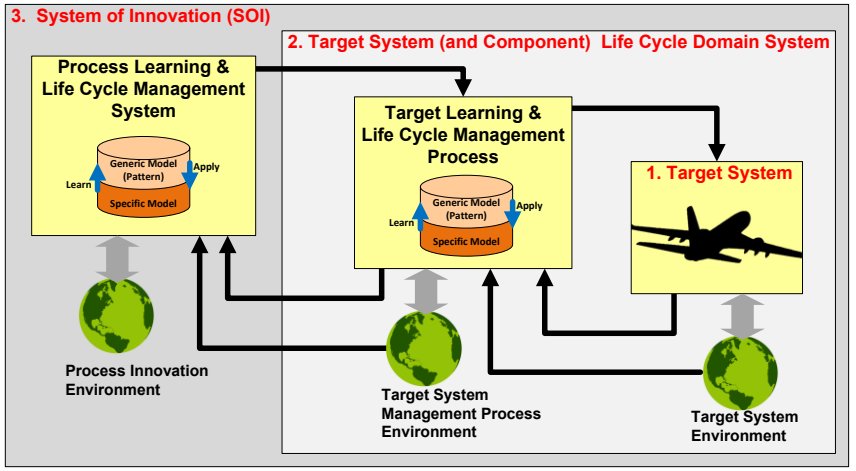
ICTT System Sciences®
Understand your systems.

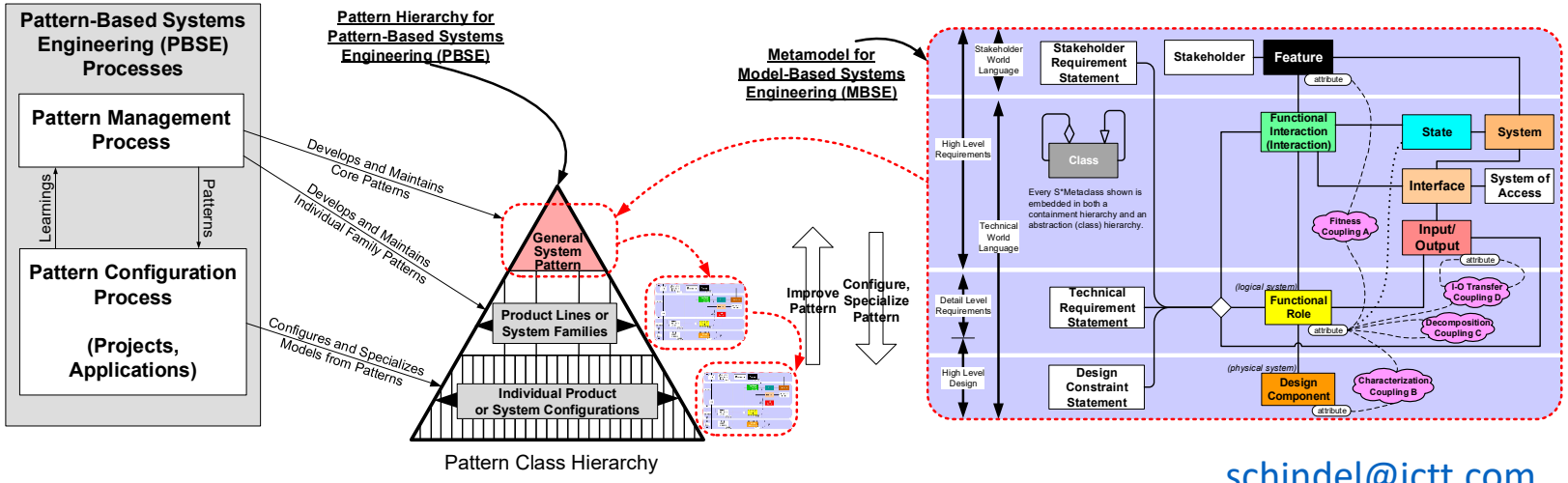
<<Company/Org. Logo>>

Phase I Session 5

Introduction to S*Models, S*Metamodel, S*Patterns, Mappings to Languages & Tools, S*Configuration Wizard, ASELCM S*Pattern



Level 0 View, INCOSE ASELCM Pattern

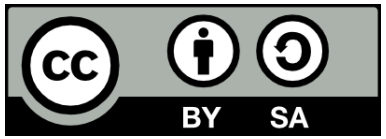


Pattern Class Hierarchy

schindel@ictt.com
lewis@ictt.com

V1.2.12

Copyright 2025, System Sciences, LLC



Licensed under a Creative Commons
Attribution Share Alike-License CC BY SA International 4.0

License Link: https://creativecommons.org/licenses/by-sa/4.0/legalcode

Uses are permitted under this license without further permission from the copyright owner, provided each use (1) is clearly marked to attribute the underlying work to "S*Patterns Community", (2) provides a link to the CC BY SA license, (3) indicates if changes were made, (4) does not suggest the licensor endorses the user or use, (5) does not apply legal terms or technological measures that legally restrict others from doing anything the license permits, and (6) if you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

Permissions beyond the scope of this license are administered through contacting:

Corporate Officer
ICTT System Sciences
378 South Airport Street
Terre Haute, IN 47803
812-232-2208

Systematica is a registered trademark of System Sciences, LLC.

Contents

- Recap of Previous Session:
  - Take-aways, discussion, questions
  - Homework
- Phase I Session 5 Class Content
- References

Recap of Previous Session:
Take-aways impacting actual practices

What consistent practices seem possible to this team, across different systems, customers, and projects?

1. Bridging **different customer's and project's semantics** by mapping to a common underlying semantics of S*MTM.
2. Bridging **different tool sets, enterprise info systems, and modeling languages** by mapping to a common underlying semantic of S*MTM.
3. **Reading and expressing mappings** between semantic frameworks.
4. Installing and navigating a model in an **S*mapped SysML environment** in Cameo Systems Modeler (CSM) or Enterprise Architect (EA).
5. Recognizing the targeted roles of **semantic technologies**.

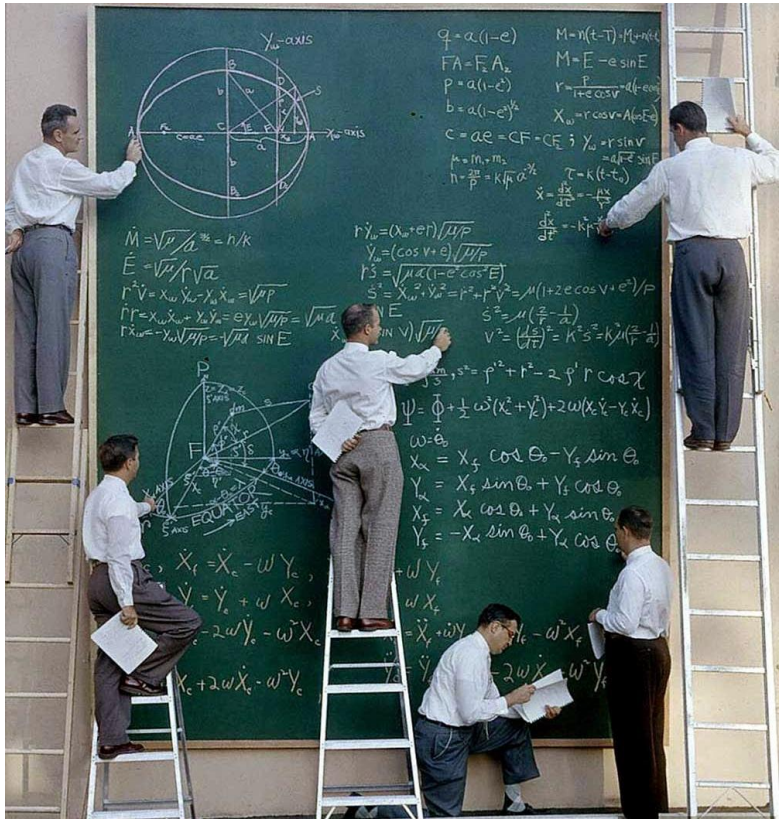


- Can a creative, talented team *align* its work?
- What should be *standard* about this, in your customer work?
- What should be *configurable* about this, in your customer work?



Previous Session Homework Discussion

- In the table of contents of the S*Metamodel Document, what is the purpose of each major section?
- In the table of contents of the “S*Metamodel Mapping for MagicDraw/Cameo Systems Modeler Versions 2019-2024xR2”, what is the purpose of each major section?
- In the table of contents of the “Guide to Systematica Patterns in Cameo Systems Modeler”, what is the purpose of each major section?
- In the S*Metamodel Document, find the metamodel segment diagrams for States (Section 2.7), Risk Analysis (Section 2.24), and Features (Section 2.4). Why does Functional Interaction appear in all three?
- If models are created by two modelers using the same modeling language and tools, why won't they automatically be compatible with each other?



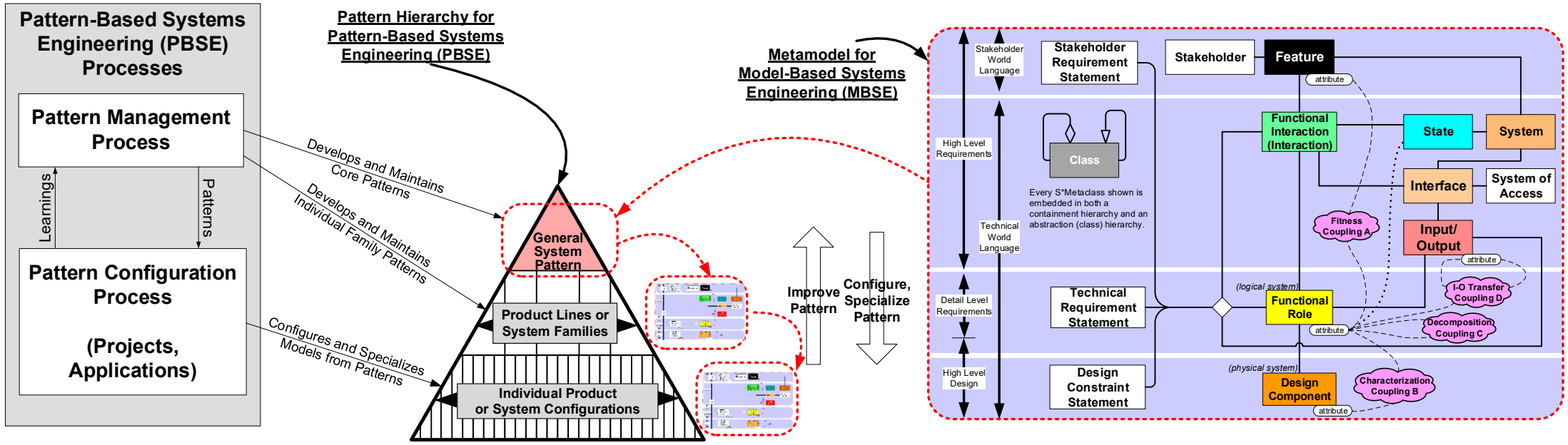
Phase I Class Sessions: Microstructure of Models and Patterns (6 Sessions, 3 hours each)

- S*Models, S*Metamodel:
  - Sessions 1-3
  - In Example Model Context
- Formal mapping of S*Metamodel into Tooling
  - Session 4
- S*Patterns, Configuration: In Example Model Context
  - Session 5 We are here
- Introduction to 3rd Generation Pattern Configurator Technology
  - Session 6
- Homework Presentation and Feedback
  - Distributed over above sessions

Patterns, configurations, frameworks, ontologies, product lines

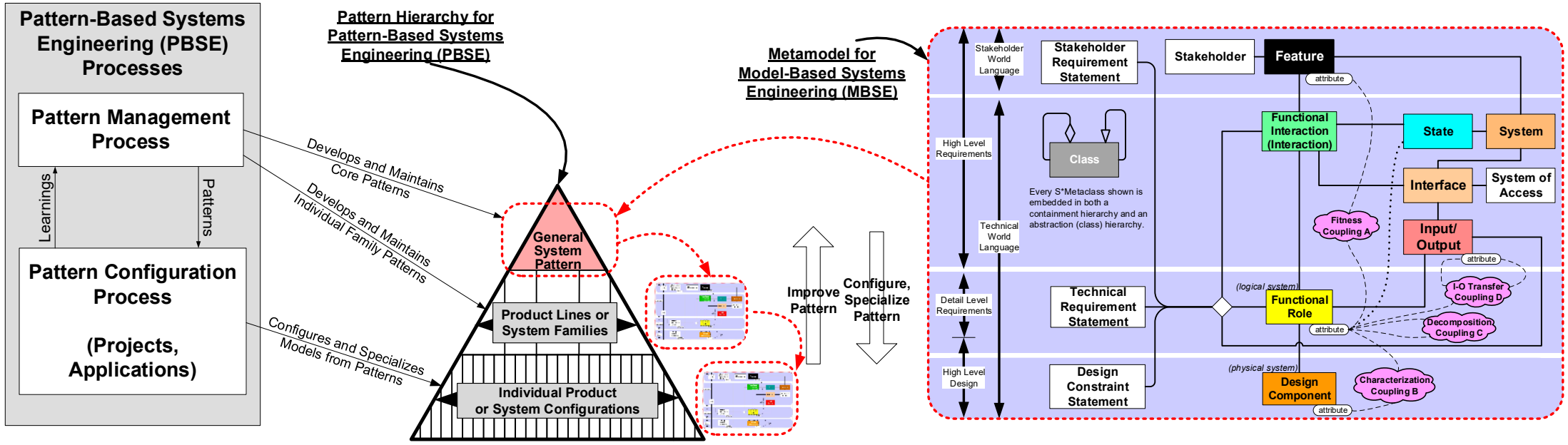
Phase I Session 5
Class Content

- S*Patterns, INCOSE MBSE Patterns Working Group
- Patterns in general
- Patterns background in physical sciences & math
- Patterns background in engineering
- Patterns background in individual & group learning
- Patterns background in machine learning
- Gestalt Rules for patterns
- S*Pattern Configuration, Specialization
- S*Pattern Configuration Algorithm
- S*Pattern Configuration Rules
- Example S*Pattern
- Impact on innovation ecosystem: ASELCM; Phase II classes
- Homework assignment 5
- Discussion, Q&A.



S*Patterns, INCOSE MBSE Patterns Working Group

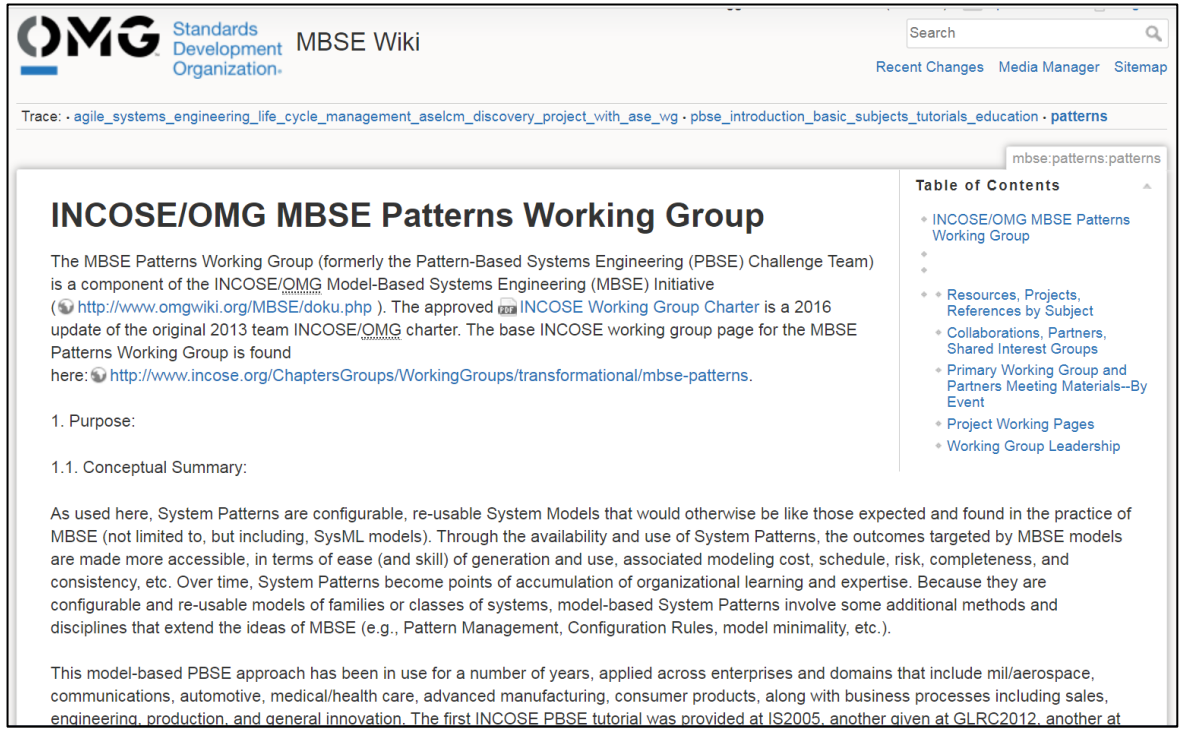
- S*Patterns are S*Models of generalized systems or system families.
- They are intended to be configurable, re-usable, and accumulate learning.
- They are often patterns of “whole systems”, as opposed to components.
- They are *model-based* patterns (a long history of other types of patterns).
- As S*Models, they are based on the S*Metamodel.



Pattern Class Hierarchy

The INCOSE MBSE Patterns Working Group

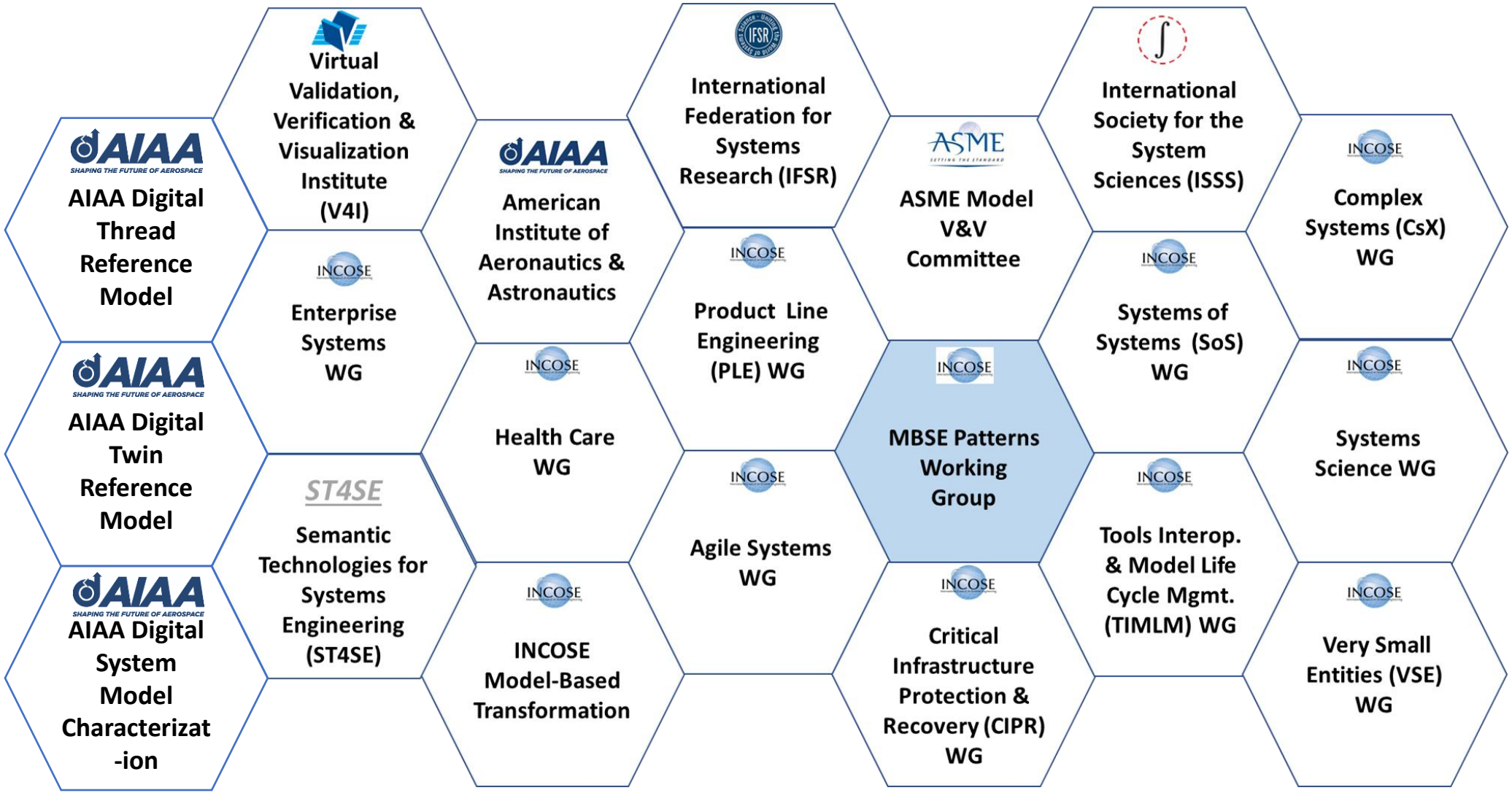
- Originated in 2013 as one of the INCOSE-OMG MBSE Initiative chartered challenge teams, advancing in 2016 to a chartered INCOSE Working Group.
- Focused on model-based representation of recurring, configurable system-level patterns.
- History of projects emphasizing collaboration with other technical societies & INCOSE Working Groups.
- Numerous publications and resources available for download from Patterns Working Group web site-- https://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns (*Note this is on OMG Wiki*)
- You are invited to participate!





https://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns

Nearly all our work includes collaboration with partner INCOSE WGs or other technical societies



Participate! Collaborate!

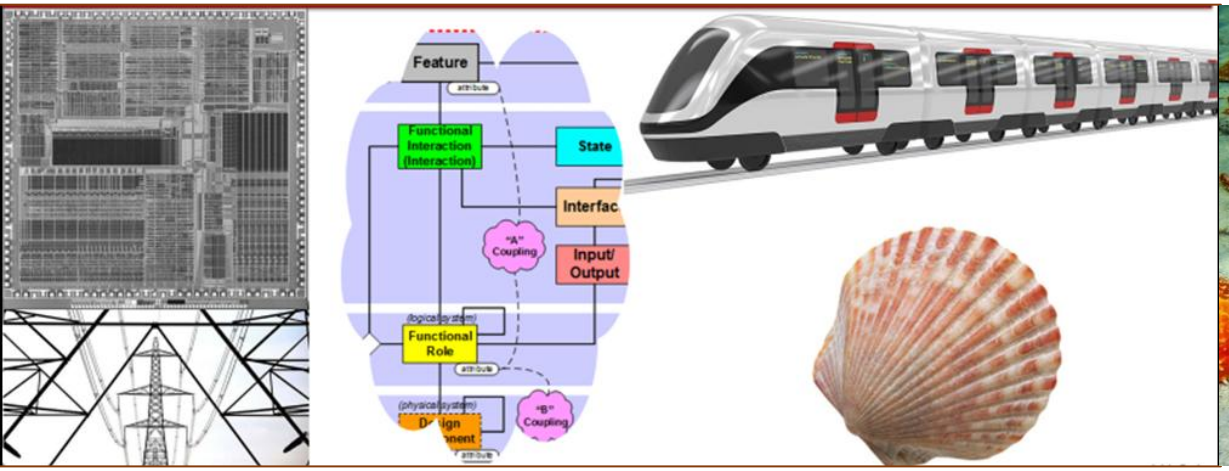
Recent / Current Patterns & Technologies Projects:

1. Rosetta Stone : Consistency Management as Foundation for SE, enhanced by DE, ST, AI
2. Adaptive Learning Ecosystem Pattern—the INCOSE ASELCM Reference Framework
3. Support for DEIX WG DE Ontology – S*Metamodel and S*ASELCM Pattern
4. S*Pattern Configuration Wizard
5. S*Patterns Starter Kit
6. Universal Model Metadata Wrapper: Model Characterization Pattern (MCP), w/AIAA DEIC DSM WG
7. Support for Vision 2035 FuSE Implementation Streams: Innovation Applications, SE Foundations, with SS WG

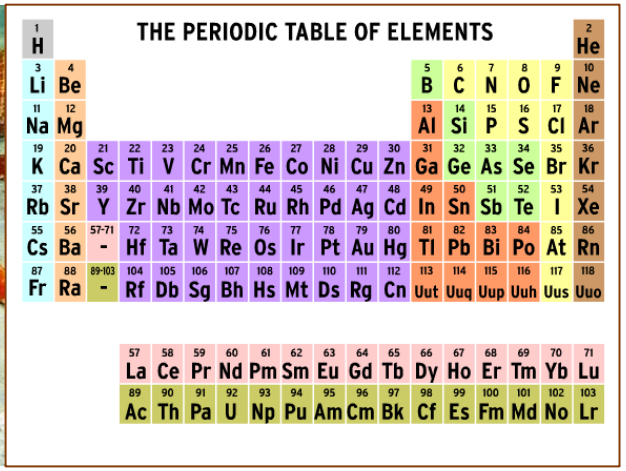
Details

Recent / Current Publications Projects:

1. INCOSE *INSIGHT* Special Editions, Aug and Oct, 2024: S*Foundations--Impacts on SE Practice
2. Minimal S*Models—A Primer (including S*Metamodel and its formal mappings to OMG SysML and tools)
3. S*Patterns Primer (second edition)
4. ASME Guideline for Managing Credibility of Models for Adv. Manufacturing, w/ASME VV50 Stds Working Grp
5. AIAA Aerospace Digital Twins Analysis and Planning Reference Pattern, w/AIAA DEIC
6. AIAA Aerospace Digital Threads Position Pub; Digital Thread Analysis & Planning Reference Pattern, w/AIAA
7. *Handbook of System Sciences*, for ISSS via Springer: Chapter: “Patterns in Science and Engineering”, w/ISSS
8. *Handbook of Model-Based Systems Engineering*, Madni & Augustine, eds, Springer, Chapter: “MBSE Patterns”
9. *INCOSE SE Handbook*, 5th Ed., for INCOSE, D. Walden et al, eds., S*Metamodel and ASELCM Pattern







Patterns in general

- More broadly, all “patterns” of any sort or representation are recurrences:
  - Recurring across time, or
  - Recurring across space, or
  - Recurring across some other index.
- All patterns have “fixed” and “variable” aspects:
  - The fixed aspects are what makes them recognizable as recurrences of something common.
  - The variable aspects are what “parameterizes” their variation.
  - We will often refer to the variable aspects as the “configuration” of an instance of the pattern.



Newton

Patterns background in physical sciences & mathematics

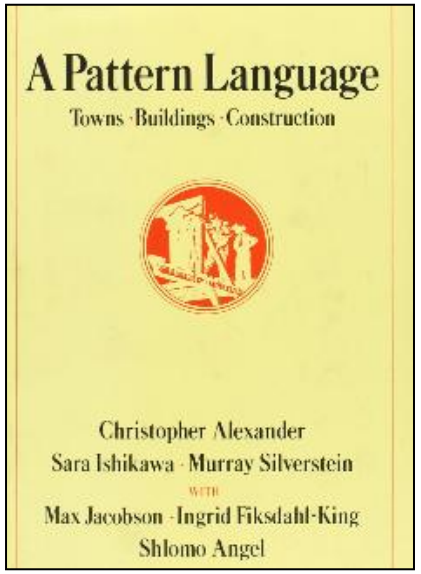


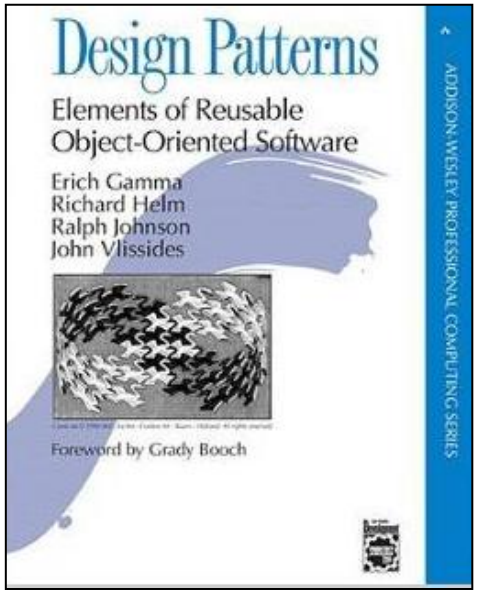
Hamilton

- Patterns are the heart of physical sciences and mathematics.
- *Think much bigger about this than just "recurrences of reusable model parts".*
  - Why?
- Because the intended uses of S*Patterns are more than just re-usable parts.
- The very word "reuse" seems to discourage thinking big enough:
  - Once Newton's Laws of Motion or Maxwell's Equations of Electrodynamics were discovered, their applications have transformed human life on Earth in ~200-300 years.
  - We would not speak of "re-using Newton's Laws" just because they apply repeatedly!
- Spirit of S*Patterns: Think bigger than just MBSE templates!

Patterns background in engineering

- The term “pattern” appears repeatedly in the history of design, such as civil architecture, software design, and systems engineering:

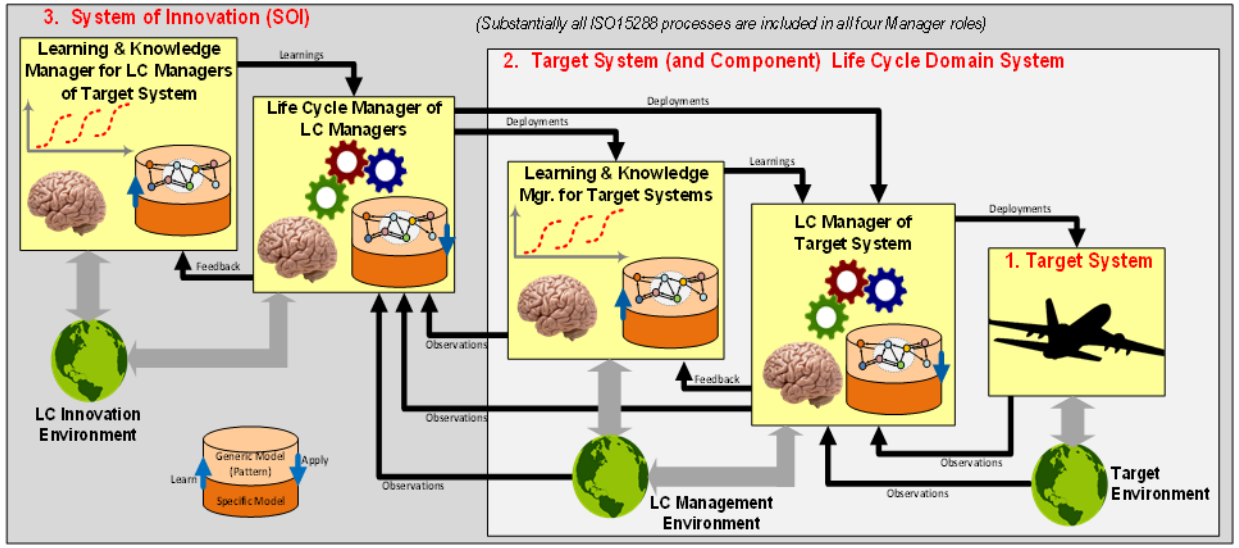






- Those “patterns” represent regularities that repeat, modulo some variable aspects, across different instances in space and time.
- However, when we refer to “PBSE” in this class series, we will usually mean the use of S*Patterns.

Patterns background in individual & group learning



- Innovation Ecosystem Pattern
- (AKA INCOSE ASELCM Pattern)
- ASELCM = Agile Systems Engineering Life Cycle Management

- Patterns are also at the heart of the mathematical and other theories of learning, including individual and group learning.
- There is great historical and contemporary applied mathematics backing this, including Bayesian learning machines that fit right into the world of S*Patterns.
- In Phase II of this series of classes, we will study the Innovation Ecosystem (aka ASELCM) Pattern, which is a generic model of a complex adaptive system, with acquisition (learning) and application of patterns right at its heart.

Patterns as compression

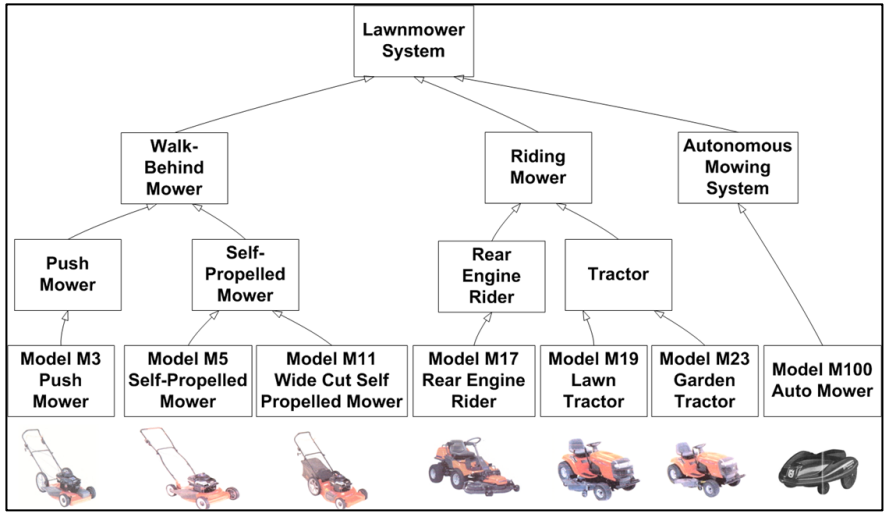
- Pattern extraction is about compression of information:
  - When biological brains or machine learning algorithms “learn” patterns from streams of data, they compress that data.
- Similar to Design of Experiments (DOE) extraction of a few dominant variables to describe some physical phenomenon.
- So, the notion of “smallest model” re-appears here.

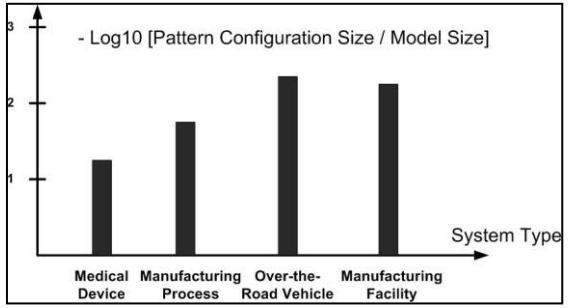
Pattern configurations are compressions

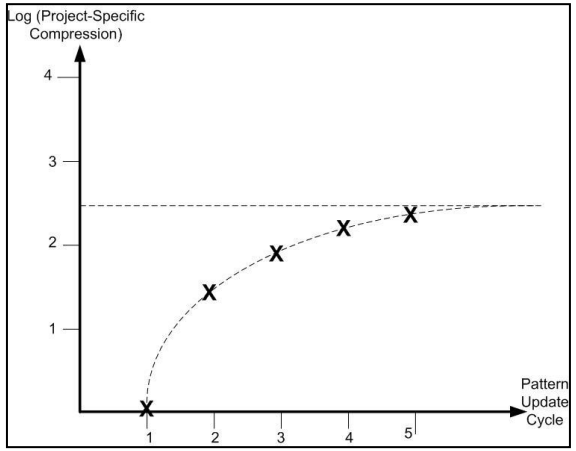
- A table of pattern configurations illustrates how patterns facilitate compression;
- Each column in the table is a compressed system representation with respect to ("modulo") the pattern;
- The compression is typically very large;
- The compression ratio tells us how much of the pattern is variable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







Getting down to PBSE Business: The Gestalt Rules for Patterns

- As systems engineers, we are interested in patterns of “whole systems”, not just their components.
- What are the essential (recurring) ideas of being an automobile? A fixed-wing aircraft? A pharmaceutical manufacturing plant? A market?
- So, we expect that this will involve something more than just using the same parts repeatedly: System Patterns are also about how the parts are organized.

Pattern-Based Systems Engineering (PBSE)

- Standard Parts have been a great aid to progress:

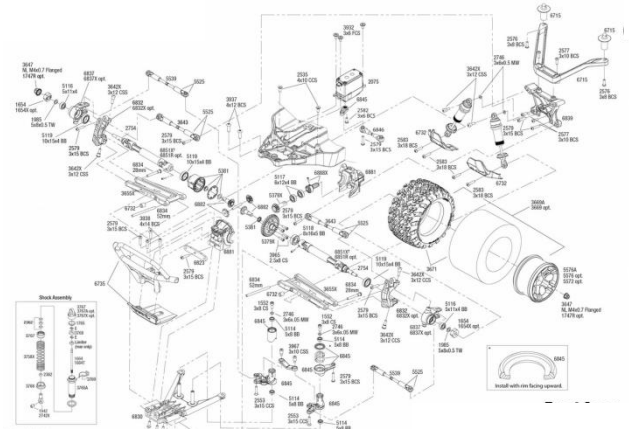


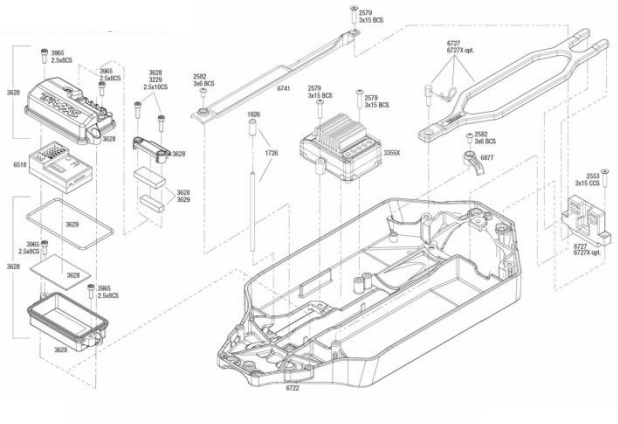


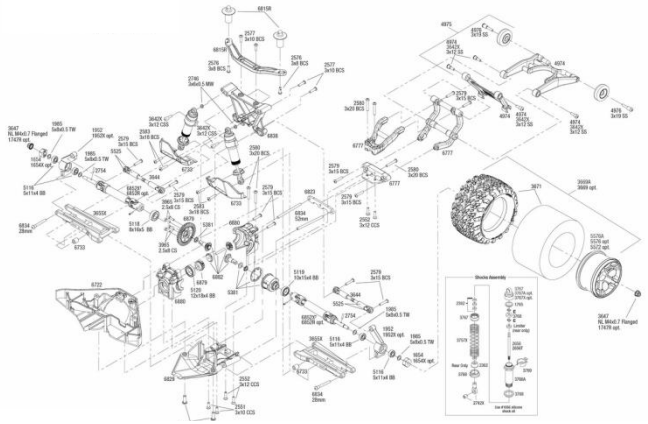


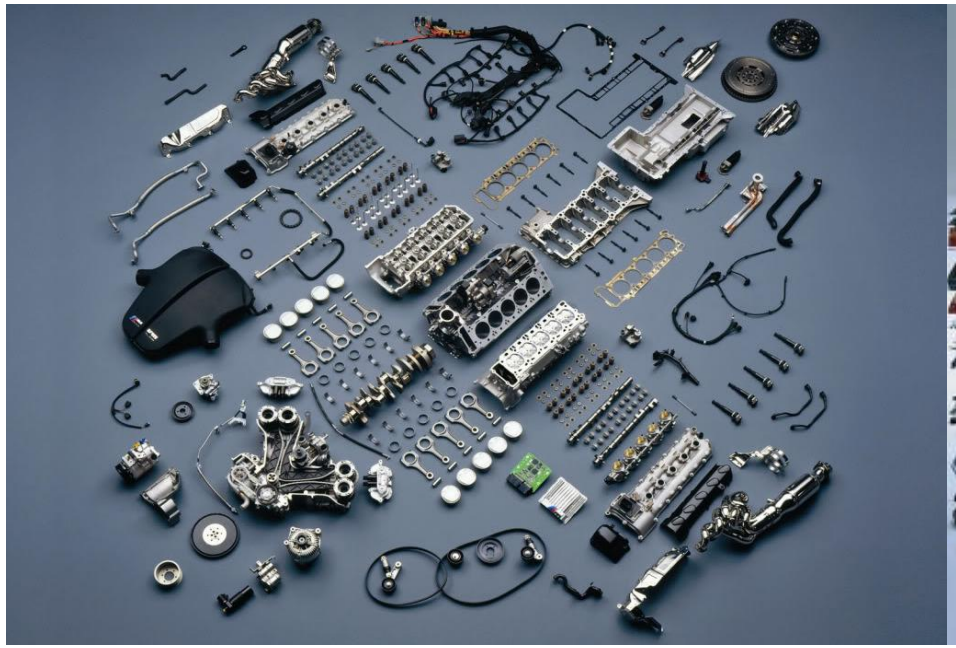
- The same part type can be used to make many things!

Showing parts in relationships helps recognition











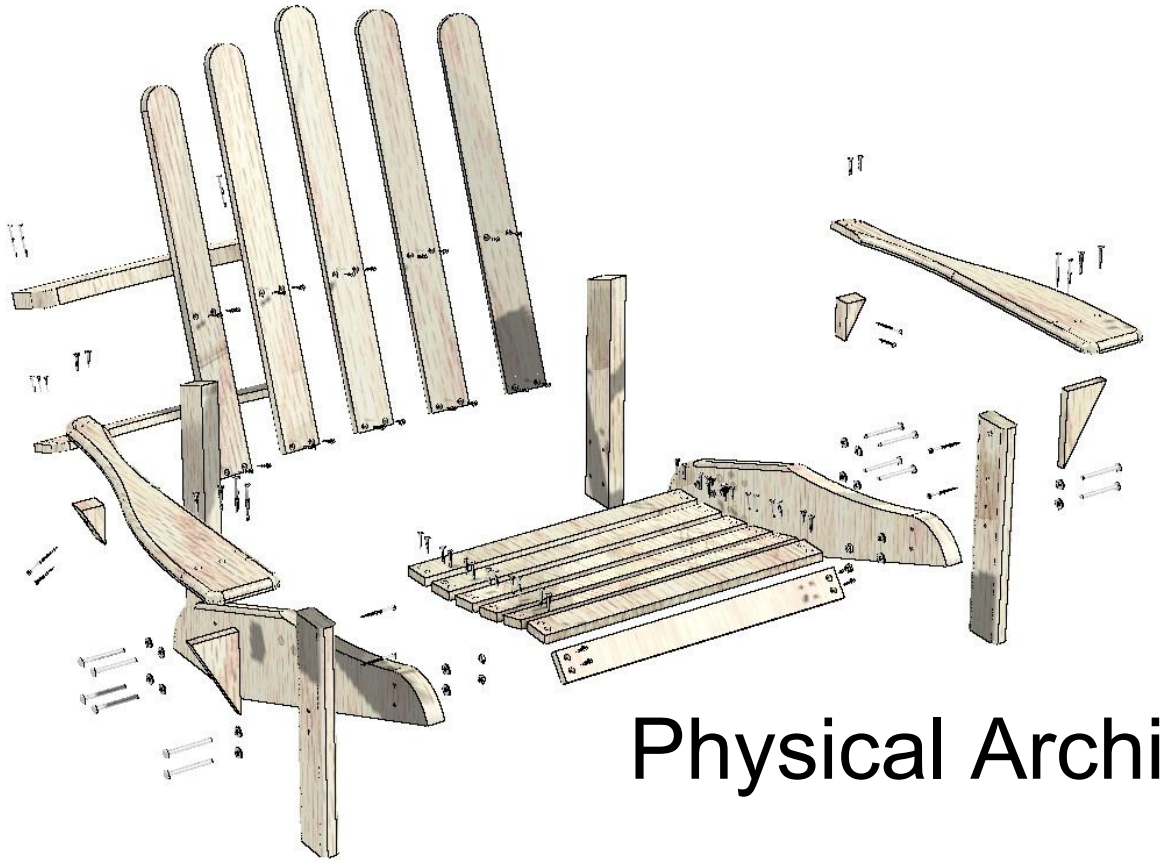
Can we identify a system from its parts alone?



Obviously not in many cases—and in all cases, the parts list alone lacks critical information . . .

Any systems engineer will tell you . . .

- We need to know the relationships between the parts to understand what the “system” they create.



Physical Architecture

But . . .

we are interested in much more than Physical Architecture:

- Stakeholders
- Interactions
- Requirements
- Design
- Interfaces
- Modes
- Performance
- Failure Modes & Effects
- Alternatives
- Configurability
- Manufacturability
- Maintainability
- Operability
- Reliability
- Risks
- Verification Plans

And, in an “information sense”, . . .

we can still think of all these as kinds of “parts”—not just physical parts of a system, but parts of a system model:

- Stakeholders
- Interactions
- Requirements
- Design
- Interfaces
- Modes
- Performance
- Failure Modes & Effects
- Alternatives
- Configurability
- Manufacturability
- Maintainability
- Operability
- Reliability
- Risks
- Verification Plans

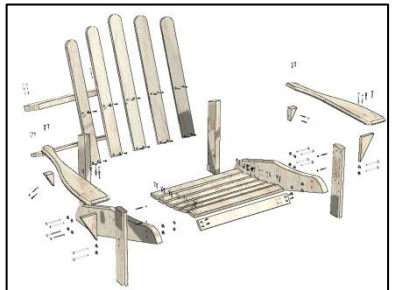
And, once again, it turns out that . . .

the relationships between these information components is just as important as the lists of them, taken alone:

- Stakeholders
- Requirements
- Design
- Interfaces
- Modes
- Performance
- Failure Modes & Effects
- Verification Plans

- Alternatives
- Configurability
- Manufacturability
- Maintainability
- Operability
- Reliability
- Risks
- Interactions

Physical Architecture



Information Architecture

??

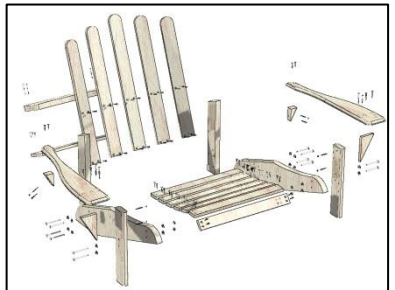
And, once again, it turns out that . . .

the relationships between these information components is just as important as the lists of them, taken alone:

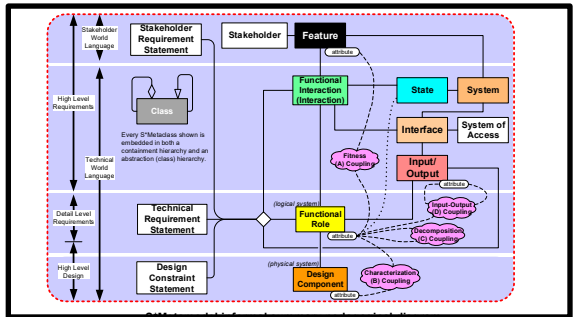
- Stakeholders
- Requirements
- Design
- Interfaces
- Modes
- Performance
- Failure Modes & Effects
- Verification Plans

- Alternatives
- Configurability
- Manufacturability
- Maintainability
- Operability
- Reliability
- Risks
- Interactions

Physical Architecture



Information Architecture

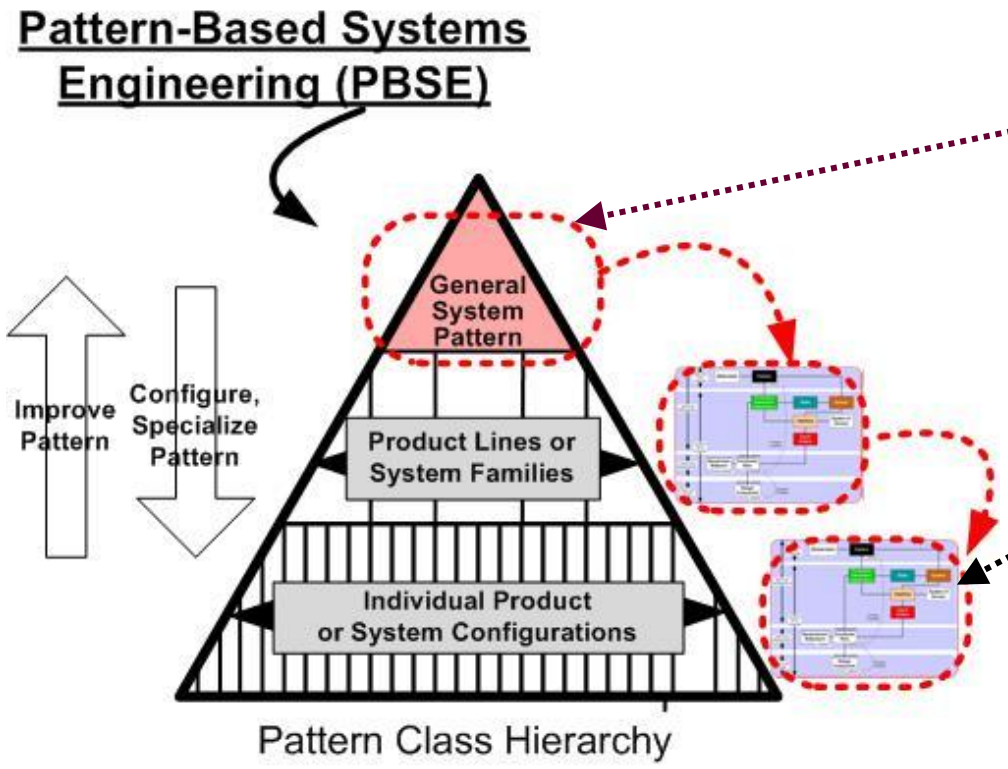


Gestalt Rules for Patterns

- What does it mean to say that “System A is an instance of Pattern B”?
  - How would you know?
- “Gestalts” are about “whole things”.
- The Gestalt Rules tell us what it means for System 1 to be an instance of Pattern B, in a holistic that way that we can test—even by automation . . .

Checking holistic alignment to a pattern

- Gestalt Rules express what is meant by *holistic* conformance to a pattern:
  - Expressing regularities of whole things, versus just "parts"



Governing pattern

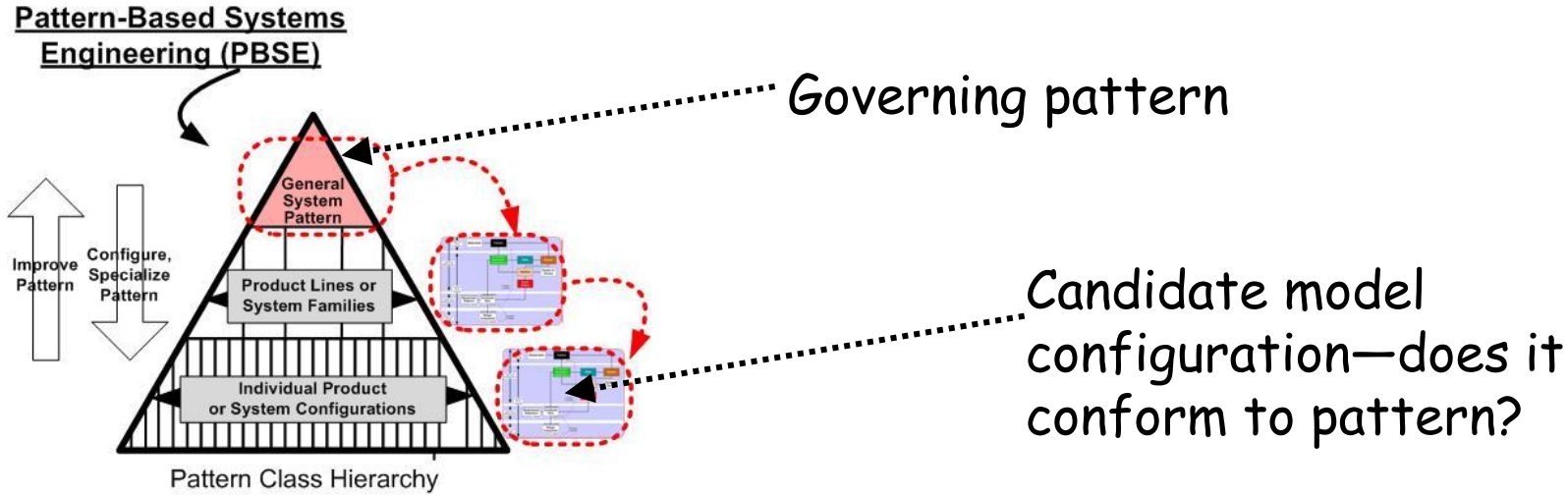
Candidate model configuration—does it conform to pattern?

The Gestalt Rules

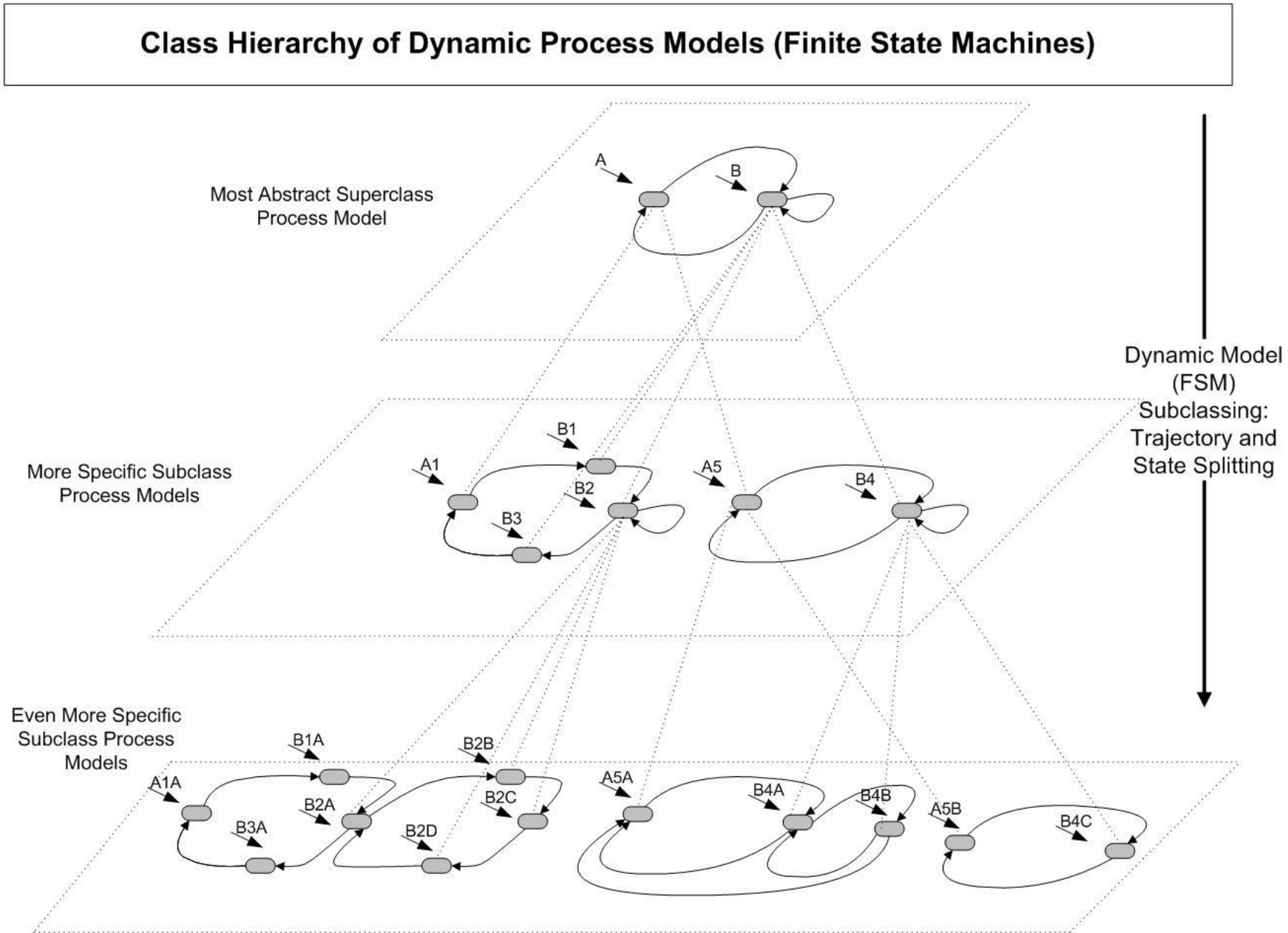
Systematica™

Do more with less

1. Every component class in the candidate model must be a subclass of a parent superclass in the pattern—no “orphan classes”.
2. Every relationship between component classes must be a subclass of a parent relationship in the pattern, and which must relate parent superclasses of those same component classes—no “orphan relationships”.
3. Refining the pattern superclasses and their relationships is a permissible way to achieve conformance to (1) and (2).



Example: State Model Pattern—illustrates how *visual* is the "class splitting" and "relationship rubber banding" of the Gestalt Rules

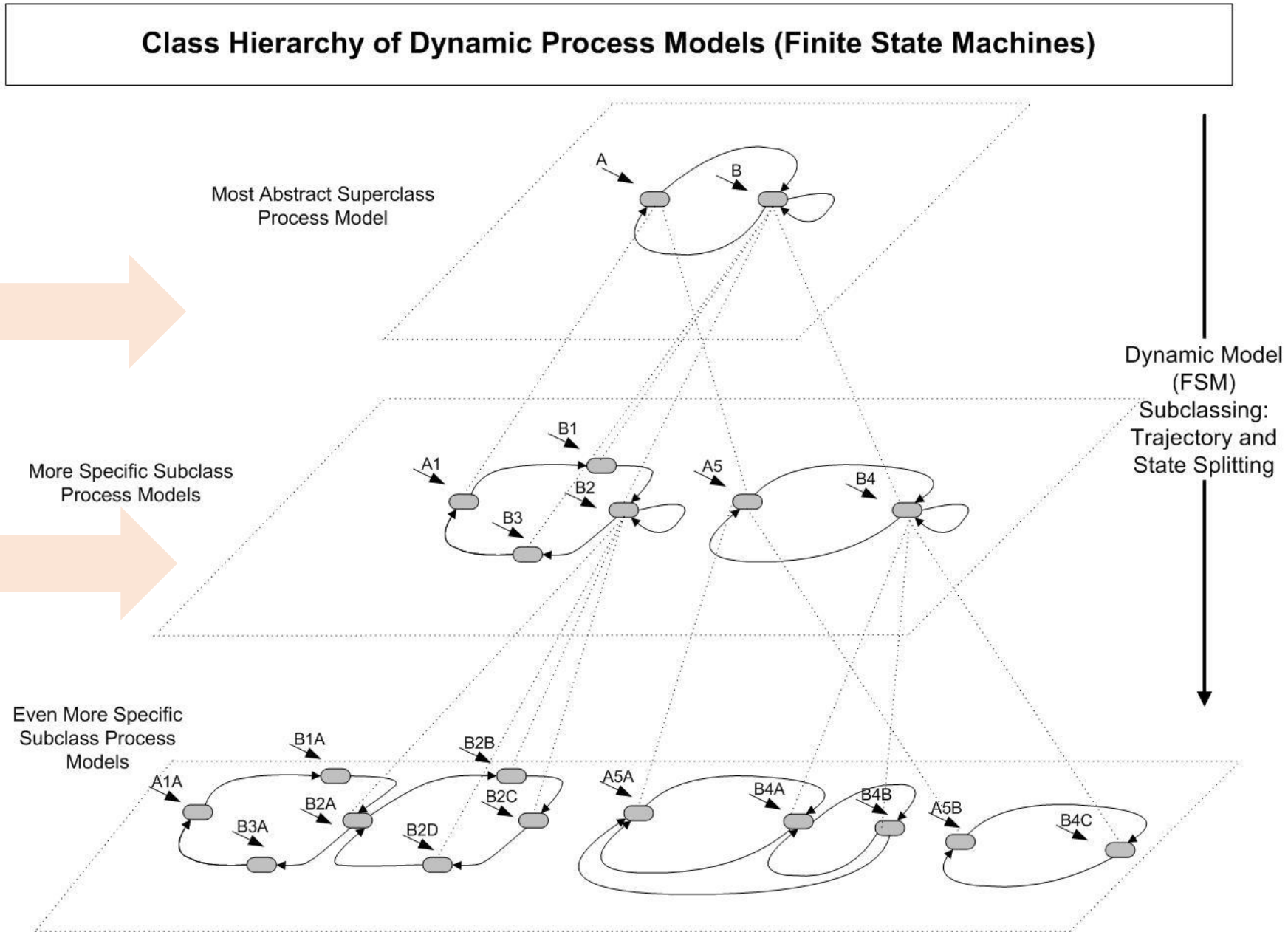


Example: State Model Pattern—illustrates how *visual* is the "class splitting" and "relationship rubber banding" of the Gestalt Rules

Example:

General Aircraft

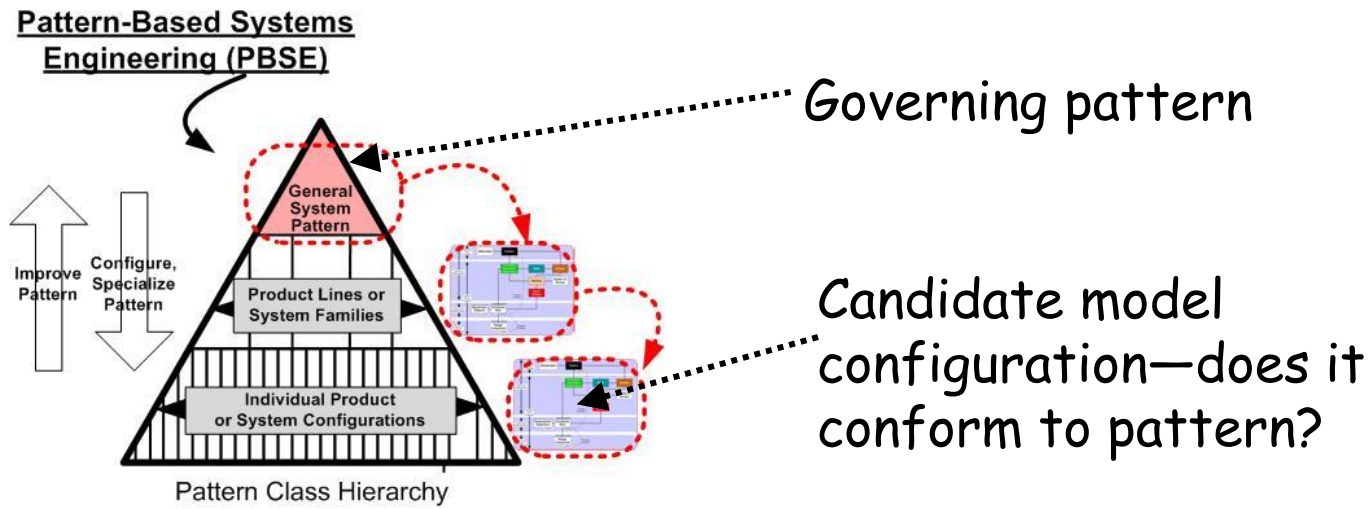
Fixed Wing vs. Vertical Lift A/C

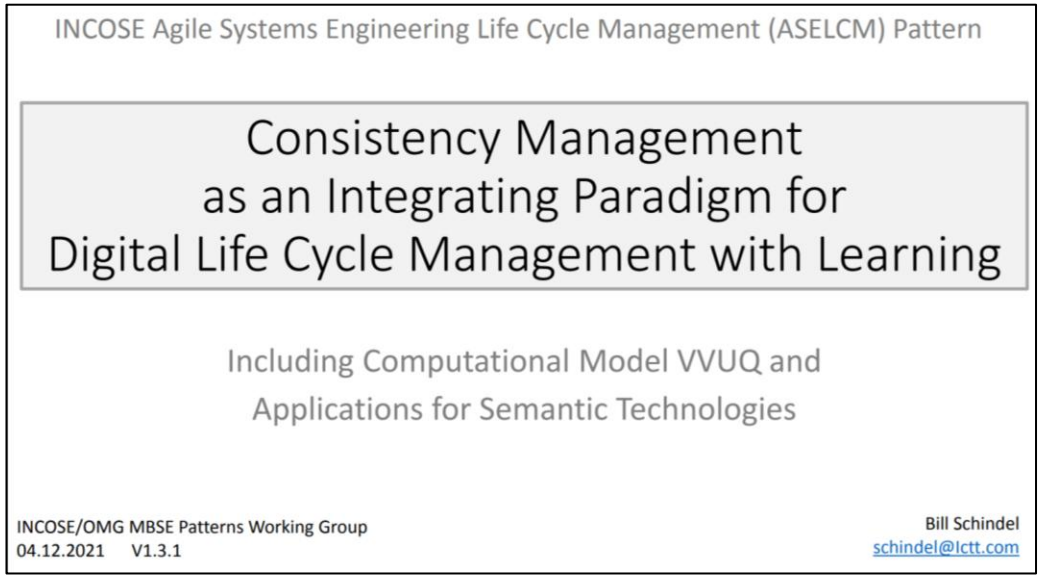


Patterns WG: ST4SE Project

- Combines pattern-based model synthesis (configuration of pattern) with pattern-based model checking

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





ST4SE = Semantic Technologies for Systems Engineering

Patterns WG: ST4SE Project

- Combines (1) pattern-based model synthesis (configuration of pattern) with (2) pattern-based model checking, using semantic technologies.

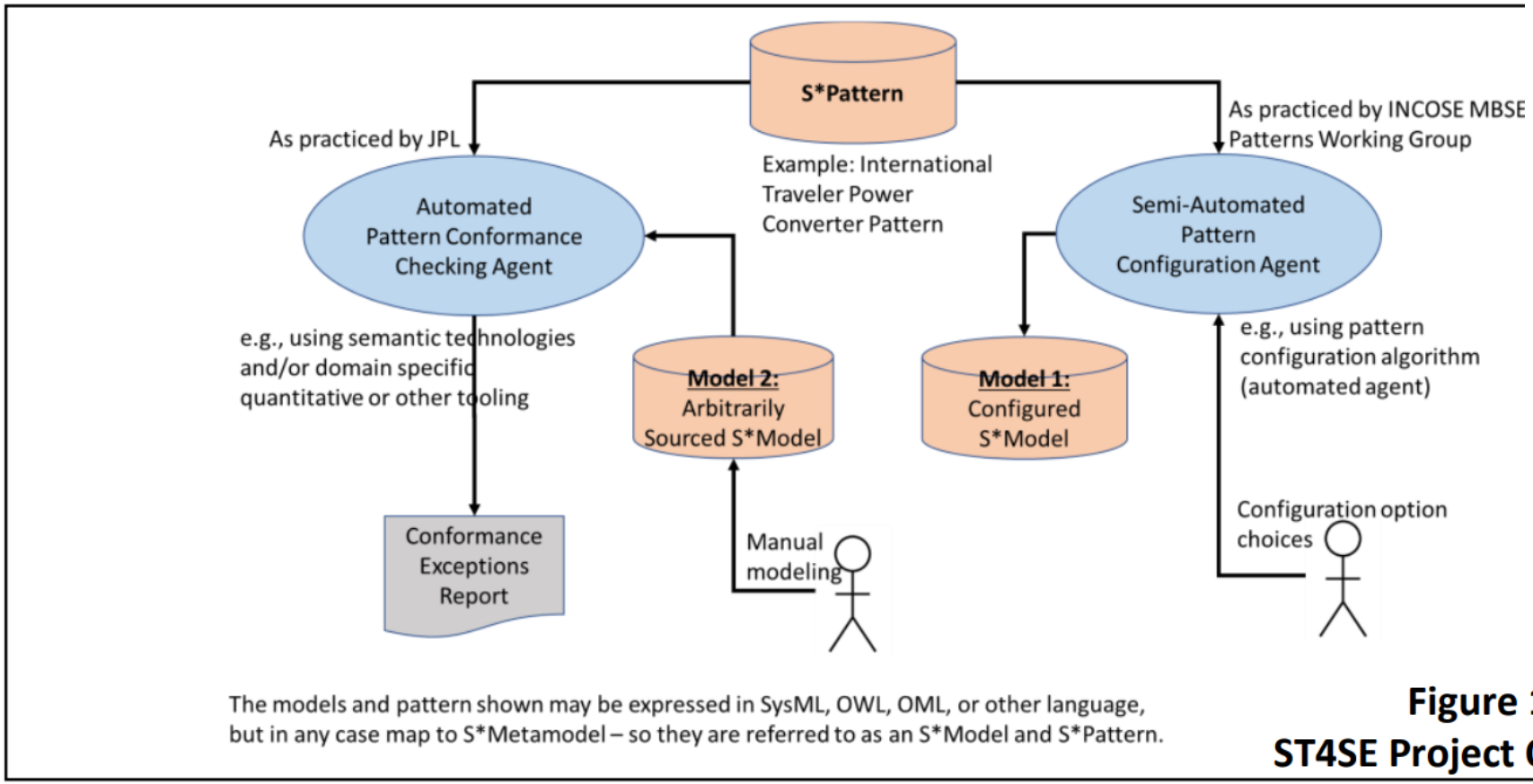
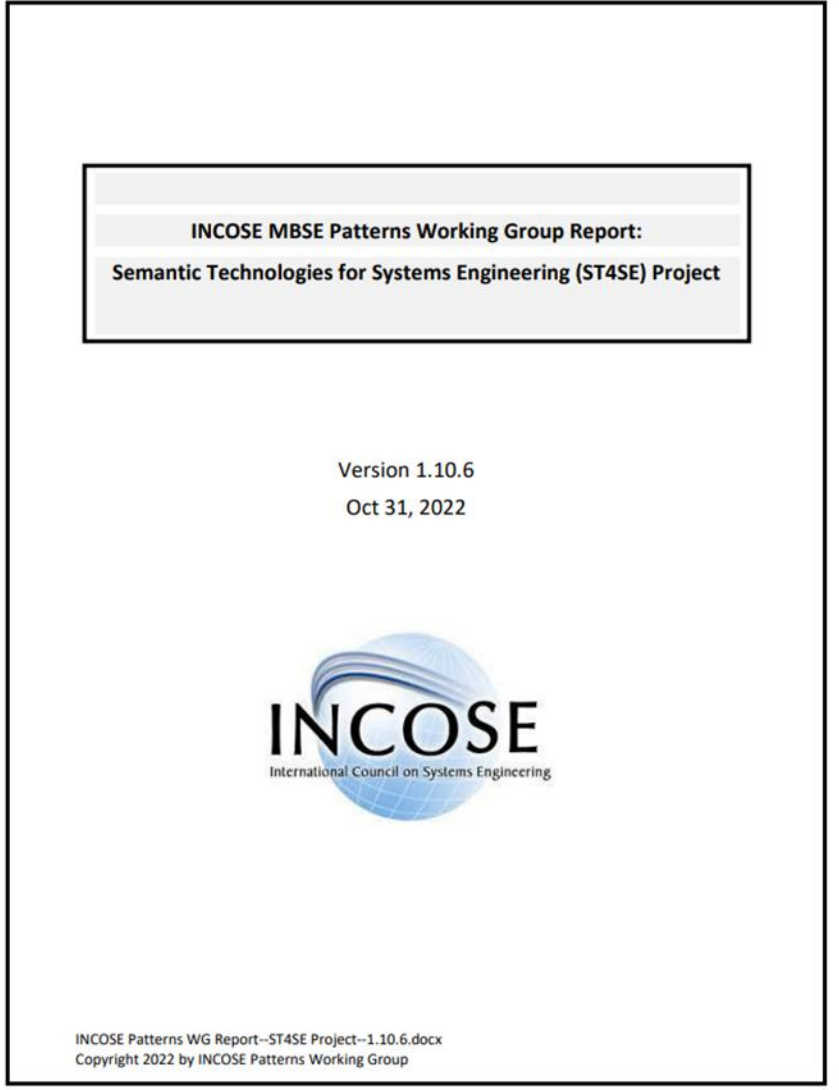


Figure 1: ST4SE Project Overview


INCOSE MBSE Patterns Working Group Report:

Semantic Technologies for Systems Engineering (ST4SE) Project

Version 1.10.6

Oct 31, 2022

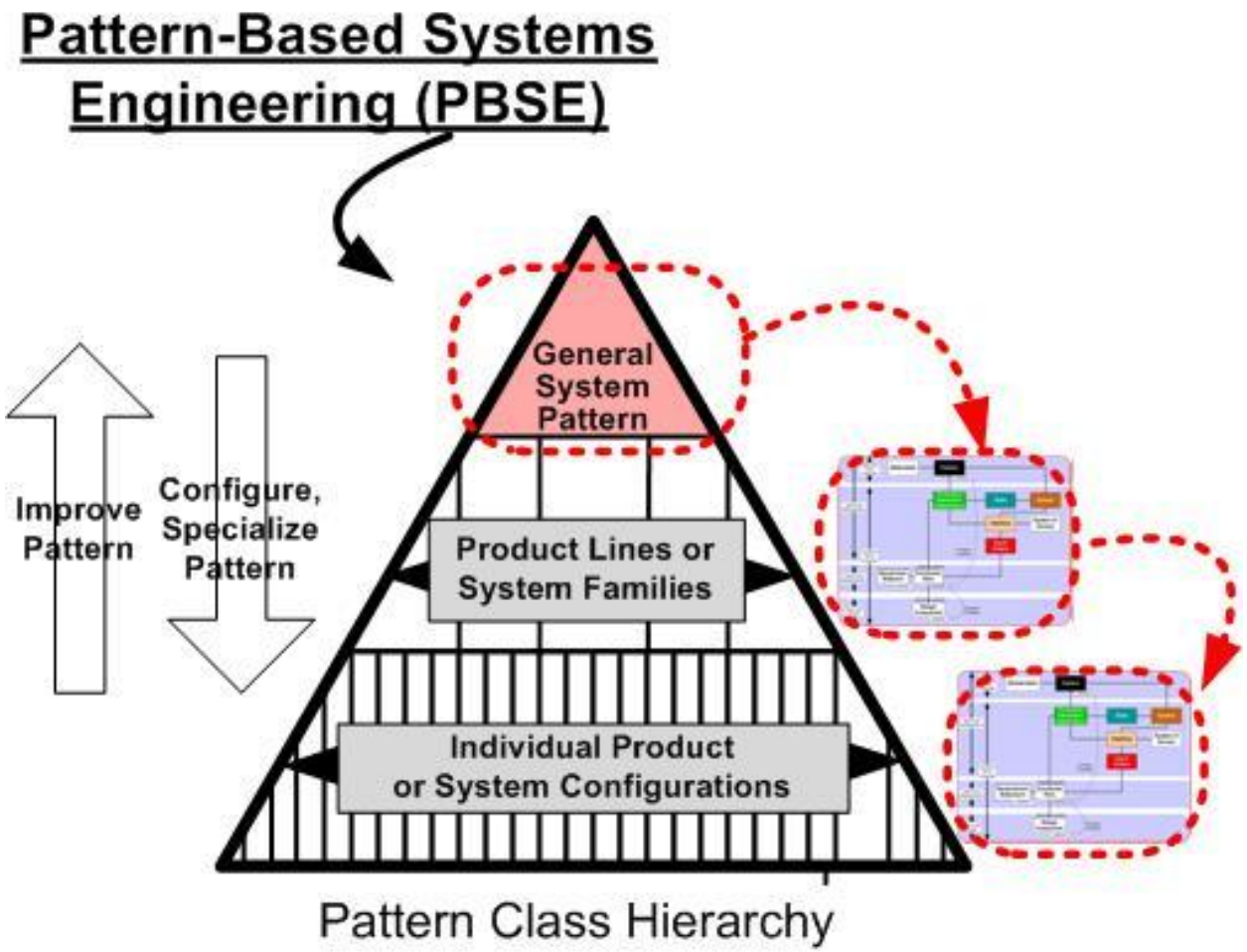
INCOSE International Council on Systems Engineering

INCOSE Patterns WG Report--ST4SE Project--1.10.6.docx

Copyright 2022 by INCOSE Patterns Working Group

https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:incose_patterns_wg_report--st4se_project--1.10.6.pdf

S*Pattern Configuration, Specialization



- Specialization transforms from an upper pattern to a more specialized (lower) pattern.
- Configuration is a "special case of specialization", requiring less modeling skill, less time, and with more "built in" rigor based on "configuration rules":
  - Populate (including multiple) or depopulate classes and relationships.
  - Set Attribute Values.

  That's all!

- This is the "sweet spot" targeted by S*Patterns—rapid, trusted configurability.

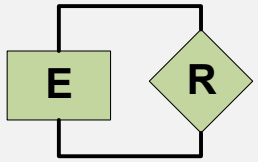
Emergence of Patterns from Patterns: S*Pattern Class Hierarchy

More General

More Specific

Definition of **Relational Modeling Paradigm**

Entity-Relationship Paradigm

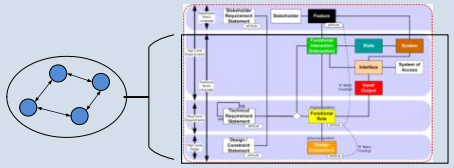


E=Entity
R= Relationship

Structured or unstructured semantic web

Minimal System S*Metamodel: Definition of (Elementary) System, Material Cause

S*Metamodel Core



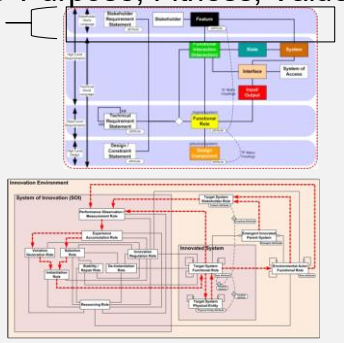
Core S*Metamodel

Smallest model of a system, for engineering or science

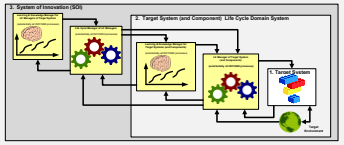
Emergence & Definition of **System of Innovation**, Fitness, Value, Purpose, Stakeholders, Agility, Final Cause, Formal Cause, Efficient Cause, Intelligence, Management, Science, Living Systems

EI Pattern, SOI Pattern, Fitness, Value

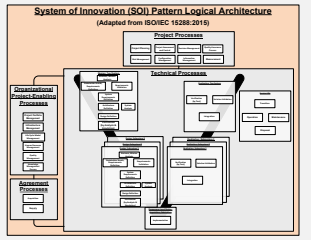
S*Purpose, Fitness, Value



System of Innovation Pattern



Agile Sys Life Cycle Pattern



ISO 15288 System Life Cycle Mgmt Pattern

Emergence & Definition of **Domain Specific Systems**

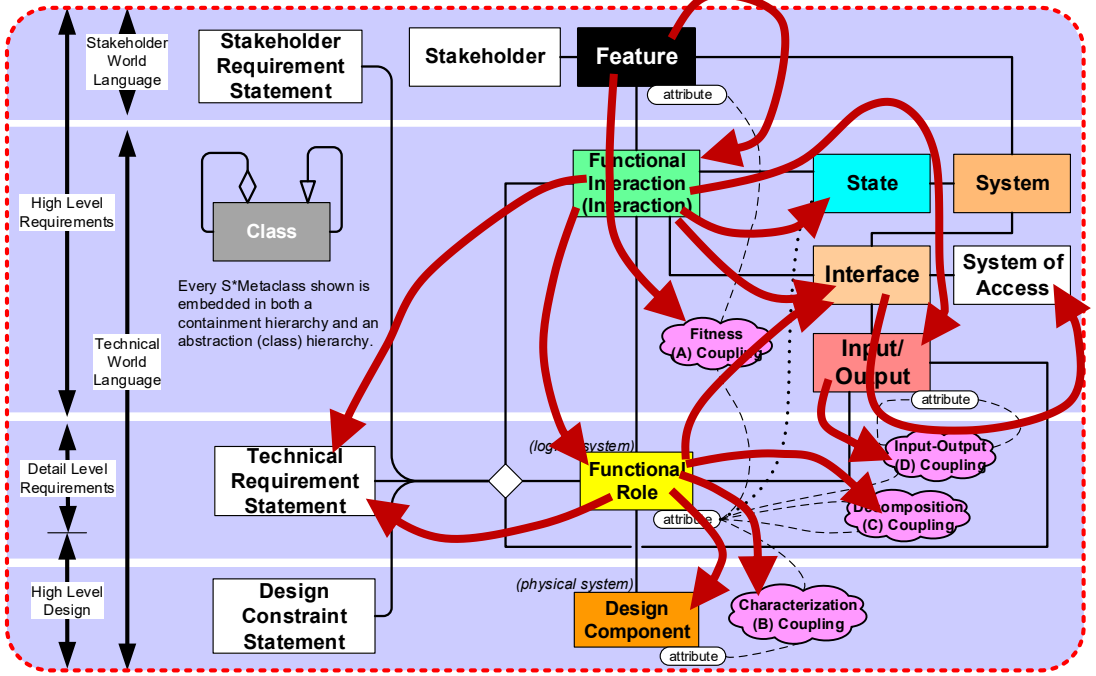
Domain Specific Pattern

- Terrestrial Vehicle Domain Pattern
- Manufacturing System Pattern
- Product Service System Pattern
- Space Tourism Pattern
- Electrical Power Domain Pattern
- Aircraft Flight Control Pattern
- Medical Device Pattern
- Distribution System Pattern
- Socio-Technical Pattern
- Critical Infrastructure Domain Pattern

Propagation of configuration population is inherent to the nature of all engineered systems

- S*Feature Space drives configuration from a smaller set of (stakeholder based) degrees of freedom / points of variation.
- Simplifies Product Line Engineering (PLE) model configuration rule-making and integrates PLE.

Inherent PLE degrees of freedom configuration propagation:



TRIGGERING METACLASSES ("IF") \ POPULATED METACLASSES ("THEN")	Feature	Interaction	Role	Design Component	Requirement Statement	State	Event	Transition	Interface	Architectural Relationship	Input/Output	Port	System of Access	Failure Impact	Counter Requirement Statement	Failure Mode	Feature Attribute	Role Attribute	Design Component Attribute	Input/Output Attribute	Fitness Attribute Coupling	Decomposition Attribute Coupling	Characterization Attribute Coupling	IO Attribute Coupling
Stakeholder Input	■																							
Feature	■	■												■			■							
Interaction		■	■		■	■	■		■	■	■	■	■	■										
Role			■	■	■	■			■	■	■	■	■					■						
Design Component				■												■			■					
Requirement Statement					■										■									
State						■	■																	
Event							■	■																
Transition								■	■															
Interface									■	■														
Architectural Relationship										■	■													
Input/Output											■	■								■				
Port												■	■											
System of Access													■	■										
Failure Impact														■	■									
Counter Requirement Statement														■	■	■								
Failure Mode															■	■								
Feature Attribute																	■				■			
Role Attribute																		■				■	■	
Design Component Attribute																			■					
Input/Output Attribute																				■				■
Fitness Attribute Coupling																					■			
Decomposition Attribute Coupling																						■		
Characterization Attribute Coupling																							■	
IO Attribute Coupling																								■

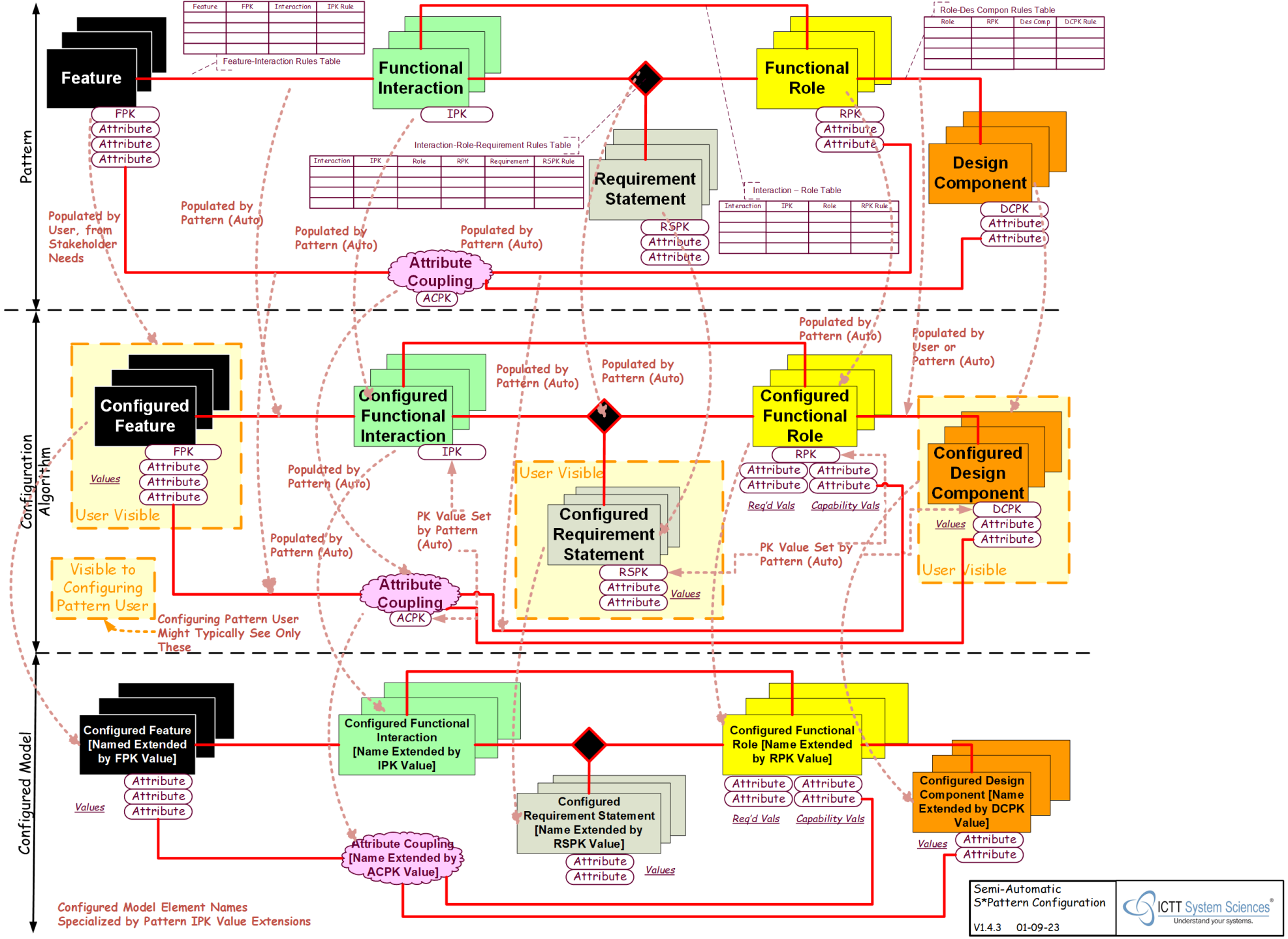
Relationship to Feature-Based PLE ala' ISO 26580

Very similar in the PLE aspects, with a few (specialization) differences:

- ISO26580 PLE specifies modeling what changes, but specifies omitting what does not change; S*Feature models include baseline capabilities.
- ISO26580 refers to all the points of variation as "Features", with rules to be established between them; S*Patterns begins with a smaller set of "Stakeholder Features" degrees of freedom in stakeholder value space, then recognizes all the other points of variation throughout the model but connects them with each other up to the Stakeholder Features points of variation.
- This shows that the number of real degrees of freedom, after considering constraints, is smaller.
- Effectively complies with ISO26580 while making its use simpler and more integrated.

Primary Key (PK) Attributes—Used in Pattern Configuration Rules

- Classes (and some relationships) can have Attributes (parameters, properties).
- Attributes are variables that can have values (numerical, enumerated lists, strings, discrete, continuous, etc.)
- To support PBSE, some attributes are special: They are Primary Key Attributes (PK Attributes):
  - A PK Attribute has the special nature that its value differentiates class instances uniquely.
  - Other attributes are "ordinary" in that they are not PK Attributes.
- **Example**:
  - US Citizens have Social Security Numbers, First Names, and Family Names.
  - All three are Attributes, but only the Social Security Number is a PK Attributes.
- In S*Patterns, PK Attributes' values are used to "configure" specializations of classes.
- **Example**: The Wireless Communication Feature has a Primary Key Attribute of Wireless Media Type. It also has other "ordinary" attributes such as Bandwidth and Security Level.
- Classes that have Primary Key Attributes are often "multiply instantiated" at the time of configuration (specialization).
- S*Metaclasses using those ideas: Feature, Interaction, Role, Design Component



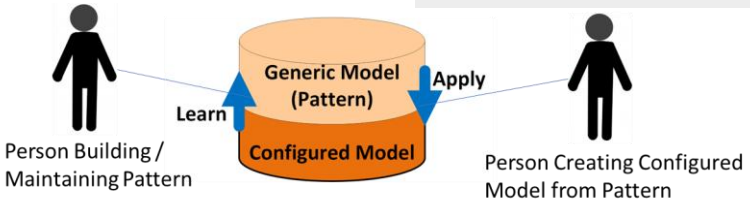
Pattern Configuration (Application) Process

Semi-Automatic
S*Pattern Configuration
V1.4.3 01-09-23

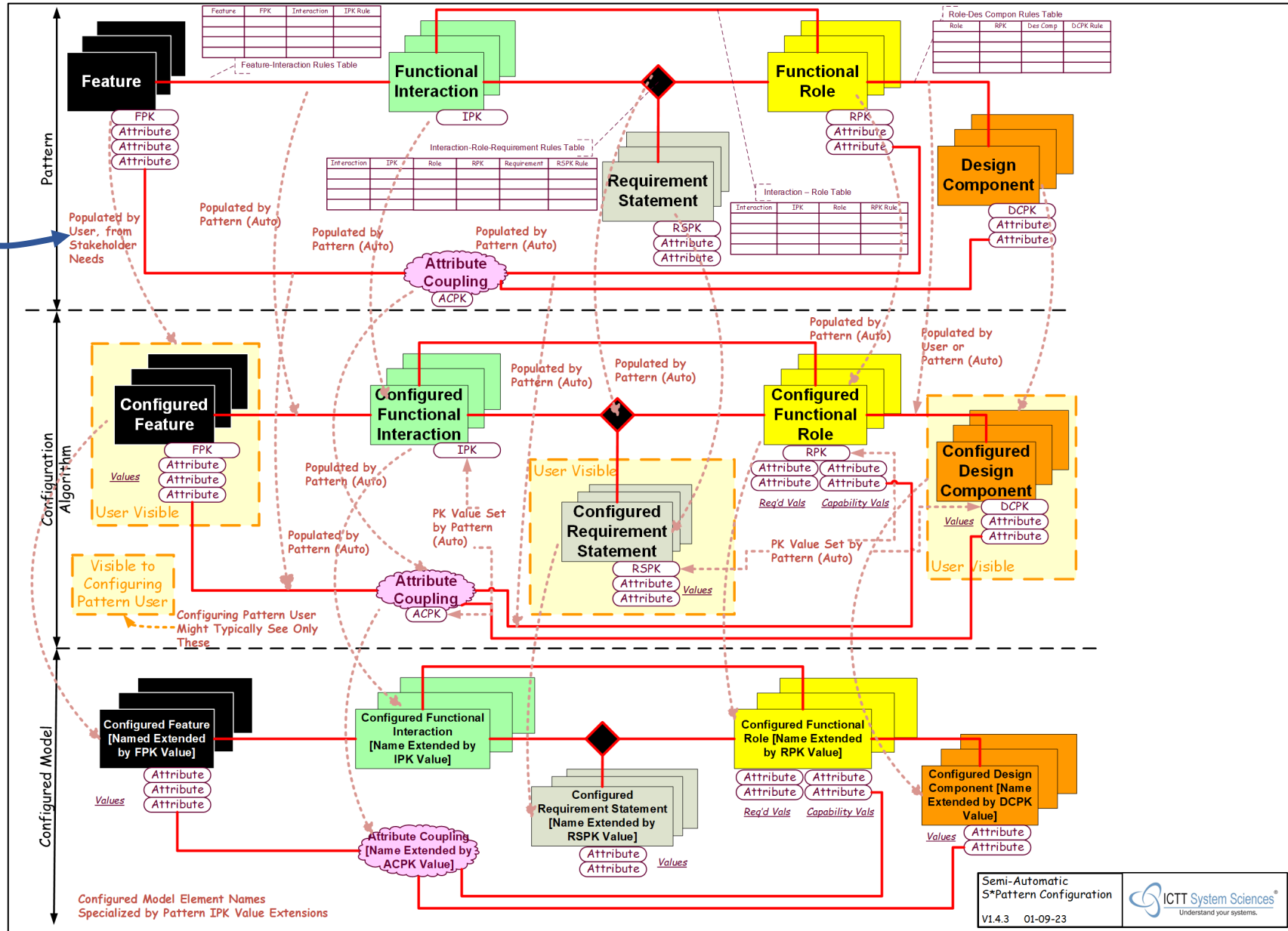
ICTT System Sciences®
Understand your systems.

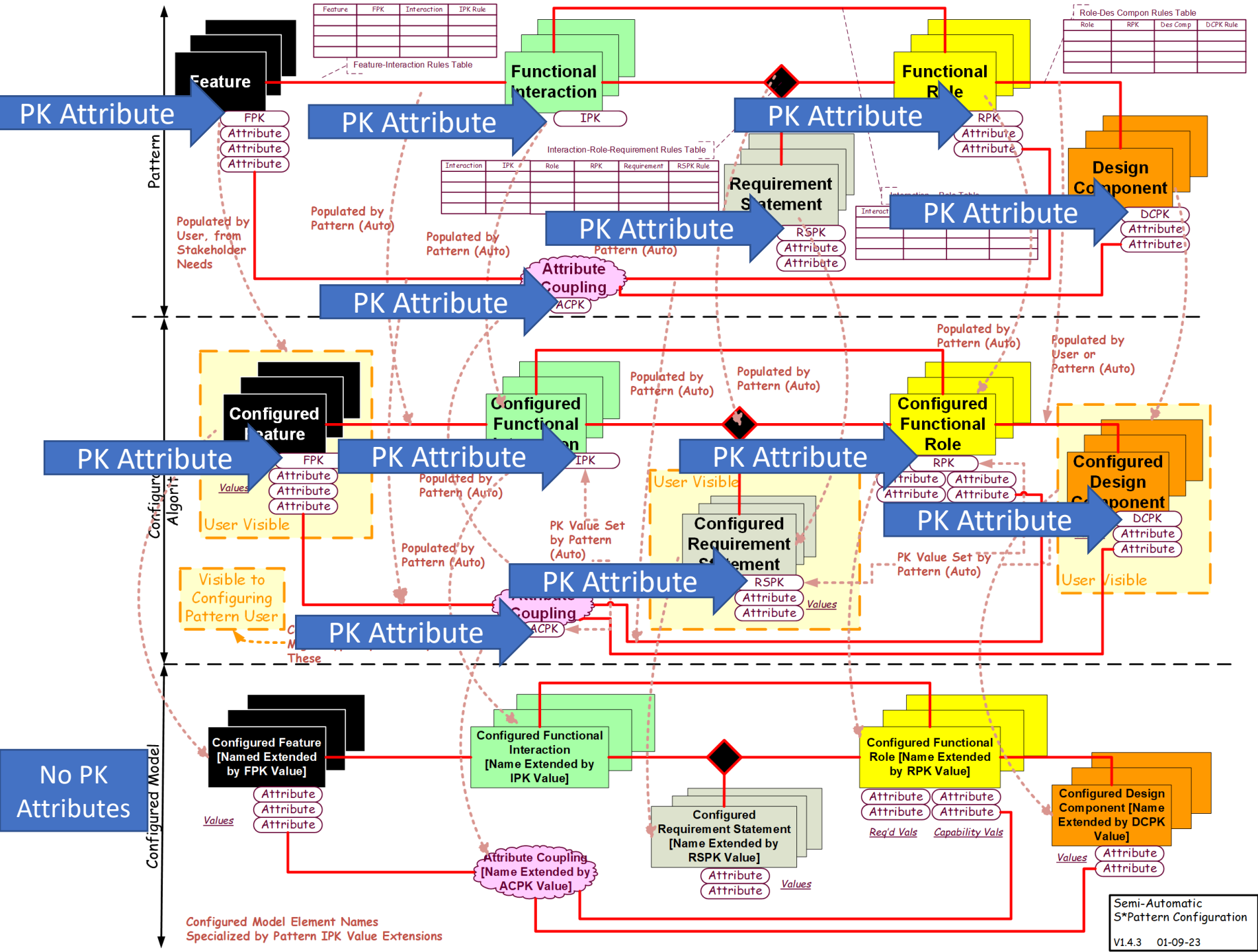
This Session's Materials

Next Session's Materials, Using Configuration Wizard



Pattern Configuration (Application) Process





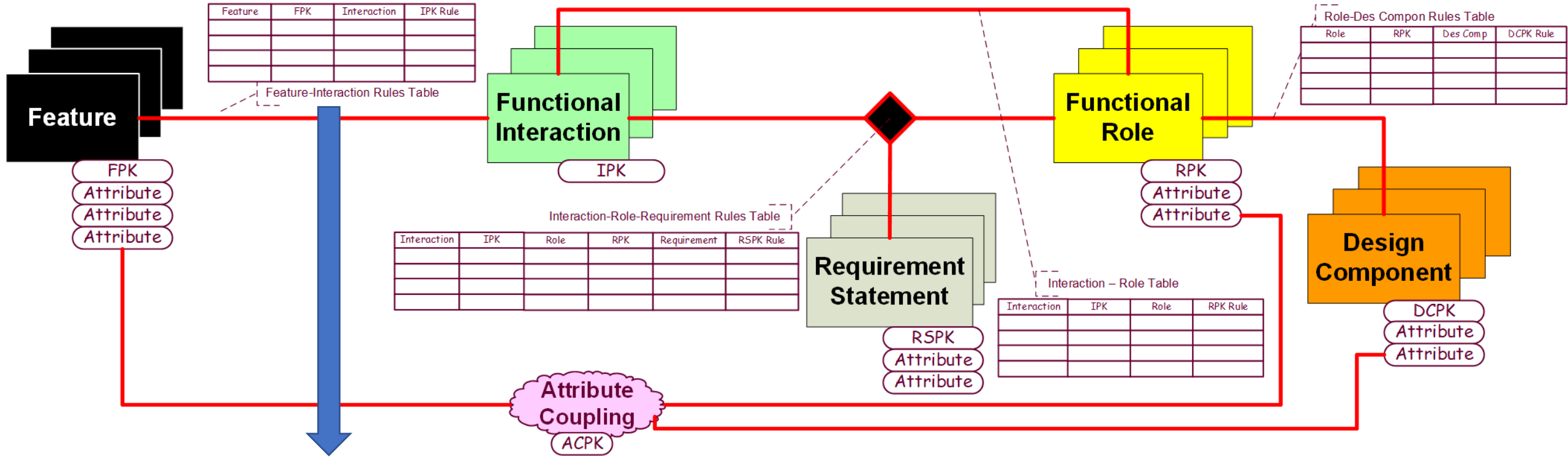
1. Pattern: PK Attributes do not have configured values, just possible values.

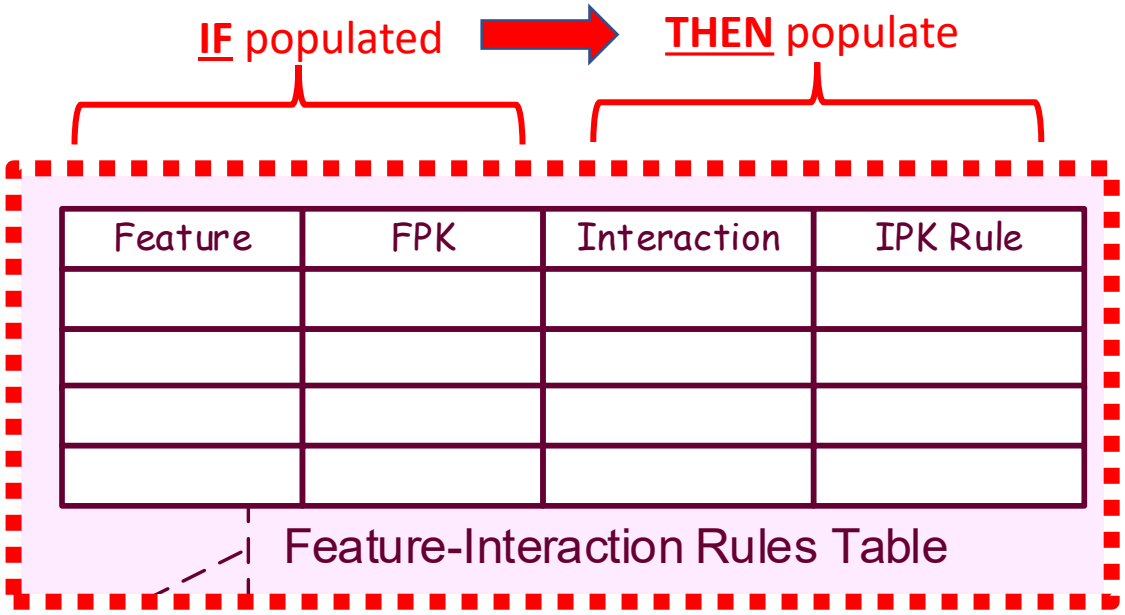
2. During Pattern Configuration Process: PK Attributes now have configured values picked by configuring user or configuration rules.

3. Configured Model, After Pattern Configuration Process: Chosen PK Attribute Values have become part of (extended) differentiated class names.

Semi-Automatic S*Pattern Configuration
V1.4.3 01-09-23

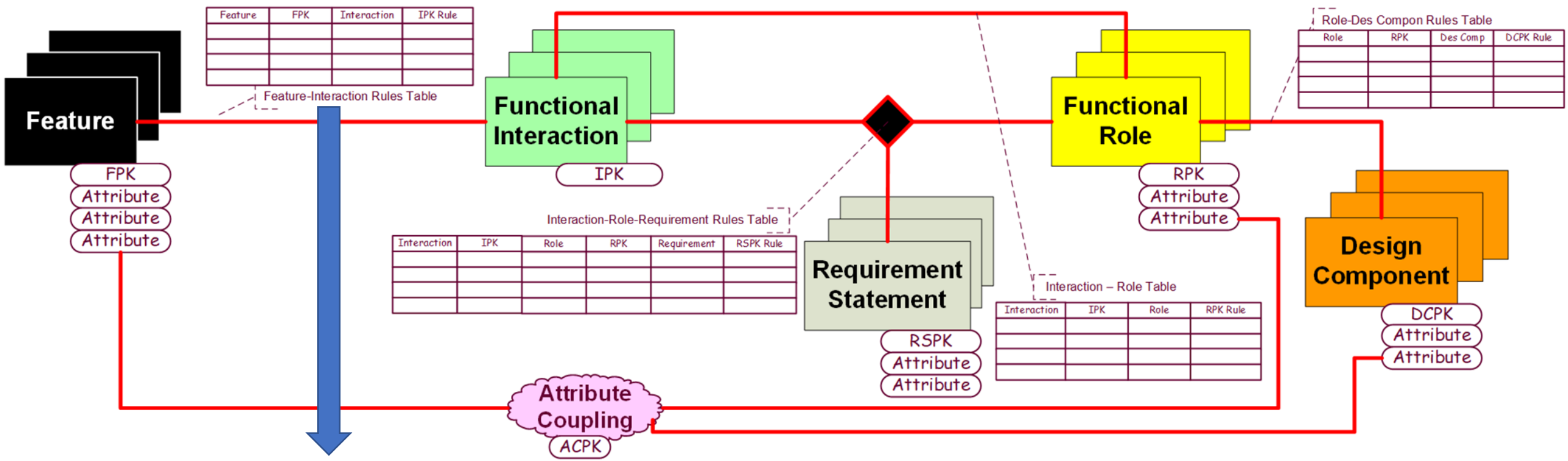
ICTT System Sciences
Understand your systems.

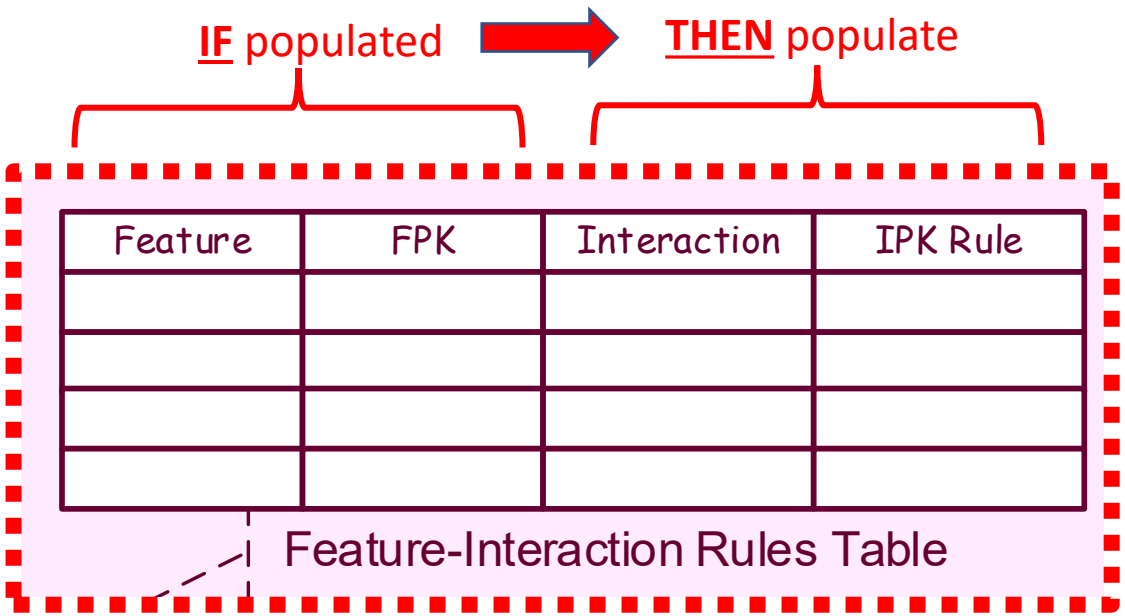


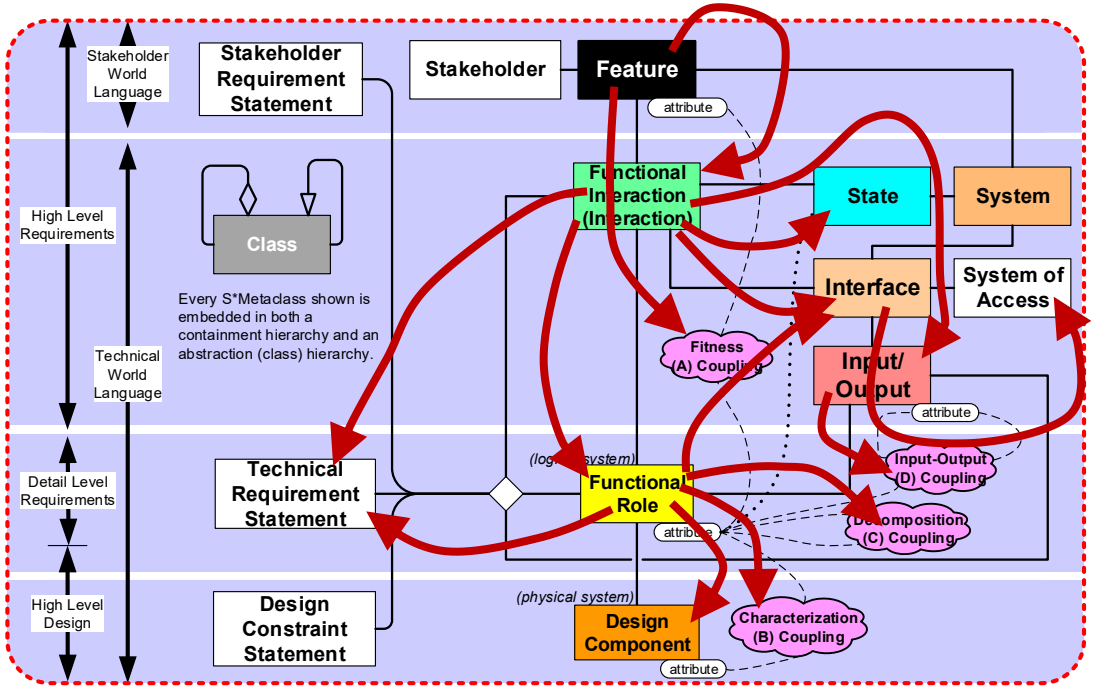


Configuration Rules

- govern whether and how pattern components (both classes and the relationships between them) become populated—using Primary Keys.
- have an “IF” part and a “THEN” part.
- can be considered to be information embedded within the relationships shown in the Pattern.


Feature
FPK
Attribute
Attribute
Attribute
Feature
FPK
Interaction
IPK Rule
Feature-Interaction Rules Table
Functional Interaction
IPK
Interaction-Role-Requirement Rules Table
Interaction
IPK
Role
RPK
Requirement
RSPK Rule
Requirement Statement
RSPK
Attribute
Attribute
Functional Role
RPK
Attribute
Attribute
Role-Des Compon Rules Table
Role
RPK
Des Comp
DCPK Rule
Interaction – Role Table
Interaction
IPK
Role
RPK Rule
Design Component
DCPK
Attribute
Attribute
Attribute Coupling
ACPK


IF populated
THEN populate
Feature
FPK
Interaction
IPK Rule
Feature-Interaction Rules Table


Stakeholder World Language
Stakeholder Requirement Statement
Stakeholder
Feature
attribute
High Level Requirements
Class
Every S*Metaclass shown is embedded in both a containment hierarchy and an abstraction (class) hierarchy.
Technical World Language
Functional Interaction (Interaction)
State
System
Interface
System of Access
Fitness (A) Coupling
Input/ Output
attribute
Input-Output (D) Coupling
Detail Level Requirements
Technical Requirement Statement
(logical system)
Functional Role
attribute
Decomposition (C) Coupling
High Level Design
Design Constraint Statement
(physical system)
Design Component
Characterization (B) Coupling
attribute

Feature	FPK	Interaction	IPK Rule

Feature-Interaction Rules Table

Role-Des Compon Rules Table

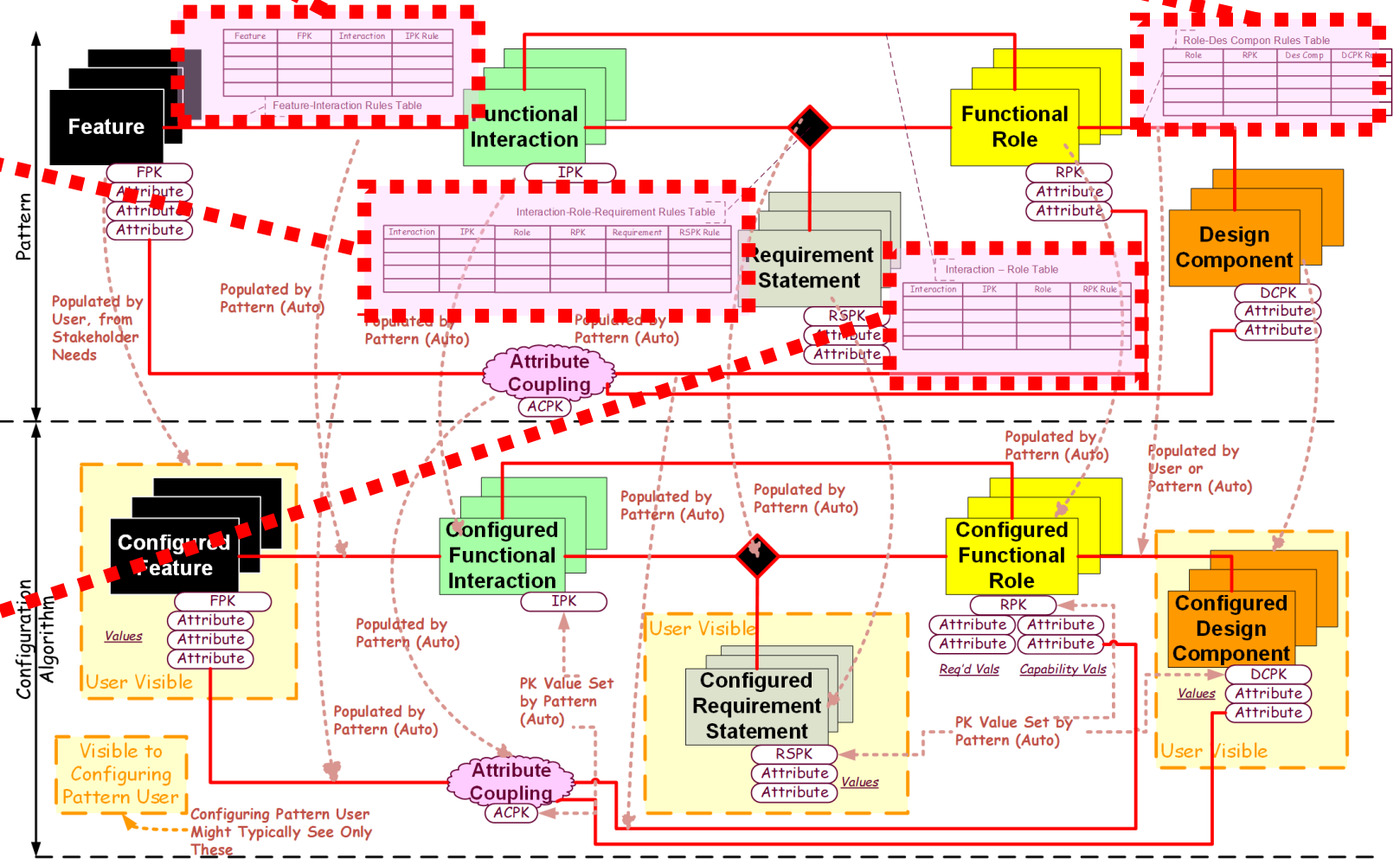
Role	RPK	Des Comp	DCPK Rule

Interaction-Role-Requirement Rules Table

Interaction	IPK	Role	RPK	Requirement	RSPK Rule

Interaction – Role Table

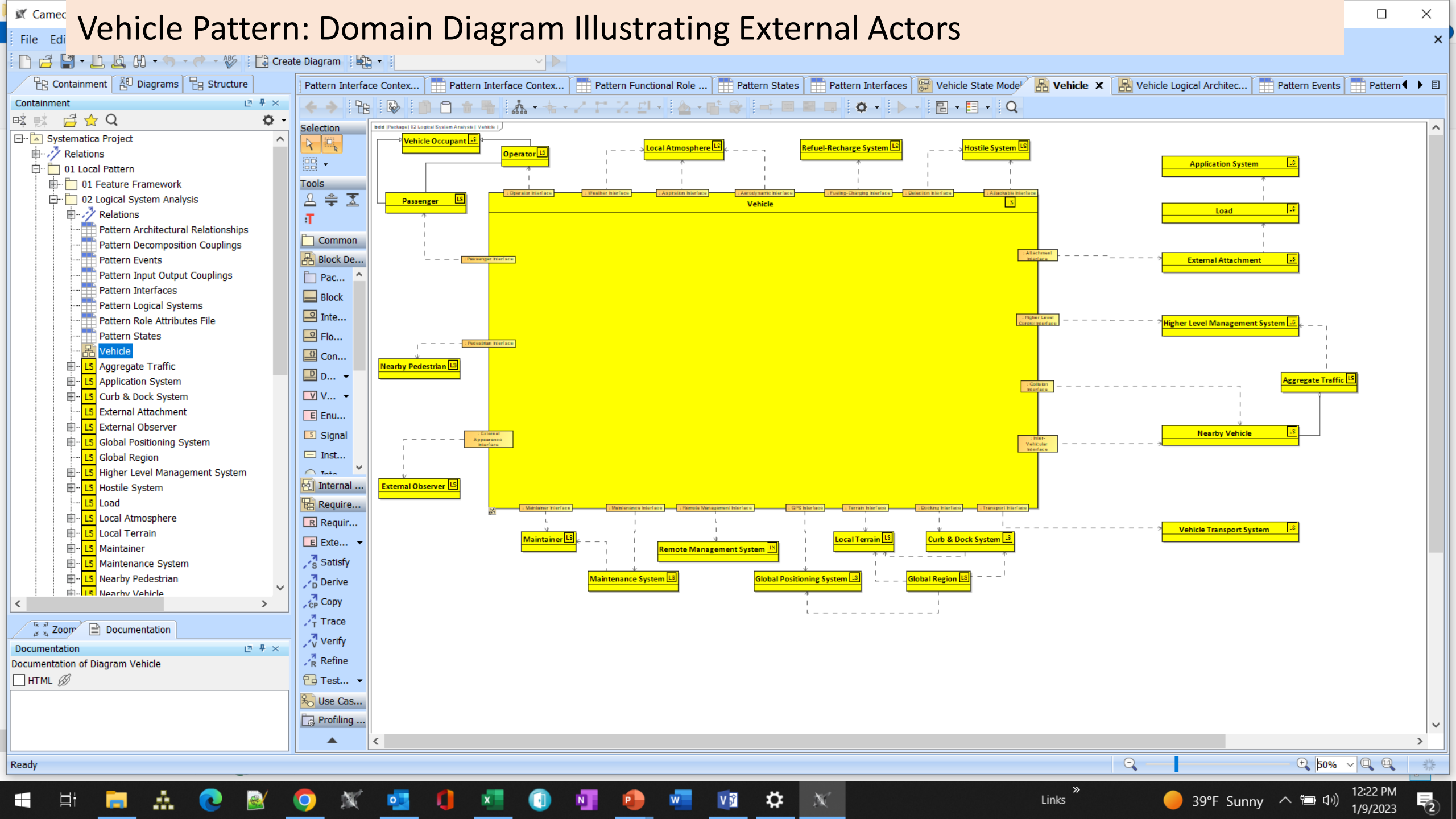
Interaction	IPK	Role	RPK Rule



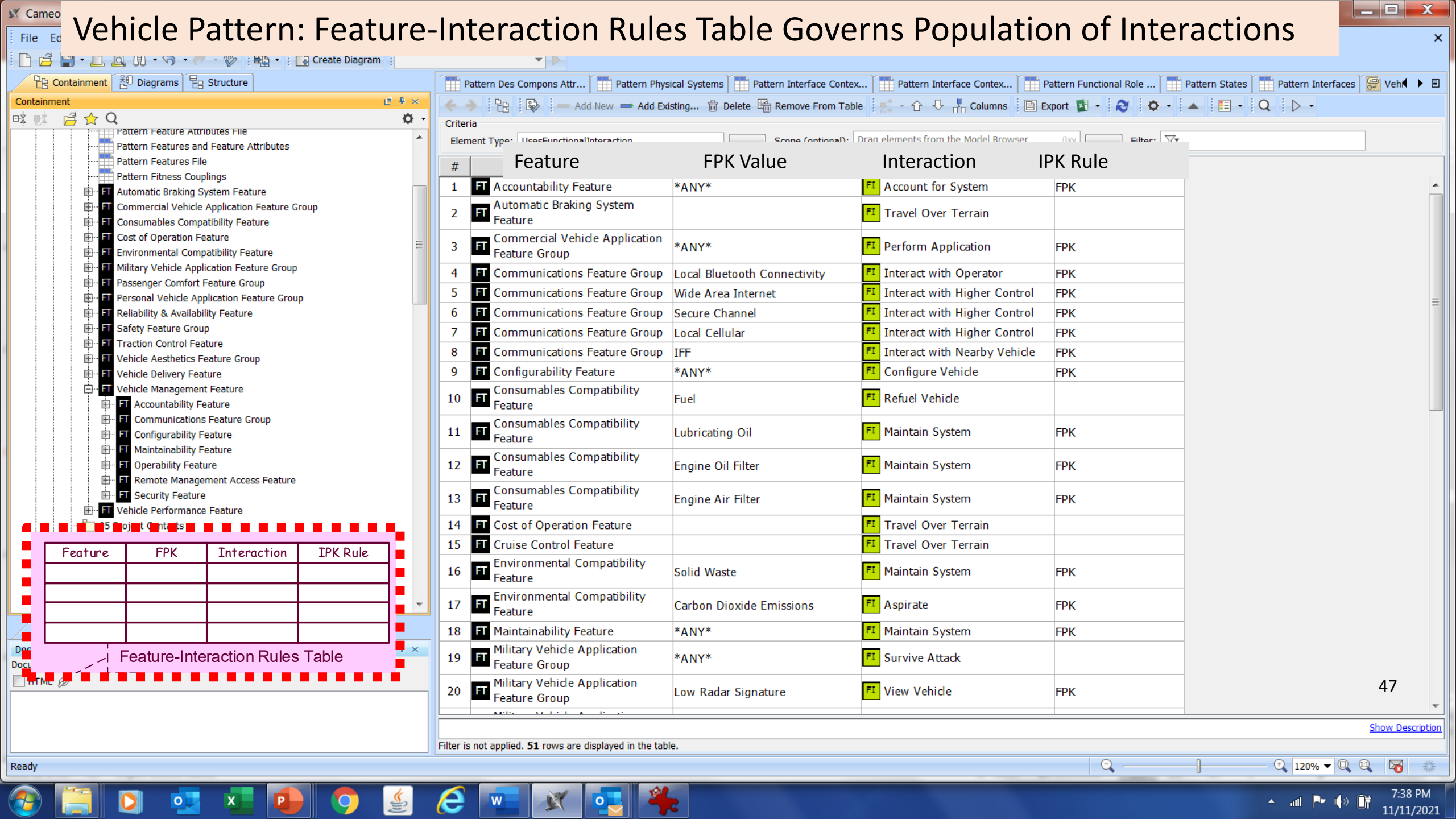
Example S*Patterns mapped into SysML tooling

- Vehicle Pattern
- International Traveler Power Converter Pattern

Vehicle Pattern: Domain Diagram Illustrating External Actors

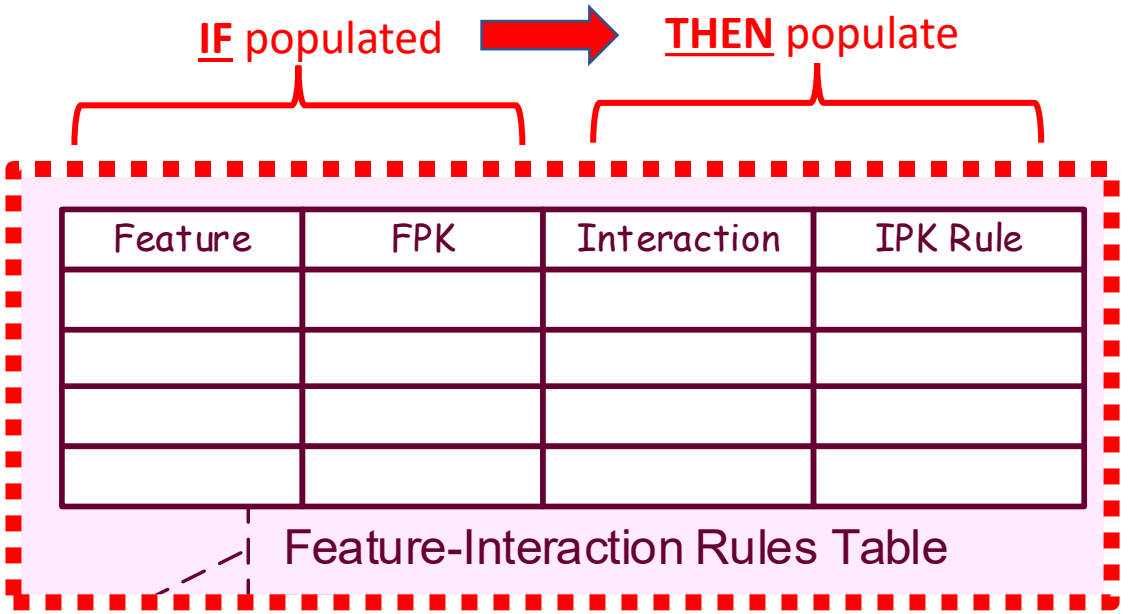


Vehicle Pattern: Feature-Interaction Rules Table Governs Population of Interactions



- Configuration Rules for Interactions are “IF-THENS” that govern the following:
  - For each possible populated Feature type, what Interaction type(s) should be populated?
    - Note that this is a Many-to-Many cardinality—a single Feature type may populate multiple Interaction types, and a single Interaction type may be populated by many Feature types.
  - What specialization may apply?
    - Limited to sub-types of the Feature, indicated by Feature Primary Key (FPK) values?
    - Should the Interactions be specialized, indicated by their Interaction Primary Key (IPK) values?

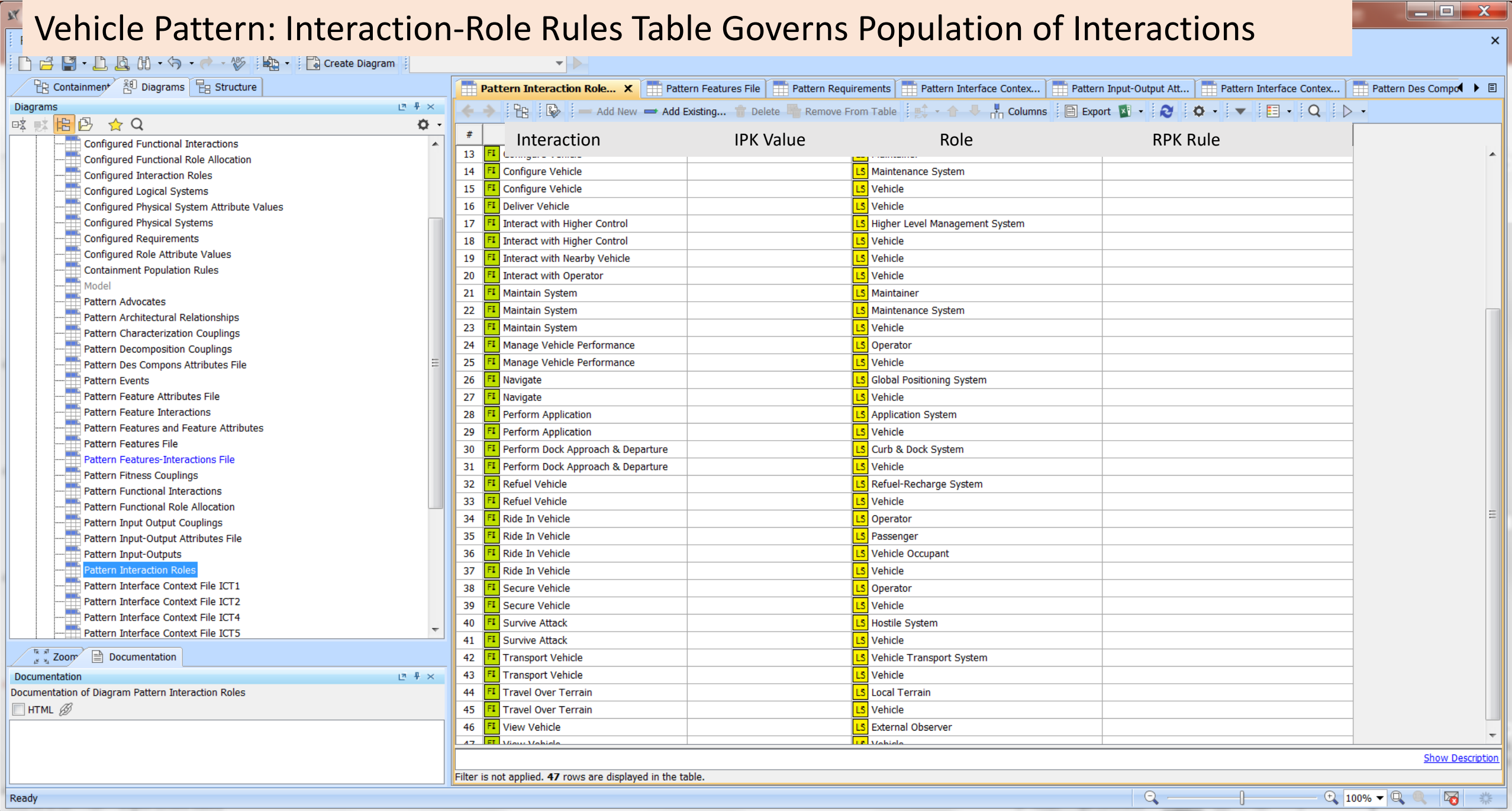
A tabular view of configuration rules for Interactions:



IPK Configuration Rules: Simple Choices

Entry in the IPK Rule Column	Interpretation
FPK	Means set the IPK of the populated Interaction to equal the value of the FPK of the Feature.
/<literal string>/	Means set the IPK of the populated Interaction to be <literal string>
FPK+/<literal string>	Means set the IPK of the populated Interaction to equal the value of the FPK of the Feature, concatenated with the <literal string>
ANY	Means associate the populated Feature with any Interactions of this type already populated, but do not populate a new Interaction.
(blank)	If no IPK Rule, then populate the Interaction with no IPK value.

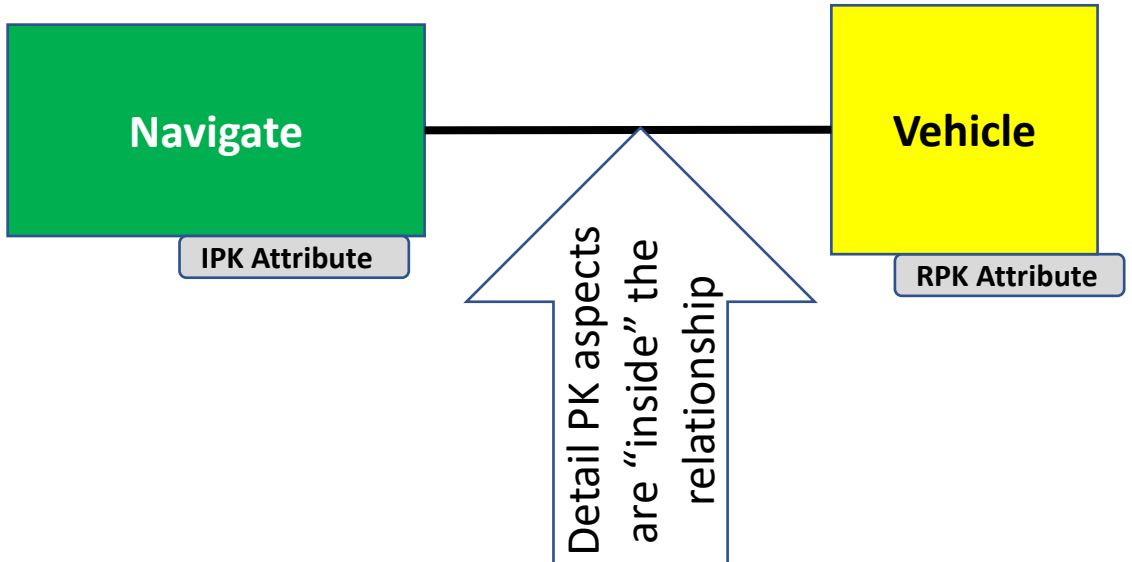
Vehicle Pattern: Interaction-Role Rules Table Governs Population of Interactions



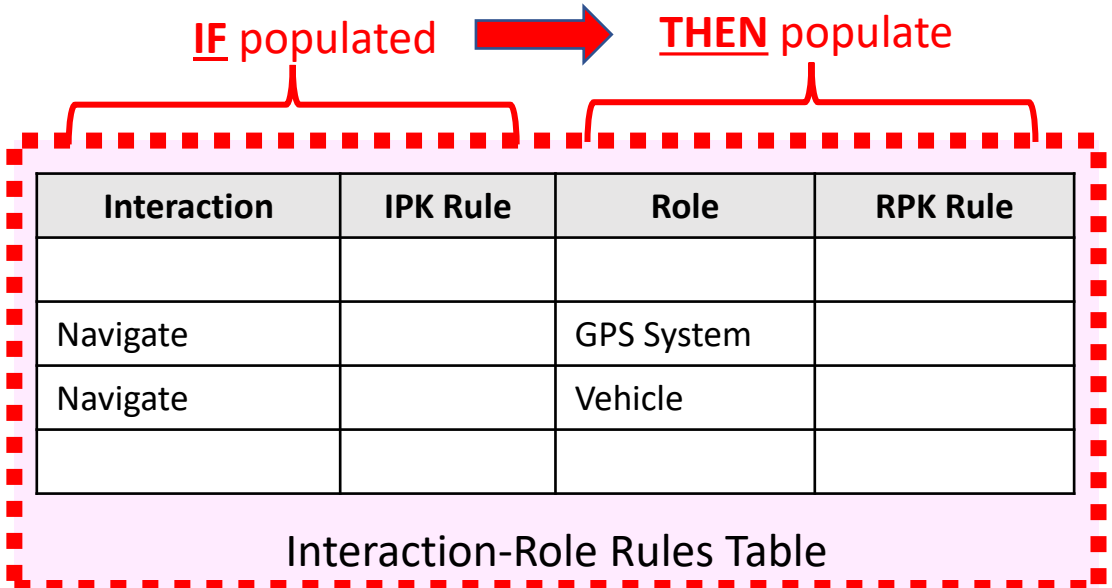
#	Interaction	IPK Value	Role	RPK Rule
13	[illegible]		[illegible]	
14	Configure Vehicle		Maintenance System	
15	Configure Vehicle		Vehicle	
16	Deliver Vehicle		Vehicle	
17	Interact with Higher Control		Higher Level Management System	
18	Interact with Higher Control		Vehicle	
19	Interact with Nearby Vehicle		Vehicle	
20	Interact with Operator		Vehicle	
21	Maintain System		Maintainer	
22	Maintain System		Maintenance System	
23	Maintain System		Vehicle	
24	Manage Vehicle Performance		Operator	
25	Manage Vehicle Performance		Vehicle	
26	Navigate		Global Positioning System	
27	Navigate		Vehicle	
28	Perform Application		Application System	
29	Perform Application		Vehicle	
30	Perform Dock Approach & Departure		Curb & Dock System	
31	Perform Dock Approach & Departure		Vehicle	
32	Refuel Vehicle		Refuel-Recharge System	
33	Refuel Vehicle		Vehicle	
34	Ride In Vehicle		Operator	
35	Ride In Vehicle		Passenger	
36	Ride In Vehicle		Vehicle Occupant	
37	Ride In Vehicle		Vehicle	
38	Secure Vehicle		Operator	
39	Secure Vehicle		Vehicle	
40	Survive Attack		Hostile System	
41	Survive Attack		Vehicle	
42	Transport Vehicle		Vehicle Transport System	
43	Transport Vehicle		Vehicle	
44	Travel Over Terrain		Local Terrain	
45	Travel Over Terrain		Vehicle	
46	View Vehicle		External Observer	
47	[illegible]		[illegible]	

- Configuration Rules for Roles are “IF-THENS” that govern the following:
  - For each possibly populated Interaction type, what Role type(s) to populate?
    - Note that this is a Many-to-Many cardinality—a single Interaction type may populate multiple Role types, and a single Role type may participate in (and be populated by) many Interaction types. (Creates an “instance mesh”.)
  - What specialization may apply?
    - Limited to sub-types of Interaction? Indicated by Interaction Primary Key (IPK Rule) values.
    - Should the Roles be specialized? Indicated by Role Primary Key (RPK Rule) values.

Graphical view of configuration rules for Roles:



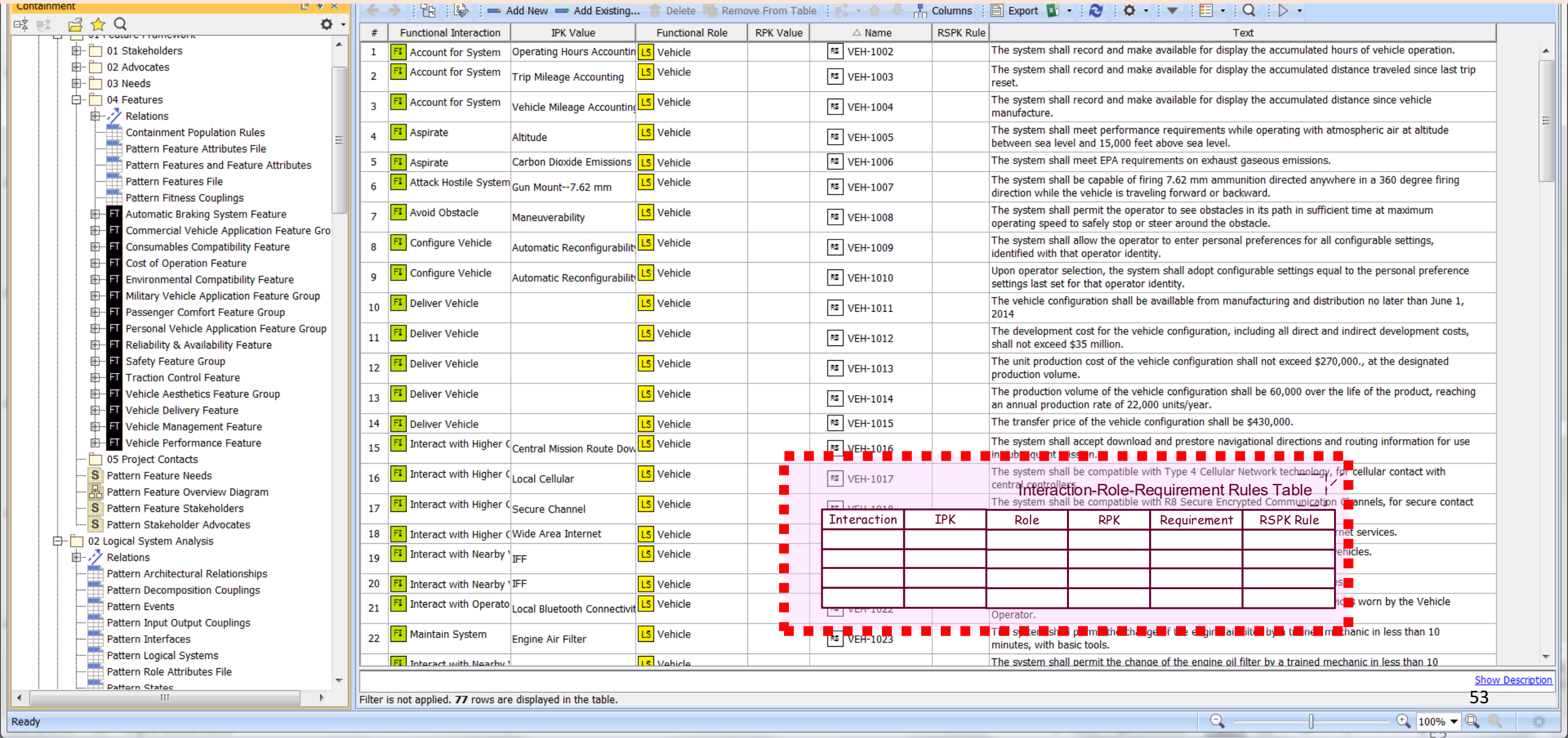
Tabular view of configuration rules for Roles:



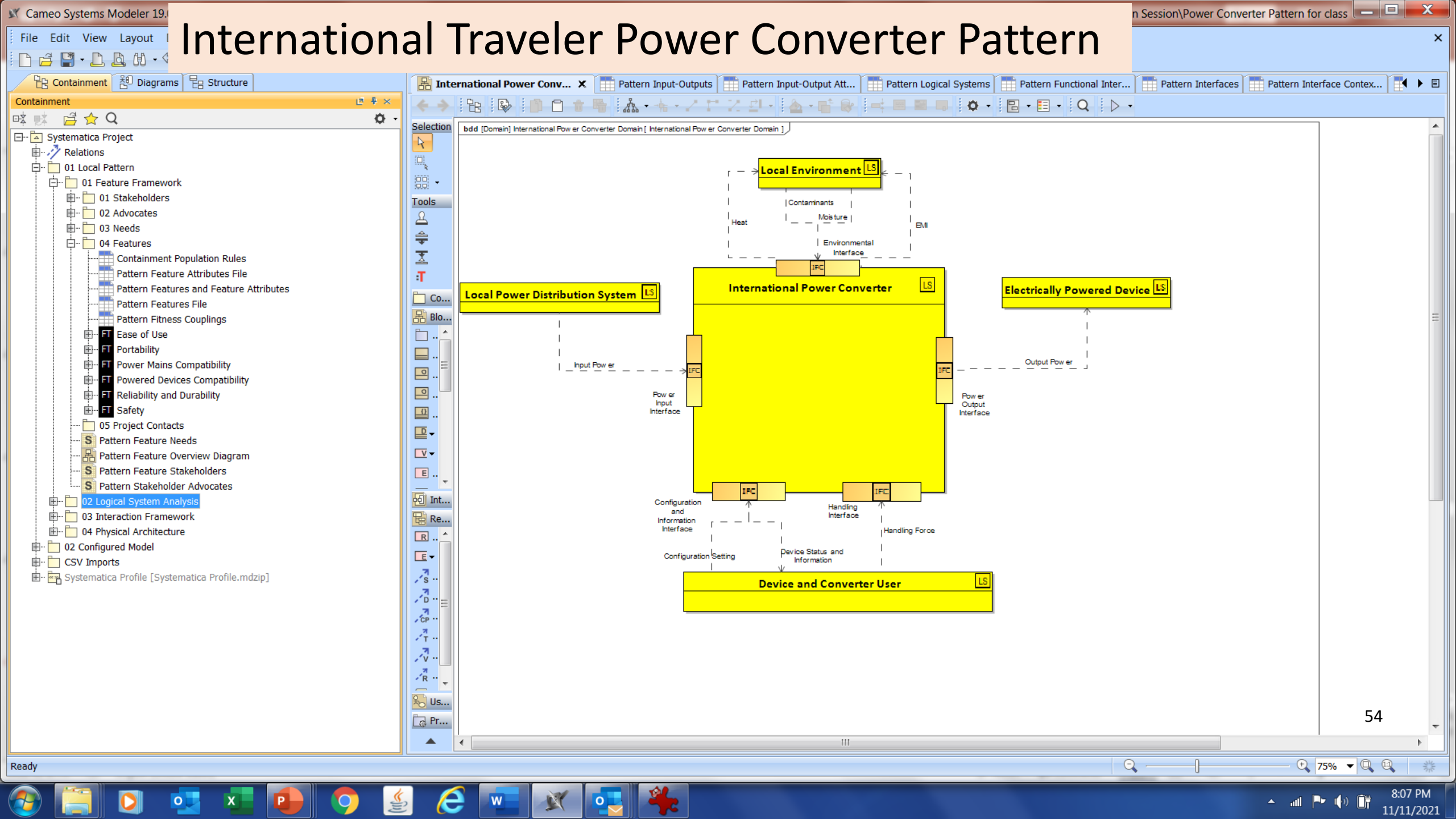
RPK Configuration Rules: Simple Choices

Entry in the RPK Rule Column	Interpretation
IPK	Means set the RPK of the populated Role to equal the value of the IPK of the Interaction.
/<literal string>/	Means set the RPK of the populated Role to be <literal string>
IPK+/<literal string>	Means set the RPK of the populated Role to equal the value of the IPK of the Interaction, concatenated with the <literal string>
(blank)	If no RPK Rule, then populate the Role with no RPK value.

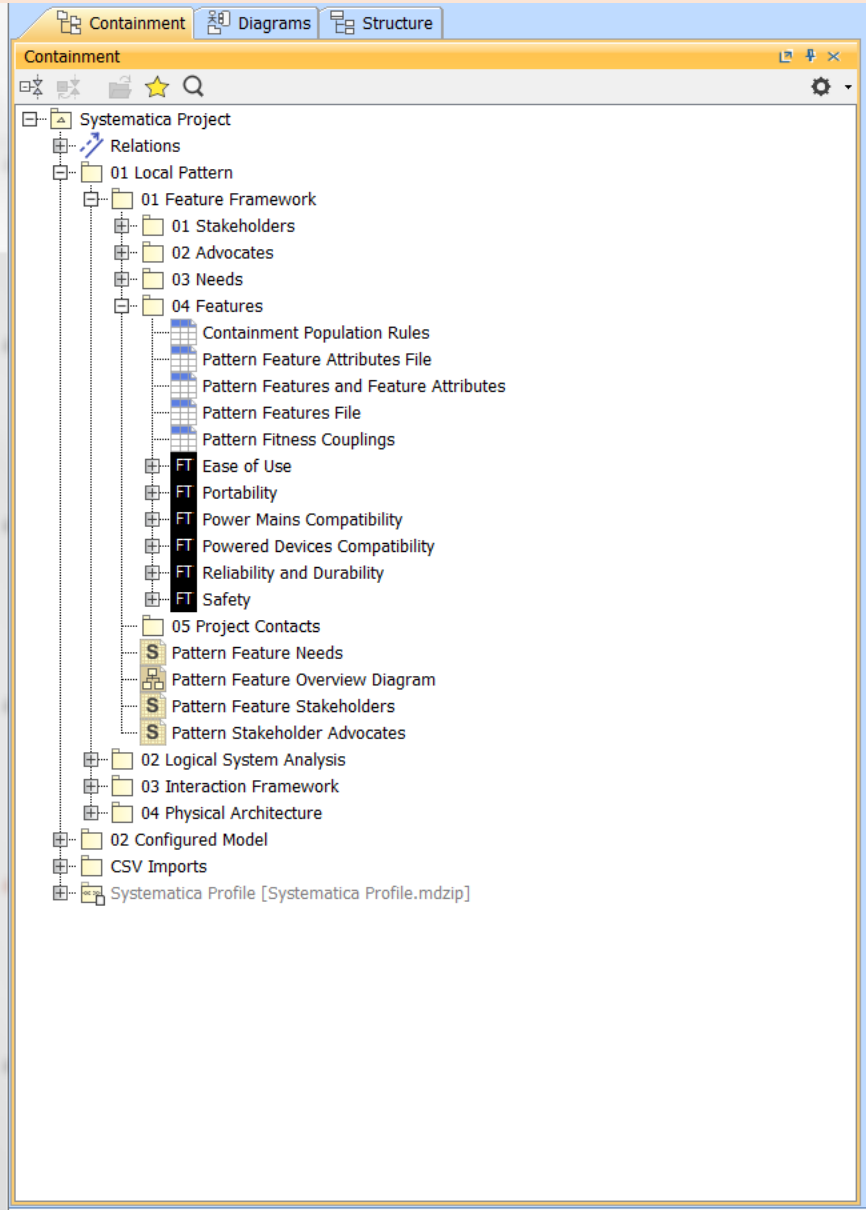
Vehicle Pattern: Interaction-Role-Requirements Table Governs Population of Requirements, based on combination "IF" of Interaction-Role pairs.



International Traveler Power Converter Pattern



International Traveler Power Converter Pattern: Feature-Interaction Rules



Pattern Interface Contex... | Pattern Interface Contex... | Pattern Interface Contex... | Pattern Interface Contex.. | Pattern Features-Interac... | Pattern Features and P

Add New | Add Existing... | Delete | Remove From Table | Columns | Export

Criteria

Element Type: UsesFunctionalInteraction | Scope (optional): 04 Features | Filter:

#	Feature	FPK Value	Interaction	IPK Rule
1	FT Ease of Use		FI Display Status and Information	
2	FT Ease of Use		FI Handle Device	
3	FT Ease of Use		FI Set Configuration	
4	FT Portability		FI Handle Device	
5	FT Reliability and Durability		FI Resist Contamination	
6	FT Reliability and Durability		FI Tolerate Moisture	
7	FT Safety		FI Handle Device	
8	FT Power Mains Compatibility		FI Consume Electrical Power	
9	FT Powered Devices Compatibility	*ANY*	FI Convert Electrical Power	FPK

Feature	FPK	Interaction	IPK Rule

Feature-Interaction Rules Table

55

Show Description

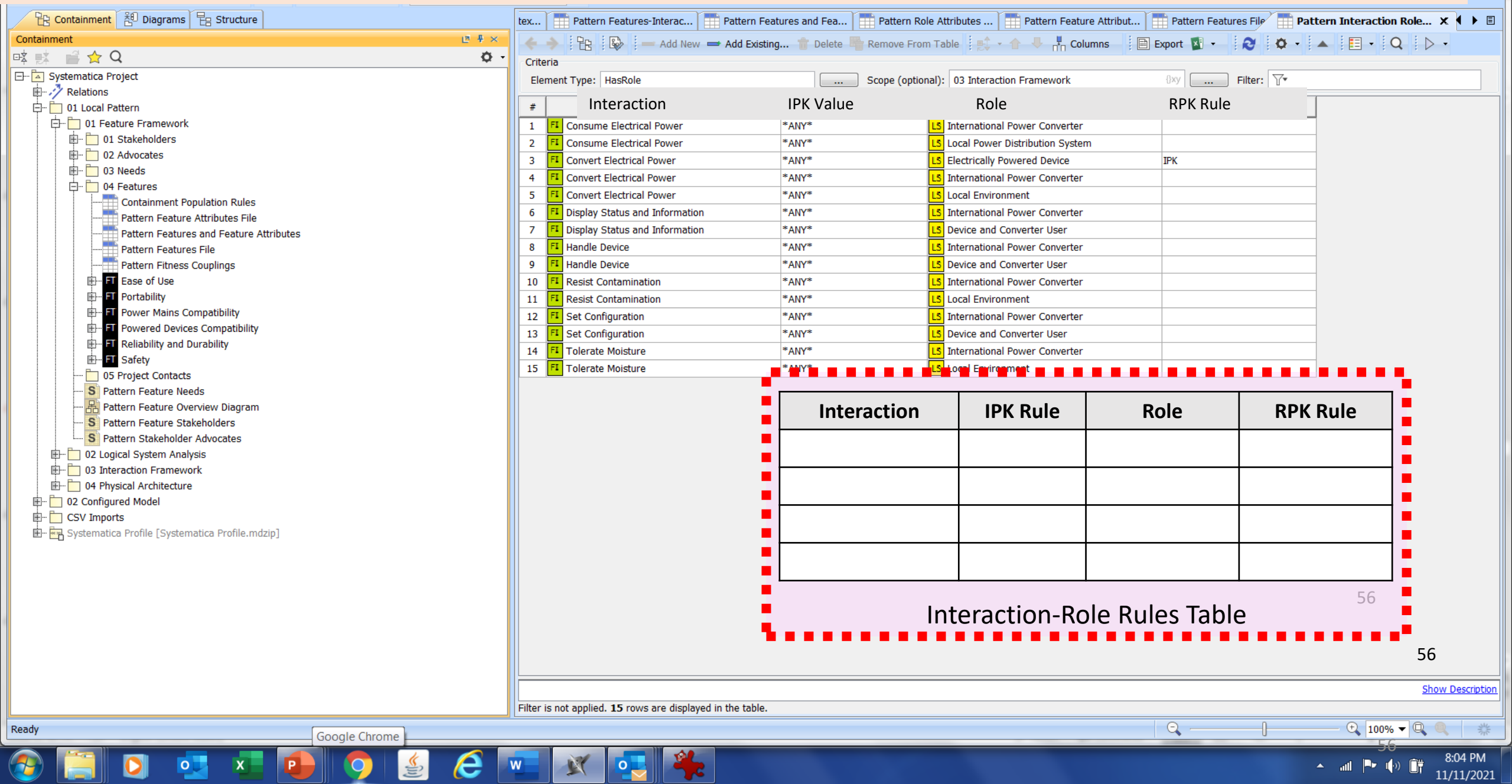
Filter is not applied. **9** rows are displayed in the table.

Ready

140%

8:02 PM
11/11/2021

International Traveler Power Converter Pattern: Interaction-Role Rules

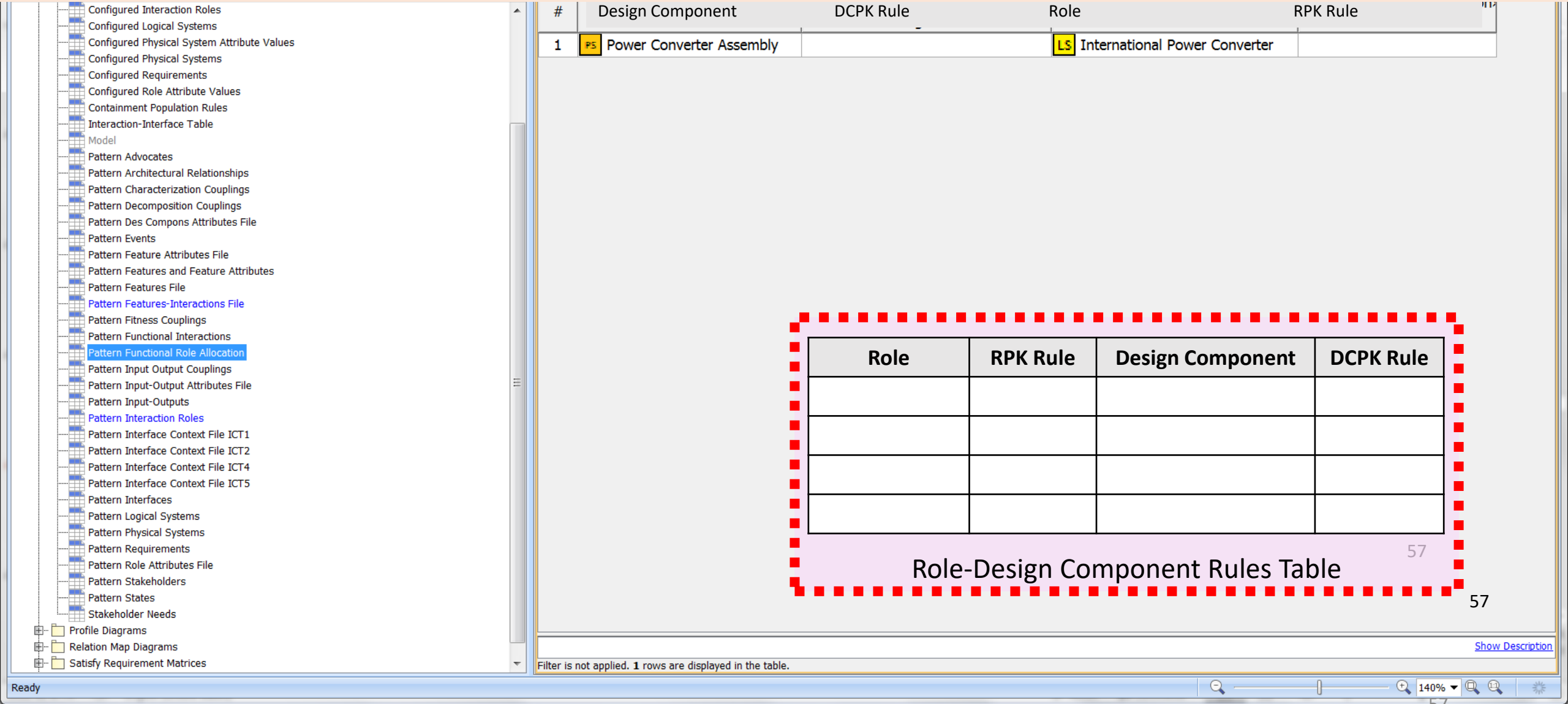


Interaction	IPK Rule	Role	RPK Rule

56

Interaction-Role Rules Table

International Traveler Power Converter Pattern: Role-Design Component Rules



Role	RPK Rule	Design Component	DCPK Rule

57

Role-Design Component Rules Table

In-class Exercises

- Open the CSM SysML model for the example teaching Pattern from prior homework assignments. Find the Features-Interactions Table for that pattern. Explain what Interactions each populated Feature will populate. Find the Interactions-Roles Table. Explain what Roles each populated Interaction will populate.
- Edit the Configuration Rules in the Pattern Feature-Interaction and Pattern Interactions-Roles tables.

Population Rules: Editing Interaction Population Rules

- The “Pattern Feature Interactions” table under the Interaction Framework package allows the user to edit the Interaction Population Rules
- The values in the FPK Value and IPK Rule columns govern when and how Interactions are populated based on which Features have been populated.
- The FPK Value may be blank, *ANY*, or <constant> and helps decide if an Interaction should be populated
- The IPK Rule may be blank, FPK, /<constant>/, or FPK+/<constant>/ and describes how to generate the IPK Value of the Interaction to be populated.
- An IPK Rule of *ANY* will tell the Configuration Agent not to populate an Interaction but to populate a relationship between the Feature and Interaction if they are both populated from other population rules.
- A detailed list of Population Rules is in the Metamodel document reference.

#	Feature	FPK Value	Interaction	IPK Rule
1	Ease of Purchase		Purchase Device	
2	Ease of Use		Display Status and Information	
3	Ease of Use		Handle Device	
4	Environmentally Friendly		Recycle Device	
5	Portability		Handle Device	
6	Power Mains Compatibility		Consume Electrical Power	
7	Powered Devices Compatibility	*ANY*	Convert Electrical Power	FPK
8	Powered Devices Compatibility	*ANY*	Set Configuration	FPK
9	Reliability and Durability		Resist Contamination	
10	Reliability and Durability		Tolerate Moisture	
11	Reliability and Durability		Provide Warranty Data	
12	Safety	*ANY*	Handle Device	

Population Rules: Editing Role Population Rules

- The “Pattern Interaction Roles” table under the Interaction Framework package allows the user to edit the Role Population Rules
- The values in the IPK Value and RPK Rule columns govern when and how Logical Systems are populated based on which Interactions have been populated.
- The IPK Value may be blank, *ANY*, or <constant> and helps decide if a Logical System should be populated.
- The RPK Rule may be blank, IPK, /<constant>/, or IPK+/<constant>/ and describes how to generate the RPK Value of the Logical System to be populated.
- A detailed list of Population Rules is in the Metamodel document reference.

#	△ Interaction	◇ IPK Value	Logical System	◇ RPK Rule
1	Consume Electrical Power	*ANY*	International Power Converter	
2	Consume Electrical Power	*ANY*	Local Power Distribution System	
3	Convert Electrical Power	*ANY*	Electrically Powered Device	IPK
4	Convert Electrical Power	*ANY*	International Power Converter	
5	Convert Electrical Power	*ANY*	Local Environment	
6	Display Status and Information	*ANY*	International Power Converter	
7	Display Status and Information	*ANY*	Device and Converter User	
8	Handle Device	*ANY*	International Power Converter	
9	Handle Device	*ANY*	Device and Converter User	
10	Provide Warranty Data	*ANY*	Product Supplier	
11	Provide Warranty Data	*ANY*	Device and Converter User	
12	Purchase Device	*ANY*	Device and Converter User	
13	Purchase Device	*ANY*	International Power Converter	
14	Recycle Device	*ANY*	Product Supplier	
15	Recycle Device	*ANY*	International Power Converter	
16	Recycle Device	*ANY*	Local Environment	
17	Resist Contamination	*ANY*	International Power Converter	
18	Resist Contamination	*ANY*	Local Environment	
19	Set Configuration	*ANY*	International Power Converter	
20	Set Configuration	*ANY*	Device and Converter User	
21	Tolerate Moisture	*ANY*	International Power Converter	
22	Tolerate Moisture	*ANY*	Local Environment	

Homework assignment 5

- Use the example teaching Pattern or other pattern from prior homework assignments.
- Edit the Configuration Rules in the Pattern Functional Role Allocation and Pattern Requirements tables.

Population Rules:
Editing Physical System (Design Component) Population Rules

- The “Pattern Functional Role Allocations” table under the Physical Architecture package allows the user to edit the Physical System (Design Component) Population Rules
- The values in the Configuration Rule and IPPK Value columns govern when and how Physical Systems (Design Components) are populated based on which Logical Systems (Roles) have been populated.
- The Configuration Rule may be blank, *ANY*, or <constant> and helps decide if a Physical System (Design Component) should be populated.
- The IPPK Rule may be blank, RPK, /<constant>/, or RPK+/<constant>/ and describes how to generate the IPPK Value of the Physical System (Design Component) to be populated.
- A detailed list of Population Rules is in the Metamodel document reference.

#	Design Component	◇ IPPK Value	Role	◇ Configuration Rule
1	Power Converter Assembly		International Power Converter	

Population Rules:
Editing Requirement Population Rules

- The “Pattern Requirements” table under the Interaction Framework package allows the user to edit the Requirement Population Rules
- The values in the IPK Value, RPK Value, and RSPK Rule columns govern when and how Requirement Statements are populated based on which Interactions and Logical Systems have been populated.
- The IPK Value may be blank, *ANY*, or <constant> and helps decide if a Requirement Statement should be populated.
- The RPK Value may be blank, *ANY*, or <constant> and helps decide if a Requirement Statement should be populated.
- The RSPK Rule may be blank, IPK, RPK, /<constant>/, or IPK+/<constant>/, or RPK+/<constant>/ and describes how to generate the RSPK Value of the Requirement Statement to be populated.
- A detailed list of Population Rules is in the Metamodel document reference.

Population Rules:
Editing Requirement Population Rules

#	Functional Interaction	IPK Value	Functional Role	RPK Value	Req ID	RSPK Rule	Text
1	FI Consume Electrical Power		LS International Power Converter		REQ 001		The system shall consume not more than [Max Drain on Mains] of Input Power from the Local Power Distribution System.
2	FI Convert Electrical Power	*ANY*	LS International Power Converter	*ANY*	REQ 002	IPK	The system shall provide up to [Max Power Drain] of Output Power to each Electrically Powered Device.
3	FI Convert Electrical Power	*ANY*	LS Electrically Powered Device	*ANY*	REQ 003	IPK	The system shall not consume more than [Max Power Drain] of Output Power.
4	FI Consume Electrical Power		LS Local Power Distribution System		REQ 004		The system shall provide up to [Max Drain on Mains] of Input Power.
5	FI Handle Device		LS International Power Converter		REQ 005		The system shall not present a shock hazard to users when operated according to its instructions.
6	FI Convert Electrical Power	*ANY*	LS International Power Converter	*ANY*	REQ 006	IPK	The system shall generate Output Power to attached Electrically Powered Devices which is compatible with the [Output Voltage-Power Profile] and [Output Frequency Profile].

Discussion, Q&A.

-
-
-
-
-
-



References

1. Schindel, W., Peterson, T. 2016. “Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques”. https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:pbse_tutorial_glrc_2016_v1.7.4.pdf
2. Patterns WG. 2019b. “Methodology Summary: Pattern-Based Systems Engineering PBSE), Based On S*MBSE Models” https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:pbse_extension_of_mbse--methodology_summary_v1.6.1.pdf
3. “What Is the Smallest Model of a System?”, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:what_is_the_smallest_model_of_a_system_v1.4.4.pdf
4. Oil Filter Product Line Family Extract, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:extract_from_--_system_requirements_document_v1.2.2a.pdf
5. Extracts from S*Patterns, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:attachment_1--example_extracts_from_s-patterns_v1.2.4.pdf
6. Patterns WG. 2021. INCOSE MBSE Patterns Working Group Web Site: https://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns
7. Schindel, W. 2021. “Variational Forces of Modularity: Coupled Macro and Micro Patterns in the Innovation Ecosystem”. Presented at PLE Momentum 2021 Conference. https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:the_forces_of_modularity_v1.3.3.pdf