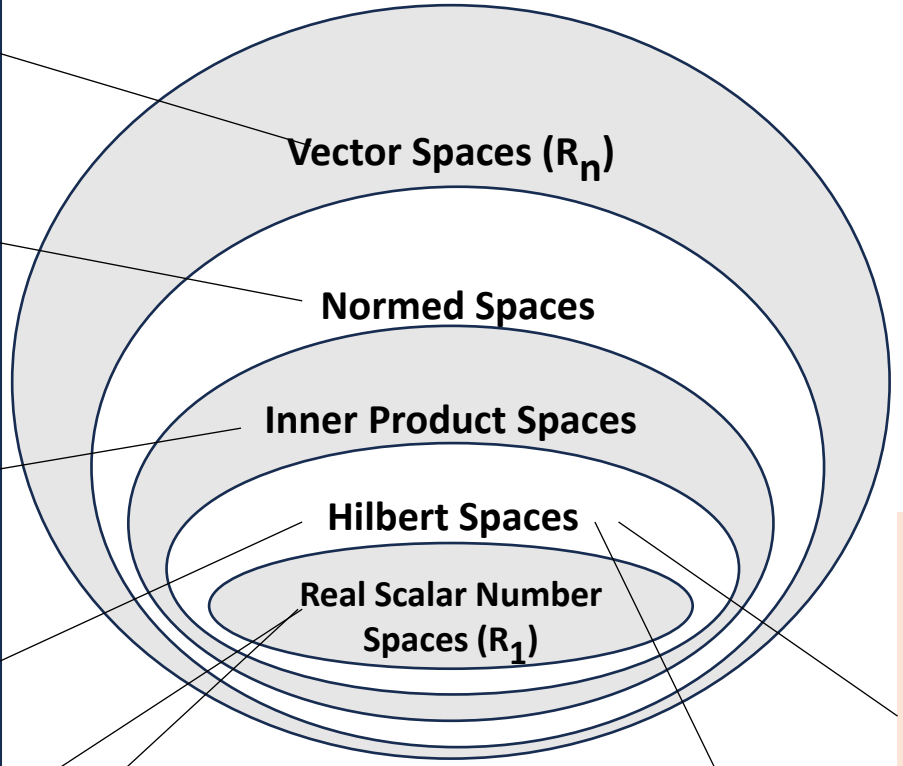
Spaces of multi-dimensional vector (V) entities, that can be added (V1+V2) and scalar multiplied (cV1).

Vector spaces with norms on vectors |V1|, describing their "size".

Normed spaces with inner products <V1,V2> that express "relative orientation" of vectors, and for which $|V1|^2 = <V1,V1>$.

Inner product spaces that are complete (closed for limit sequences).

Simple one-dimensional numeric quantities.





David Hilbert
1862 - 1943

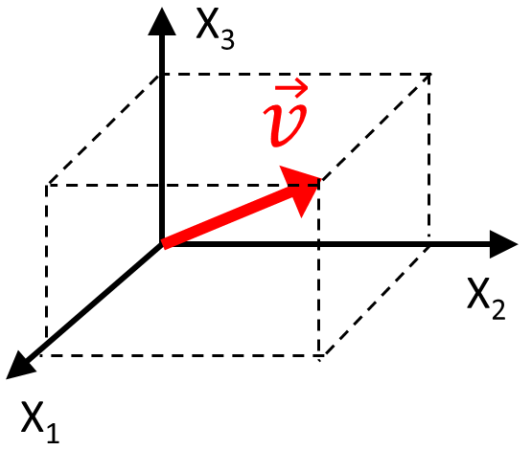
3. "Vector Embedding" software can currently embed MBSE Models in this space, quantifying their "nearness" & "gaps".

1. We express Computational Model numerical output in this space.

2. "Area Metric" expresses Computational Model "nearness" in this space.

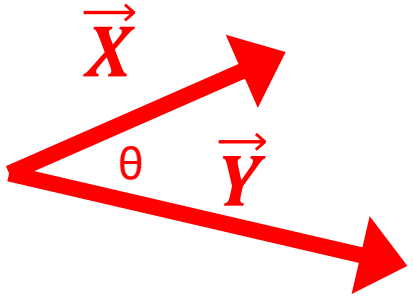
Examples of Inner Products & Hilbert Vector Spaces

- Euclidean 3-D vector space R_3, with vectors **X** = $\{x_1, x_2, x_3\}$, and inner product <**X**,**Y**> :



$$\langle \mathbf{X},\mathbf{Y} \rangle = [x_1 \cdot y_1 + x_2 \cdot y_2 + x_3 \cdot y_3]^{1/2} \quad ,$$

or



$$\langle \mathbf{X}, \mathbf{Y} \rangle = |X||Y| \cos(\theta) \, ,$$

where θ is angle between X and Y.

Notice that the same "3-D" ideas and forms can apply when the dimension of the space is smaller than 3 (as in a 1-D line or 2-D plane) or larger than 3 (as in higher-dimensional vector spaces). The vectors in R_n are still of the form $\{x_1, x_2, \ldots x_n\}$, and their norms and inner products are still calculated using the same form of equations.

- Hilbert space of infinite dimensional vectors of continuous functions y = f(x) on the interval [a, b], and with inner product:

$$\langle f(x), g(x) \rangle = \int_a^b f(x)g(x)dx \quad ,$$ as applied in spectral analysis, etc.

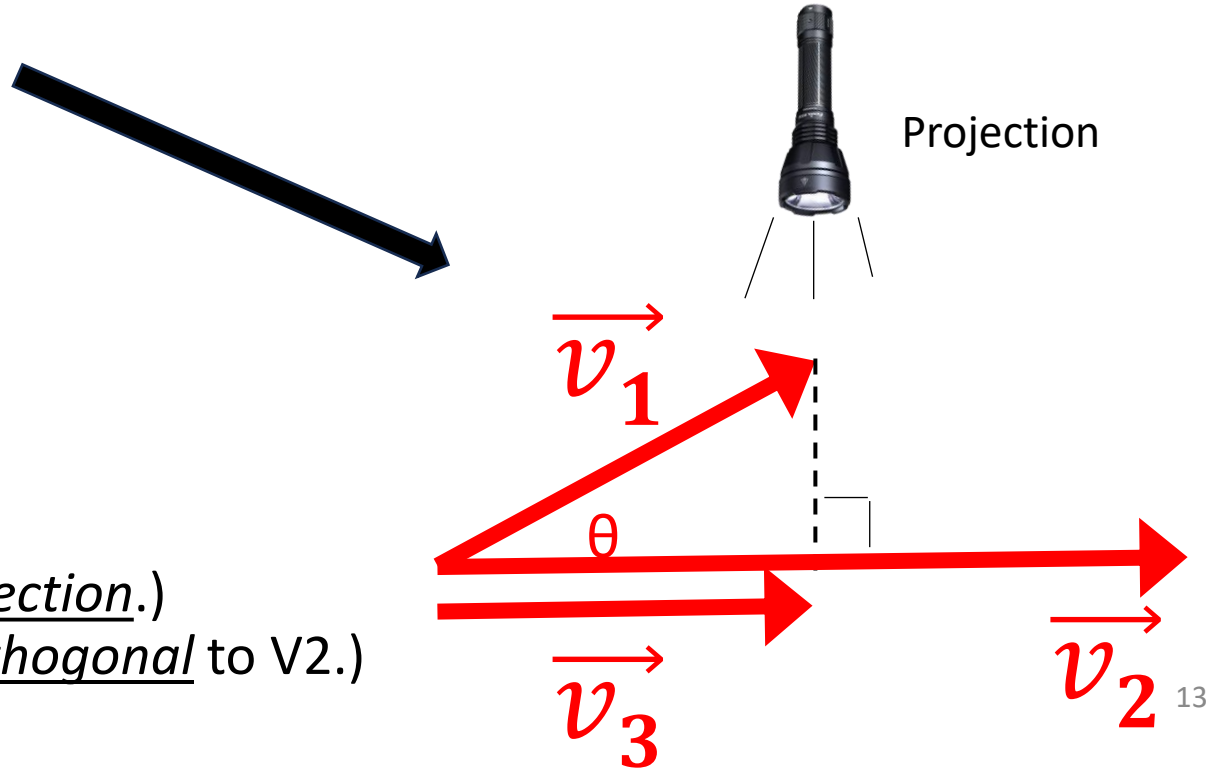
Projections: Vectors, PCA, S*Models, Configured S*Patterns

- Geometrization of mathematical models does not always mean drawing geometric diagrams (as in 2D & 3D geographic maps), but instead provides geometry-based intuitive basis for more abstract mathematical operations: computing size and distance (metric spaces), inner products, *projections*, and evolutionary paths in configuration space.

- Projection of vector V1 onto vector V2, resulting in vector V3:

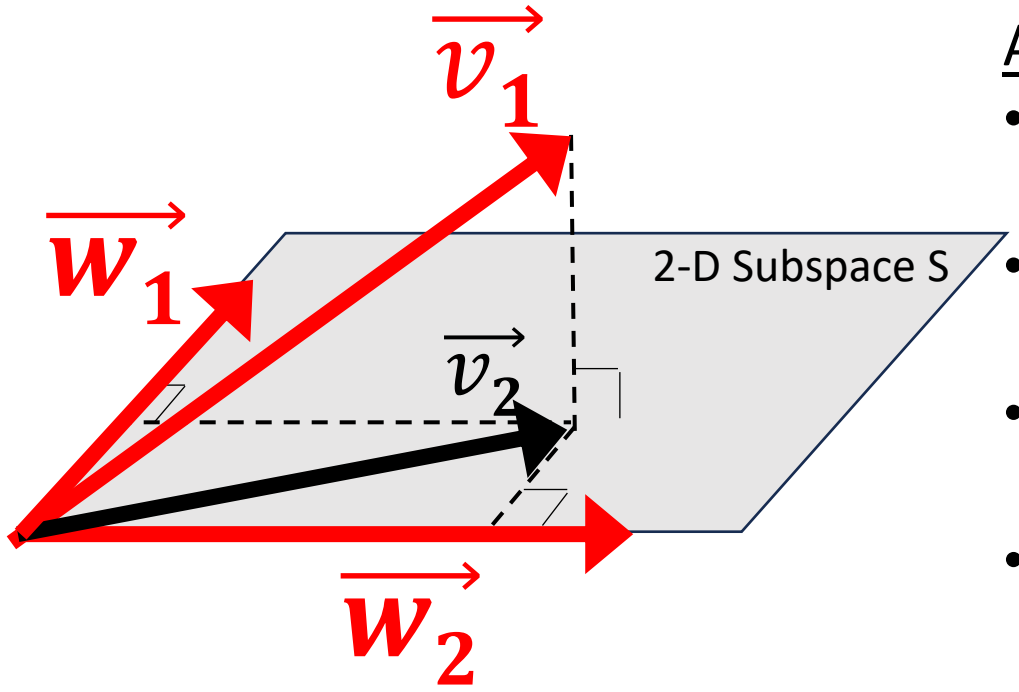
$$\vec{v}_3 = \frac{\langle \vec{v}_1, \vec{v}_2 \rangle}{|\vec{v}_2|} \vec{v}_2$$

(Projection is "shadow" of V1 on V2.)
(*Inner product is fundamental to projection*.)
(Note that projection *difference* is *orthogonal* to V2.)



Projections: Vectors, PCA, S*Models, Configured S*Patterns

- Projection of a vector V_1 onto a subspace S, spanned by other vectors W_1 and W_2, resulting in vector V_2, which is "inside" that subspace.
  - If W_2 and W_3 span the subspace S, then any vector inside S can be expressed as a sum of multiples of W_2 and W_3.
  - Projecting onto the subspace is equivalent to projecting onto its basis vectors W_2 and W_3, using inner products

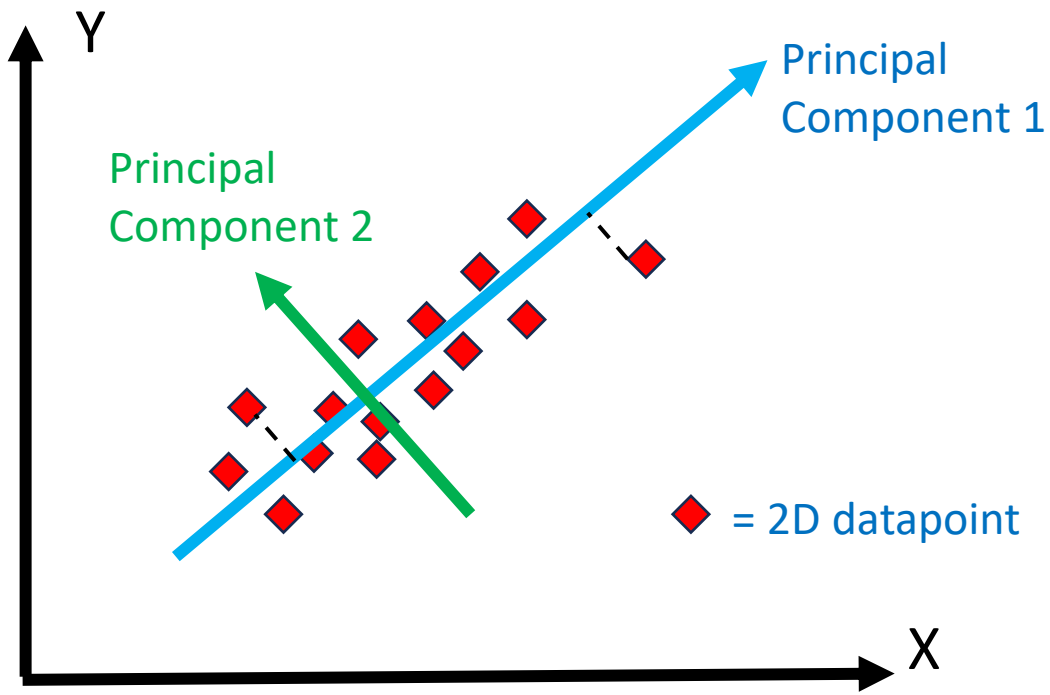


Applications:

- Saving legacy System Requirements by projecting them onto a new MBSE Pattern's set of External Interactions.
- Translating a prose artifact into an S*Model by projecting the artifact onto the space of the S*Metamodel.
- Mapping an artifact into an S*Model by projecting the artifact onto the space of an S*Pattern.
- Configuring an S*Pattern by suppressing its inapplicable dimensions, projecting it onto a configured S*Model.

Projections: Vectors, PCA, S*Models, Configured S*Patterns

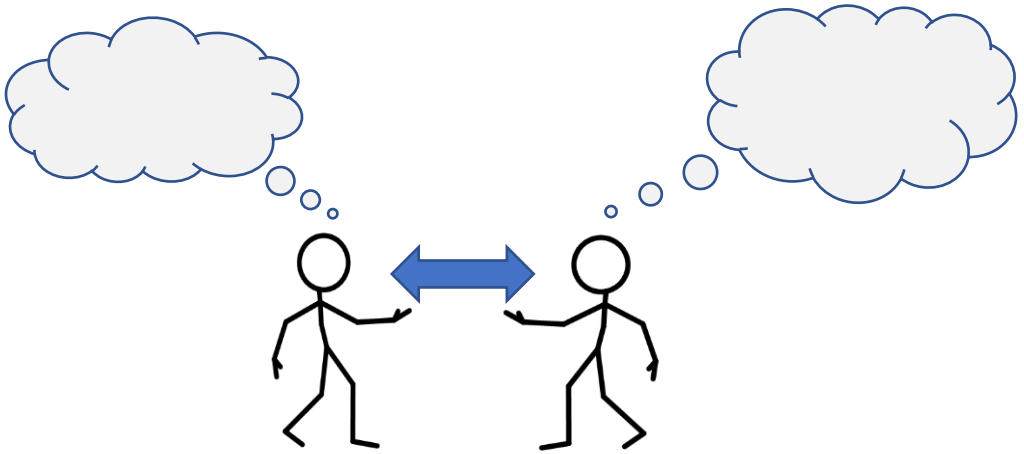
- Principal Component Analysis (PCA) is a statistical method that *extracts the main components of variation* across a set of data, by asking what subspaces ("dimensions") account for the main ("principal") variation in the data, thereby reducing degrees of freedom and complexity in a model of that data [18]:

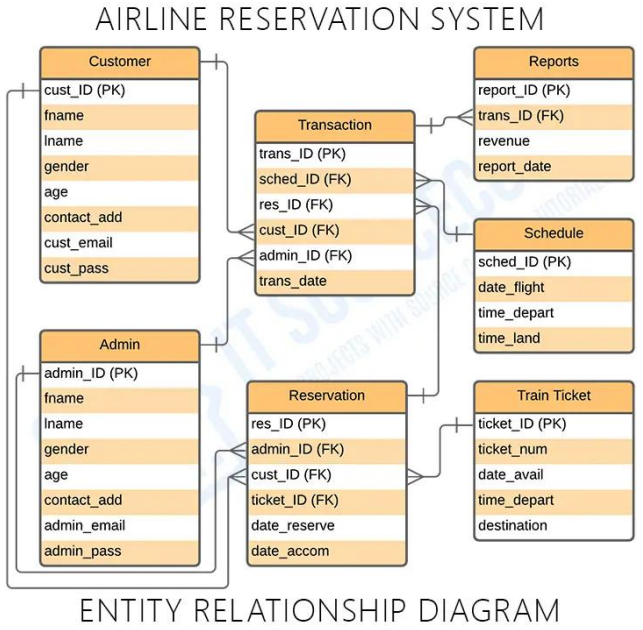


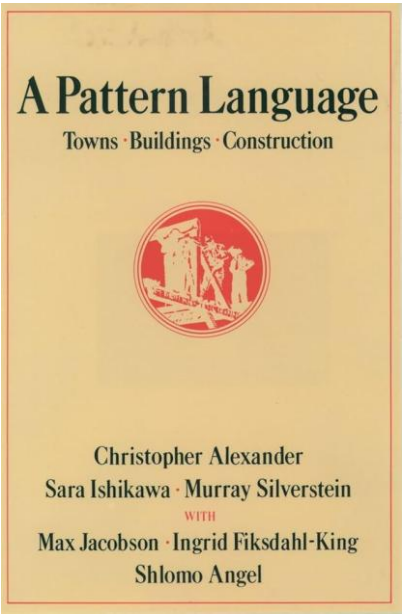
- Orthogonal projection of data points ♦ onto different candidate subspaces (lines shown).
- For a fixed set of data, one can rotate the lines to find an optimum line direction that "soaks up" maximum variation across the data points.
- Here, PC1 accounts for the most variation; PC2 accounts for most remaining variation.
- PCA works by projecting observations onto lower dimension subspaces, while finding the subspaces that minimize unprojected variation.
- Ideas most pattern extraction methods use.

Semantics of MBSE models, DSLs, LLMs, vector spaces

- The term “semantics” is used in the contexts of human natural language prose, MBSE and other entity-relationship language models, domain specific languages (DSLs), pattern languages, ontologies, database models, and the large language models (LLMs) of AI—how is the idea similar across them?
  - Semantics in all those cases refers to the web of allowed relationships between concepts in those spaces.



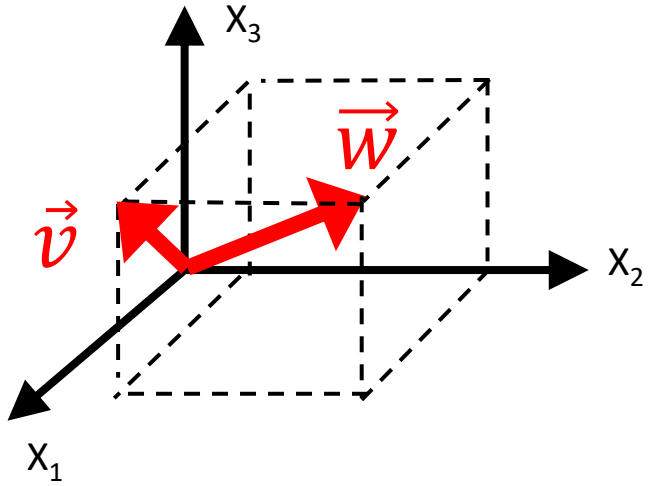




- All the vectors in a given vector space have the same number of components—equal in number to the dimension of that vector space:

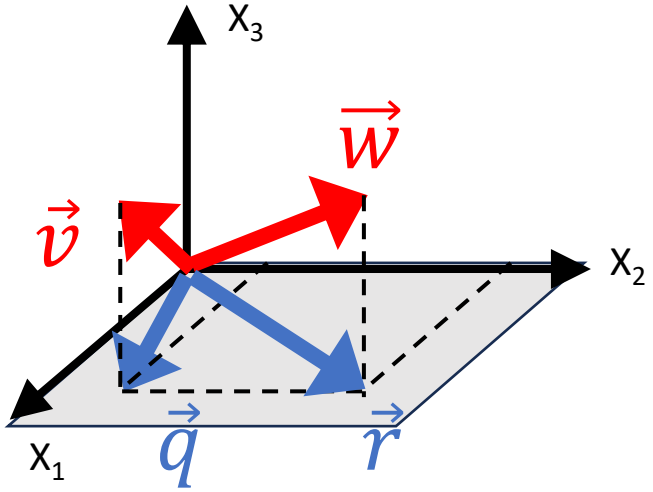
Vector $\vec{v}$ = {v_1, v_2, v_3}

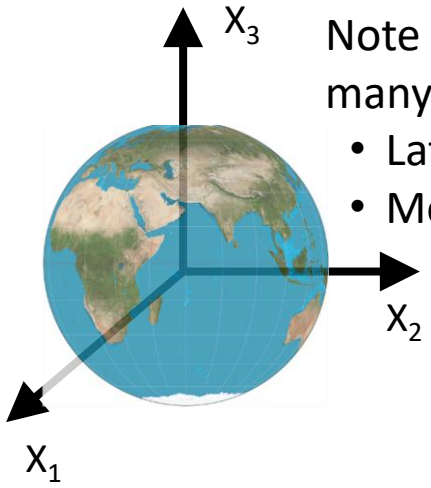
Vector $\vec{w}$ = {w_1, w_2, w_3}



- Vectors in a subspace of the "parent" vector space have fewer components—possibly eliminating a parent space component—a simple form of projection:

	In Parent Space	In Subspace
Vector $\vec{q}$	= {v_1, v_2, 0}	{v_1, v_2}
Vector $\vec{r}$	= {w_1, w_2, 0}	{w_1, w_2}



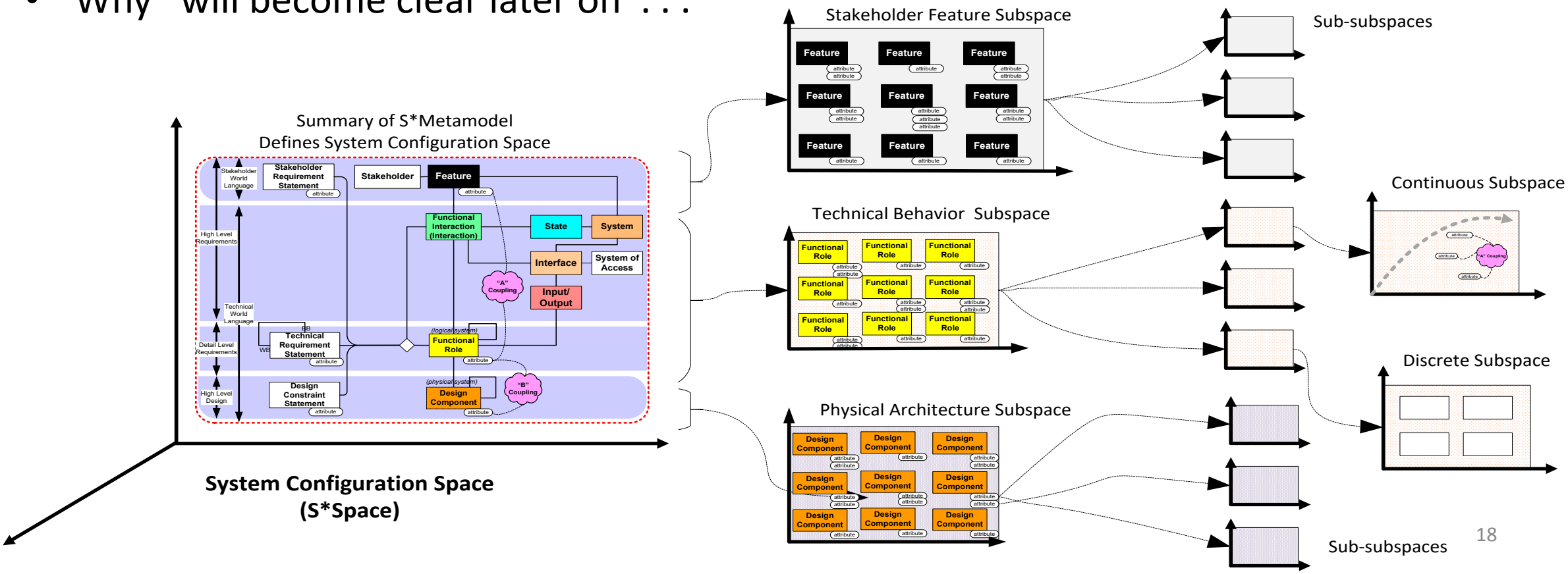


Note that there are many other subspaces!

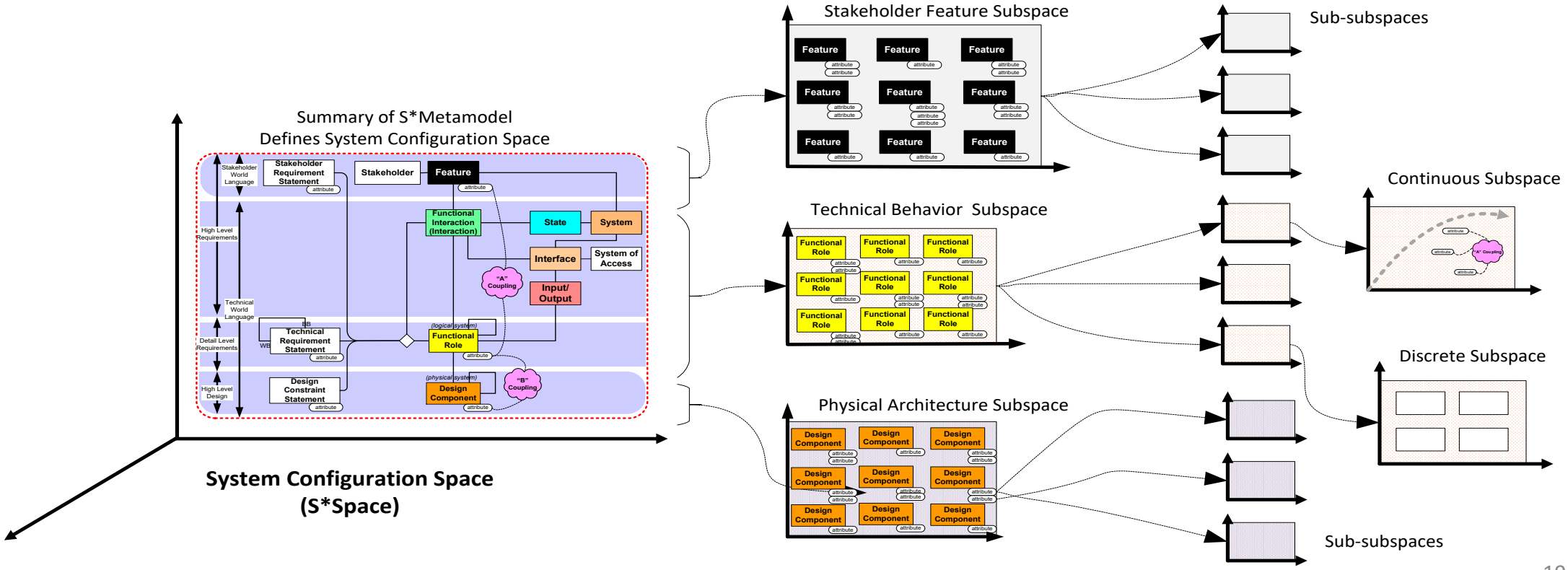
- Lat/Long Projection
- Mercator Projection

Models in Vector Spaces

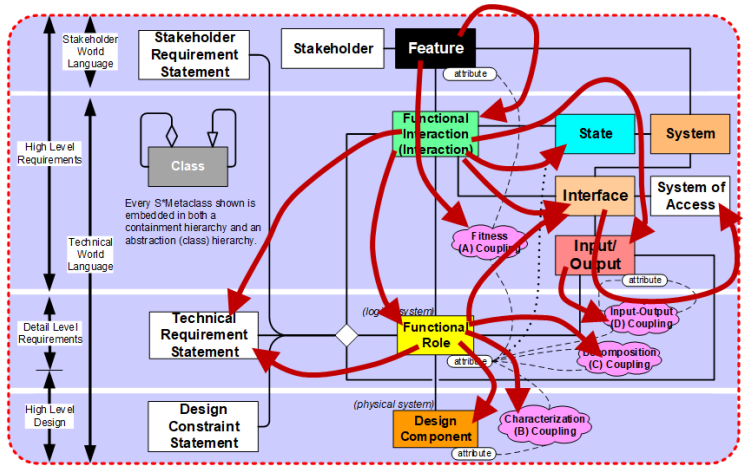
- A vector can represent an occurrence of a modeled class (entity) type:
  - "Why" we'd do so is less obvious for now than "how"—but postpone that a moment.
- Likewise, subspaces of relationship types can be formed from the cross products (the "power" set of pairs) of the model's entity subspaces they can connect:
  - "Why" will become clear later on . . .

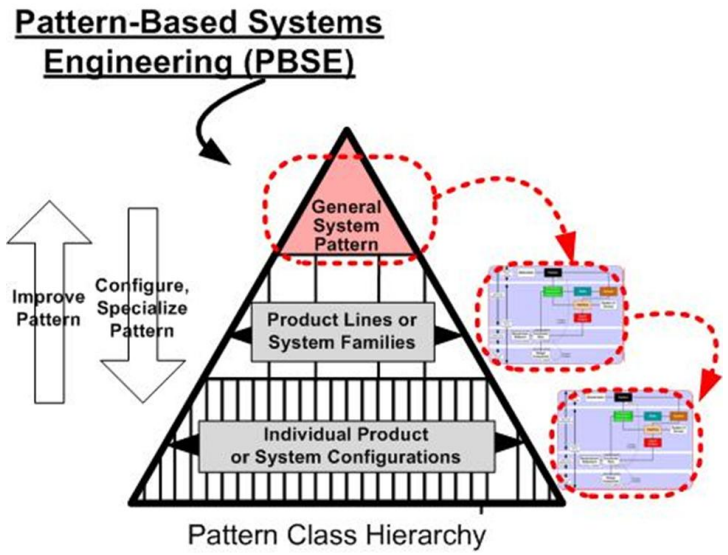


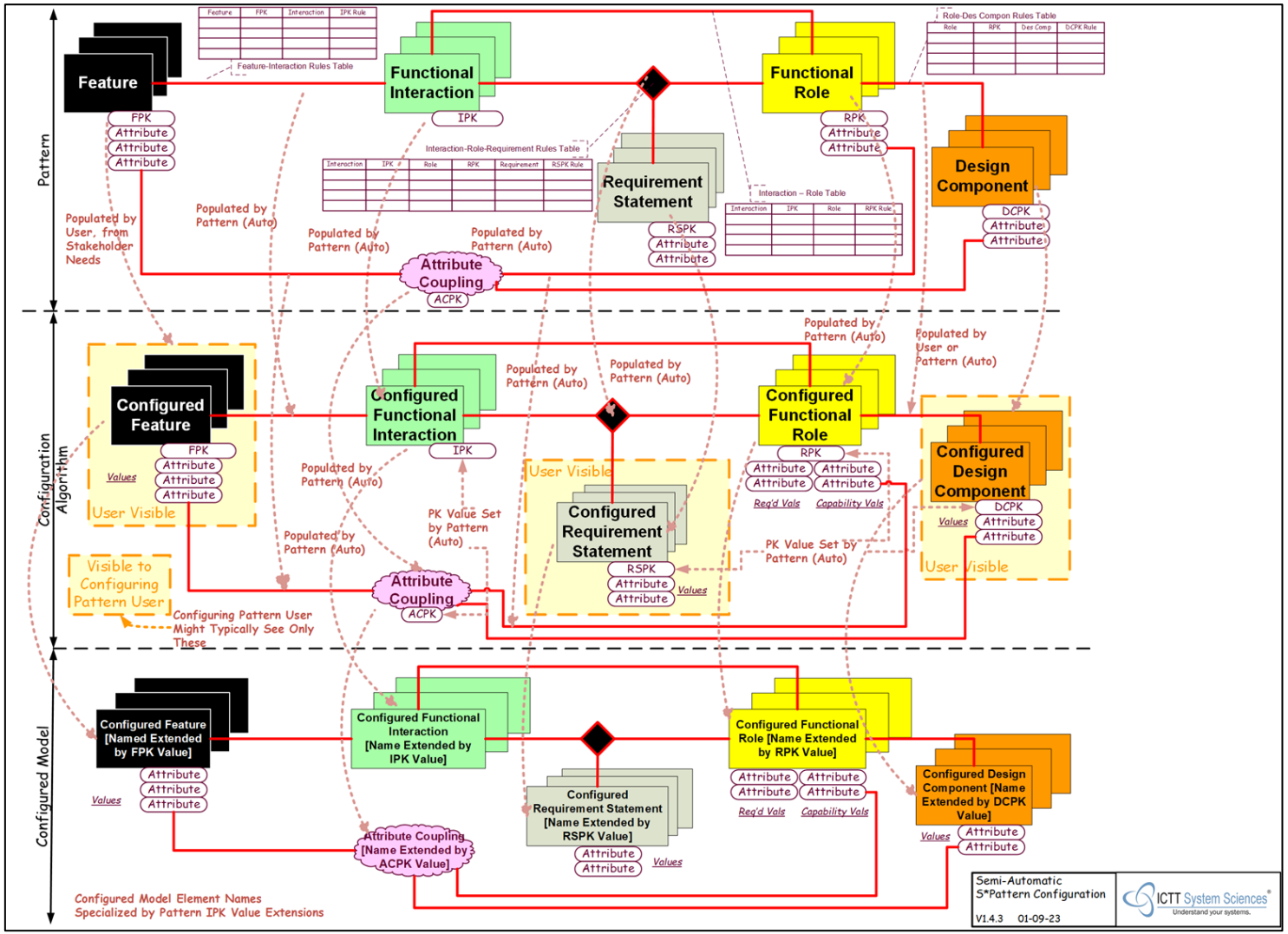
- For the attributes of a class (not of a reified relationship), those attributes and their values can be used to create a multi-dimension subspace for each entity type, whose subspace dimensionalities are the number of attribute types of each class type.
- For attributes of reified relationship classes pointing to classes they relate ("foreign keys" in database language), those attributes and their values create a subspace for each relationship type, whose dimension is the number of attributes (role linkages) of the relationship type.



The S*Pattern configuration algorithm produces S*Models by projection and generative inference [7-8]

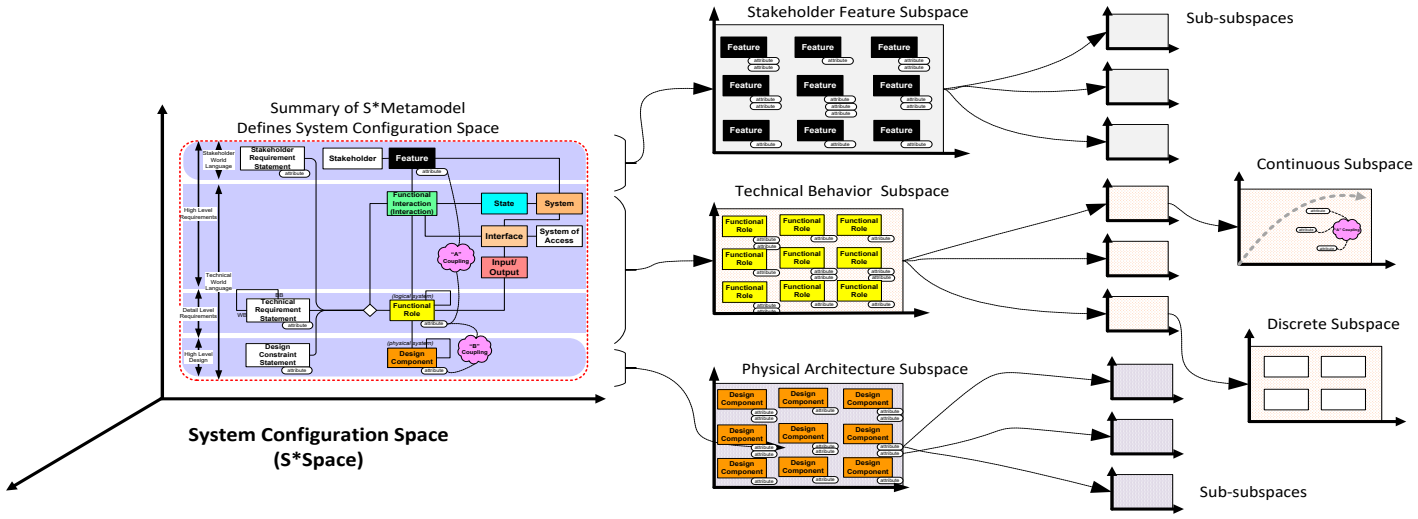


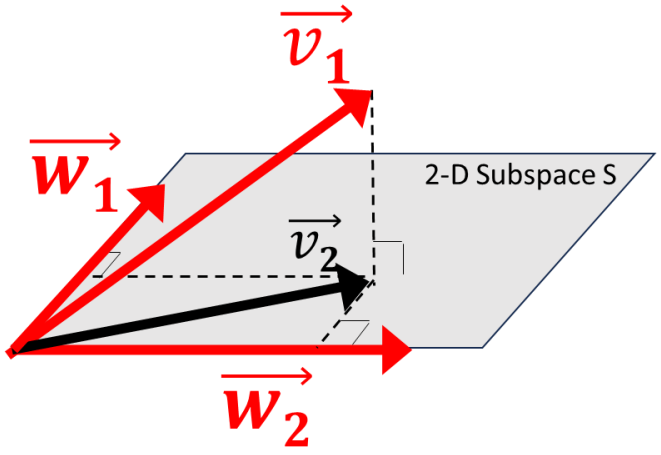




Multiple visualizations: E-R model diagrams versus vector spaces

- For *visualizing entity-relationship semantics*, traditional E-R model diagrams (whether in UML or SysML or otherwise) will seem more powerful / natural than visualizing vector space.
- However, for expressing a rigorous concept of "*closeness*" (similarity, consistency) of models, *projecting* a model (or data) onto another model, or forming *inner products* of models, it will turn out that *vector space representations* (whether as tables, diagrams, JOINs, or otherwise) are more powerful / natural than visualizing (or computing) with E-R model style diagrams:
  - That is why vector space representations arise in the technology of ML and AI . . . .
  - It is also why, in generative pattern database representations, JOINs become important.





More recent applications and contributions from AI

- If you are interacting with an AI--say, a large language model (LLM) through a human-oriented “Chat” user interface--it is not particularly necessary to be conscious of the underlying theory and technologies, including vector spaces and vector database technology.
- However, if you are interested in the capabilities of the mathematics that enables and describes machine learning, generative AI, and related functions, it is fundamental to have awareness of use of vector representations, projection (inner product) metrics on closeness (similarity), etc.
- The mathematical concepts of embedding in a vector space and applying inner products and projects can be more fundamental and important to understand (such as, what is possible) than the IT concept of using a vector database—*except for software specialists.*
- The process of transforming information of interest into vector representation in these applications is called “embedding” the information into vector space.

More recent applications and contributions from AI [12]

- Learning phase:
  - Neural network as a universal function approximator (for classification, prediction, other uses).
  - Processes observation data to generate vector subspace function minimizing (loss function) error.
  - Gradient-based optimization search, suitable for parallelized hardware search algorithms.
  - Network-based function representation with (very numerous) optimized coupling parameters.
- Generative inference phase:
  - Applies the learned network inference function to infer values from offered inputs.
- Additional aspects:
  - May include use of vector space representation of additional reference data (Retrieval Augmented Generation).
  - In the case of LLMs and similar sequential cases, encoder/decoder and transformer architectures and attention models for time series representation and inference.
  - Inclusion of stochastic (Boltzmann Machine) aspects to further optimize performance.
- Key aspects of the underlying approach can be seen as generalizations of PCA and other vector space inner product and projections.
- Although natural language models have captured public interest in Large Language Model (LLM) applications, they were not the original and are not the only kinds of information neural networks have represented: Sound, Vision, Structured Models, other forms of information.
- There are direct applications in systems engineering models. [21]

Physics-Informed Neural Networks (PINNs)

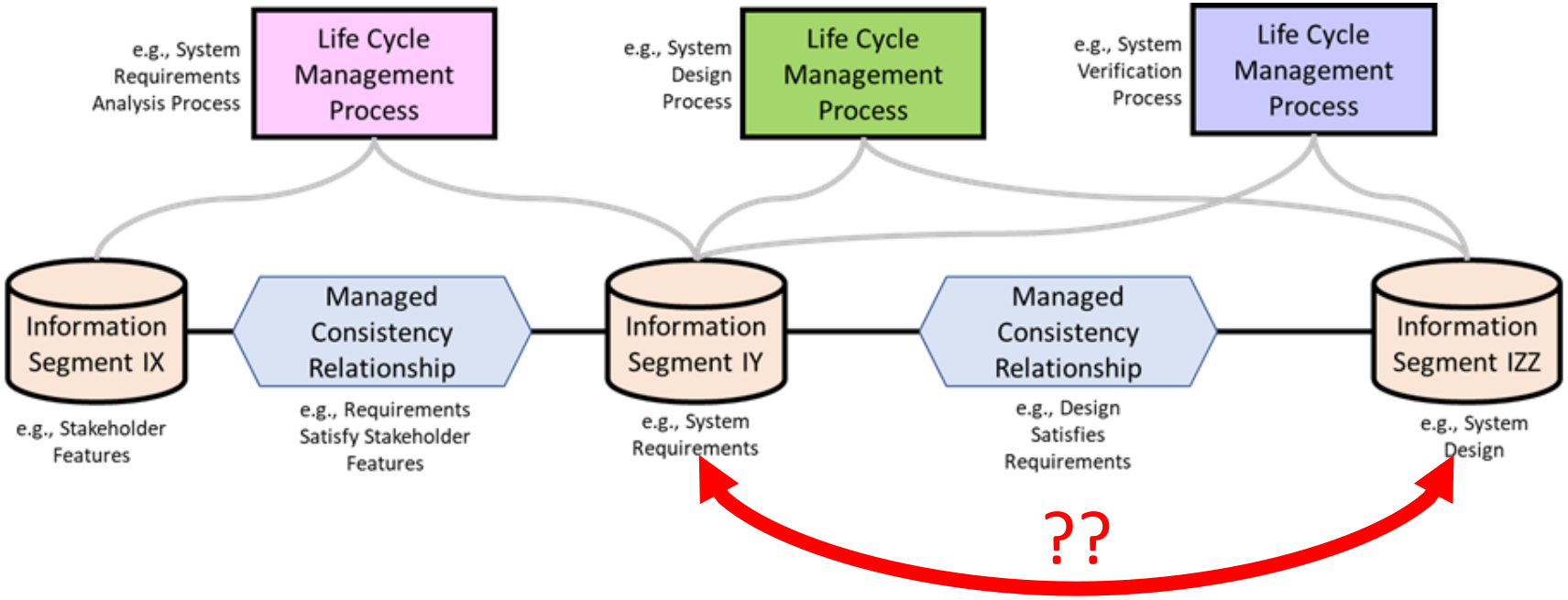
- Historically, classical physical science sought out the smallest models (hypotheses) that “soak up” (reproduce) the maximum amount of variation observed in Nature.
- Over 300 years of such work, the framework of physical laws have evolved to capture that progress in STEM models, which can be described by the states-interactions-based framework of the S*Metamodel.
- When a learning machine is trying to compress observational data into the smallest number of explanatory dimensions, projecting it onto the S*Metamodel can be used to seek out a minimal S*Model in the tradition of smallest models.
- Physics-Informed Neural Networks (PINNs) illustrate the concept of giving the learning machine a “leg up” using known modeling frameworks such as the S*Metamodel. [19]
- One such path is to use the S*Metamodel and/or more specific S*Patterns in a Retrieval-Augmented Generation (RAG) AI architecture, pre-embedded into a vector database. [20]

Specialized hardware and database software

- Because of the vector mathematics nature of image processing and neural networks, it was found that special graphic processing unit (GPU) hardware was significantly faster for certain (e.g., learning) aspects of neural network processing:
  - This has led to even further optimized neural processors [24, 25]
- Because of the high-dimensional vector mathematics nature of neural networks, it was found that specialized “vector databases” could be developed to efficiently operate on (query, etc.) high-dimension vector spaces:
  - This has led to commercial “vector database” products. [26,27]
- With or without these aids, it is important to recognize that awareness of the underlying mathematics is key to fuller understanding what is going on and what is possible.

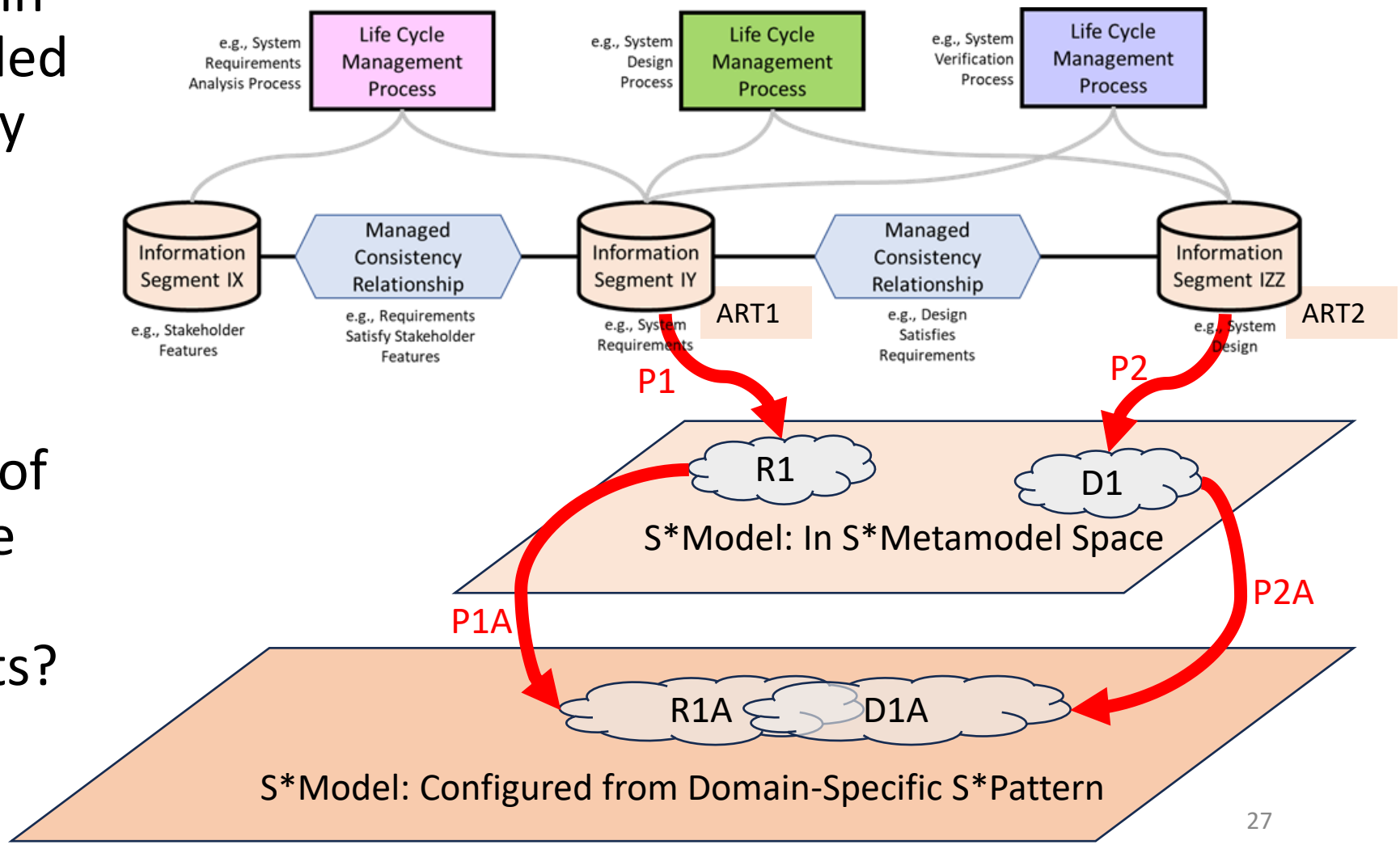
An application: Consistency estimation for information artifacts

- The above vector space concepts can be applied to Consistency Management, at the heart of the Innovation Ecosystem [9, 22]
- Suppose we want to estimate the degree of consistency of two artifacts:



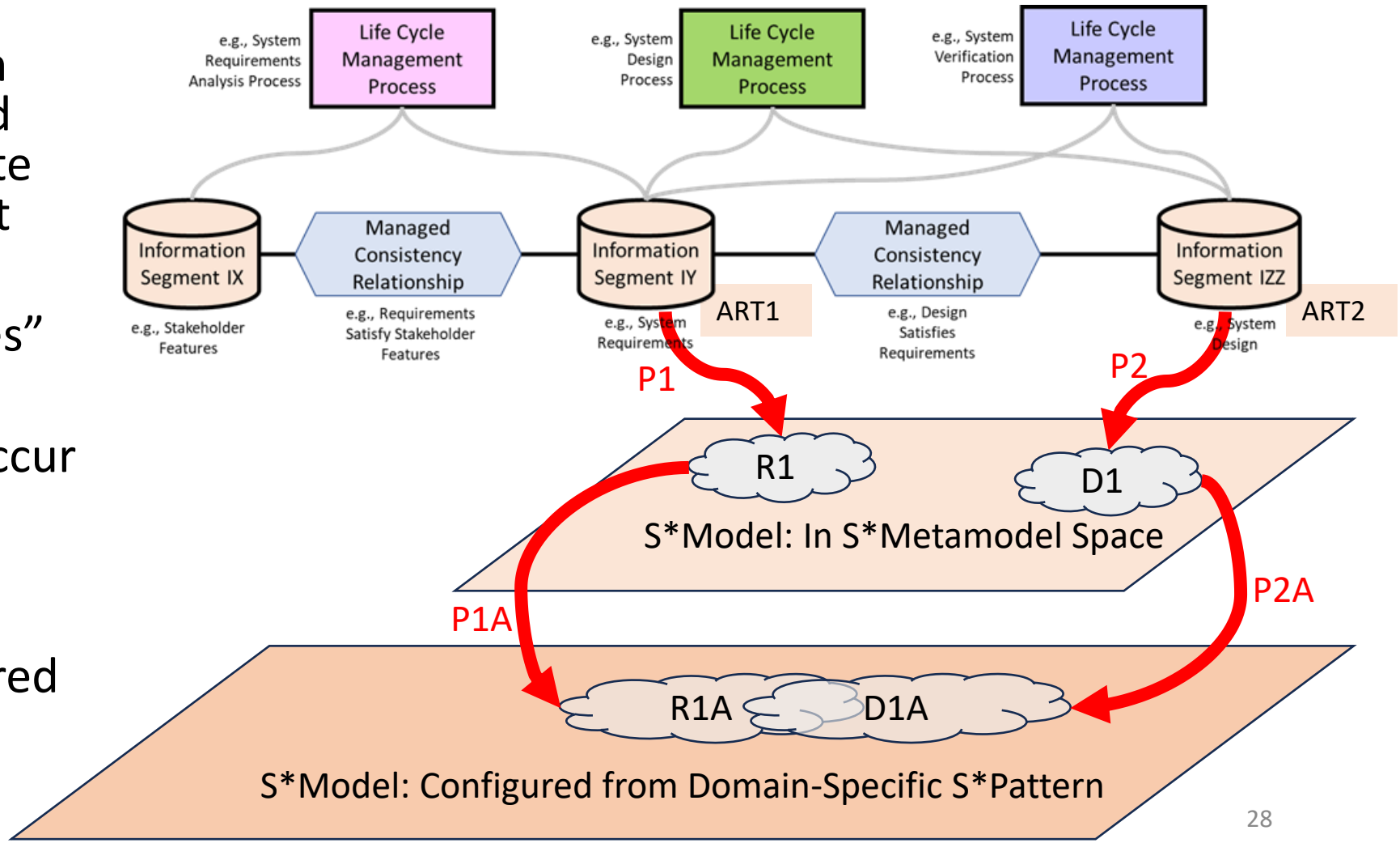
An application: Consistency estimation for information artifacts

- In general, additional information (SME domain knowledge, etc.) is needed to assess the consistency of two artifacts.
- For example, assessing consistency of a design with requirements depends on knowledge of the phenomena that the design employs—will it conform to requirements?



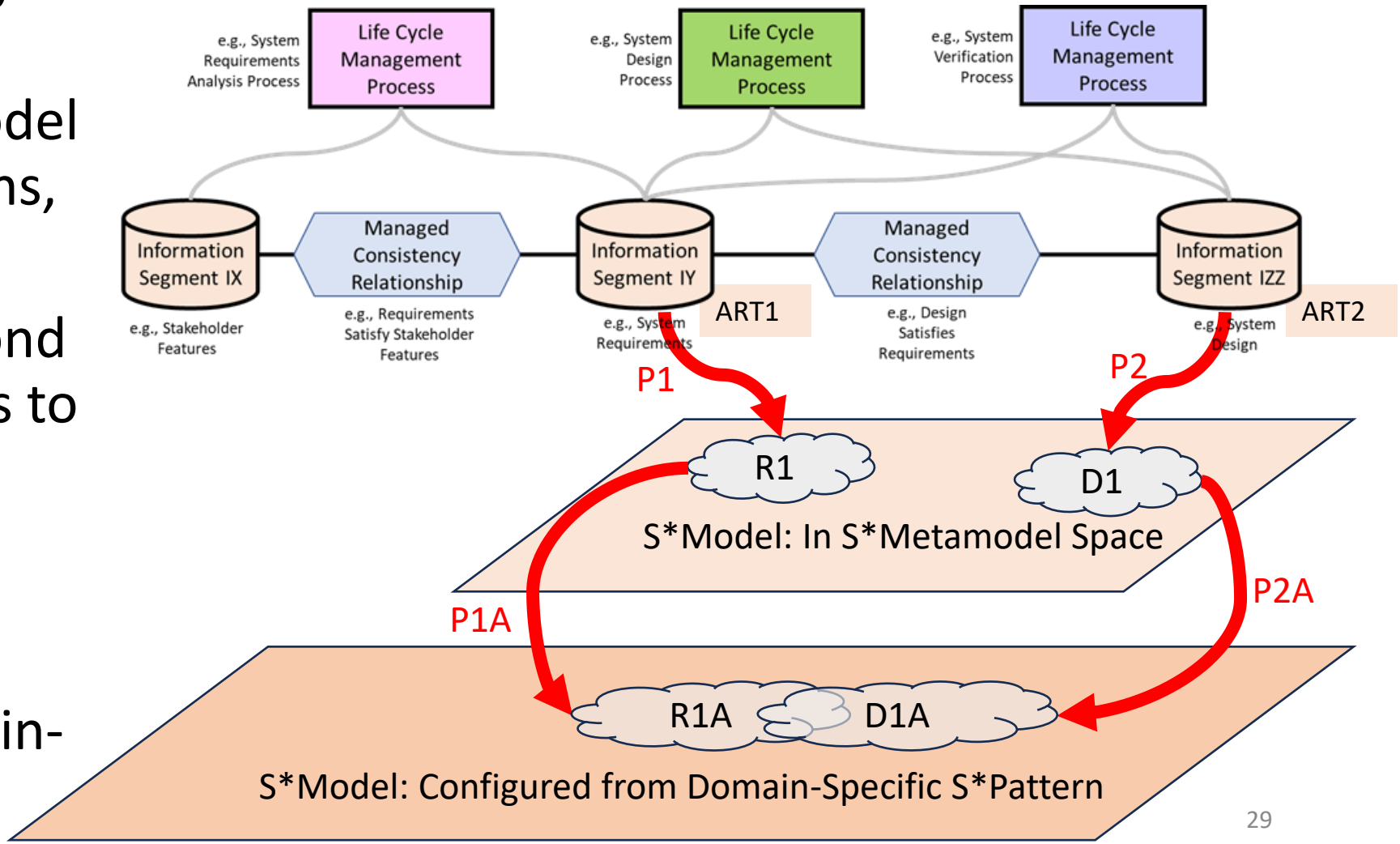
An application: Consistency estimation for information artifacts

- If we can project each artifact into a shared pattern's vector space containing that additional information, then we can more rigorously (and more automatically) estimate their consistency within that space.
- The models become "proxies" for the projected artifacts.
- Such projections can best occur in two sequential stages:
  - From Artifact to S*Model Space: P1 and P2.
  - From S*Model to Configured Domain-Specific S*Model Space: P1A and P2A.



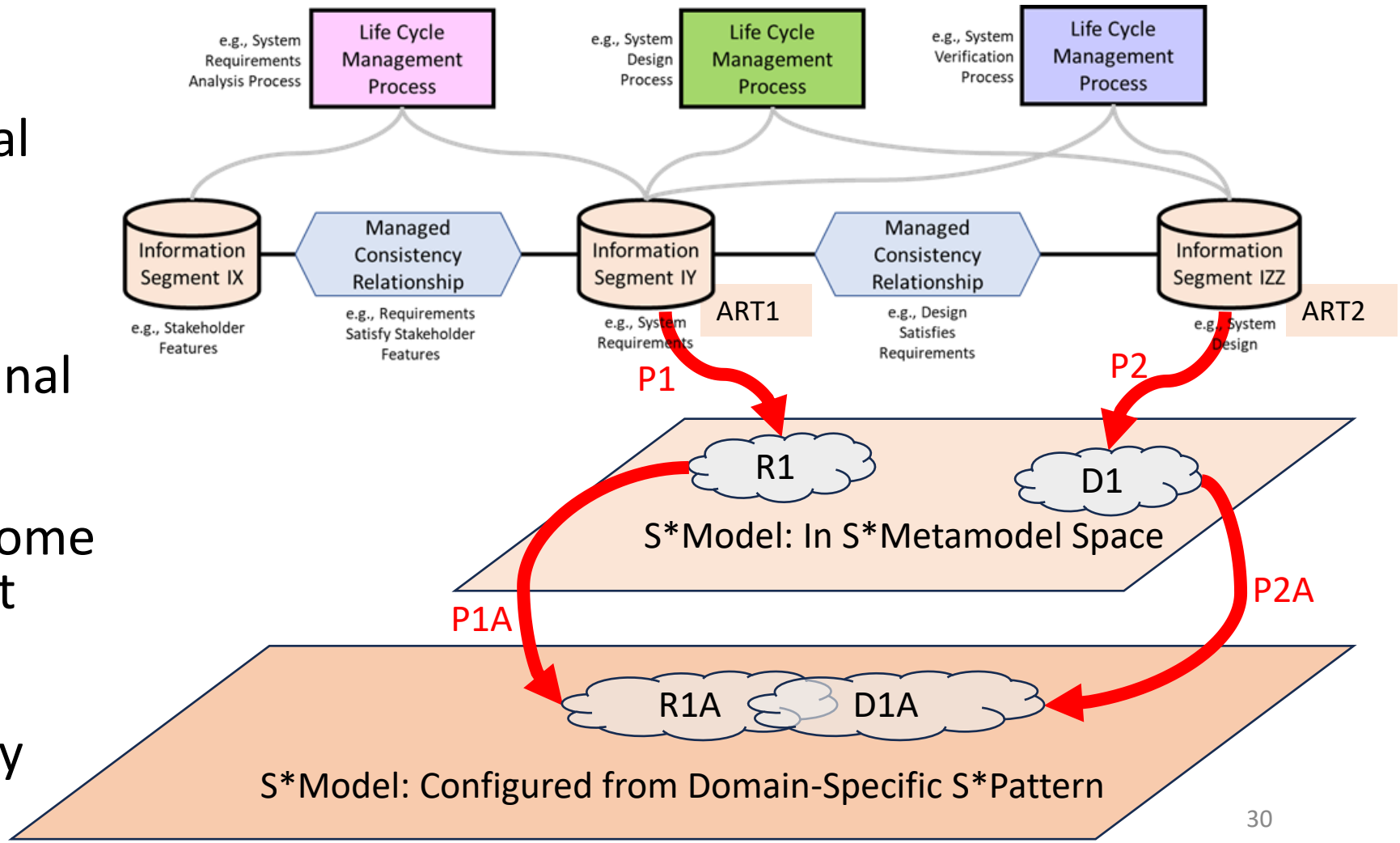
An application: Consistency estimation for information artifacts

- The purpose of the first projections (P1, P2) is to translate arbitrary form artifacts into S*Metamodel components (Interactions, Roles, Features, etc.).
- The purpose of the second projections (P1A, P2A) is to translate to the more specific semantics of a domain-specific model (which is itself a single configuration of a domain-specific S*Pattern).



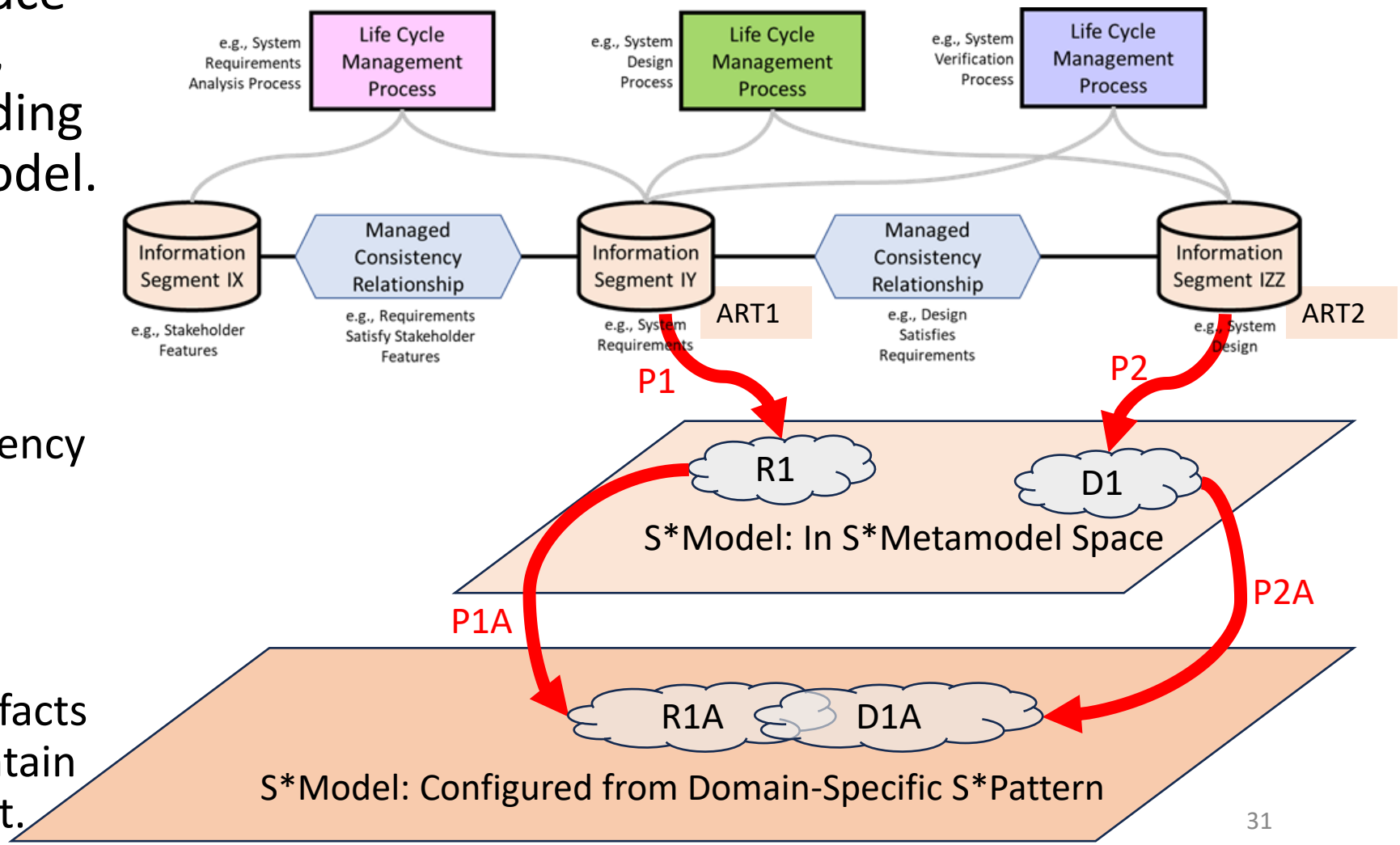
An application: Consistency estimation for information artifacts

- If either of the original artifacts (ART1, ART2) are already in the form of S*Models, then the first projection (P1, P2) is trivial and no translation error occurs as part of the first translation.
- But if either of those original artifacts is not already an S*Model, then the first projection may result in some error, caused by imperfect projection, or by artifact content that lies outside S*Metamodel space, or by artifact ambiguities.



An application: Consistency estimation for information artifacts

- The second projection (P1A, P2A) should rarely introduce any additional error itself, other than misunderstanding of the domain-specific model.
- This brings us to the consistency of the two projections R1A and D1A.
  - It <u>appears</u> that the consistency of R1A and D1A measures consistency of interest.
  - <u>But</u> P1 and P2 can also introduce additional error, caused by “non-ideal” artifacts that are ambiguous or contain non-S*Metamodel content.



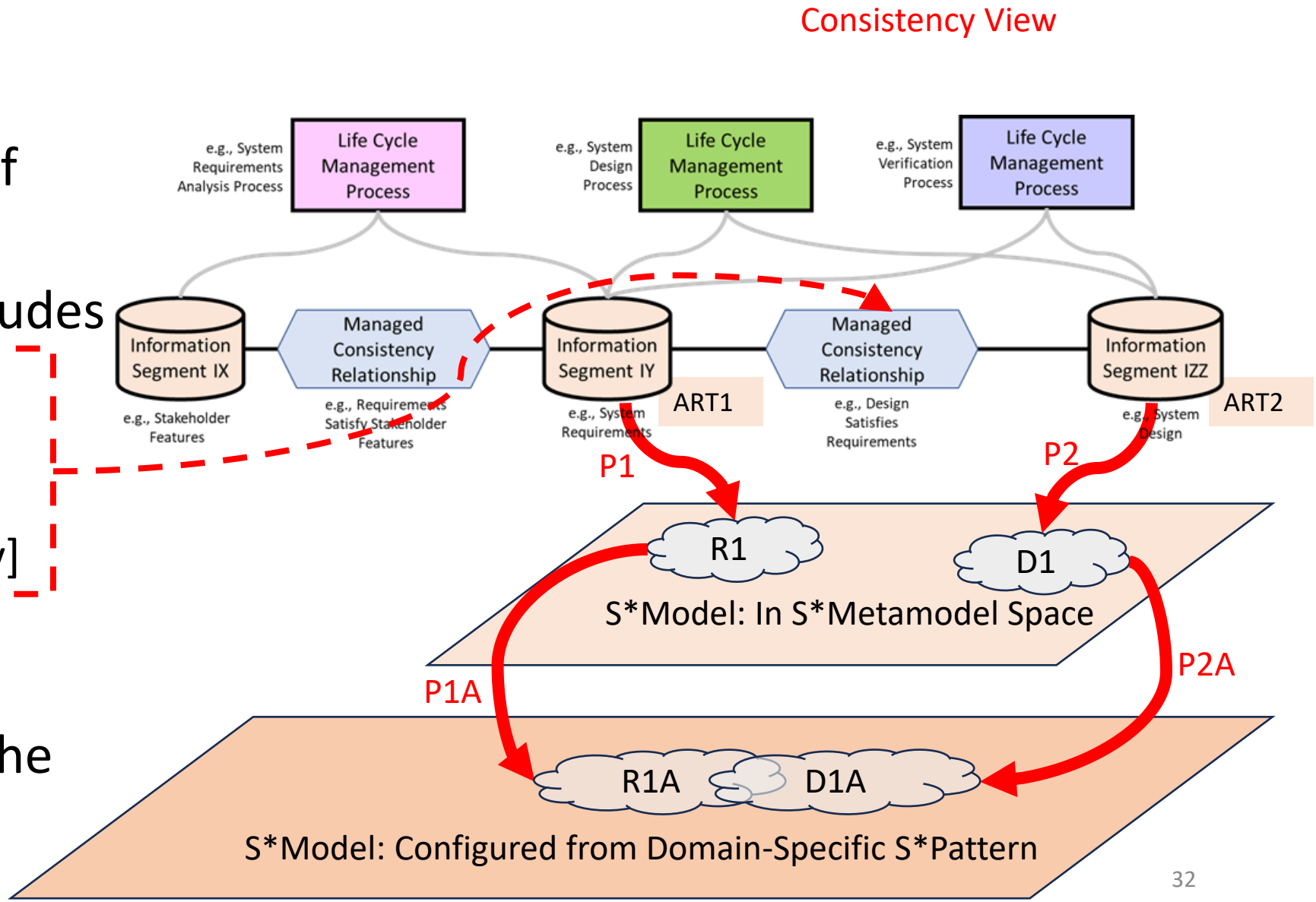
An application: Consistency estimation for information artifacts

- So, we can see that the general expression of inconsistency consists of three parts:

Overall Inconsistency includes

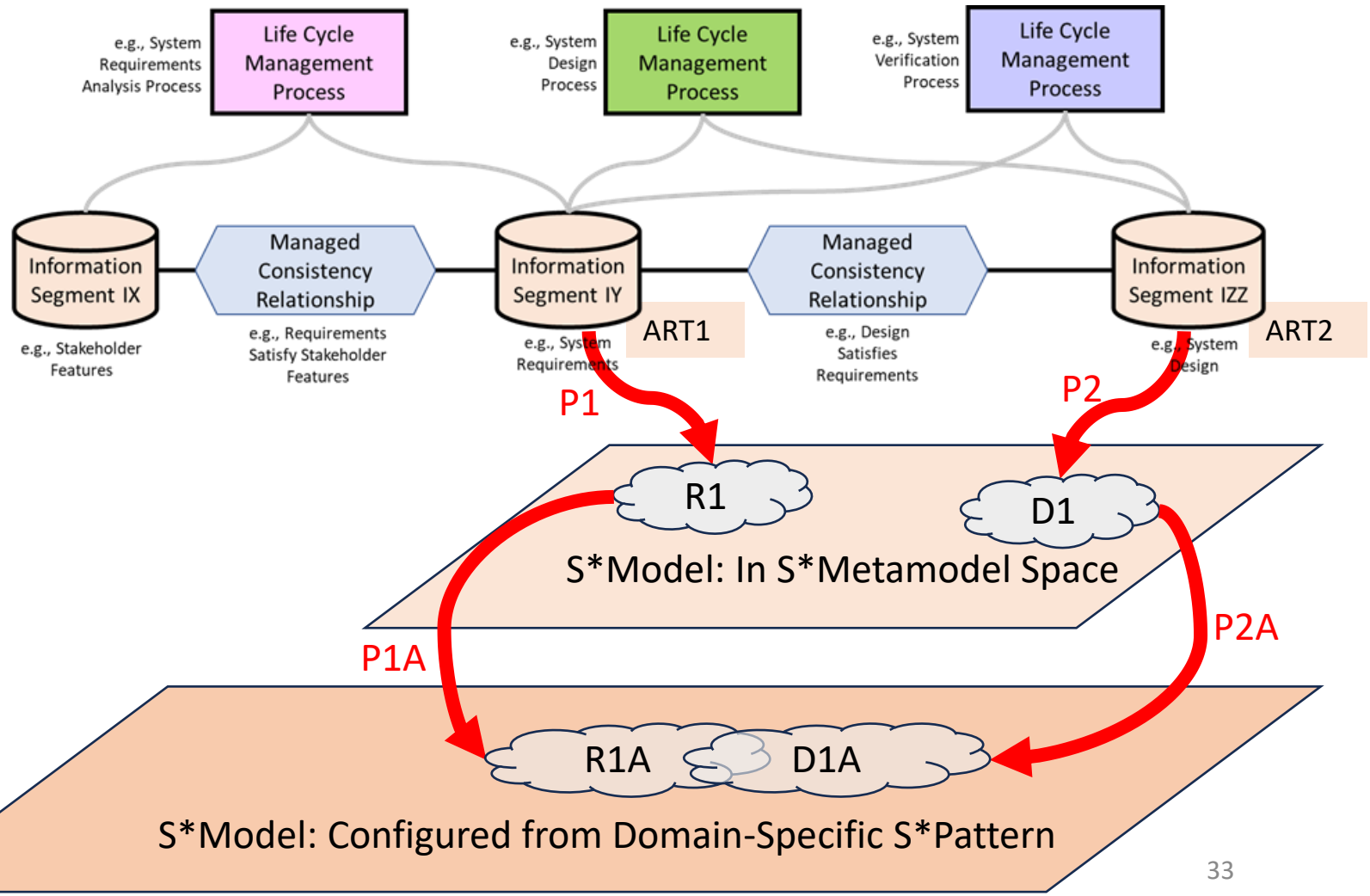
- [P1 Projection Error] ;
- [P2 Projection Error] ;
- [P1A P2A Domain Projection Inconsistency]

- All three inconsistency components should be separately reported in the “consistency view”



An application: Consistency estimation for information artifacts

- Notice that these errors should “shrink over time”.
- The reason is that, during a project, the artifact consistencies will be evaluated multiple times, in a form of “continuous checking”.
- This means that the manual, semi-automatic, or automated projection agents should accumulate learning that improves their projection errors over time.



Architectural Implications for consistency management

In checking the consistency of a Subject Artifact Set and a Referent Artifact Set: (see diagram next)

1. Generate a configuration of a reference pattern intended to match up with the referent artifact set. This is the referent proxy model. It is possible that there may be a significant difference between that pattern referent configuration and what the referent artifacts say. This may lead to more immediate (a) update to the referent artifacts and/or (b) update to the reference pattern or (c) update to its configuration. After some degree of alignment (quite possibly not complete alignment), the difference between what the referent artifacts say and what the configured pattern says should be recorded as a part of the consistency view set. A wide enough gap there may suggest de-rating the influence of the referent artifacts on the judged fidelity of the subject artifacts—because the referents themselves may need adjustment in an "upstream feedback" case. Notice that the language of the artifacts may be very far from S*Metamodel concepts, so that even judging whether they are aligned may be challenging.
2. Generate a (possibly different) configuration of the same reference pattern intended to match up with the subject artifact set. This is the subject proxy model. It is quite possible that there may be a significant different between the pattern subject configuration and what the subject artifacts say. This may lead to more immediate (a) update to the subject artifacts and/or (b) update to the reference pattern or (c) its configuration. After some degree of alignment (quite possibly not complete alignment), the difference between what the subject artifacts say and what the configured pattern says should be recorded as a part of the consistency view set. A wide enough gap there may suggest doubt about either the subject artifacts or the reference pattern or both.
3. The two configurations of (1) and (2) are not necessarily the same. At this point, analyze the difference between the subject proxy model and referent proxy model. If the gaps detected in (1) and (2) are small enough, then this third gap may be an indicator of the "actual" degree of inconsistency of the referent and subject artifacts. This third difference becomes the third part of the consistency view.

Notice the following diagram shows this is effectively an extended neural network . . .

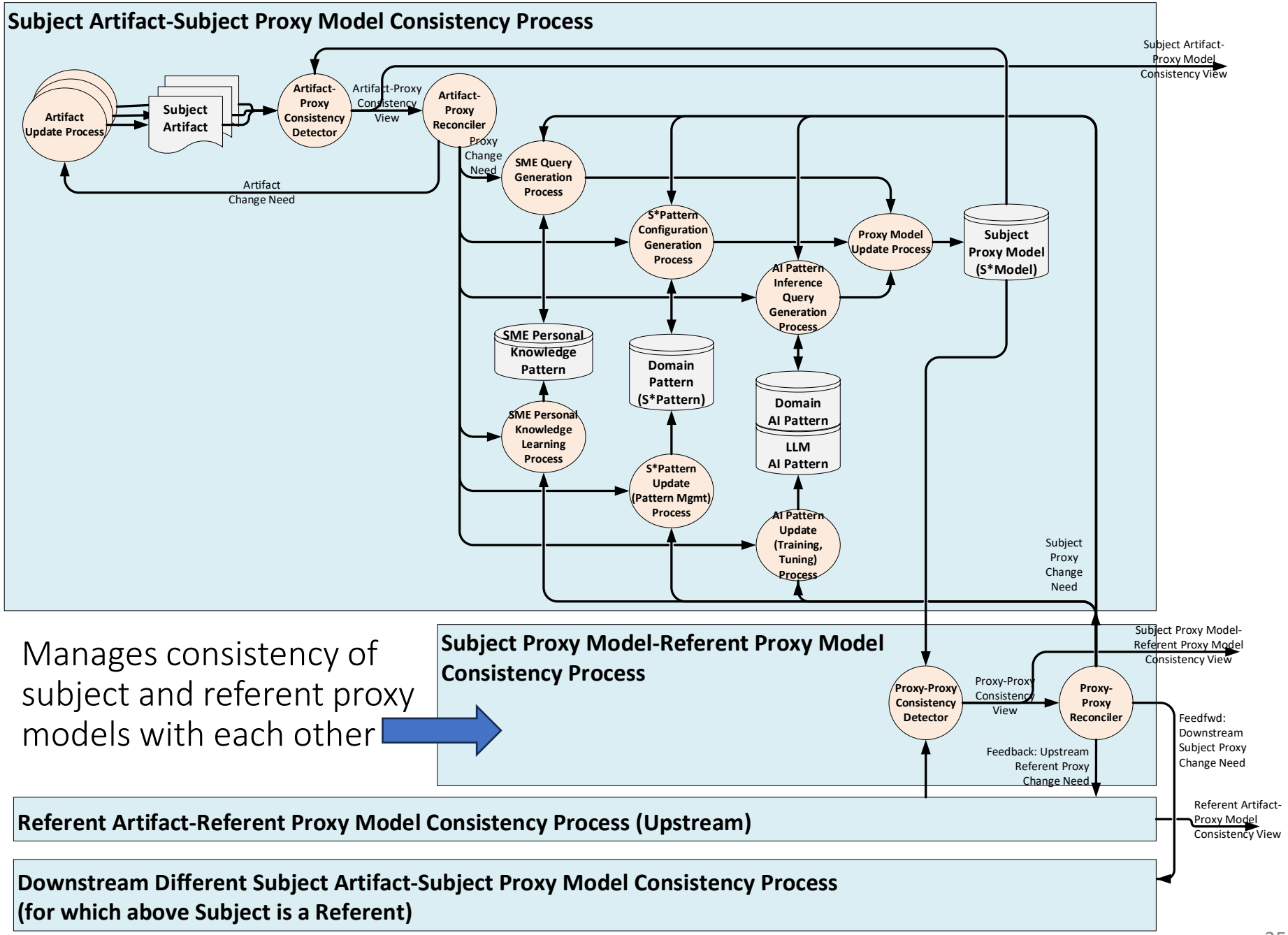
Architectural implications for consistency management (see discussion above)

Manages consistency of subject artifacts with a subject proxy model

Manages consistency of subject and referent proxy models with each other

Manages consistency of referent artifacts with a referent proxy model

More of same



References

1. Crowe, M. J., *A History of Vector Analysis: The Evolution of the Idea of a Vectorial System*. Dover Publications. New York, 1985.

2. Simmons, G. F., *Introduction to Topology and Modern Analysis*, McGraw-Hill, New York, 1963.

3. Lass, H., *Vector and Tensor Analysis*, McGraw-Hill, New York, 1950.

4. Ananthaswamy, A. *Why Machines Learn: The Elegant Math Behind Modern AI*. Dutton, Penguin, Random House Publishers, New York. 2024.

5. INCOSE MBSE Patterns Working Group Web Site. Access from-- https://www.incose.org/communities/working-groups-initiatives/mbse-patterns

6. INCOSE MBSE Patterns Working Group. "S*Models and the S*Metamodel: A Primer". 2024. Down-loaded from: https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:s-models_and_the_s-metamodel--a_primer_wg_review_draft_v1.2.2.pdf

7. INCOSE MBSE Patterns Working Group. S*Patterns: A Primer". 2024. https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:s-patterns--a_primer_wg_review_draft--v1.2.1.pdf

8. "The Systematica Metamodel", Version 8.0. ICTT System Sciences, May 1, 2025. Download from -- www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:systematica_8_metamodel_v8.0.61.pdf

9. INCOSE Patterns Working Group. "Consistency Management as an Integrating Paradigm for Digital Life Cycle Management with Learning." Download from -- https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:aselcm_pattern_--_consistency_management_as_a_digital_life_cycle_management_paradigm_v1.3.1.pdf

10. Schindel, W., "Realizing the Promise of Digital Engineering: Planning, Implementing, and Evolving the Ecosystem", *INCOSE 2022 International Symposium*, July 2022. Upload paper from https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:realizing_the_vision_of_digital_engineering_is2022_v1.3.4.pdf

11. Schindel, W., "Realizing the Promise of Digital Engineering: Planning, Implementing, and Evolving the Ecosystem", INCOSE 2022 International Symposium, July 2022. Upload presentation from https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:incose_is2022_realizing_the_promise_of_digital_engineering_v1.1.3.pdf

12. LeCun, Y., Bengio, Y., Hinton, G. "Deep Learning", in *Nature*, pp 436-444. May 2015. https://www.researchgate.net/publication/277411157_Deep_Learning

13. Fischer, French, Hightower, Karl, Matlik, Pullum, Schindel, Shao, and Taylor, "A Cross-Society Collaboration Project: Mapping Consistency Confirmation Frameworks of Different Communities—Phase I Project Report", presented at AIAA SciTech 2025, January, 2025. Download from -- https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:rosetta_stone_project_report_01.08.2025_v2.1.5.pdf

References

14. Schindel, W., “Innovation Ecosystem Dynamics, Value, Learning II: Can V&V Frameworks Tell Us More?”, Presentation to INCOSE 2025 International Symposium, July, 2025. Download from -- https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:v_and_v_in_innovation_ecosystems_ver1.2.1.pdf

15. Oberkampf, W. and Roy, C., *Verification and Validation in Scientific Computing*, Cambridge U. Press, 2010.

16. ASME VVUQ 1-2022: “Verification, Validation, and Uncertainty Quantification Terminology in Computational Modeling and Simulation”, 2022.

17. Boyer, C. B., “Descartes and the Geometrization of Algebra”, *The American Mathematical Monthly*, Vol. 66, No 5., May, 1959, pp. 390 – 393.

18. “Principal component analysis”. Wikipedia. Download from -- https://en.wikipedia.org/wiki/Principal_component_analysis

19. Raissi, M., Perdikaris, P., and Karniadakis, G. E., “Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations”. *Journal of Computational Physics*, Volume 378, 1 February 2019, Pages 686-707.

20. Rothman, D., *RAG-Driven Generative AI*, Packt Publishing, Birmingham, UK, 2024.

21. Schindel, W., “Examples of AI (Generative Pretrained Transformer LLM) Generation and Analysis of S*Models and S*Patterns”, INCOSE Patterns Working Group, 2024. Download from – https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:examples_of_ai_generation_and_analysis_of_models_and_patterns_v1.3.3.pdf

22. Schindel, W. “Innovation Ecosystem Dynamics, Value and Learning I: What Can Hamilton Tell Us?”. *Proc. of INCOSE 2024 International Symposium*. Dublin. 2024. https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:incose_is2024_hamiltonian_ecosystem_schindel_v1.6.2.pdf

23. ISO, "ISO/IEC/IEEE 15288-2023: ISO/IEC/IEEE International Standard - Systems and software engineering -- System life cycle processes", 2023.

24. NVIDIA, “RTX PRO: Accelerating professional AI, graphics, rendering, and compute workloads.” NVIDIA. Download from https://www.nvidia.com/en-us/products/workstations/

25. “AMD Instinct™ Accelerators: Supercharging AI and HPC to Propel New Discoveries.” AMD. Download from https://www.amd.com/en/products/accelerators/instinct.html

26. “Domain-specific AI agents at scale: CustomGPT.ai serves 10,000+ customers with Pinecone.” Pinecone. Download from: https://www.pinecone.io/customers/customgpt-ai/

27. “The High-Performance Vector Database Built for Scale.” Milvius. Download from -- https://milvus.io/