What Is the Smallest Model of a System?

Big — Models — Little

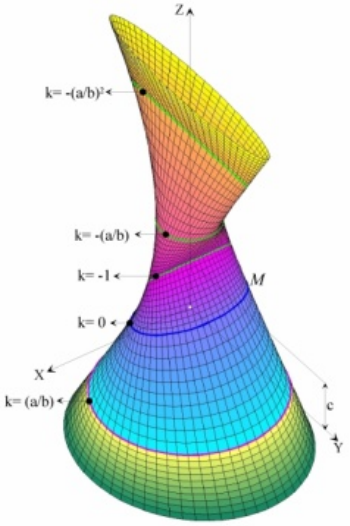
Small — Systems — Large

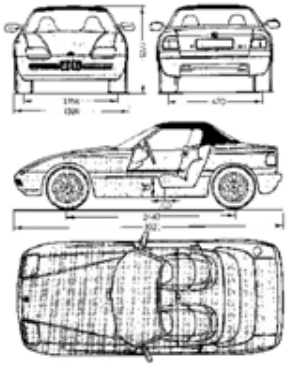
William D. Schindel
ICTT System Sciences
schindel@ictt.com

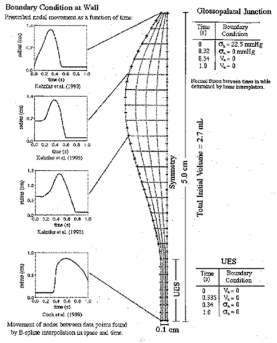
Contents

- Problem statement: Size matters!
- Constructing an efficient representation
- Using patterns to compress models
- Results and implications

Additional Information





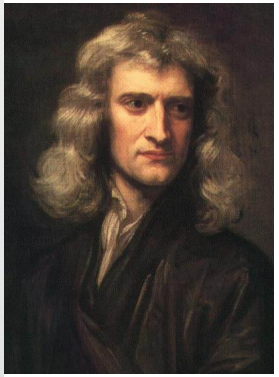


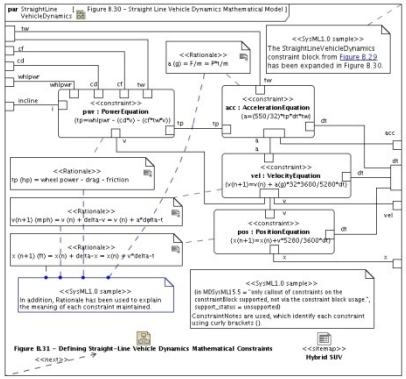
Problem Statement

- How we <u>represent</u> systems is fundamental to the history of mathematics, science, and engineering.
- Why is <u>minimality</u> of representation of interest?
  - *<u>Scientific</u>* interests: The size of a system's minimal representation is used to define that system's <u>complexity</u>.[1]
  - *<u>Practical</u>* interests: The size and redundancy of engineering specifications <u>challenge the effectiveness of real-world engineering processes</u>.
- What is the <u>smallest</u> representation of a system?

Size matters!

- We describe a (possibly least) upper bound on size of effective representations for systems engineering (SE) purposes:
  - Consistent with current model-based SE trends, extending their power;
  - Drawn more directly from scientific traditions for representing systems based on physical interactions, compared to typical SE sources;
  - When used for system families (product lines, ensembles), this representation also facilitates compression by use of system patterns.

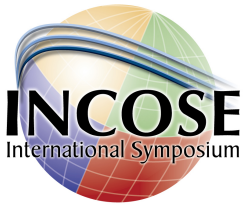




What do we mean by “size” of a model?

Not this!

Practical challenges in traditional SE representations



- Traditional task-specific representations document systems:

CONOPS	Requirements	Architecture
Design Specs	FMEAs	Test Plans
O&M SOPS	Use Cases	Other . . .

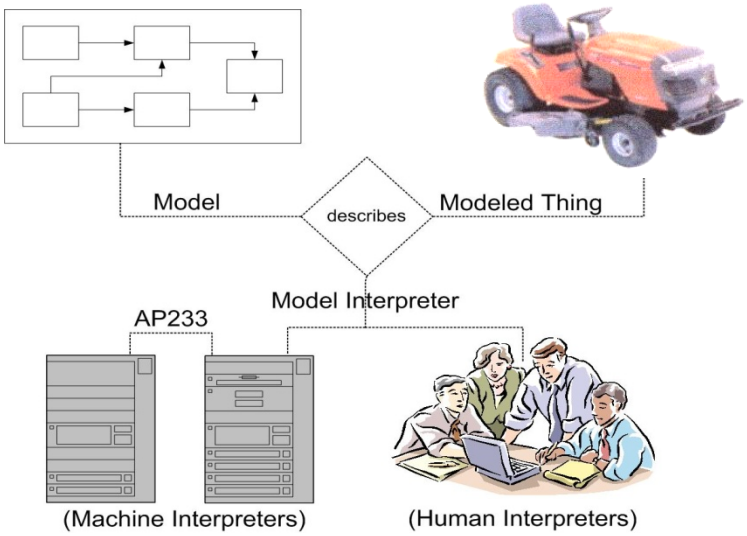
- Can run hundreds or thousands of pages during life cycles.
- Typical: Provide the same system document to three different expert readers, and get back three different interpretations:
  - This would be considered unacceptable for an electronic schematic—so why accept it for "systems engineering" artifacts?
- Subjective expert judgments are typically required to assess artifact completeness and consistency;
- These are among reasons cited for model-based methods.

Complexity science

- Subject of formal study for both natural and human-engineered systems;
- Initial efforts sought a theoretical basis for measuring and understanding complexity [Li & Vitany, Chaiten, Kauffmann];
- More recently, practical implications for engineering processes [Bar-Yam, Braha, Kuras & White, Schindel]
- Terminology of Complex adaptive systems (CAS), Complex systems engineering (CSE), various INCOSE working groups, etc.
- Growing awareness of the connection between systems science and systems engineering

Model-based methods

- Math-Physics models have longer-standing historical roles in design verification and other prediction of system behavior [Karayanakis];
- As described by other conference speakers, modeling ideas were later extended, using model languages to represent system requirements and design [Mellor, INCOSE, MBSE, SysML Partners, Schindel];
- In all these cases, "model" implies formal, explicit, and unambiguous—potentially a big improvement on prose alone:



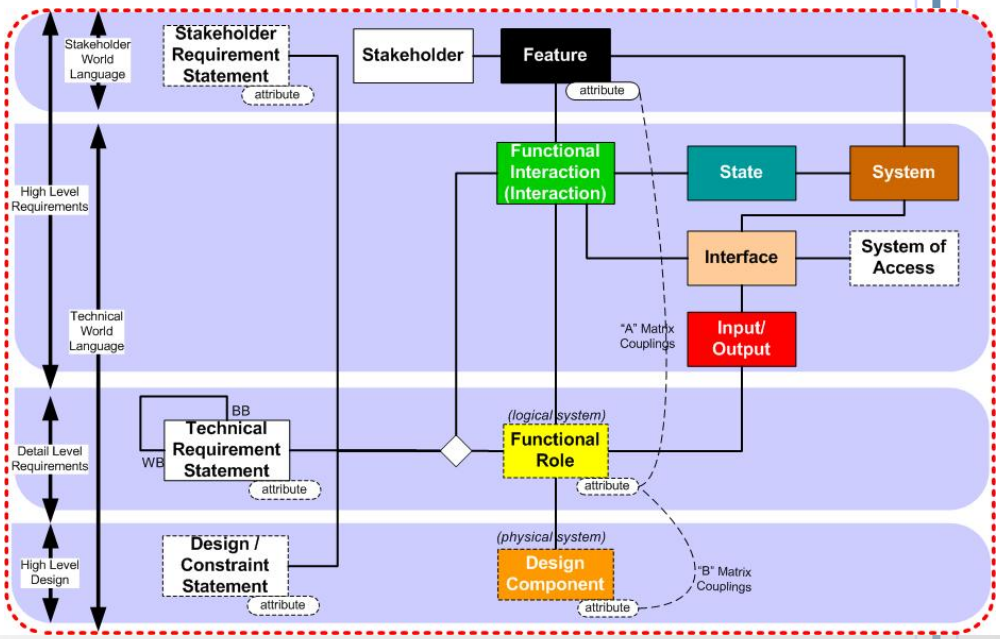
Constructing an efficient representation

- A metamodel is a model of other models;
  - Sets forth how we will represent Requirements, Designs, Verification, Failure Analysis, Trade-offs, etc.;
  - We utilize the (language independent) S* Metamodel from Systematica™ Methodology:

- The resulting system models may be expressed in SysML™, other languages, DB tables, etc.

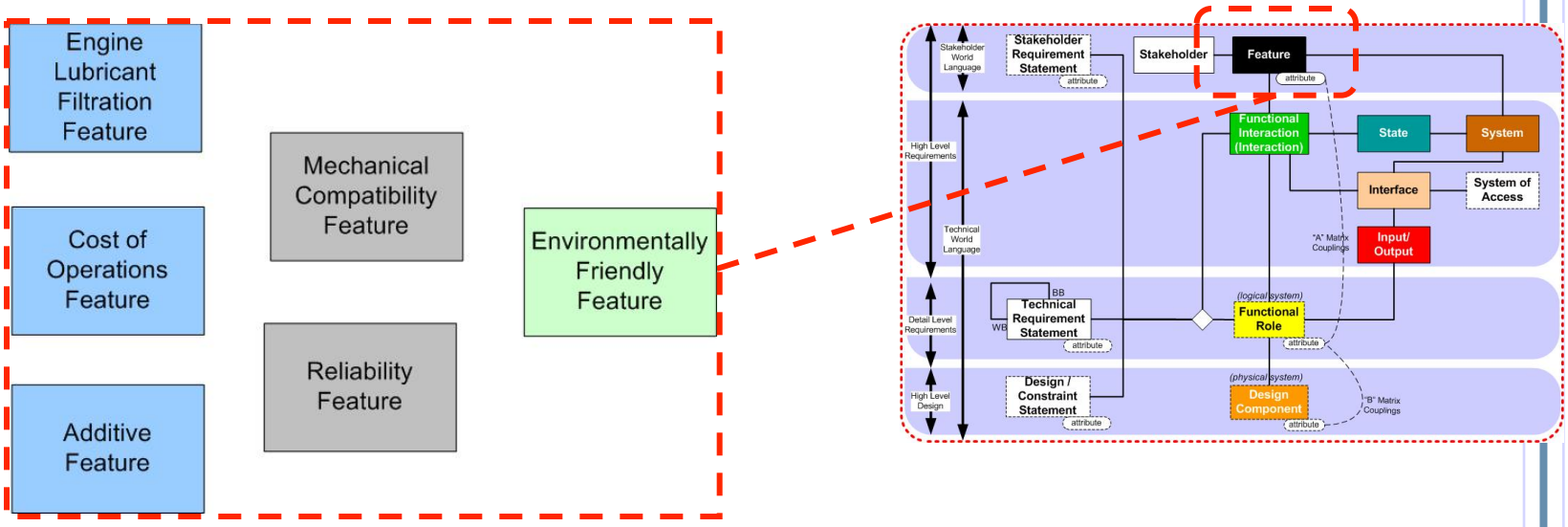
- Has been applied to systems engineering in aerospace, transportation, medical, advanced manufacturing, communication, construction, other domains.

Simple summary of detailed S* Metamodel.



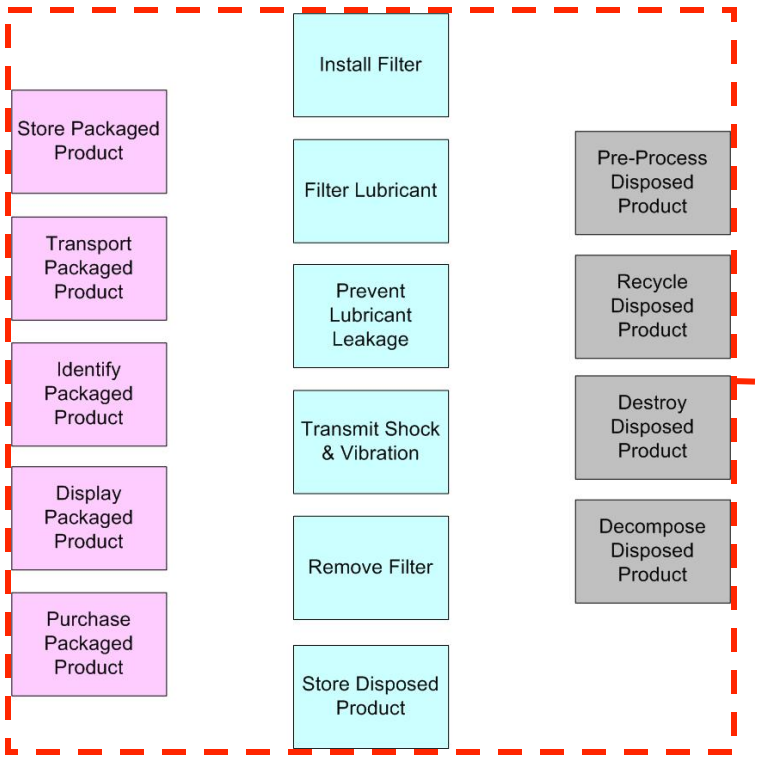
Stakeholder Features: Expressing Value, for many SE Purposes

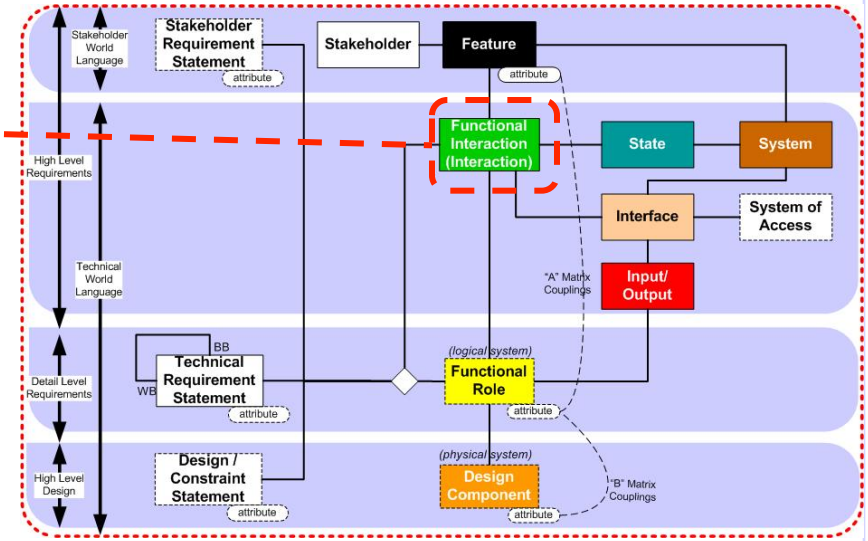
- S* models represent Value as explicit objects & attributes:
  - Essential for representing goals, innovation progress, trade space, effects of failure modes, expression of risk, etc.
  - By covering all Stakeholders and their Features, these become the scoreboard for all decision-making and risk management.
  - Example: Oil Filter Features



Physical Interactions: At the heart of S* models

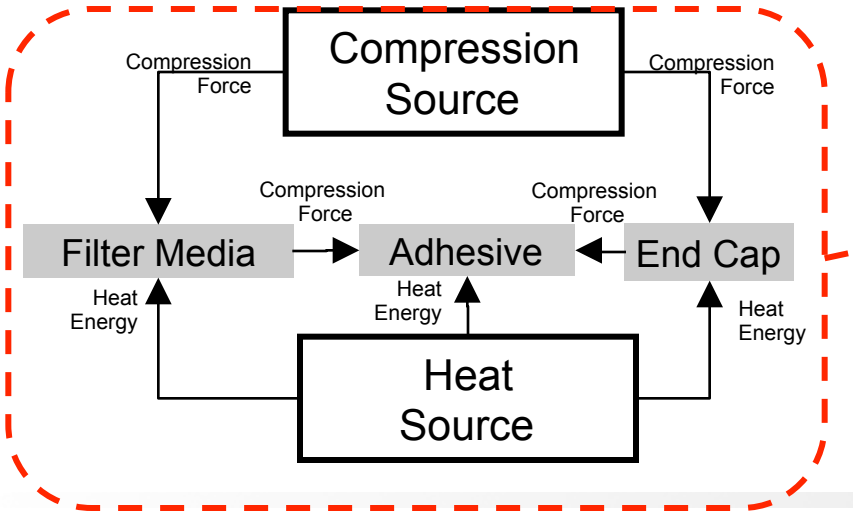
- S* models represent Physical Interactions as explicit objects:
  - Example: Oil Filter Interactions:

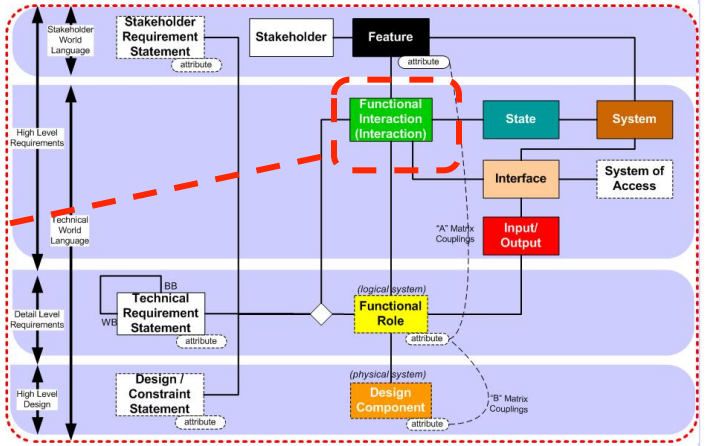




Physical Interactions: At the heart of S* models

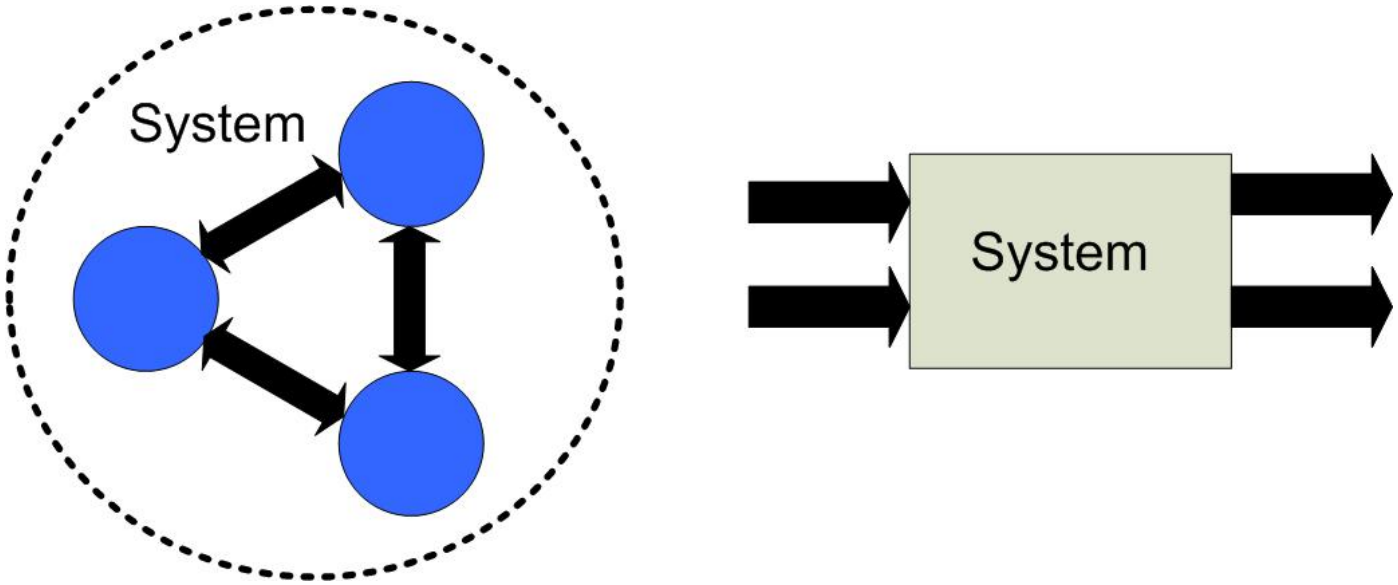
- S* models represent Physical Interactions as explicit objects:
  - Goes to the heart of 300 years of natural science of systems as a foundation for engineering, including emergence.
  - Link to traditional mathematical-physical modeling.
  - Interacting elements perform Functional Roles, based on allocated Requirements.
  - *All functional requirements are revealed as external interactions* [Schindel 2005].
  - Example: Oil Filter Mfg Process Bonding:





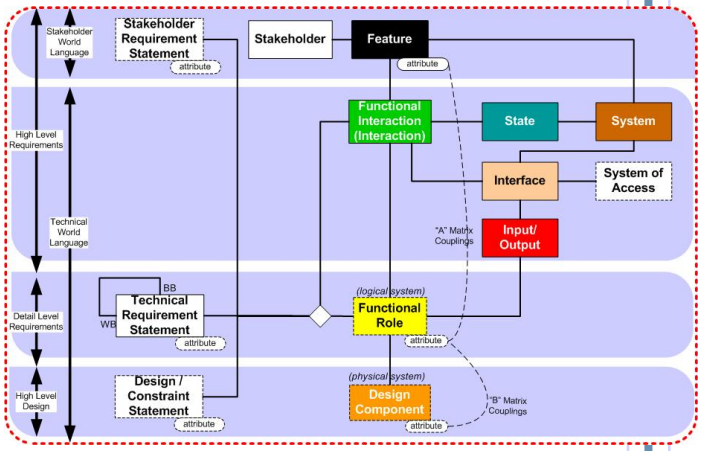
Physical Interactions: At the heart of S* models

- Emergence, emergent properties are based on Interactions
- Two different mental starting points for thinking about systems
  - Systems as interacting components, versus SIPOC perspective:



S* Metamodel

- Other Metaclasses and Relationships include:
  - States (Modes, Situations)
  - Interfaces
  - Input-Outputs
  - Systems of Access
  - Design Components (Physical Elements)
  - Other classes (see the References)
  - Relationships between them
  - Attributes of the classes and relationships

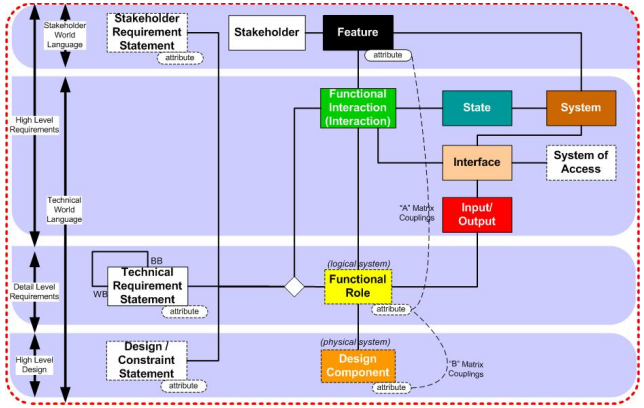


- Modeling Language?
  - None if this is specific to modeling language (e.g., SysML, etc.)
  - Rather, it is about underlying information that must be addressed.

Model minimality: Summary of the argument

- Summary of the formal argument of S* model minimality:
  - Sufficiency Argument: This part of the argument demonstrates that the information in S* models is sufficient for the needs of various systems engineering processes;
  - Minimality Argument: This part of the argument demonstrates that removal of any class of S* information results inability to adequately perform an SE process.
    - Example: The use of States in representing Black Box Requirements—"when" does each Requirement apply?
- This argument makes use of a mapping of which S* model components are needed for different SE tasks; e.g.:

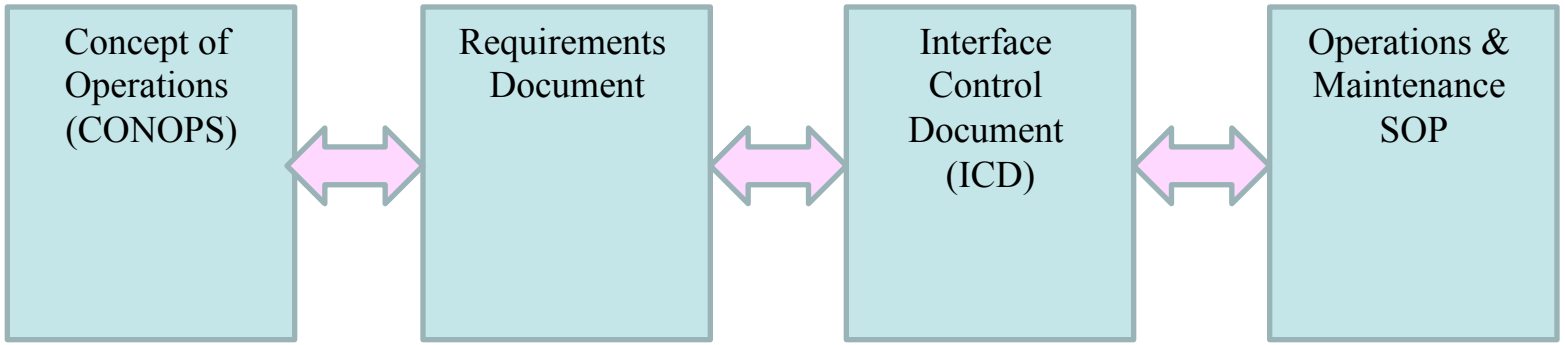
SE Area	Grp1	Grp 2	Grp 3	Grp 4	Grp 5
HLR	X				
DLR/BB	X	X			
DLR/WB	X	X			
HLD	X	X	X		
FMEA	X	X	X	X	
TST	X	X	X	X	X



- This argument is constructive: It not only tells us that such a model exists—it also tells us how to construct it.
- However, the argument does not include uniqueness: Other data structures could represent the same system.

Model views; useful redundancy

- A familiar challenge is that different "SE Documents" may be inconsistent with (contradict) each other:
  - This is because they contain redundant information
  - As documents evolve, that consistency must be maintained to be consistent across the "documents":

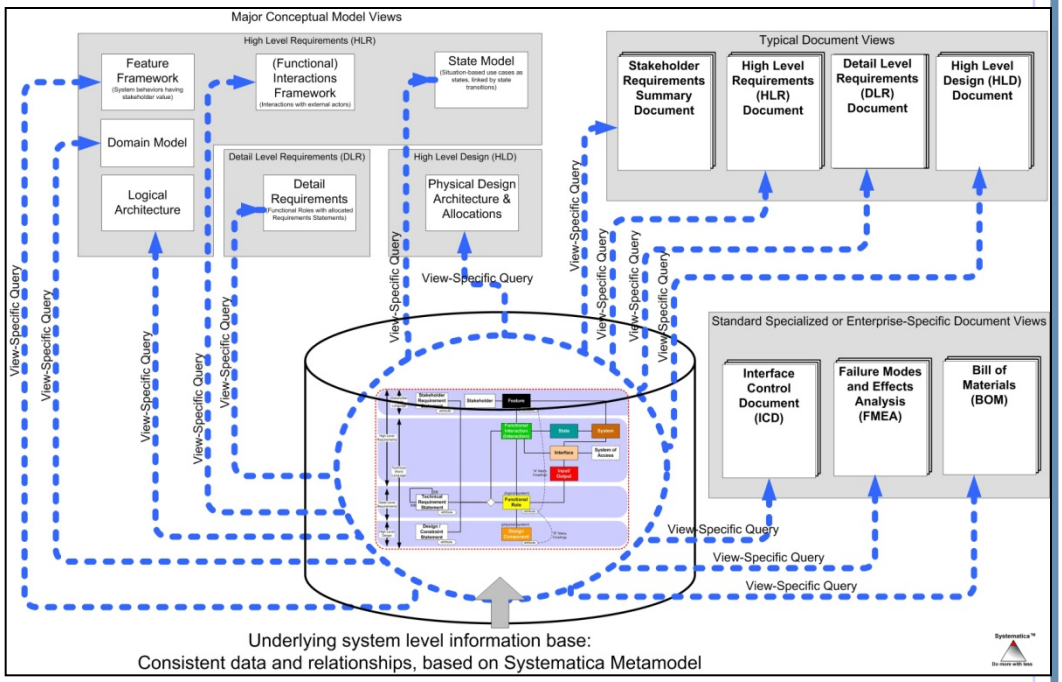


  - This issue also occurs <u>within</u> single documents (self-consistency)

Model views; useful redundancy

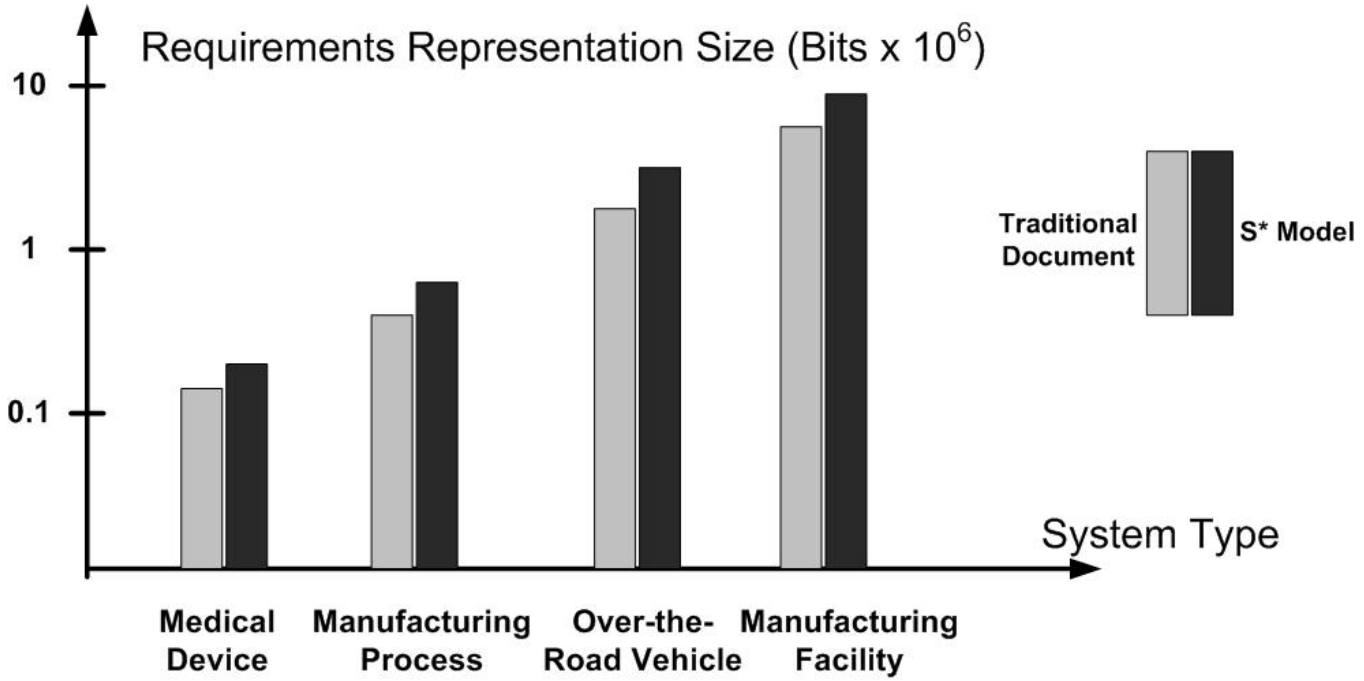
- This is one reason why DB tools are powerful in systems engineering:
  - Properly used, they can generate different "views" (documents, etc.) from the common underlying data model, thereby maintaining their consistency:

- The S* Model goes farther, by pointing out redundancies not always recognized; e.g.:
  - FMEA Failure Effects vs. Stakeholder Features
  - FMEA Functional Failures vs. Requirements (Counter-Requirements)
  - ICDs vs. System Requirements
  - CONOPS and Use Cases vs. System Requirements, Features
  - Such "redundancies" are really deep insights that make model construction easier & reinforcing.



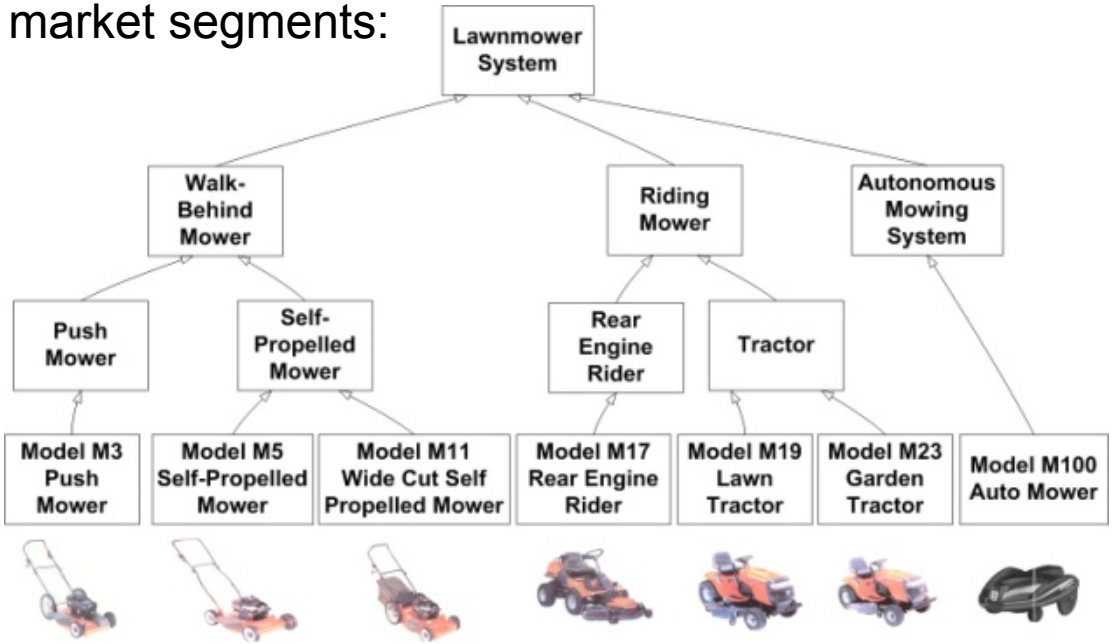
Practical issues of size

- So, how big are S* models compared to other models?
  - A practical discovery is that a typical S* model of requirements is more complete than a corresponding traditional description—it is bigger, not smaller!



Using patterns to compress models

- Descriptions of SE processes typically appear to describe engineering a "new" system "from scratch" [e.g., ISO 15288, INCOSE SE Handbook]:
  - However, real projects are often concerned with engineering similar (but different) systems across different product generations, applications, configurations, or market segments:



- How should SE processes be adjusted to <u>explicitly</u> address "Variable Sameness"?

Pattern-Based Systems Engineering (PBSE)

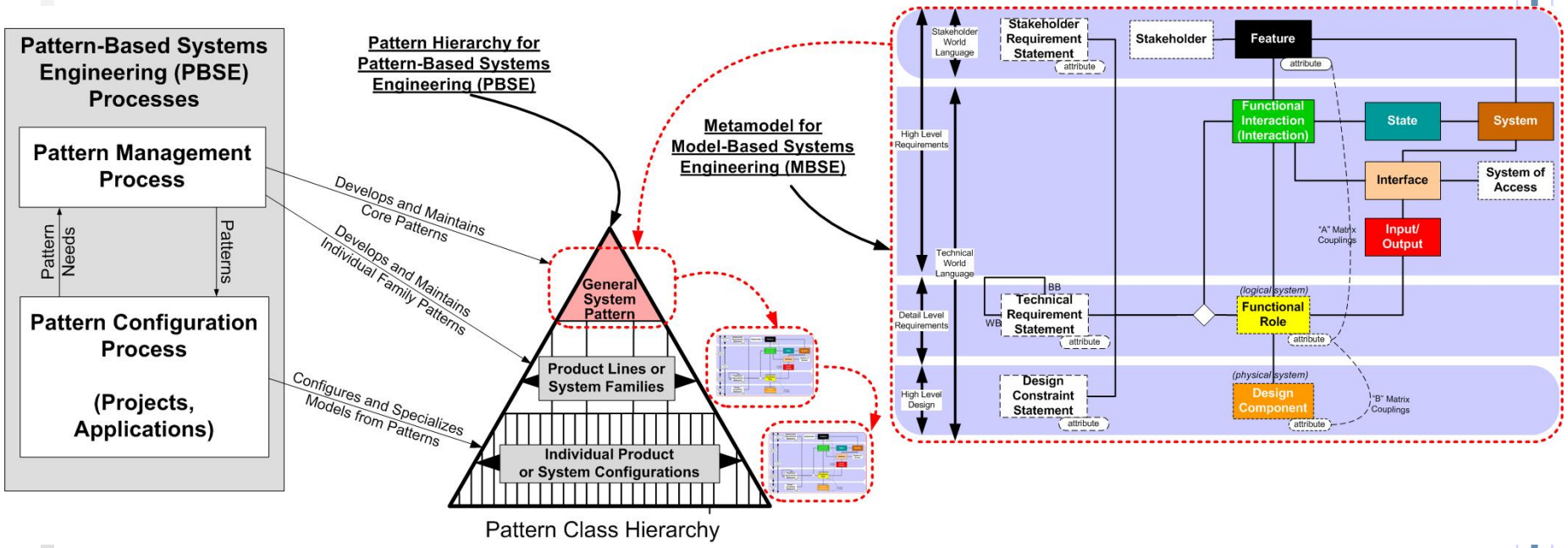
- Model-based Patterns:
  - In this approach, S* Patterns are reusable, configurable S* Models of families (product lines, sets, ensembles) of systems.
- These Patterns are ready to be configured to serve as Models of individual systems in projects.
- Configured here is specifically limited to mean that:
  - Pattern model components are populated / de-populated, and
  - Pattern model attribute (parameter) values are set

. . . both based on configuration rules that are part of the Pattern.

- S* Patterns are based on the same S* Metamodel as "ordinary" S* Models

Pattern-based systems engineering (PBSE)

- Pattern-Based Systems Engineering (PBSE) has two overall processes:
  - **Pattern Management Process**: Generates the underlying family model, and periodically updates it based on application project discovery and learning;
  - **Pattern Configuration Process**: Configures the pattern into a specific model for application in a project.

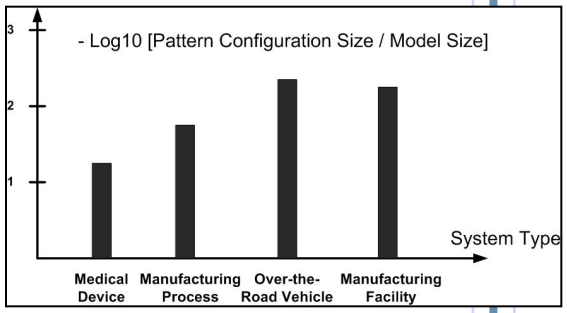


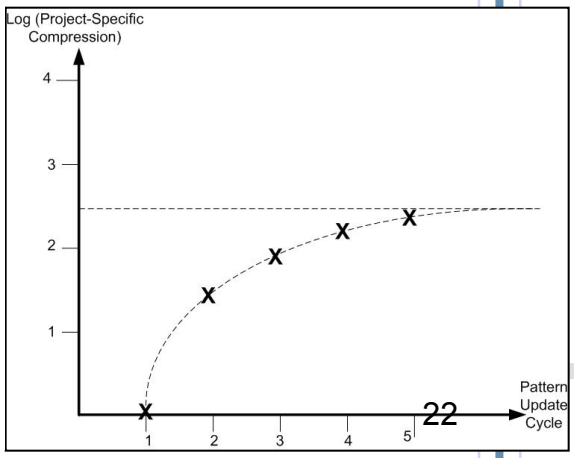
Pattern configurations

- A table of configurations illustrates how patterns facilitate compression;
- Each column in the table is a compressed system representation with respect to ("modulo") the pattern;
- The compression is typically very large;
- The compression ratio tells us how much of the pattern is variable and how much fixed, across the family of potential configurations.

Lawnmower Product Line: Configurations Table

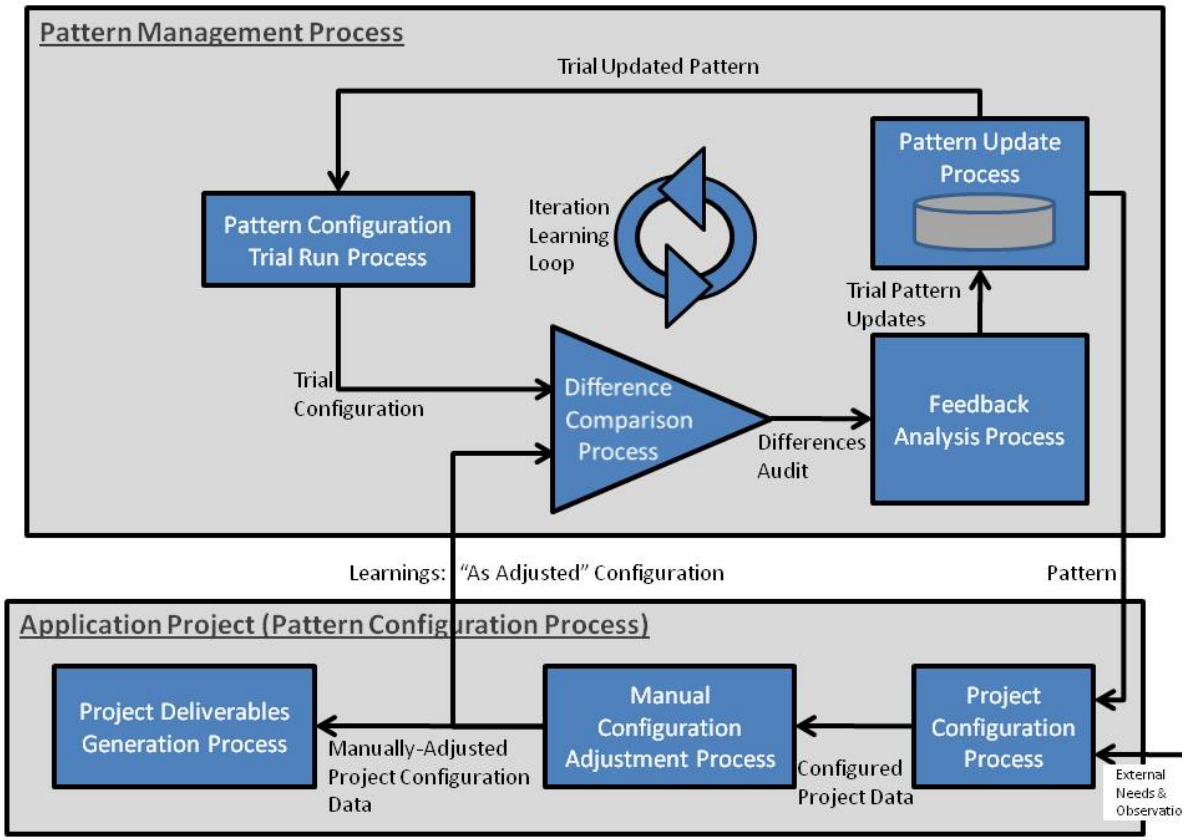
		Units	Walk-Behind Push Mower	Walk-Behind Mower	Walk-Behind Self-Propelled	Riding Rider	Riding Tractor	Riding Mower Tractor	Autonomous Autonomous
			Push Mower	Self-Propelled	Wide Cut	Rider	Lawn	Garden	Auto Mower
	Model Number		M3	M5	M11	M17	M19	M23	M100
	Market Segment		Sm Resident	Med Resident	Med Resident	Lg Resident	Lg Resident	Home Garden	High End Suburban
Power	Engine Manufacturer		B&S	B&S	Tecumseh	Tecumseh	Kohler	Kohler	Elektroset
	Horsepower	HP	5	6.5	13	16	18.5	22	0.5
Production	Cutting Width	Inches	17	19	36	36	42	48	16
	Maximum Mowing Speed	MPH	3	3	4	8	10	12	2.5
	Maximum Mowing Productivity	Acres/Hr			1.6				
	Turning Radius	Inches	0	0	0	0	126	165	0
	Fuel Tank Capacity	Hours	1.5	1.7	2.5	2.8	3.2	3.5	2
	Towing Feature						x	x	
	Electric Starter Feature				x	x	x	x	
	Basic Mowing Feature Group		x	x	x	x	x	x	x
Mower	No. of Anti-Scalping Rollers		0	0	1	2	4	6	0
	Cutting Height Minimum	Inches	1	1.5	1.5	1.5	1	1.5	1.2
	Cutting Height Maximum	Inches	4	5	5	6	8	10	3.8
	Operator Riding Feature					x	x	x	
	Grass Bagging Feature		Optional	Optional	Optional	Optional	Optional	Optional	
	Mulching Feature		Standard	Factory Installed	Dealer Installed				
	Aerator Feature					Optional	Optional	Optional	
	Autonomous Mowing Feature								x
	Dethatching Feature					Optional	Optional	Optional	
Physical	Wheel Base	Inches	18	20	22	40	48	52	16
	Overall Length	Inches	18	20	23	58	56	68	28.3
	Overall Height	Inches	40	42	42	30	32	36	10.3
	Width	Inches	18	20	22	40	48	52	23.6
	Weight	Pounds	120	160	300	680	705	1020	15.6
	Self-Propelled Mowing Feature			x	x	x	x	x	x
	Automatic TransmFeature							x	
Financials	Retail Price	Dollars	360	460	1800	3300	6100	9990	1799
	Manufacturer Cost	Dollars	120	140	550	950	1800	3500	310
Maintenance	Warranty	Months	12	12	18	24	24	24	12
	Product Service Life	Hours	500	500	600	1100	1350	1500	300
	Time Between Service	Hours	100	100	150	200	200	250	100
Safety	Spark Arrest Feature		x	x	x	x	x	x	





Pattern Management as Learning Feedback Loop:

An error-correcting loop, as might be practiced in physical sciences, study of markets, or other learning processes.



From: D. E. Williams, "How Concepts of Self-Regulation Explain Human Knowledge", *The Bent of Tau Beta Pi*, Winter, 2011.

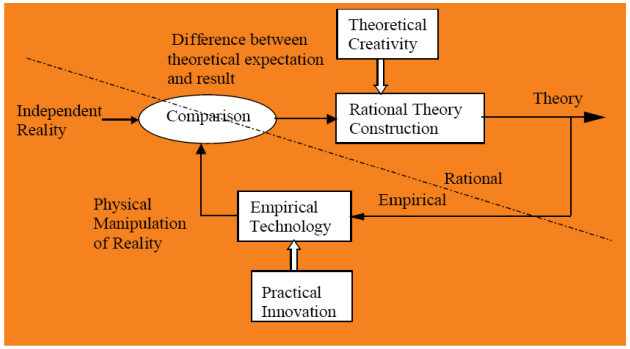


Figure 2: The scientific closed-loop

Results and Implications

1. These methods have been successfully applied across a wide range of domains: Transportation, Mil/Aero, Communications, Medicine/Healthcare, Advanced Manufacturing, Consumer Products.
2. The minimum base of information required to perform SE tasks is clarified by MBSE.
3. Minimal MBSE models contain information missing from many projects.
4. Minimal underlying models generate the redundancies needed across different task-based artifacts, with greater consistency or less effort to maintain that consistency.
5. Formalization of Patterns as configurable Models leads to further size compression: Configurations.
6. All models are actually configurations of more abstract patterns.

References

1. Ahmed, J., Hansen, J., Kline, W., Peffers, S., Schindel, W. 2011. All innovation is innovation of systems: An integrated 3-D model of innovation competency. To appear in *Proceedings of the 2011 American Society for Engineering Education Annual Conference*, Vancouver, BC.
2. Alexander, Christopher; Sara Ishikawa, Ingrid Fiksdahl-King, Shlomo Angel. 1977. *A pattern language: Towns, buildings, construction.* New York: Oxford U. Press.
3. Ashby, W. Ross. 1957. *An introduction to cybernetics*. London: Chapman & Hall.
4. Bar-Yam, Y. 2003b. When systems engineering fails—toward complex systems engineering. *Proceedings of the International Conference on Systems, Man & Cybernetics*, Vol 2, 2021-2028. Piscataway, NJ: IEEE Press.
5. ———. 2005. About engineering complex systems: multiscale analysis and evolutionary engineering. ESOA 2004, LNCS 3464, pp 16-31, Spinger-Verlag, 2005.
6. Bradley, J, Hughes, M, Schindel, W. 2010. Optimizing delivery of global pharmaceutical packaging solutions, using systems engineering patterns. *Proceedings of the INCOSE 2010 Symposium.*
7. Braha, D., A. Minai, Yaneer Bar-Yam, eds. 2006. *Complex engineered systems: Science meets technology*, City: Springer.
8. Chaitin, Gregory. 2005. *Metamath: The quest for omega*, New York: Pantheon, 2005.
9. Cloutier, Robert J., Dinesh Verma. 2007. Applying the concepts of patterns to systems architecture. *Systems Engineering.* Wiley. Vol 10, No. 2. pp 138-154.
10. Duda, Richard. O., Peter E. Hart, David G. Stork. 2001. *Pattern classification*, (2nd ed.), New York: Wiley.
11. Estafan, J. 2008. Survey of model-based systems engineering (MBSE) methodologies. INCOSE MBSE Initiative.
12. Gamma, E., R. Helm, Ralph Johnson, J. Vlissides. 1995. *Design patterns: Elements of reusable object-oriented software*. Reading, MA: Addison-Wesley.
13. Gould, S. J. 2003. *The hedgehog, the fox, and the magister's pox: Mending the gap between science and the humanities*. New York: Three Rivers Press.
14. Grunwald, P. 2007. The minimum description length principle. Cambridge, MA: MIT Press.
15. Gunyon, R., and Schindel, W. 2010. Engineering global pharmaceutical manufacturing systems in the new environment. *Proceedings of the INCOSE 2010 Symposium.*
16. Haskins, Cecilia. 2005. Application of patterns and pattern languages to systems engineering. Paper presented at the 15th annual international symposium of the international council on systems engineering, Rochester, NY.
17. Haskins, Cecilia, ed. 2010. *INCOSE systems engineering handbook, Version 3.2.* Seattle, WA: International Council on Systems Engineering.

References

18. INCOSE HSIG web site. http://www.incose.org/practice/techactivities/wg/hsi/
19. INCOSE MBSE web site: http://www.incose.org/practice/techactivities/modelingtools/mdsdwg.aspx.
20. INCOSE SSWG web site: http://www.incose.org/practice/techactivities/wg/syssciwg/
21. ISO 10303 AP233 web site. http://www.ap233.org/
22. ISO/IEC 15288: 2002. Systems engineering – System life cycle processes. Geneva: International Organization for Standardization.
23. Karayanakis, N., *Computer-assisted simulation of dynamic systems with block diagram languages.* CRC Press, 1993.
24. Kauffman, Stuart. 2000. *Investigations* New York: Oxford University Press.
25. Kuras, M. L., B. E. White. 2005. Engineering enterprises using complex-system engineering. Paper presented at the annual international symposium of the International Council on Systems Engineering, July, Rochester, NY.
26. Li, Ming, Vitany, Paul. 1997. *An introduction to Kolmogorov complexity and its applications*. Second edition. Springer.
27. Mellor, Stephen; Marc J. Balcer. 2002. *Executable UML: A foundation for model-driven architecture.* Boston: Addison-Wesley.
28. Schindel, W. 1996. Systems engineering: An overview of complexity's impact. Tech Paper 962177, SAE International.
29. _____. 1997. The tower of Babel: Language and meaning in system engineering. Technical Report No. 973217 SAE International.
30. ———. 2005a. Requirements statements are transfer functions: An insight from model-based systems engineering. Paper presented at the annual international symposium of the International Council on Systems Engineering, July, Rochester, NY.
31. ———. 2005b. Pattern-based systems engineering: An extension of model-based systems engineering. INCOSE TIES tutorial presented at 2005 INCOSE Symposium.
32. _____. 2006. Feelings and physics: Emotional, psychological, and other soft human requirements, by model-based systems engineering. *Proceedings of the INCOSE 2006 International Symposium.*
33. _____. 2010. Failure analysis: Insights from model-based systems engineering. *Proceedings of the INCOSE 2010 International Symposium.*
34. _____. 2011. Systems engineering for advanced manufacturing: Unit op insights from model-based methods. To appear in *Proceedings of the INCOSE 2011 International Symposium.*
35. Schindel, William D., Vern R. Smith. 2002. Results of applying a families-of-systems approach to systems engineering of product line families. Technical Report 2002-01-3086. SAE International.
36. Shannon, Claude. 1963. *A mathematical theory of communication*. Champaign, IL: University of Illinois Press.
37. Snow, C.P. 1960. *The two cultures.* Cambridge: University Press. pp. 181. ISBN 978-0521457309 (second edition; 1993 reissue).
38. SysML Partners web site. http://www.sysml.org/

Speaker background

Bill Schindel (schindel@ictt.com) is president of ICTT System Sciences (www.ictt.com), a systems engineering company, and developer of the Systematica™ Methodology for model and pattern-based systems engineering. His 40-year engineering career began in mil/aero systems with IBM Federal Systems, Owego, NY, included service as a faculty member of Rose-Hulman Institute of Technology, and founding of three commercial systems-based enterprises.

He has consulted on improvement of engineering processes within automotive, medical/health care, advanced manufacturing, telecommunications, aerospace, and consumer products businesses. Schindel earned the BS and MS in Mathematics, and was awarded an Hon. D.Eng by Rose-Hulman Institute of Technology for his systems engineering work. At the 2005 INCOSE International Symposium, he was recognized as the author of the outstanding paper on Modelling and Tools.