Document Instructions

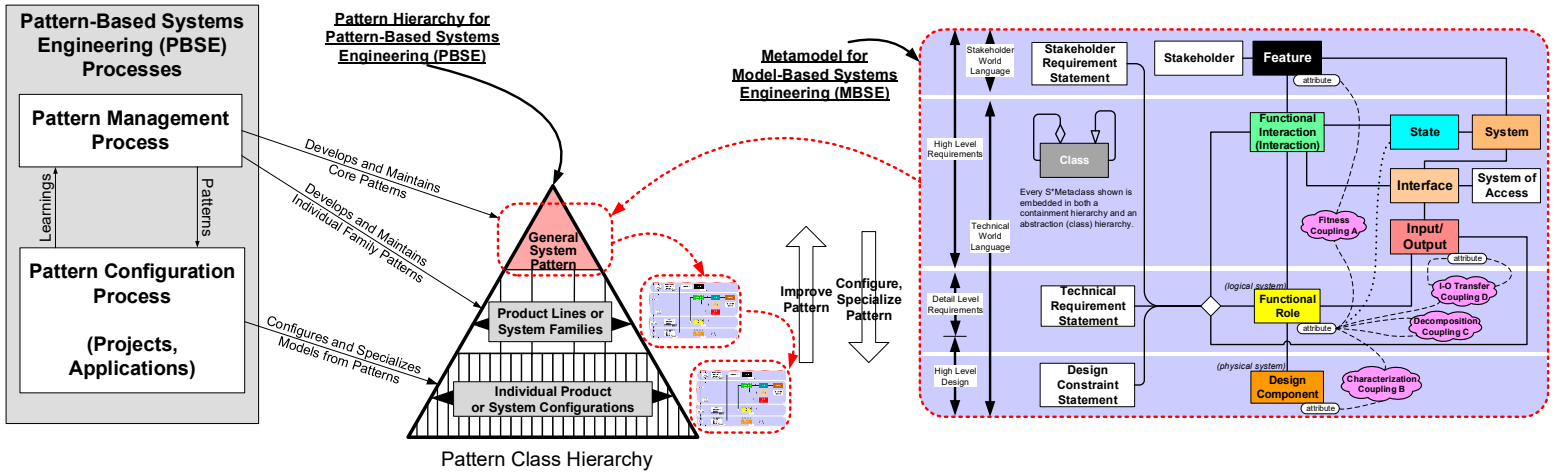
- Remove this page before use in the ATP Session.
- This is a configurable class template, not for use without configuring first.
- Globally replace items identified in arrows as shown: <<XYZ>>
  - includes company-specific names, logos, and homework pattern file names and locations

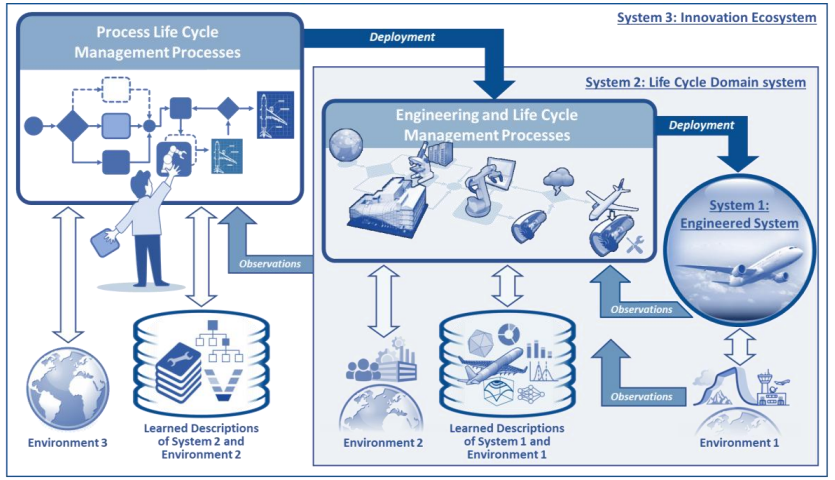
ICTT System Sciences®
Understand your systems.

<<Company/Org. Logo>>

Phase I Session 1

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

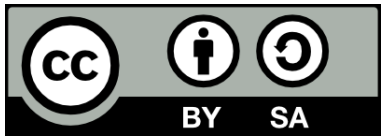




schindel@ictt.com
lewis@ictt.com

V1.2.27

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.

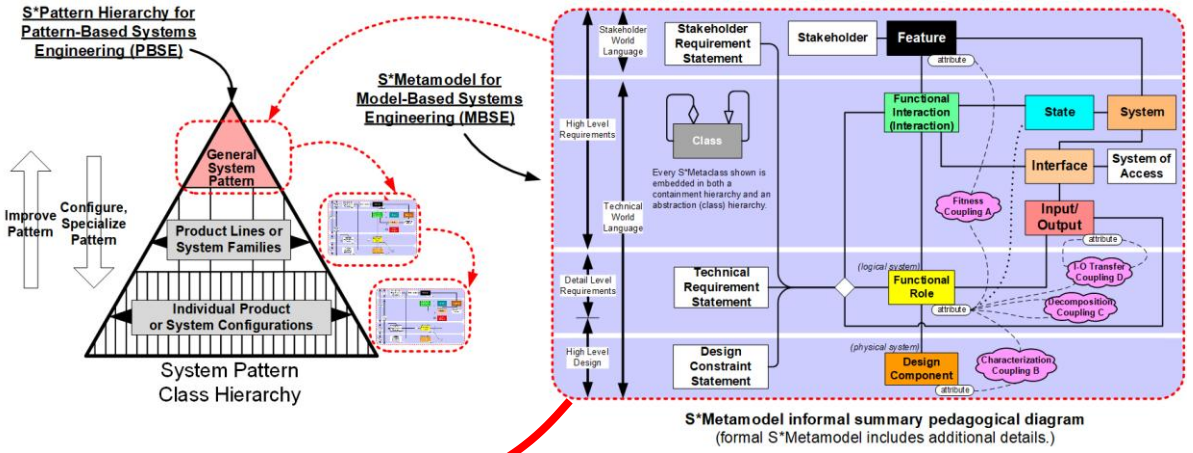
Permissions beyond the scope of this license are administered through contacting:

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.

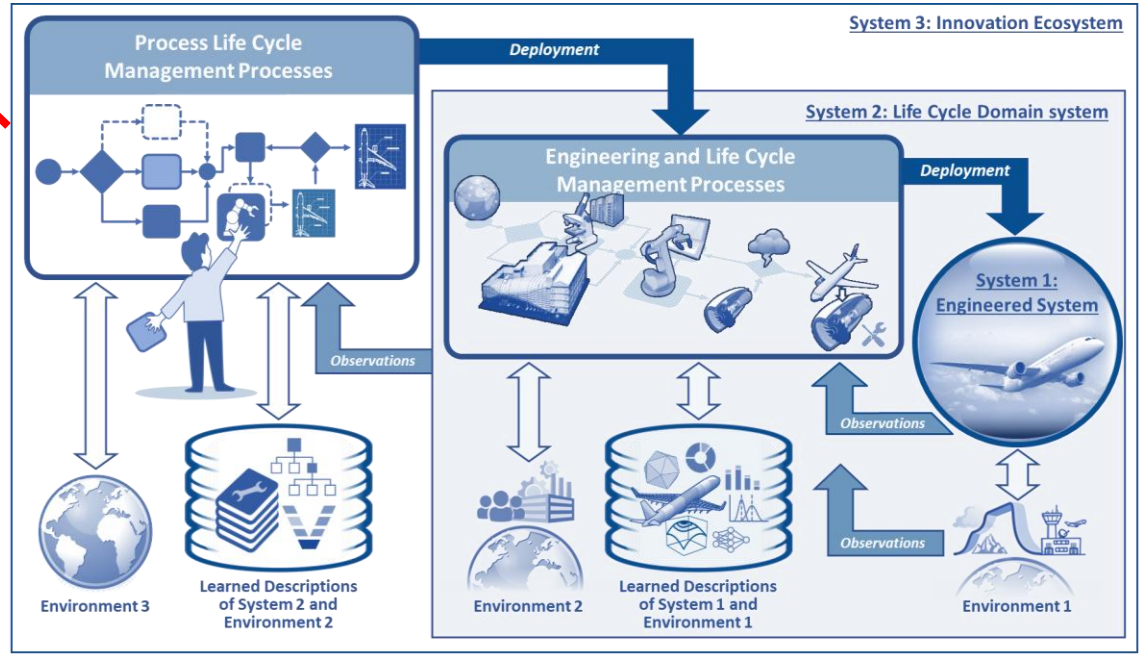
Contents

- Course Sequence Overview:
  - Goals, sessions & content plan, schedule
  - Class Phase I: Content outline
  - Class Phase II: Content outline
- Phase I Session 1: Class Content
- Additional References



S*Models, S*Patterns

S*Ecosystem Pattern



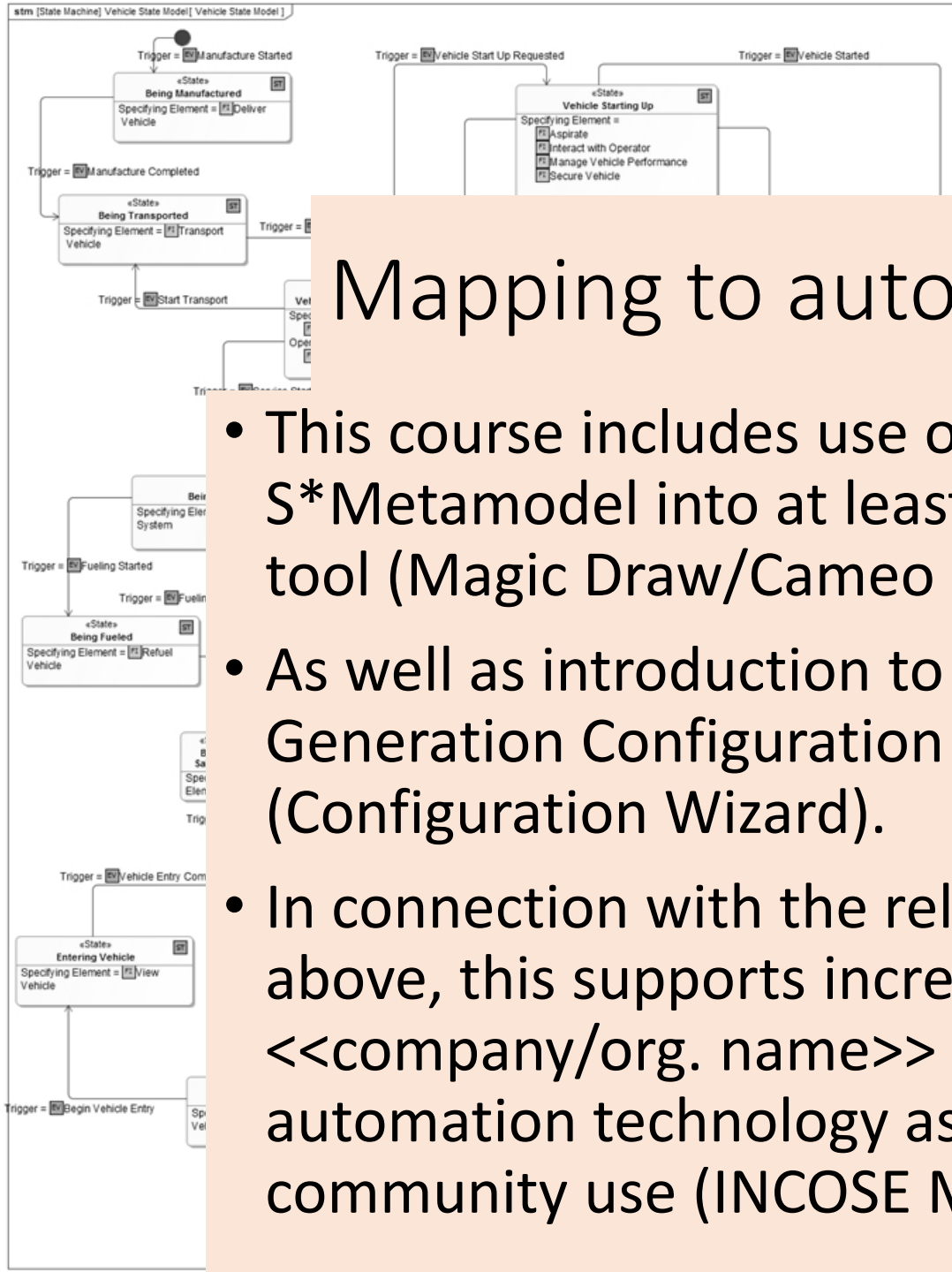
Goals

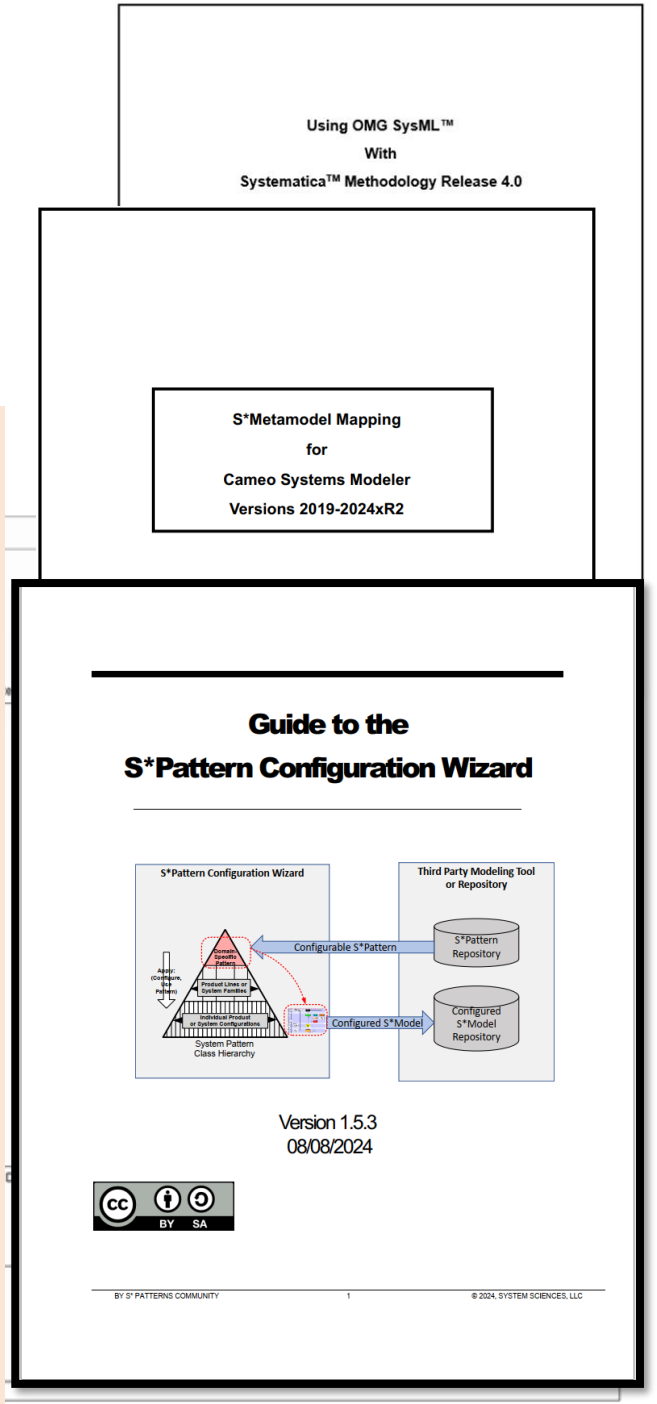
A. Improve industry performance of systems engineering: Enable <<company/org. name>> staff use of S*Model, S*Patterns, related methods, to lift customer industry practices, outcomes.

B. Focus on:
   1. S*Metamodel, S*Models created from it;
   2. Leverage of S* MBSE Patterns;
   3. Use of ASELCM Ecosystem S*Pattern as a neutral integrating reference framework for Digital Engineering and SE Transformation by customers and industry;
   4. Mapping of S*Metamodel and S*Pattern configuration automation to third party COTS or similar tooling, and their effective use in supporting the above;
   5. Displaying cross-industry Patterns Community activity and advancement that encourage similar progress by others, increasing unification of systems efforts.

Mapping to automated agents

- This course includes use of the mapping of the S*Metamodel into at least one SysML modeling tool (Magic Draw/Cameo Systems Modeler).
- As well as introduction to the (new) Third Generation Configuration Agent Technology (Configuration Wizard).
- In connection with the related Goals listed earlier above, this supports increased involvement by <<company/org. name>> in the related automation technology assets for shared community use (INCOSE MBSE Patterns WG).





Expectations of participants

- Engage, be active, raise questions, push back/object, make suggestions
- Do the homework assignments; bring back questions, observations
- Apply what is learned in customer projects
- Support internal <<company/org. name>> convergence to a configurable framework of <<company/org. name>> and/or Customer deliverables and recommendations
- Advance the practice across the community

Assumed background of participants

- This is not a general class on tool or language specifics of modeling as a whole.
- Rather, it is focused on certain aspects of modeling, and use of models, that appear to need advancement in systems engineering practice, based upon:
  - broad observation of industry practices of system modeling and model use, compared to . . .
  - what earlier generations of physical sciences and their related engineering disciplines and models accomplished over the last 300 years;
  - and based on years of applying those same historical lessons to advancing the relatively younger practice of systems engineering and its evolving use of models.
- Based on the targeted participants in this offering of the class material, the general level of content assumes some experience and expertise on the part of those participants, but . . .
- Because many of the points of emphasis are so fundamental, this material should also be accessible and of value to less experienced participants:
  - The main thing is to engage, interact, and be homework diligent.

Plan of Phased class sessions and homework

Deliverable	Notes
Pre-class Preparation: Homework model(s) identification, review, configuration of class content. Planning of class to support and reinforce <<company/org. name>> strategic goals.	Assumes class illustrations use existing examples; participants' homework use models of interest to <<company/org. name>> .
Phase I Class Sessions: Microstructure of Models and Patterns	6 sessions in Phase I
S*Models, S*Metamodel: In Example Models Context (Class and Homework)	3 sessions of 3 hours each
Formal Mapping of S*Metamodel into Tooling	3 sessions of 3 hours each
S*Patterns, Configuration: In Example Models Context (Class and Homework)	
Introduction to 3rd Generation Pattern Configurator Technology	
Participants' Homework Presentation and Feedback	Distributed over above, other sessions
Phase II Class Sessions: Accountability, Macrostructure of Model-Based Ecosystem	4 sessions in Phase II
ASELCM Ecosystem S*Pattern: Nature, Intended Uses, Levels 0-4 Logical Arch	1 session of 3 hours
ASELCM Features, Interactions, Roles, Resources, Use in Consist. Mgmt.	2 sessions of 3 hours each
ASELCM Configuration Stages and Accountability	1 session of 3 hours
Participants' Homework Presentation and Feedback Sessions	Distributed over above, other sessions

Participants' time

Deliverable	Instructor Effort (Hrs)	Each <<company/ org. name>> Person (Hrs)	Notes
Pre-class Preparation: Homework model(s) identification, review, configuration of class content. Planning of class to support and reinforce <<company/org. name>> strategic goals.	**30**	**2**	Assumes class illustrations use existing examples; participants' homework use models of interest to <<company/org. name>> or example models.
Phase I Class Sessions: Microstructure of Models and Patterns	**30**	**34**	**6 sessions in Phase I, 3 hours each**
S*Models, S*Metamodel: In <<company/org. name>> Example Model Context		9	3 sessions of 3 hours
Formal mapping of S*Metamodel into Tooling		3	1 sessions of 3 hours
S*Patterns, Configuration: In <<company/org. name>> Example Model Context		3	1 sessions of 3 hours
Introduction to 3rd Generation Pattern Configurator Technology		3	1 sessions of 3 hours
Homework Preparation (Presented and discussed in above sessions)		16	Shows participant hours outside class
Phase II Class Sessions: Accountability, Macrostructure of Model-Based Ecosystem	**30**	**24**	**4 sessions in Phase II, 3 hours each**
ASELCM Ecosystem S*Pattern: Nature, Intended Uses, Levels 0-4 Logical Arch		3	1 session of 3 hours
ASELCM Features, Interactions, Roles, Resources, Use in Consist. Mgmt.		6	2 sessions of 3 hours each
ASELCM Configuration Stages and Accountability		3	1 session of 3 hours
Homework Preparation (Presented and discussed in above sessions)		12	Shows participant hours outside class
Project Totals	**90**	**60**	

Schedule

Class sessions to occur weekly as scheduled by <<company/org. name>>.

Phase	Session	Date
I	1	Week 1
I	2	Week 2
I	3	Week 3
I	4	Week 4
I	5	Week 5
I	6	Week 6
II	1	Week 7
II	2	Week 8
II	3	Week 9
II	4	Week 10

Plan for example System 1 models

- The class sessions will utilize some basic teaching models:
  - Vehicle Pattern; International Power Converter (IPC) Pattern
  - Both illustrated in Cameo Systems Modeler (CSM) tooling
- By Session 1, you should have installed and begin using the provided S*Metamodel Profile in your own <<company/org. name>> CSM tooling:
  - Session 4 includes more on the mapping of the S*Metamodel into tooling (such as CSM)
  - ICTT providing the Profile early enough to do Sessions 1+ homework in that environment.
- Your homework should be performed using example models of interest to <<company/org. name>> and <<company/org. name>> customer current or potential systems, or the example teaching models:

International Power Converter	International Power Converter Example Pattern xx May 2025.mdzip	<<Pattern/Model File Location>>

- The above refers to the choice of “System 1” model, used in Phase I; Phase II is based on models of enterprises (“System 2 and 3”):
  - Including customer System 2/3 (e.g., defense, medical devices, government, manufacturing, etc.)
  - Including <<company/org. name>> System 2/3 (<<company/org. name>> standard practice archetypes).
  - Including supplier System 2/3, completing the supply chain.

Metamodel Pattern (Phase I) versus the Ecosystem Pattern (Phase II)

- The foundation of this class is practice with the representation of S*Models and S*Patterns—models (in SysML or otherwise) which are based on the S*Metamodel (see Phase I).
- The ASELCM Ecosystem S*Pattern is also part of this class (see Phase II):
  - Because of its importance to effective understanding, analyzing, planning, and implementing digital engineering and model-based methods accountable to the business processes of real enterprises.
- Important to understand: The ASELCM Pattern does not assume that the enterprise(s) it describes are users of S*Models or the S*Metamodel:
  - Quite the opposite: It is for describing any enterprise's innovation information ecosystem.
  - But, the ASELCM Pattern is itself an S*Pattern, making use of the S*Metamodel without expecting it is (necessarily) in use by the modeled enterprise(s).
  - The ASELCM Pattern introduces the Consistency Management paradigm as foundational to accountable systems of engineering and innovation.

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
- Introduction to 3rd Generation Pattern Configurator Technology
  - Session 6
- Homework Presentation and Feedback
  - Distributed over above sessions

S*Models, S*Metamodel: Session 1 of 3

- Motivating background
- S*Models
- S*Metamodel: Pedagogical and formal versions
- Systems
- Functional Interactions: Phenomena; clarifying SE views of behavior
- Functional Roles; attributes; clarifying SE views of function; non-functional
- Stakeholder Features; clarifying SE views of value, selection, risk, FMEA, configuration
- Input-Outputs
- States
- Three paths to finding all the Interactions
- Design Components: Clarifying SE views of allocation, logical, physical
- Homework assignment 1
- Discussion, Q&A.

Motivating background

- Systems Engineering has a different kind of history than the other engineering disciplines—and SE is also a lot younger (50-75 years).
- ME, CE, EE, ChE are supported by phenomena-based physical science providing theoretical foundations based in science and mathematics (300 years+).
- The resulting three century acceleration of STEM has changed the nature and possibilities of human life on Earth—a key test of efficacy that we want to use.
- In contrast, the formalized methodological aspects of SE have a shorter history, with early (successful) focus on improving the technical alignment of work of the other engineering disciplines and the needs of stakeholders—because that was the need.
- Viewed another way:
  - The physical sciences emphasize agreement of models with Nature, empirical observable phenomena, and languages (including mathematics) suited for that purpose[1].
  - Systems engineering (and many other human endeavors) emphasized agreement of people with each other, and languages suited for that purpose.
- INCOSE *Vision 2025* and *Vision 2035* summarize this.

(1) Acknowledging an important component physical science is about agreement between people—but with the important addition of empirical experiment as the primary "referee".

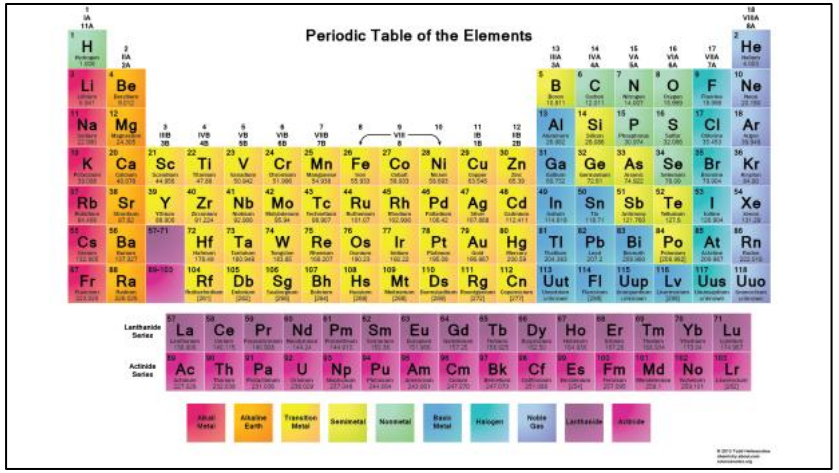
"Traditional" Perspective on SE—as often practiced

- Specialists in individual engineering disciplines (ME, EE, CE, ChE—without them, we would be living as in 1500) sometimes argue that their fields are based on:
  - "real physical phenomena",
  - physical laws based in the "hard sciences", and first principles, . . .
- while claiming Systems Engineering lacks equivalent "phenomena-based" foundation.

$$\nabla \cdot \mathbf{D} = \rho$$
$$\nabla \cdot \mathbf{B} = 0$$
$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$
$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}$$

$$\frac{N_b}{N_a} = \left(\frac{g_b}{g_a}\right)\left(e^{-(E_b - E_a)/kT}\right)$$

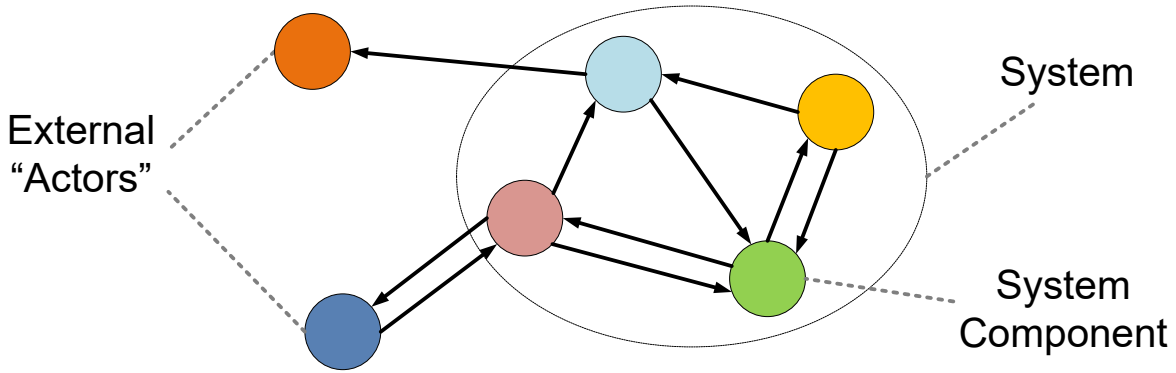
$$H(t)|\psi(t)\rangle = i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle$$

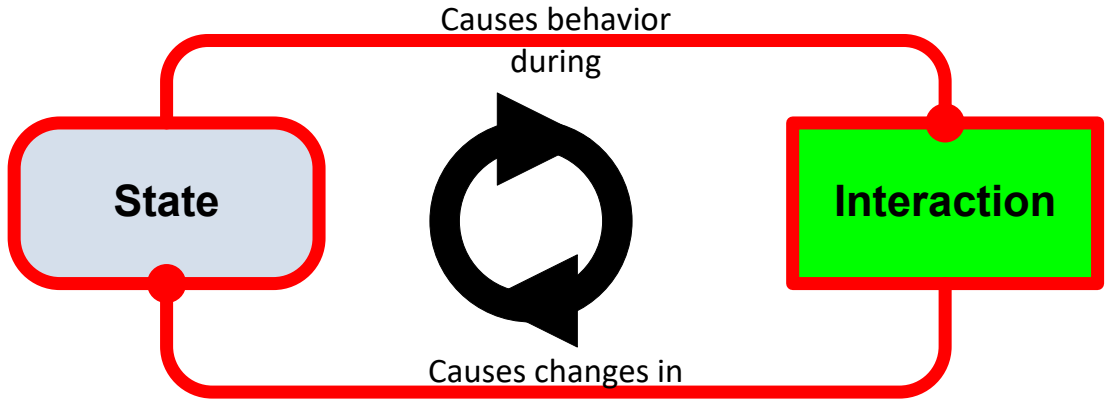


- Instead, Systems Engineering is sometimes viewed as:
  - Emphasizing process and procedure,
  - Critical thinking and good writing skills,
  - Organizing and accounting for information (and lately, "modeling it"),
  - Integrating the work of the other engineering disciplines and stakeholder needs.
- But not based on an underlying "hard science" like other engineering disciplines
- ***We assert otherwise***--this class is about practicing SE with engineering foundation.

Formalizing System Terms and Representations

- Definition*: In the perspective described here**, by "System" we mean a collection of interacting system components:



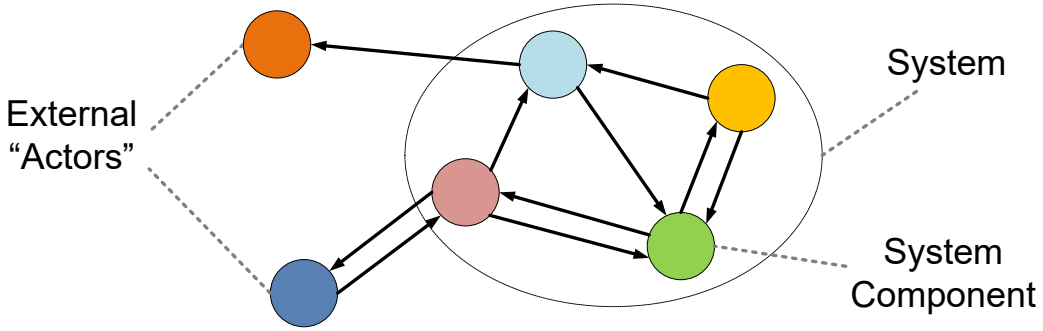


- By "interacting" we mean the exchange of energy, force, material, or information (all of these are "input-outputs") between system components, . . .
- . . . through which one component impacts the state of another component.
- By "state" we mean a property of a component that impacts its input-output behavior during interactions. (Note the circular cause-effect definition chain here.)
- So, a component's "behavior model" describes input-output-state relationships during interaction—*there is no "naked behavior" in the absence of interaction*.
- The behavior of a system involves emergent *states of the system as a whole*, exhibited in its behavior during its own external interactions, resulting in observable holistic aspects.

(* Other world view definitions of "System" are acknowledged; there are reasons for our minimalist choice of definitions.)

The System Phenomenon

- Phenomena of the hard sciences in all instances occur in the presence of special cases of the (generalized) "System Phenomenon":
  - *The System Phenomenon: System behavior emerges from the interaction of behaviors (phenomena themselves) of system components a level of decomposition lower.*
- Each emergent phenomenon is then visible through the subsequent interaction-based behavior of the larger system with its own external environment:



The combinatorial nature of emergent phenomena can be unpredictably diverse, as well as unlike the component behaviors.

- The resulting "patterns" of recurring larger-scale behavior become the basis for recognition, mathematical laws of motion or other hard science, heuristics, rules of thumb, intuition, prediction, or other exploitation of those regularities.
- Phenomena in the "softer" domains in all instances likewise occur in the presence of cases of the above System Phenomenon, even though the domain-specific phenomena, input-outputs, states, and behaviors are different.



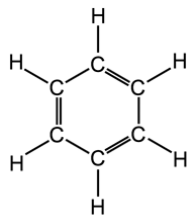


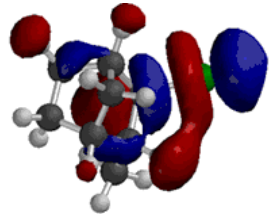


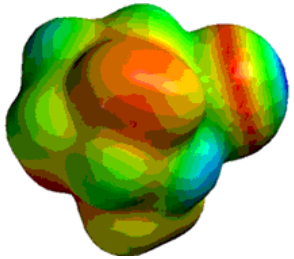

Patterns: The heart of scientific laws, rules of thumb, intuition

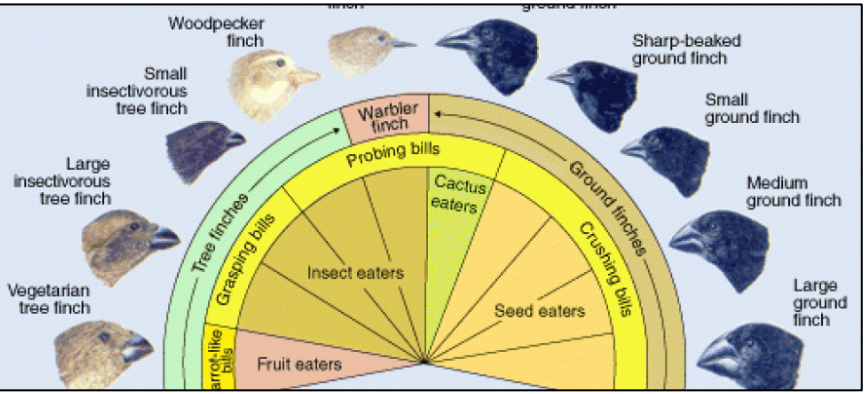
- All “patterns” are recurrences, having both fixed and variable (configurable) aspects.
- The heart of physical science’s life-changing 300 year success in prediction and explanation lies in recognition, representation, exploitation of recurring patterns.
- *Also* at the heart of deep human intuition, expertise, and heuristics.

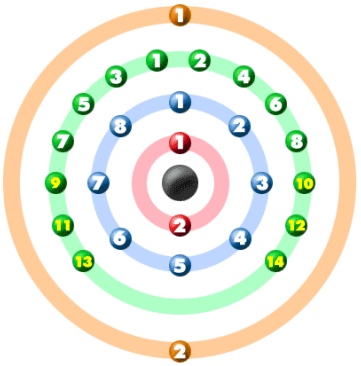




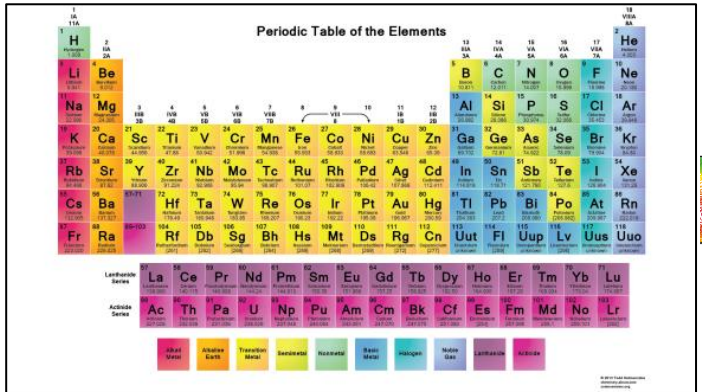


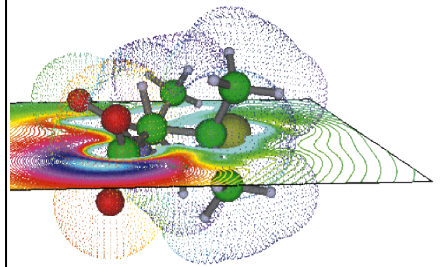






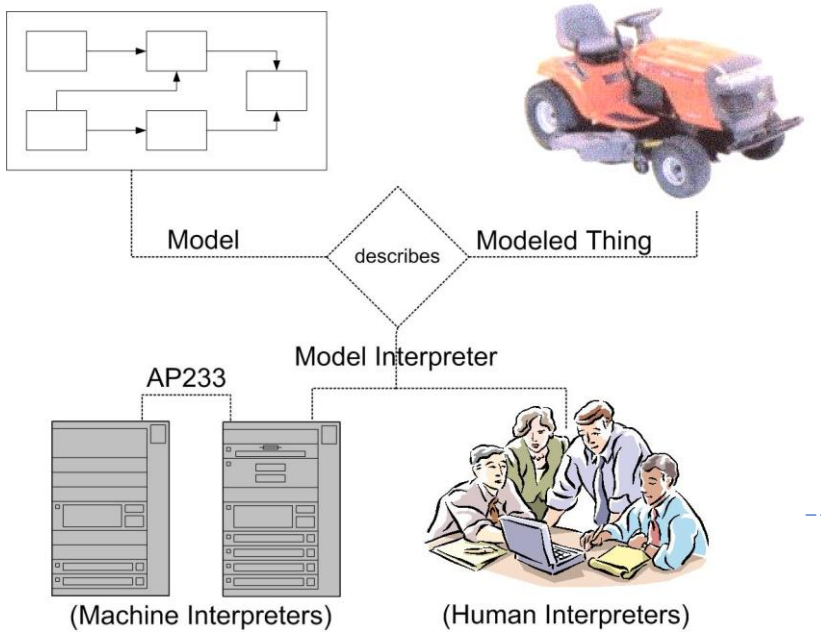


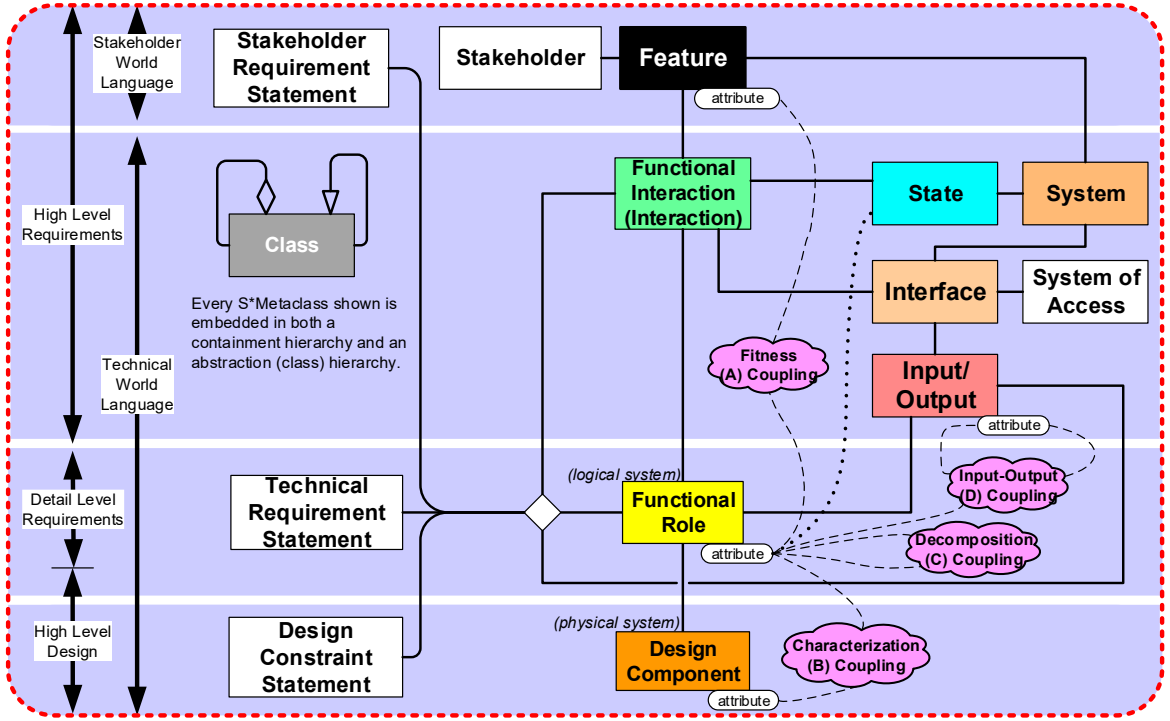




S*Models

- An S*Model is any model (descriptive information construct) *of a system*, in any language, view, or tooling, which conforms to the S*Metamodel:



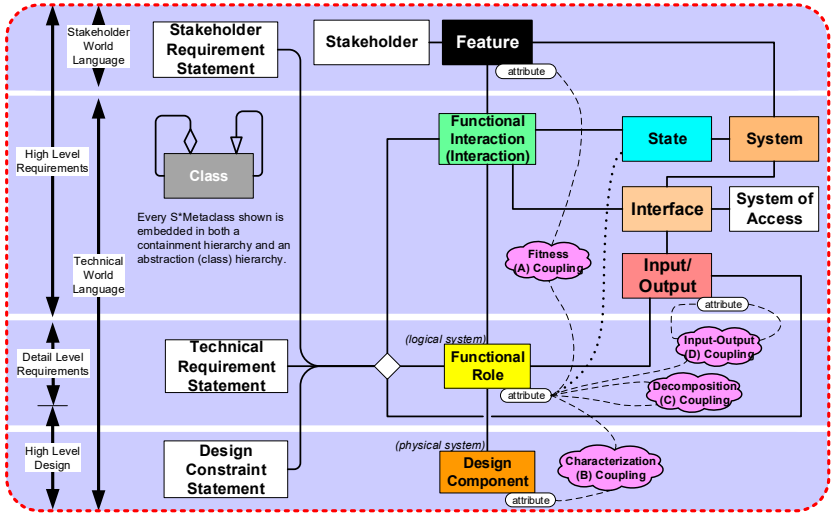


S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

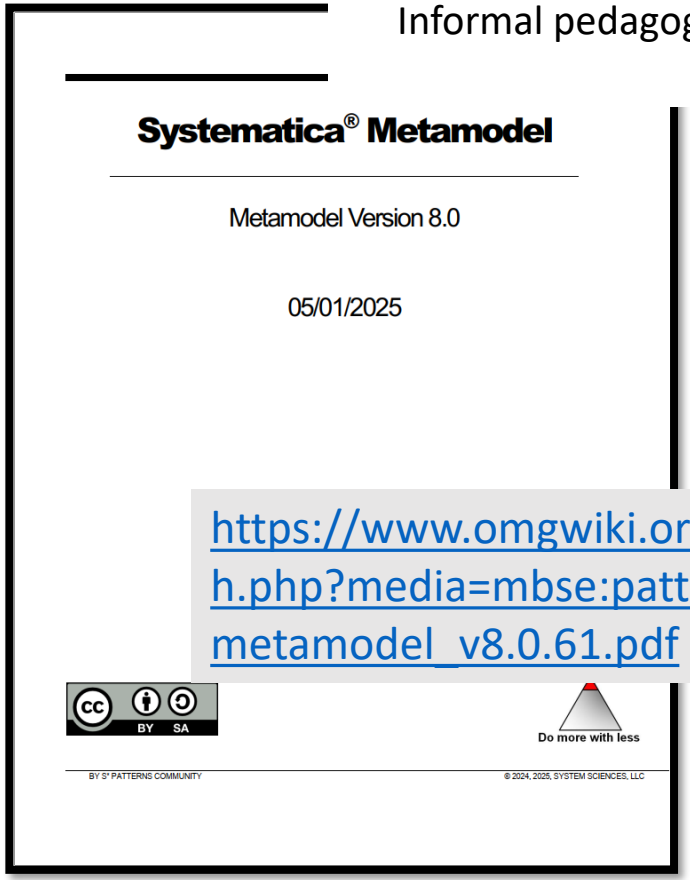
So what is the S*Metamodel, and more important why is it?

S*Metamodel: A reference model of models

- The S*Metamodel is intended to answer:
  - What is the smallest amount of information necessary to describe a system over its life cycle, for the purposes of science and engineering?
- Important because contemporary MBSE models often:
  - Are missing key aspects (are too small)
  - Contain redundant conflicting aspects (are too big)
  - At the same time!
  - We will be discussing prominent examples of both.
- Much of this course will refer to the "informal pedagogical" S*Metamodel diagram above, as a partial intuitive guide.
- Session 4 of this course will look more at the formal S*Metamodel (~100 pages of UML and prose), to understand its mapping to modeling languages like OMG SysML, third party modeling tools, etc.)
- Not an alternative modeling language or tool!



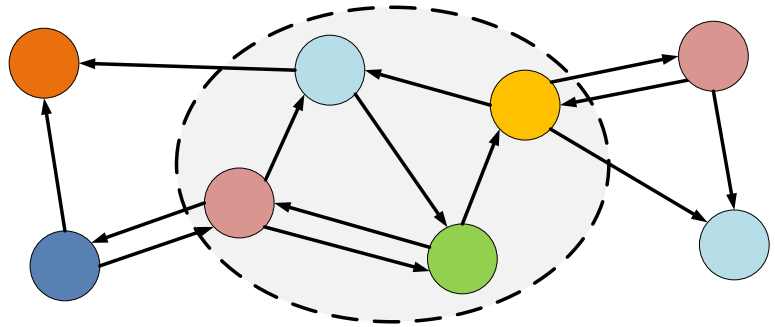
Informal pedagogical S*Metamodel subset diagram

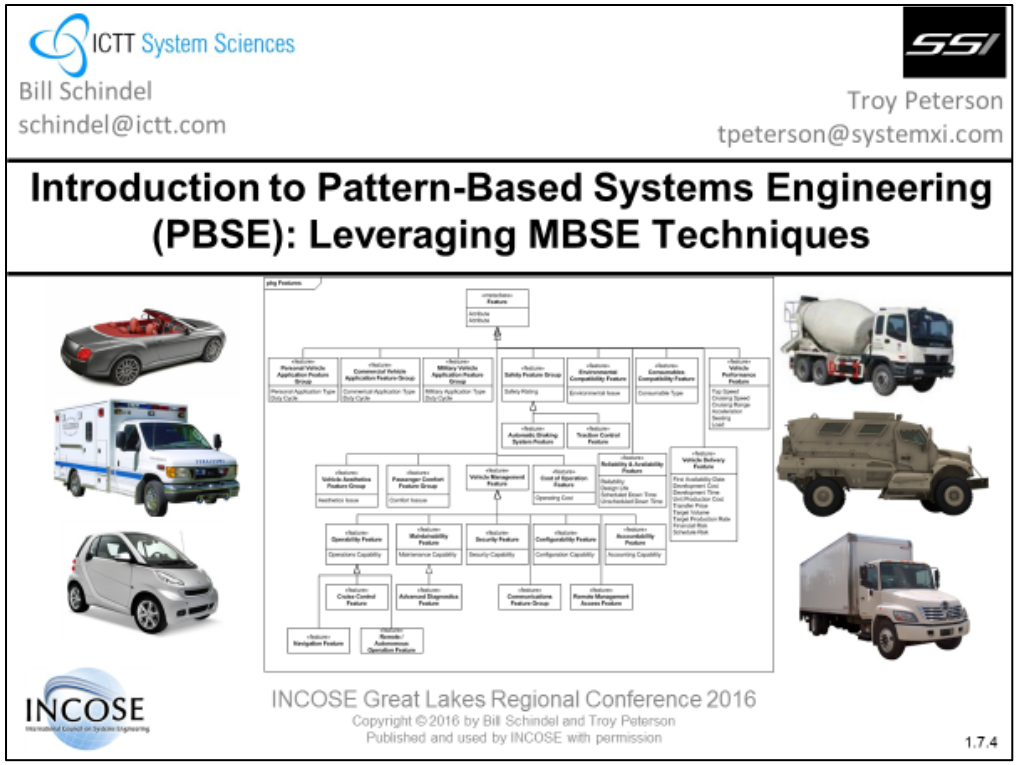


Formal S*Metamodel (~100 pages UML & prose)

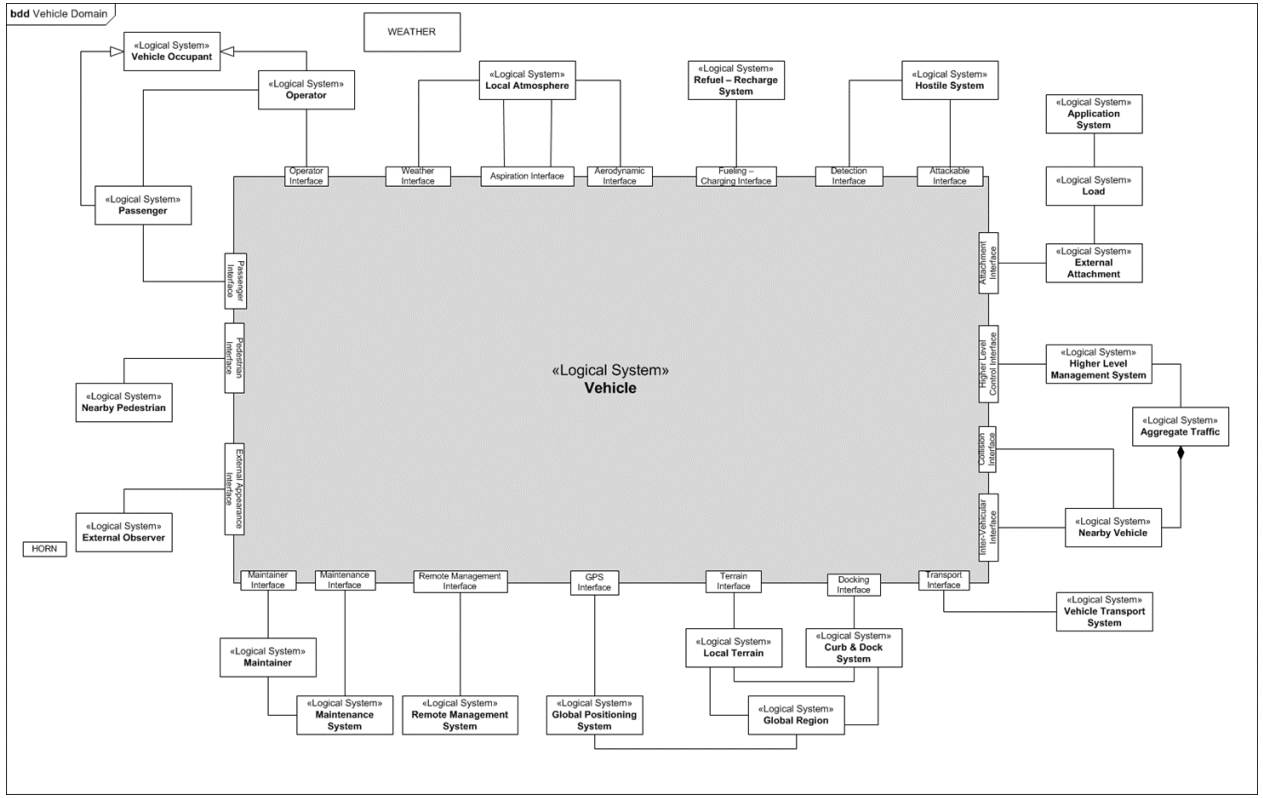
https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:systematica_8_metamodel_v8.0.61.pdf

Systems: Vehicle Examples



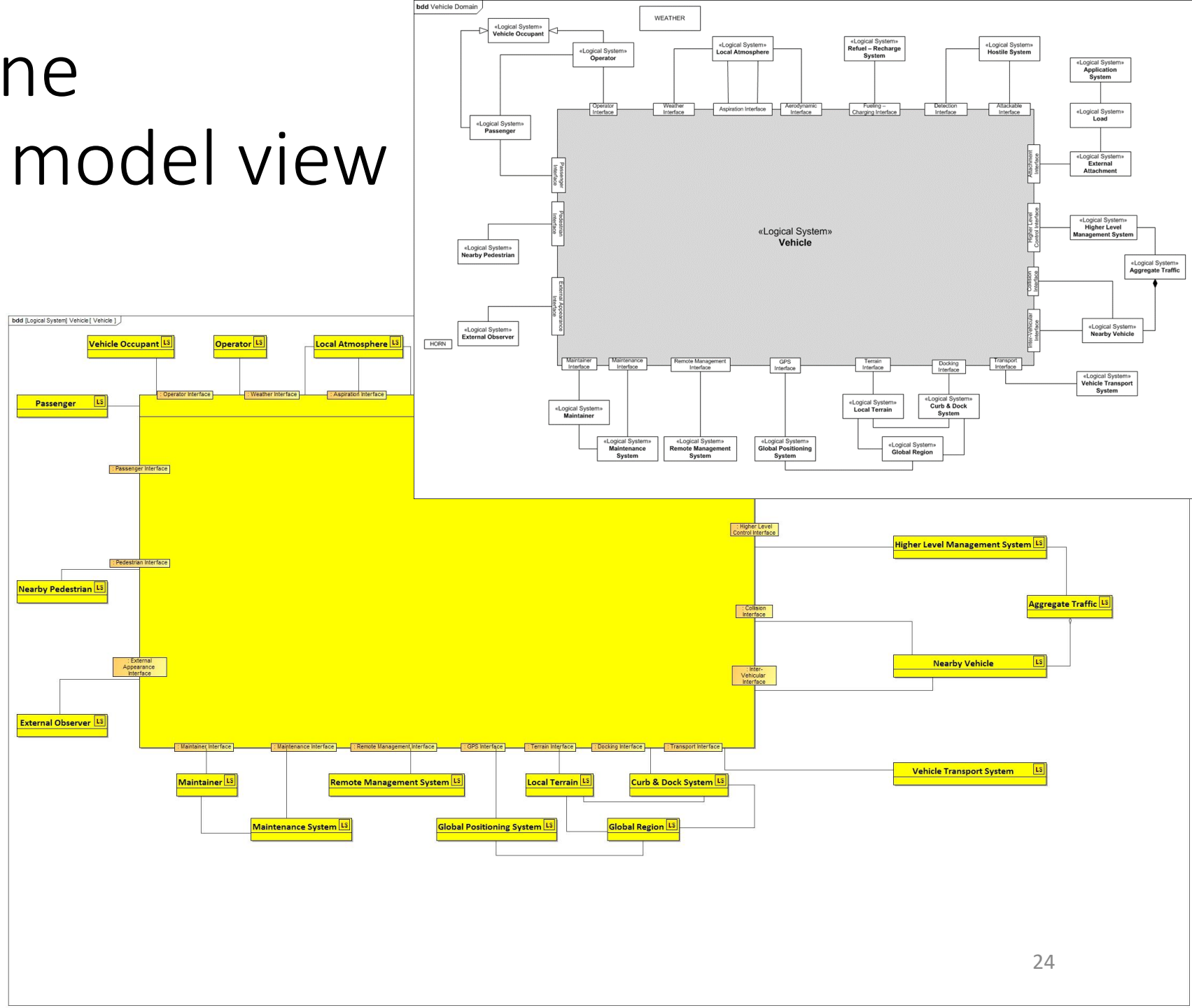


https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:pbse_tutorial_glrc_2016_v1.7.4.pdf

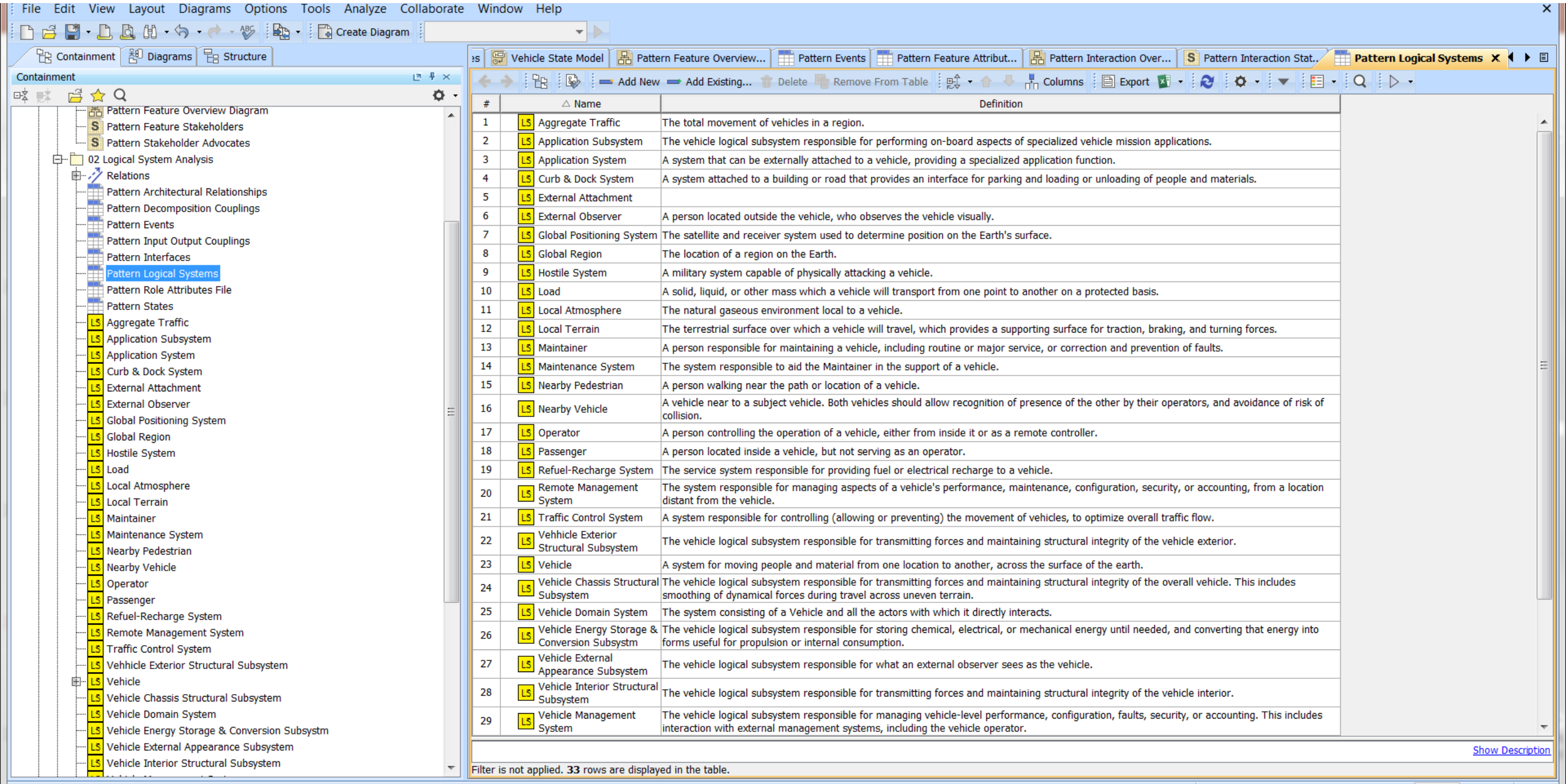


Domain Model: One important system model view

- All the external actors with which a system of interest interacts directly, forming a “Domain System”.
- The (larger) system that is the context of the System of Interest.
- Example Domain Systems:
  - Total life cycle domain
  - Operational or In-Service Domain
  - Maintenance or Sustainment Domain
  - Distribution Domain



Logical Systems

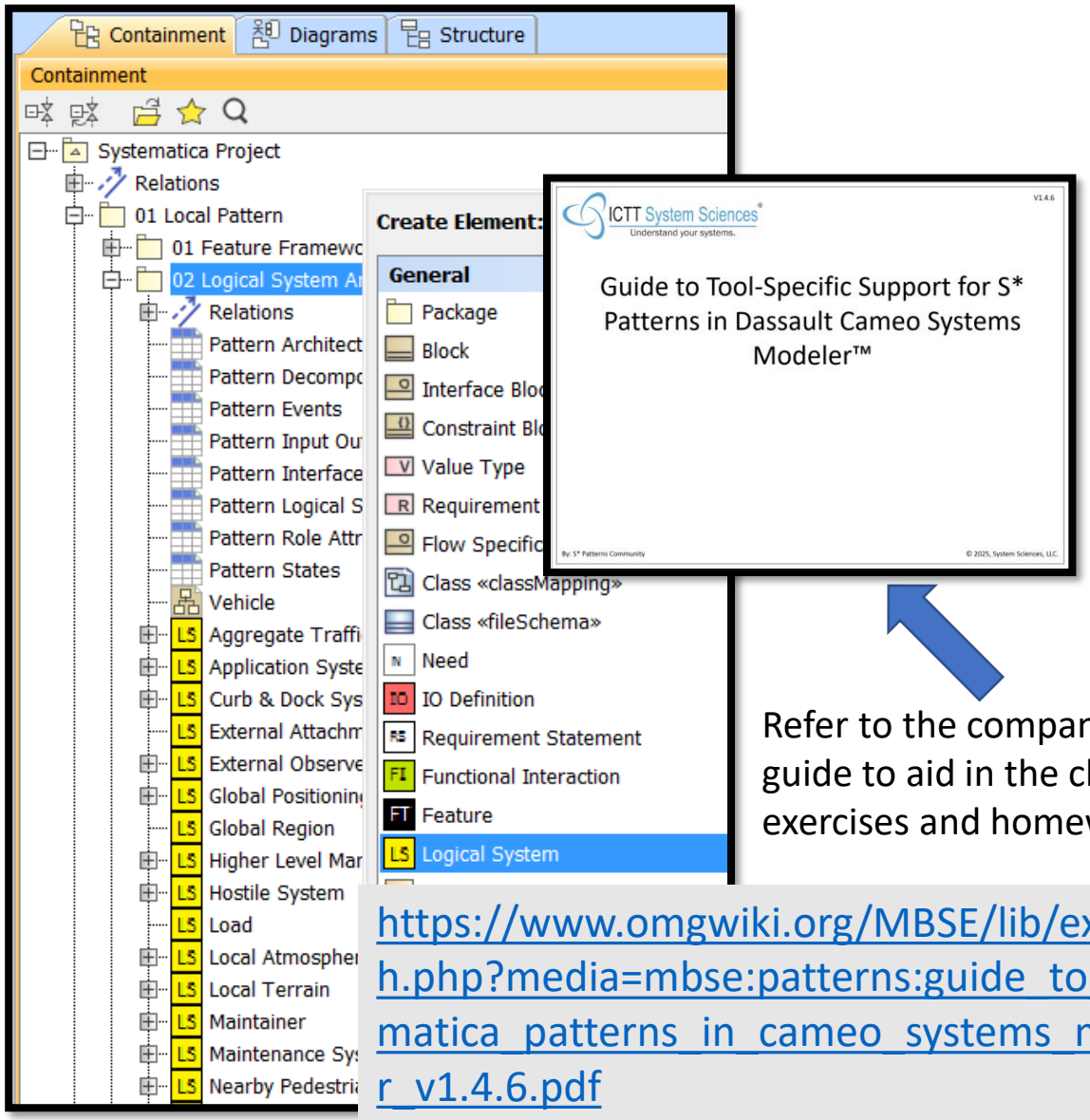


#	Name	Definition
1	Aggregate Traffic	The total movement of vehicles in a region.
2	Application Subsystem	The vehicle logical subsystem responsible for performing on-board aspects of specialized vehicle mission applications.
3	Application System	A system that can be externally attached to a vehicle, providing a specialized application function.
4	Curb & Dock System	A system attached to a building or road that provides an interface for parking and loading or unloading of people and materials.
5	External Attachment	
6	External Observer	A person located outside the vehicle, who observes the vehicle visually.
7	Global Positioning System	The satellite and receiver system used to determine position on the Earth's surface.
8	Global Region	The location of a region on the Earth.
9	Hostile System	A military system capable of physically attacking a vehicle.
10	Load	A solid, liquid, or other mass which a vehicle will transport from one point to another on a protected basis.
11	Local Atmosphere	The natural gaseous environment local to a vehicle.
12	Local Terrain	The terrestrial surface over which a vehicle will travel, which provides a supporting surface for traction, braking, and turning forces.
13	Maintainer	A person responsible for maintaining a vehicle, including routine or major service, or correction and prevention of faults.
14	Maintenance System	The system responsible to aid the Maintainer in the support of a vehicle.
15	Nearby Pedestrian	A person walking near the path or location of a vehicle.
16	Nearby Vehicle	A vehicle near to a subject vehicle. Both vehicles should allow recognition of presence of the other by their operators, and avoidance of risk of collision.
17	Operator	A person controlling the operation of a vehicle, either from inside it or as a remote controller.
18	Passenger	A person located inside a vehicle, but not serving as an operator.
19	Refuel-Recharge System	The service system responsible for providing fuel or electrical recharge to a vehicle.
20	Remote Management System	The system responsible for managing aspects of a vehicle's performance, maintenance, configuration, security, or accounting, from a location distant from the vehicle.
21	Traffic Control System	A system responsible for controlling (allowing or preventing) the movement of vehicles, to optimize overall traffic flow.
22	Vehhicle Exterior Structural Subsystem	The vehicle logical subsystem responsible for transmitting forces and maintaining structural integrity of the vehicle exterior.
23	Vehicle	A system for moving people and material from one location to another, across the surface of the earth.
24	Vehicle Chassis Structural Subsystem	The vehicle logical subsystem responsible for transmitting forces and maintaining structural integrity of the overall vehicle. This includes smoothing of dynamical forces during travel across uneven terrain.
25	Vehicle Domain System	The system consisting of a Vehicle and all the actors with which it directly interacts.
26	Vehicle Energy Storage & Conversion Subsystm	The vehicle logical subsystem responsible for storing chemical, electrical, or mechanical energy until needed, and converting that energy into forms useful for propulsion or internal consumption.
27	Vehicle External Appearance Subsystem	The vehicle logical subsystem responsible for what an external observer sees as the vehicle.
28	Vehicle Interior Structural Subsystem	The vehicle logical subsystem responsible for transmitting forces and maintaining structural integrity of the vehicle interior.
29	Vehicle Management System	The vehicle logical subsystem responsible for managing vehicle-level performance, configuration, faults, security, or accounting. This includes interaction with external management systems, including the vehicle operator.

Simple exercise: Creating Domains and Logical Systems

- Try inserting a Logical System into the Local Pattern part of your Cameo project space.
- Pattern Domains and Logical Systems are created in package 01 Local Pattern::02 Logical System Analysis
  - Right click on that package.
  - Use ~Create Element and choose Logical System
- Name the new system (nouns recommended) and verify that it has been added.

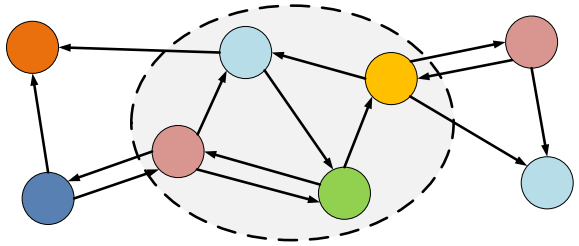
(Later, we'll learn to add Attributes and Relationships.)



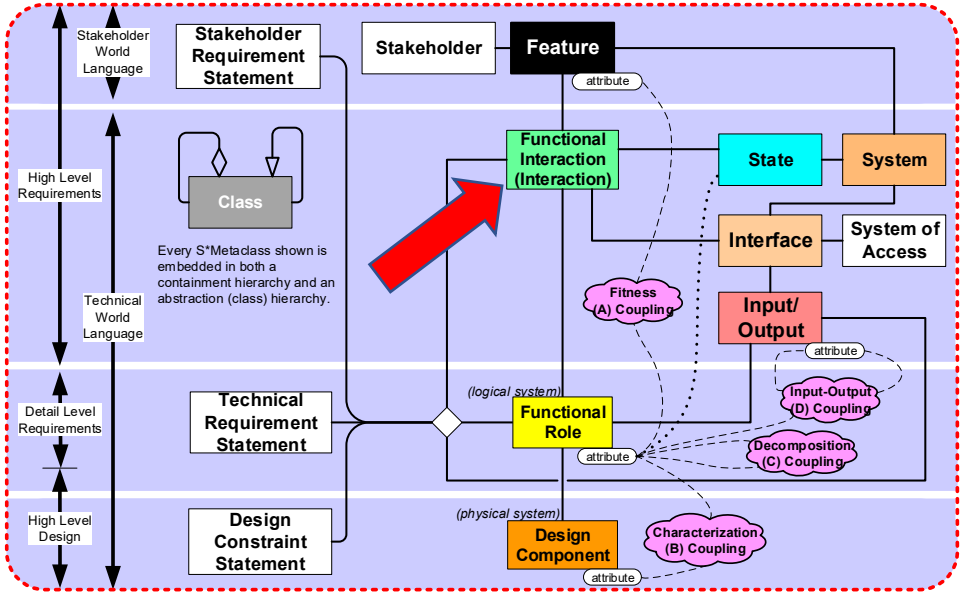
Refer to the companion guide to aid in the class exercises and homework.

https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:guide_to_systematica_patterns_in_cameo_systems_modeler_v1.4.6.pdf

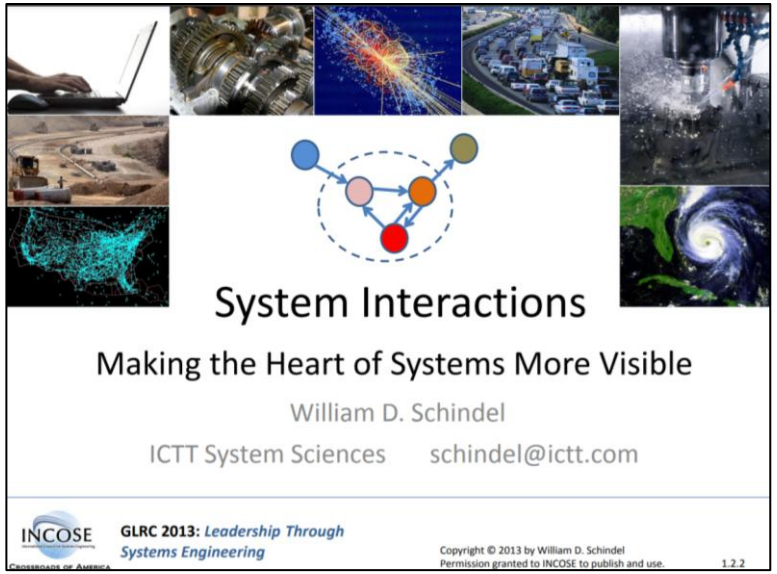
Functional Interactions: Phenomena; clarifying SE views of behavior



- A Functional Interaction (or simply, an Interaction) is an exchange of Input-Outputs (energy, force, material, information) between two or more system components, resulting in component changes of state.
- Two such components might be within a product you are designing—but they also might be that product (viewed as a "black box") and actors in its external environment, in which case the overall system is the Domain System.
- By "state" we mean a property of a component that impacts its input-output behavior during interactions. (Note the circular cause-effect definition chain here.)
- So, a component's "behavior model" describes input-output-state relationships during interaction—*there is no "naked behavior" in the absence of interaction*.
- Interactions are not an important "side issue"—they are at the heart of engineering and science:
  - All the known physical laws of the hard sciences are about or in the context of interactions.
- It will turn out to be very important to identify "all" the interactions—a subject to which we'll return.

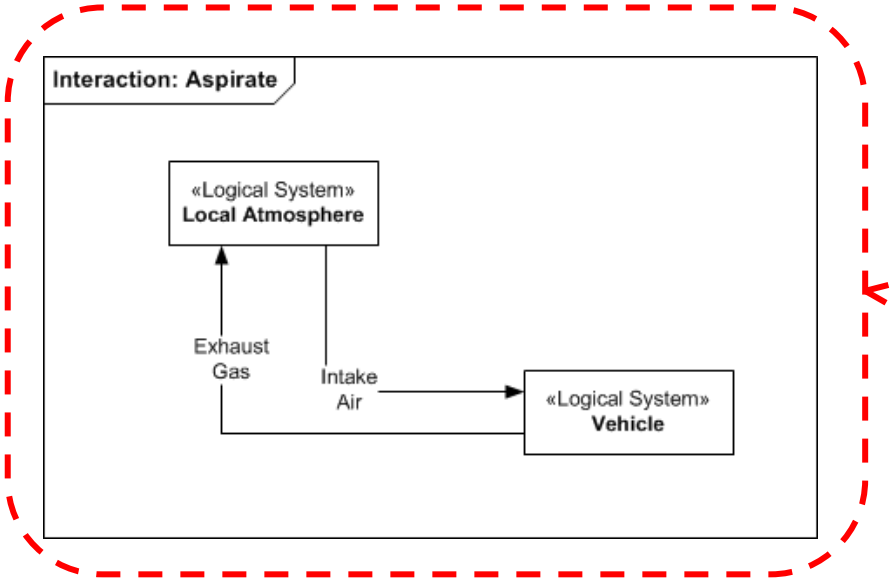


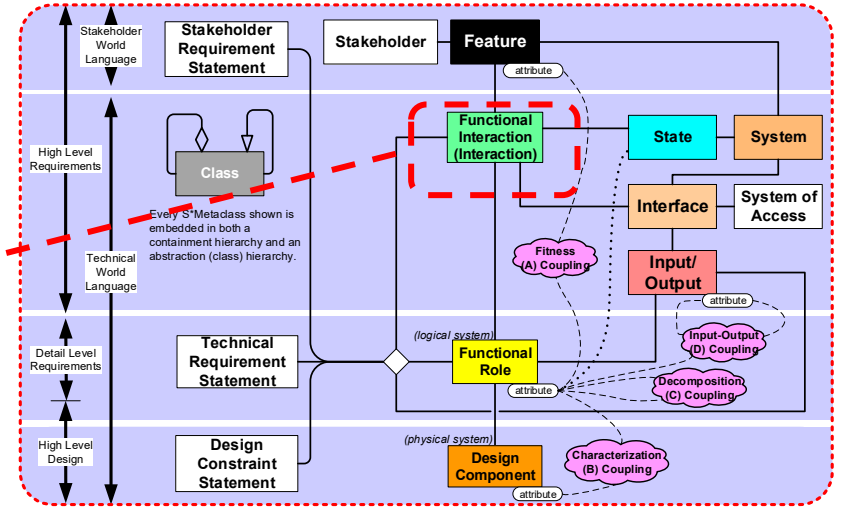
S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)



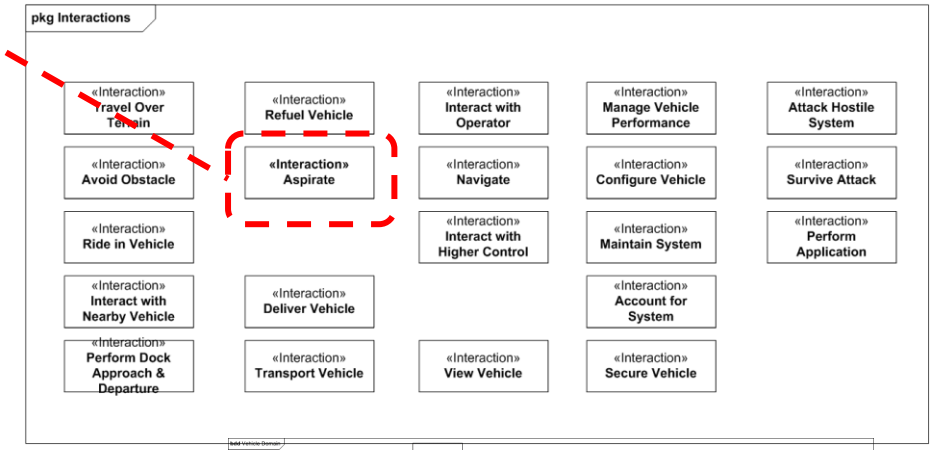
https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:system_interactions--making_the_heart_of_systems_more_visible_v1.2.2.pdf

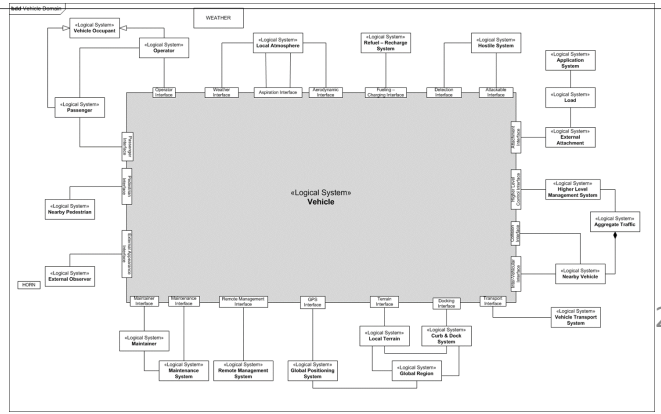
Interactions: Vehicle example





S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)



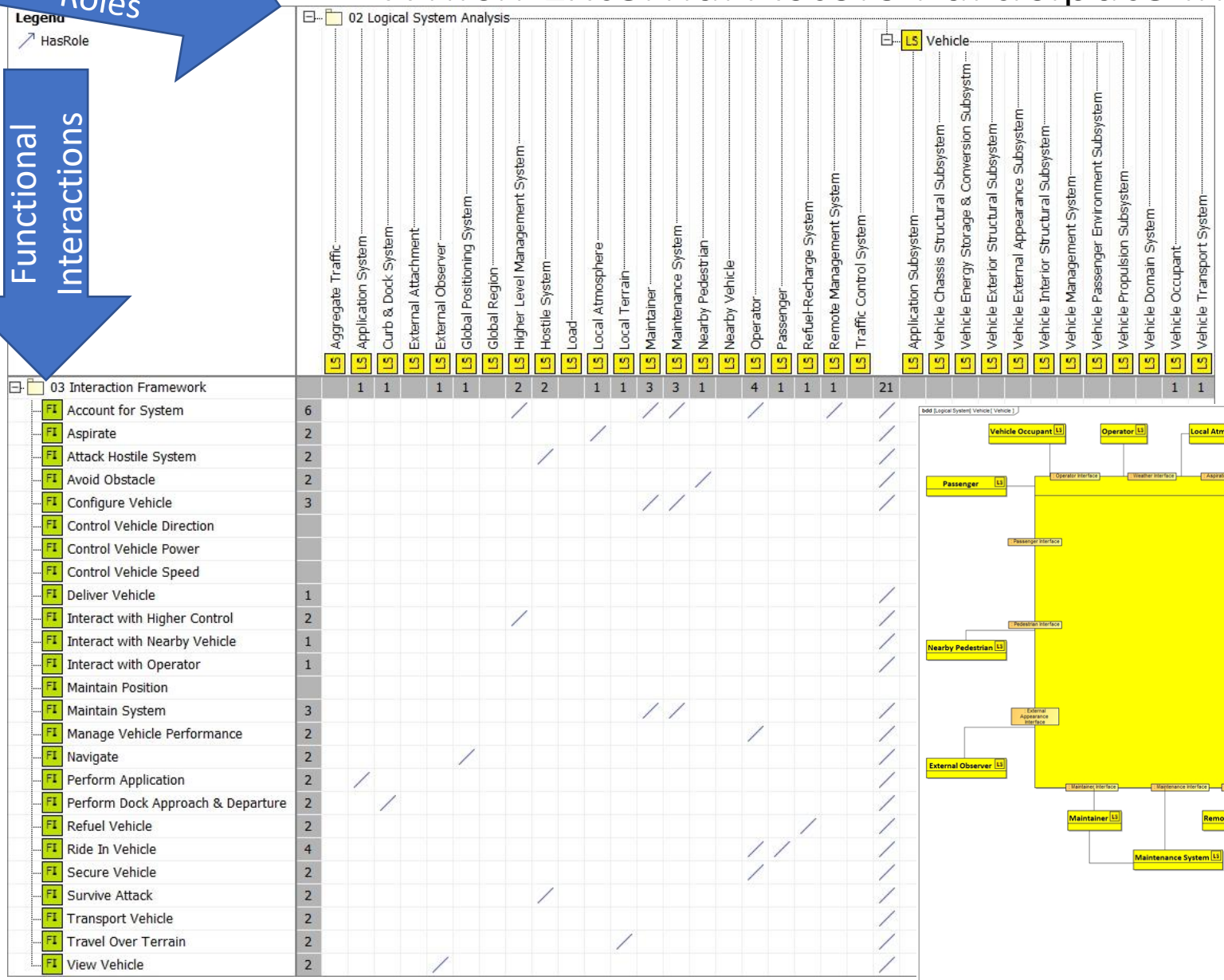


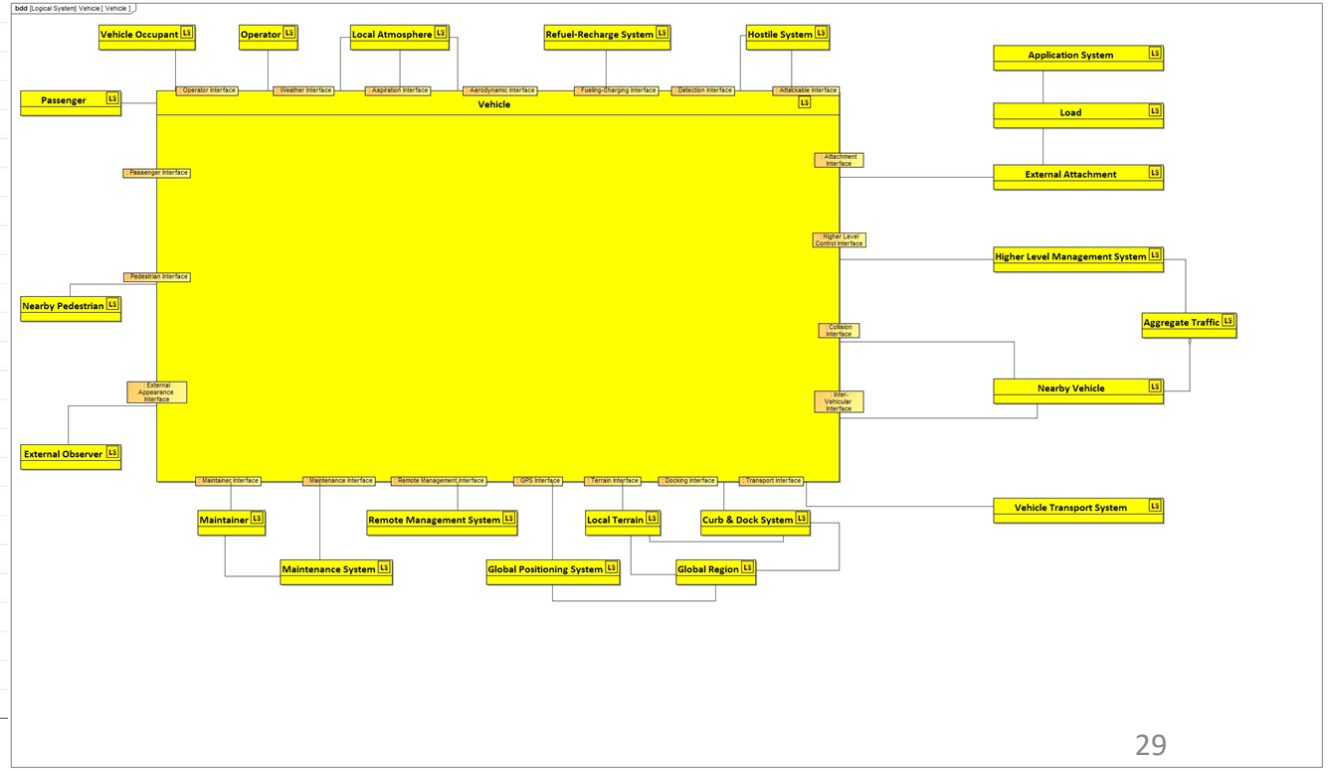
- A key point for systems engineers is not to over-emphasize “my system” as opposed to its interactions with external actors.
- Sometimes engineers object that “I am not responsible for and cannot control those other actors”; however, . . .
- The fact is, the only externally visible behaviors your product will exhibit are its interactions with those external actors.
- The technical requirement specifications for your product are all manifest in its interactions with external actors.
- You do not have to design or control those external actors, but *<u>you do have to understand their behaviors in interaction with your product.</u>*
- Interactions are shown as diverse types of model and tabular diagrams and views: Collaboration Diagrams, Sequence/Timing Diagrams, FFBDs, Free Body Diagrams, etc.

Vehicle Interactions: Which External Actors Participate in Which Interactions?

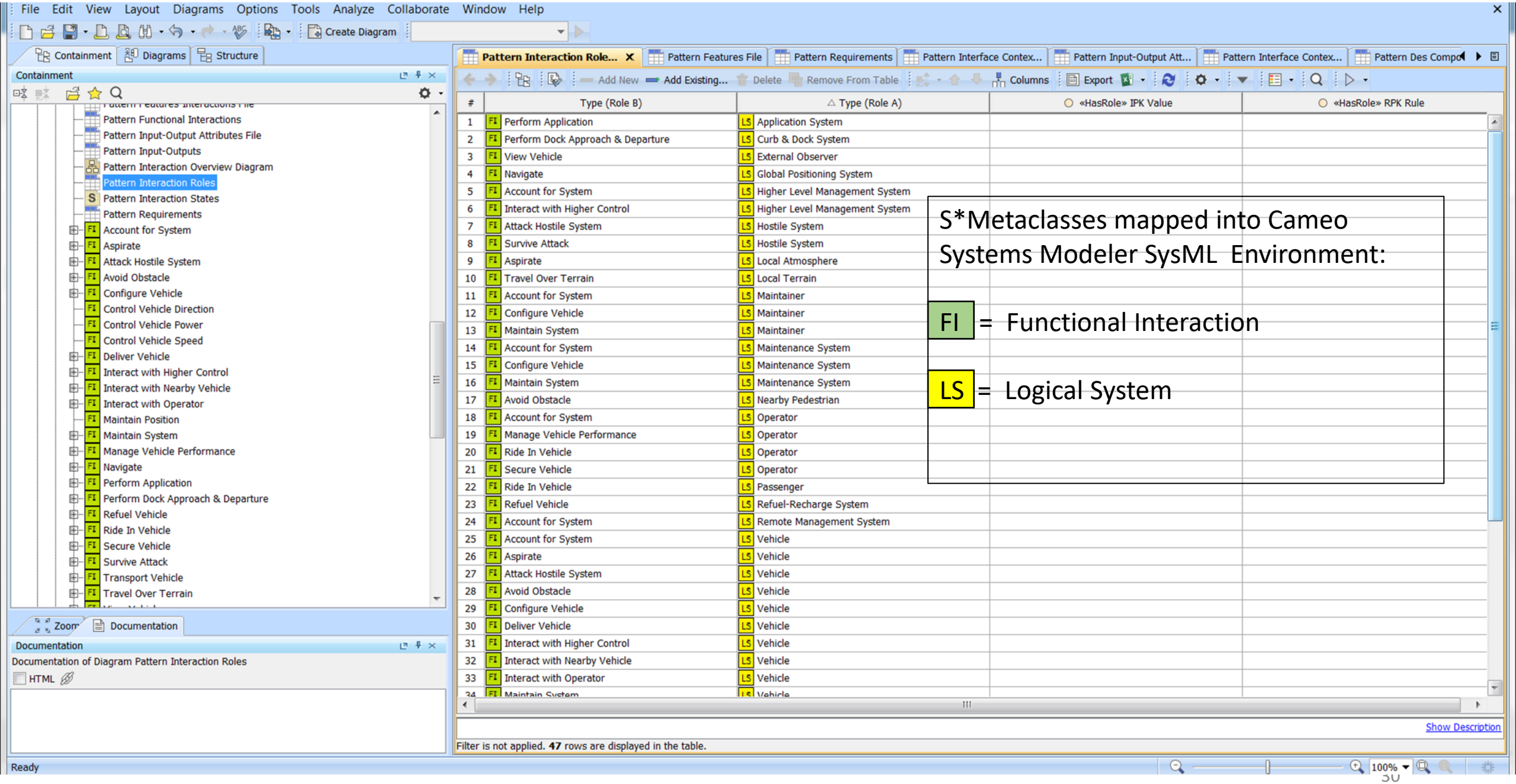
Functional Roles

Functional Interactions





Example checks: Some of these Interactions are missing their external Actors. What should they be?



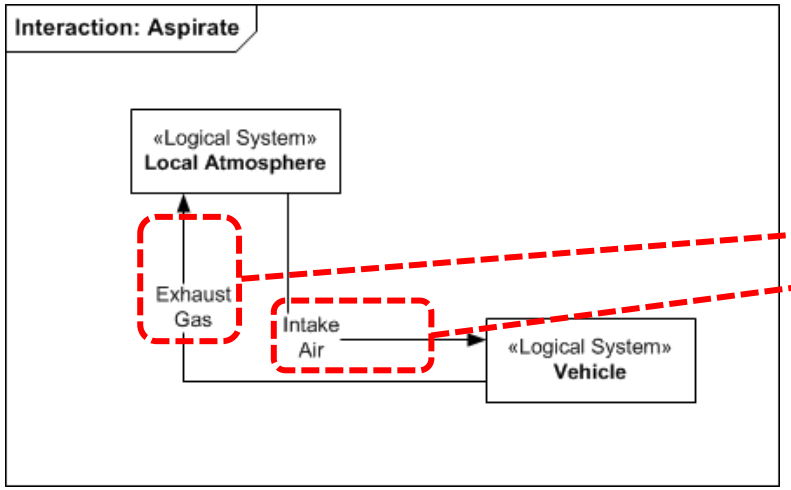
S*Metaclasses mapped into Cameo Systems Modeler SysML Environment:

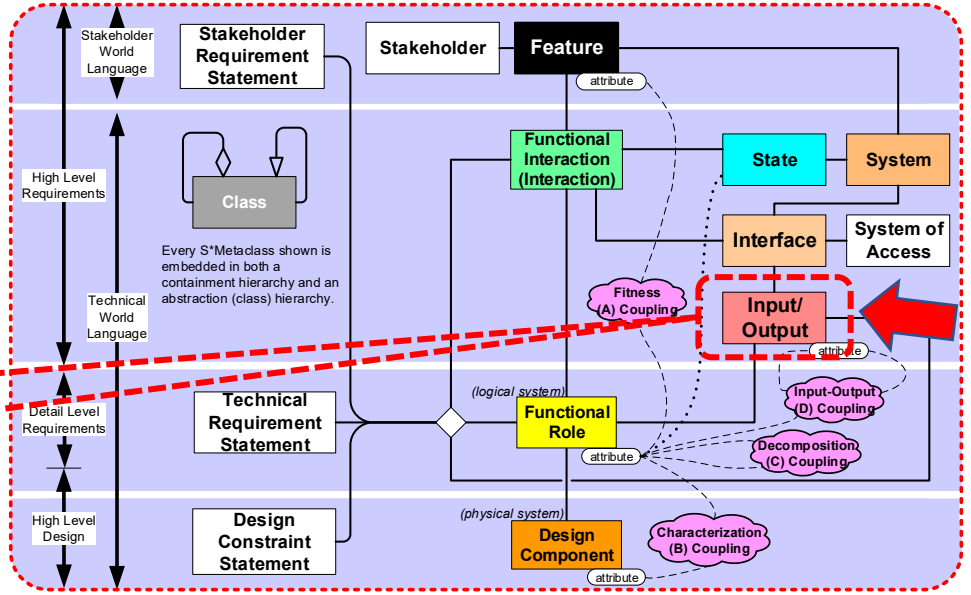
FI = Functional Interaction

LS = Logical System

Input-Outputs (I-Os) are not Interactions

From the Vehicle Model:





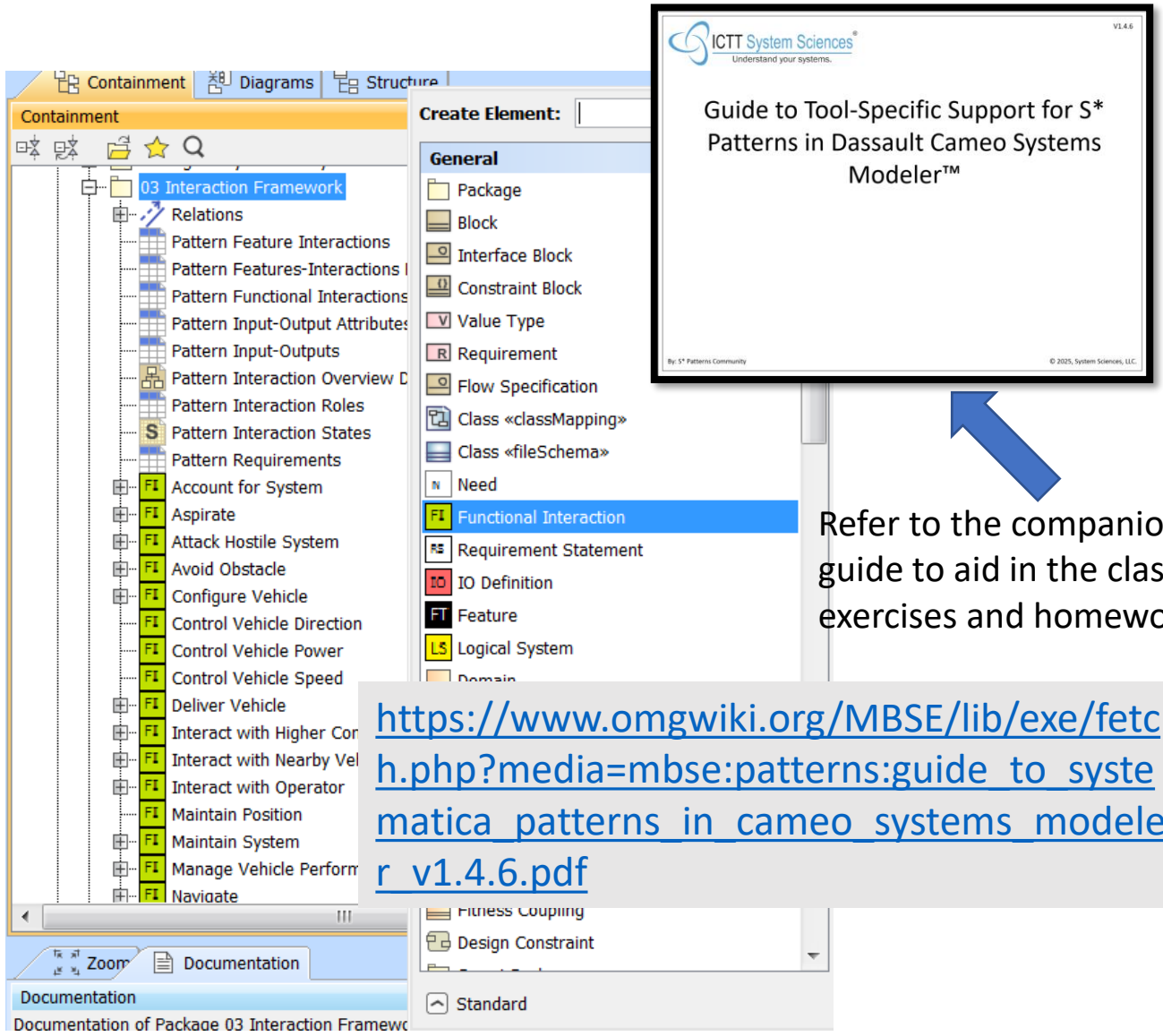
S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

- It helps to keep in mind that I-Os are exchanges (during Interactions), typically of Energy, Force, Material, Information, or some variant of those—think about the Interactions and what they exchange—chemical energy, data, etc.
- The IO is not the Interaction.
- Some IOs are discrete (e.g., messages, manufactured parts, etc.) and some are continuous and possibly varying in time (e.g., energy flows, forces, etc.).
- Many IOs have direction, but don't forget Newton: Forces are bi-directional!
- Some IOs occur not at a "point", but in continuum mechanics or equivalent fields (e.g., electromagnetics, etc.)—if helpful, you can create models that show both the Calculus "differential element" aspects as a "point" model, and show how it connects to the larger continuum model (e.g., Navier-Stokes, Euler-Bernoulli, etc.)
- Don't confuse Input-Outputs with Attributes. Often continuous state variables are attributes of Roles, but you an also include attributes of IOs, such as flow rate, etc. (More on Attributes in Session 2.)

Simple exercise: Creating Functional Interactions

- Try inserting a Functional Interaction into your Cameo project space, in the Local Pattern Functional Interactions Package.
- Pattern Functional Interactions are created in package 01 Local Pattern::03 Interaction Framework
- Right click there and select ~Create Element and choose Functional Interaction
- Name the new interaction (action verb or verb-noun phrase names recommended) and verify it has been added.

(Later, we'll learn to add relationships.)

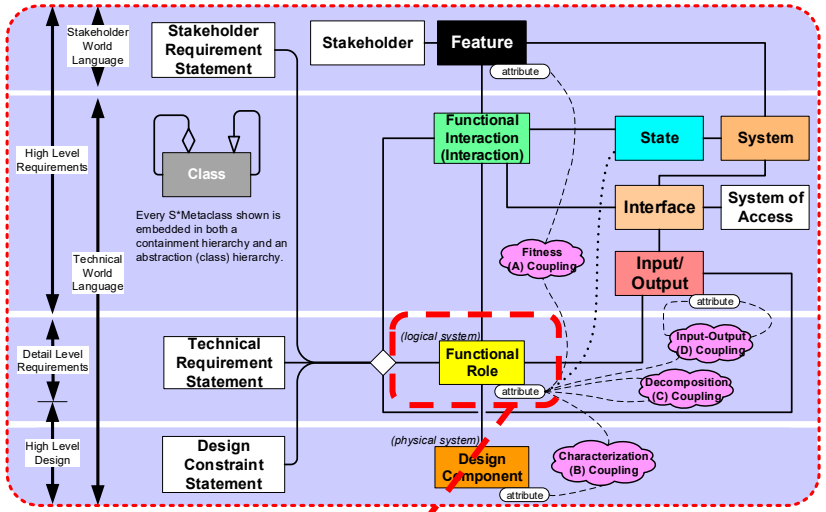


Refer to the companion guide to aid in the class exercises and homework.

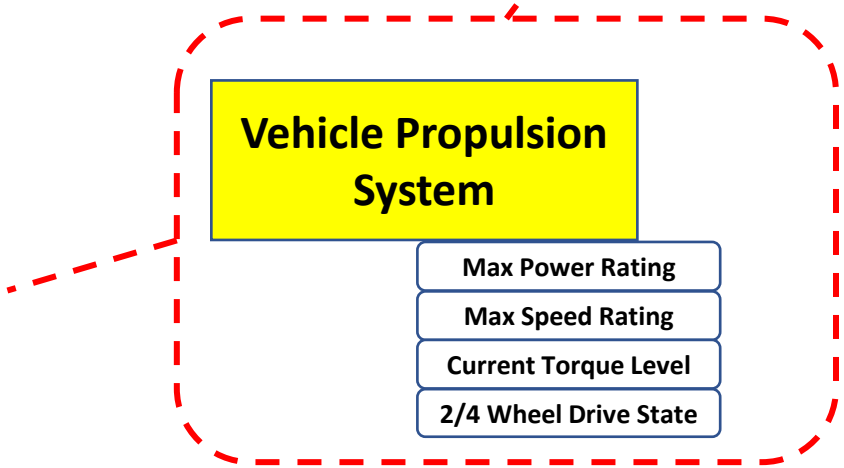
https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:guide_to_systematica_patterns_in_cameo_systems_modeler_v1.4.6.pdf

Functional Roles

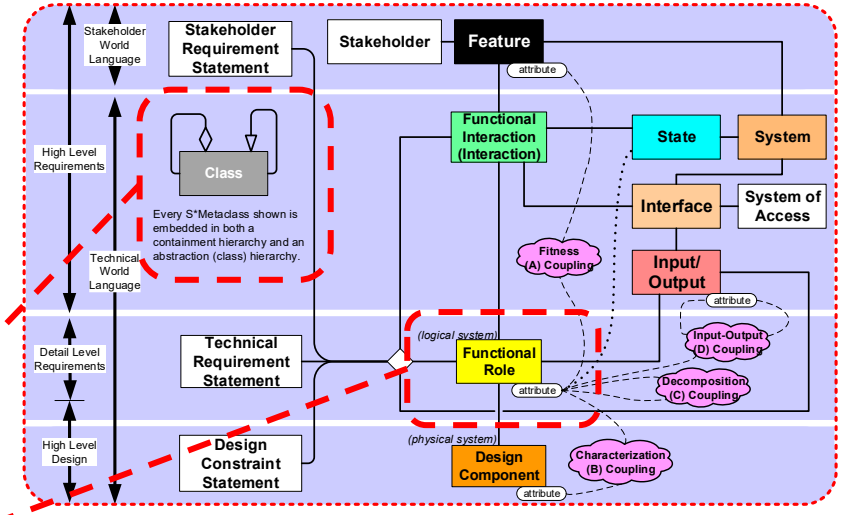
- A Functional Role (or simply Role) is the behavioral part played by a system component in an Interaction.
- It is the input-output behavior seen by the other roles in the interaction.
- Examples: Heat Sink, Control System, Vehicle Passenger
- Every Interaction must have two or more Roles—else it would not be an Interaction!
- A functional role is *described entirely in behavioral terms*.
- These are also called “logical systems” (more on this later).
- Roles (behaviors) may be further parameterized by Role Attributes, which may be static parameters, continuous dynamic state variables, state variables modeled by finite state machines, etc. These variables are further descriptions of aspects of the Behavior that the Role represents. (More on attributes later.)



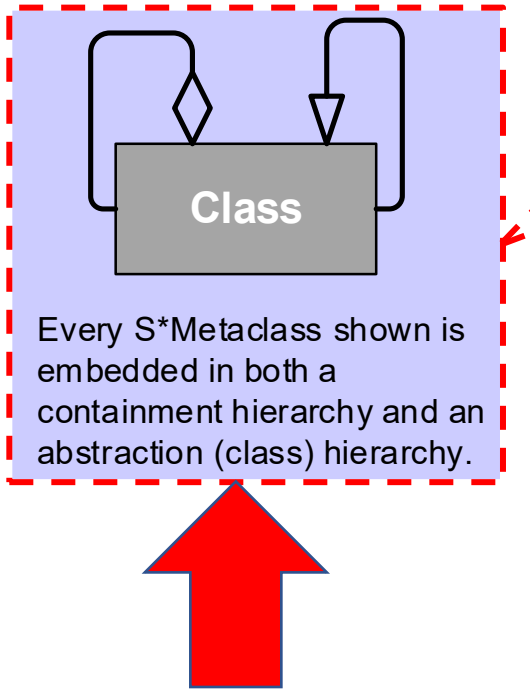
S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)



Dual Hierarchies: There are containment and class hierarchies of logical systems, as well as other classes



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)



<u>Containment (Part-Whole) Hierarchy</u>:

- Vehicle System
- Vehicle Propulsion System
- Braking System
- Brake

Functional Roles (Logical Systems)

Important to traditional engineering decomposition and Bill-of-Materials

<u>Class (General-Special) Hierarchy</u>:

- Vehicle System
- Ambulance Vehicle System
- Military Ambulance Vehicle System
- Mil Ambulance Vehicle, configured for Desert.

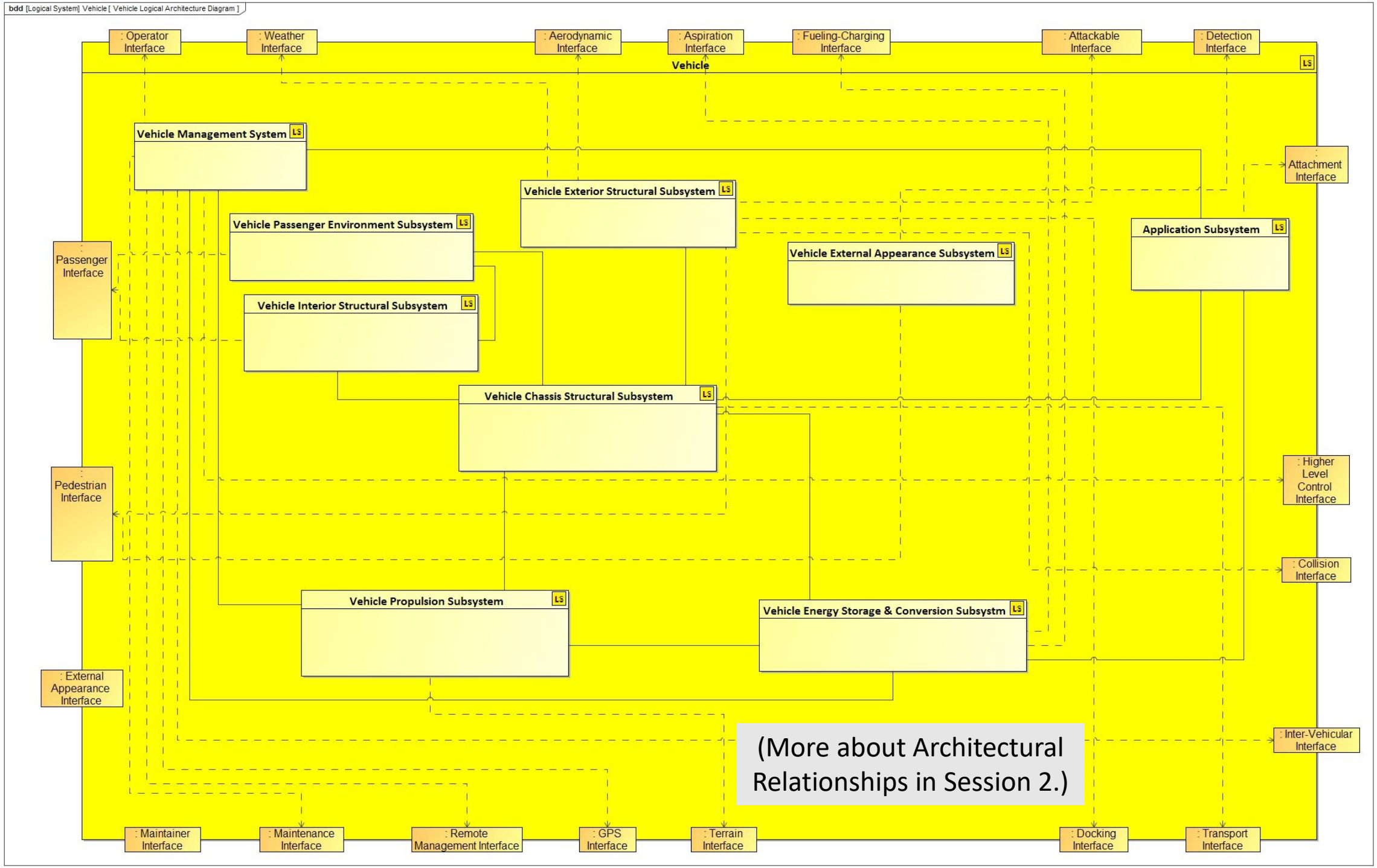
Functional Roles (Logical Systems)

Important to pattern management, product line engineering, economics of re-use

Logical Architecture

- One of the uses of containment hierarchy for roles / logical systems is that we can build up “larger” logical systems (behaviors) that contain (interconnected, interacting) “smaller” logical systems behaviors).
- At any given level of such a hierarchy (for example, the whole vehicle level), we can view the interacting smaller logical systems one level down, understanding the “logical architecture” described by the relationships summarizing their interactions:

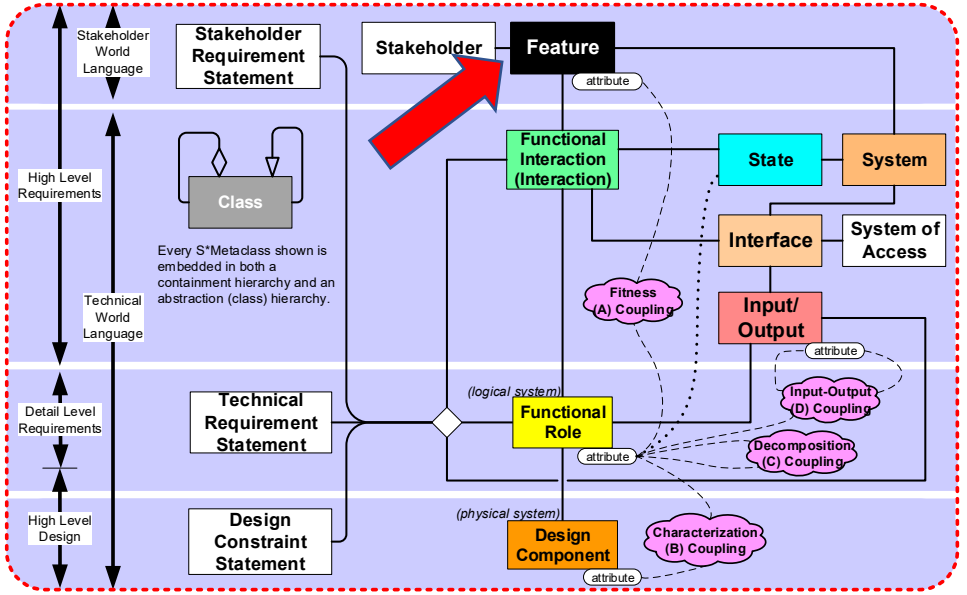



bdd [Logical System] Vehicle [Vehicle Logical Architecture Diagram]
: Operator Interface
: Weather Interface
: Aerodynamic Interface
: Aspiration Interface
: Fueling-Charging Interface
: Attackable Interface
: Detection Interface
Vehicle
LS
Vehicle Management System
: Attachment Interface
Vehicle Exterior Structural Subsystem
Vehicle Passenger Environment Subsystem
Application Subsystem
: Passenger Interface
Vehicle External Appearance Subsystem
Vehicle Interior Structural Subsystem
Vehicle Chassis Structural Subsystem
: Pedestrian Interface
: Higher Level Control Interface
: Collision Interface
Vehicle Propulsion Subsystem
Vehicle Energy Storage & Conversion Subsystm
: External Appearance Interface
: Inter-Vehicular Interface
(More about Architectural Relationships in Session 2.)
: Maintainer Interface
: Maintenance Interface
: Remote Management Interface
: GPS Interface
: Terrain Interface
: Docking Interface
: Transport Interface

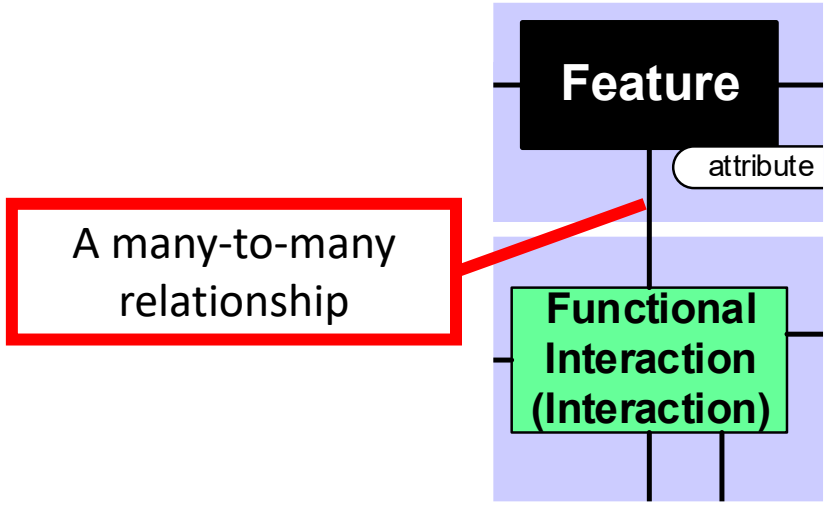
BREAK

Stakeholder Features; clarifying SE views of value, selection, risk, FMEA, configuration

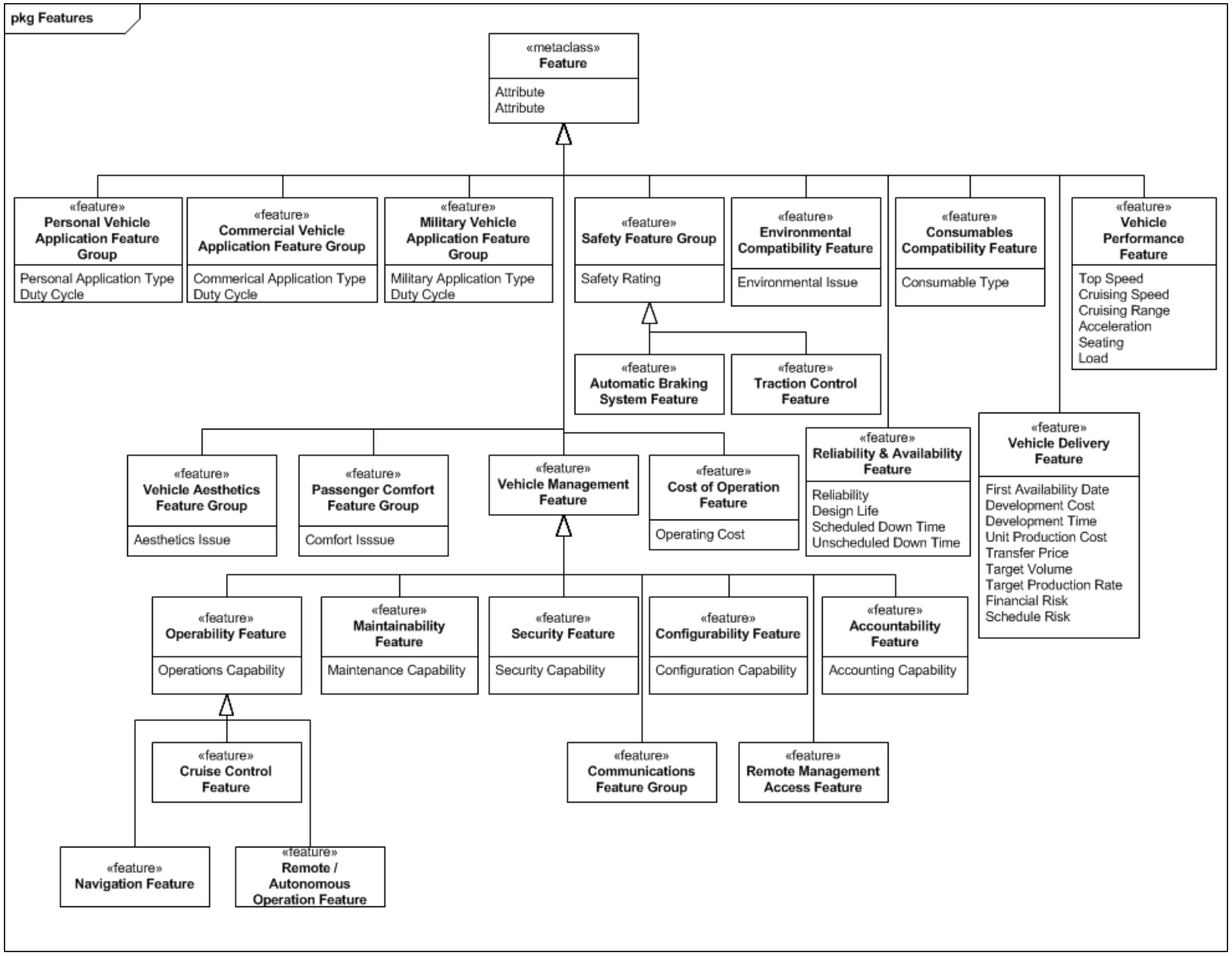
- Stakeholder Features model, in the language and conceptual values framework of the respective Stakeholders, chunks of value:
  - what is “at stake”
  - Often may be quite subjective
- Notice that we are describing twice the external behavior exhibited by the system of interest:
  - Interactions (and the Technical Requirements that will go with them) describe what is wanted in objective testable terms common to engineers.
  - Features describe the same system, but in terms of what is valued, Measures of Effectiveness (MOEs), etc.
- Analogous to pre-model engineering practice of “Customer Requirements” and “Technical Requirements” (other terms also used included “Product Requirements”, “System Requirements”, etc.)
- Two different ontologies, in a many-to-many mesh!



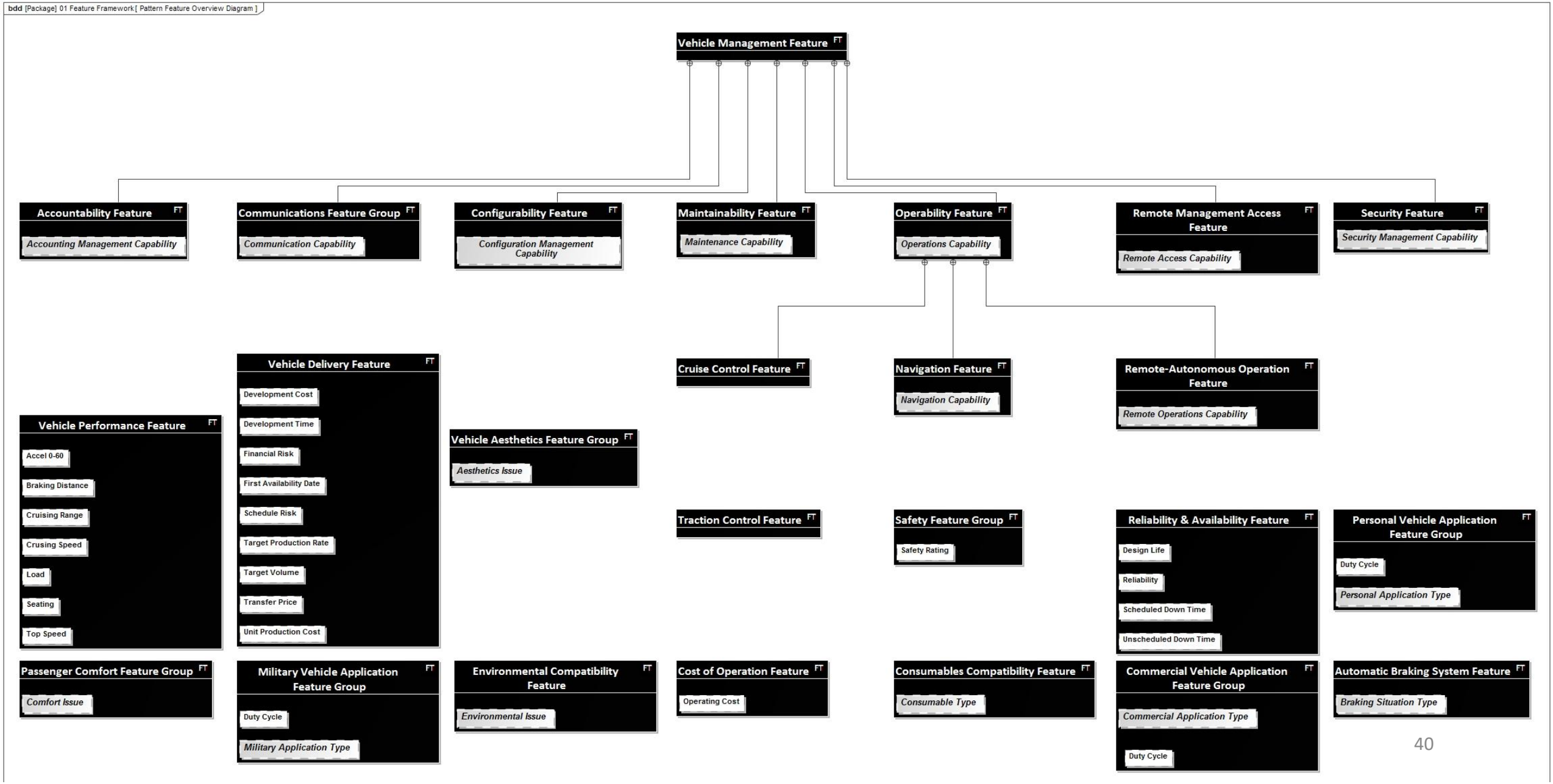
S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)



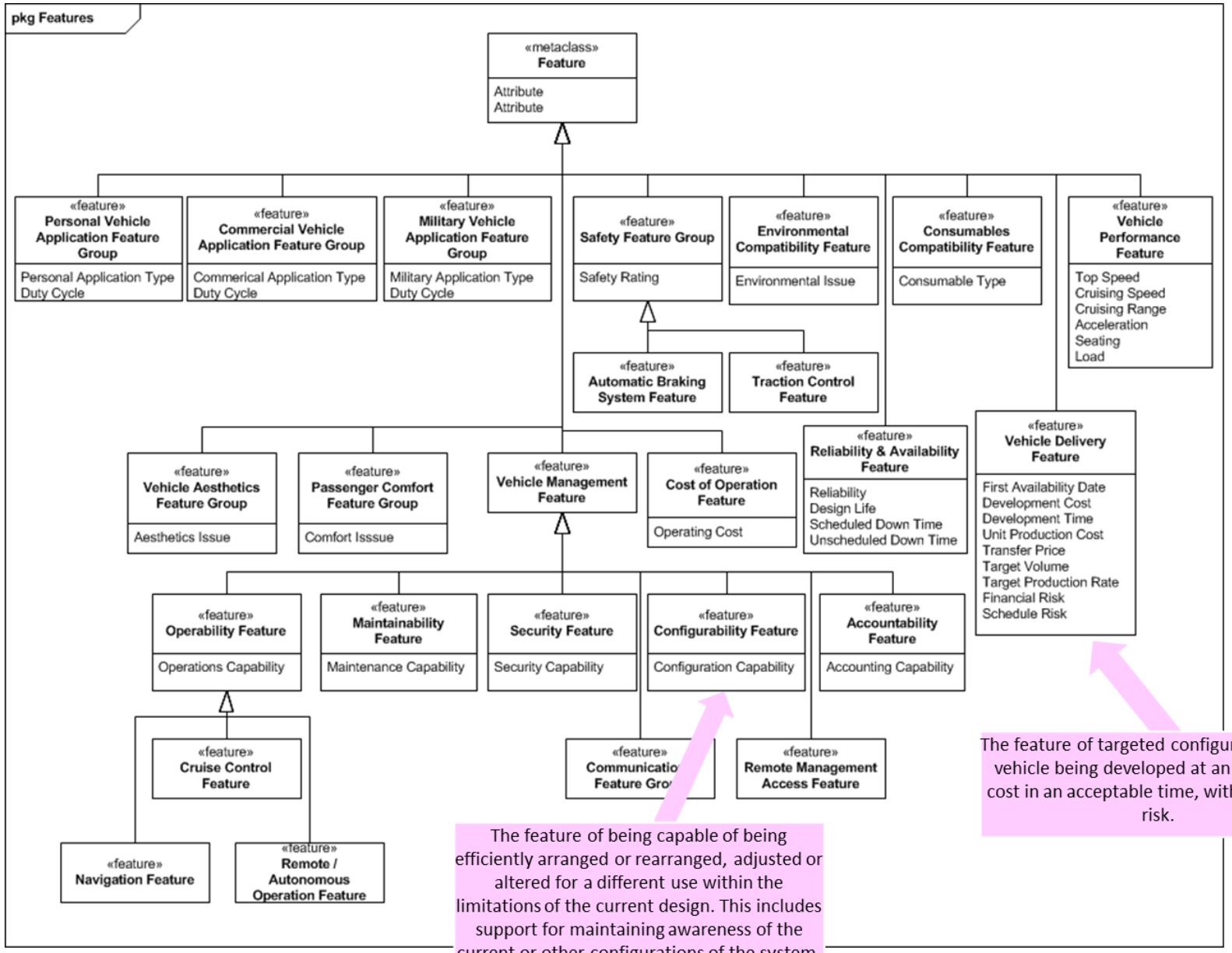
Stakeholder Features: Vehicle example



Stakeholder Features: Vehicle example



Stakeholder Features: Vehicle example



Feature Attributes, in Cameo SysML

(more about Feature Attributes and their Couplings in Session 2)

#	Owner	Name	Definition
1	Accountability Feature	Accounting Management Capability	An internal management control capability (active or passive) provided by the system, in the area of accounting for system utilization. More than one capability may be instantiated.
2	Automatic Braking System Feature	Braking Situation Type	The class of braking situations for which the system can provide automatic braking. Multiple types can be populated.
3	Commercial Vehicle Application Feature Group	Commercial Application Type	The type of commercial application(s) the vehicle is configured to perform. More than one value may be populated.
4	Commercial Vehicle Application Feature Group	Duty Cycle	The percent of time during the system's service life that it is to be in operation.
5	Communications Feature Group	Communication Capability	The class of communications provided by the system. Multiple types can be populated.
6	Configurability Feature	Configuration Management Capability	An internal management control capability (active or passive) provided by the system, in the area of management of system configuration. More than one capability may be instantiated.
7	Consumables Compatibility Feature	Consumable Type	The type of supply item that the system consumes. More than one type may be populated.
8	Cost of Operation Feature	Operating Cost	The cost of operating the vehicle in its intended use.
9	Environmental Compatibility Feature	Environmental Issue	The type of risk to the natural environment that the system potentially represents. More than one may be populated.
10	Maintainability Feature	Maintenance Capability	An internal management control capability (active or passive) provided by the system, in the area of management (prevention or correction) of system faults. More than one capability may be instantiated.
11	Military Vehicle Application Feature Group	Duty Cycle	The percent of time during the system's service life that it is to be in operation.
12	Military Vehicle Application Feature Group	Military Application Type	The type of military application(s) the vehicle is configured to perform. More than one value may be populated.
13	Navigation Feature	Navigation Capability	A type of automatic navigation capability. More than one type may be instantiated.
14	Operability Feature	Operations Capability	An internal management control capability (active or passive) provided by the system, in the area of management of its main purpose performance outcomes. More than one capability may be instantiated.
15	Passenger Comfort Feature Group	Comfort Issue	The type of passenger comfort issue that the system may be configured to address. More than one value may be populated.
16	Personal Vehicle Application Feature Group	Duty Cycle	The percent of time during the system's service life that it is to be in operation.
17	Personal Vehicle Application Feature Group	Personal Application Type	The type of personal application(s) the vehicle is configured to perform. More than one value may be populated.
18	Reliability & Availability Feature	Design Life	The length of time over which the system should be available for intended use, if properly maintained.
19	Reliability & Availability Feature	Reliability	The likelihood that the system will operate without fault during its operational service life.
20	Reliability & Availability Feature	Scheduled Down Time	The amount of time that the system can be unavailable, if scheduled in advance.
21	Reliability & Availability Feature	Unscheduled Down Time	The amount of time that the system can be unavailable, if not scheduled in advance.
22	Remote Management Access Feature	Remote Access Capability	
23	Remote-Autonomous Operation Feature	Remote Operations Capability	
24	Safety Feature Group	Safety Rating	The rating granted to vehicles of this type by an authoritative agency, concerning its ability to be operated without undue risk to people or property.
25	Security Feature	Security Management Capability	An internal management control capability (active or passive) provided by the system, in the area of management of security of the system. More than one capability may be instantiated.
26	Vehicle Aesthetics Feature Group	Aesthetics Issue	The type of issue(s) that the configured system is to address, in the area of subjective experience of a person in the presence of the system, as observer, passenger, or operator. More than one issue may be populated.
27	Vehicle Delivery Feature	Development Cost	The total cost of development of the configured version of the vehicle, including R&D and production tooling costs.
28	Vehicle Delivery Feature	Development Time	The total time interval within which the development of the configuration version of the vehicle must be completed, including production tooling installation and qualification.
29	Vehicle Delivery Feature	Financial Risk	The expected (probability weighted) amount of additional cost over target that the system is likely to require to deliver.
30	Vehicle Delivery Feature	First Availability Date	The targeted introduction date by which the configured version of the system must be available for production delivery.
31	Vehicle Delivery Feature	Schedule Risk	The expected (probability weighted) amount of additional time over target schedule that the system is likely to require to deliver.
32	Vehicle Delivery Feature	Target Production Rate	The quantity of the vehicle which is targeted to be produced and sold annually, after production ramp-up.
33	Vehicle Delivery Feature	Target Volume	The quantity of the vehicle which is targeted to be produced and sold, over the lifetime of the product configuration.
34	Vehicle Delivery Feature	Transfer Price	The targeted price to be received by the manufacturer from the distribution channel.
35	Vehicle Delivery Feature	Unit Production Cost	The targeted cost to manufacture a single unit of the configured vehicle, including direct and indirect costs, at the targted production volume and rate.
36	Vehicle Performance Feature	Accel 0-60	The amount of time in which the vehicle must be capable of accelerating from 0 to 60 MPH.
37	Vehicle Performance Feature	Braking Distance	The distance in which the vehicle must be capable of stopping from 60 MPH.
38	Vehicle Performance Feature	Cruising Range	The minimum distance that the vehicle must be capable of traveling on one tank of fuel over level terrain in nominal conditions.
39	Vehicle Performance Feature	Crusing Speed	The average speed at which the vehicle must be capable of traveling on a sustained basis over long distances.
40	Vehicle Performance Feature	Load	The amount of weight that the vehicle is capable of transporting, not including people.
41	Vehicle Performance Feature	Seating	The number of people that the vehicle is capable of transporting over long distances in the intended seating or standing pattern.
42	Vehicle Performance Feature	Top Speed	The highest speed at which the vehicle must be capable of traveling under its own power on level terrain.

Feature Attributes, in Cameo SysML

"FPK" attributes are Feature Primary Keys indicating multiple instantiation configuration

				qualification.	
16	Commercial Vehicle Application Feature Group	Duty Cycle	Feature Attribute [Property]	The percent of time during the system's service life that it is to be in operation.	
17	Military Vehicle Application Feature Group	Duty Cycle	Feature Attribute [Property]	The percent of time during the system's service life that it is to be in operation.	
18	Personal Vehicle Application Feature Group	Duty Cycle	Feature Attribute [Property]	The percent of time during the system's service life that it is to be in operation.	
19	Environmental Compatibility Feature	Environmental Issue	Feature Primary Key [Property]	The type of risk to the natural environment that the system potentially represents. More than one may be populated.	Carbon Dioxide Emissions: Solid Waste:
20	Vehicle Delivery Feature	Financial Risk	Feature Attribute [Property]	The expected (probability weighted) amount of additional cost over target that the system is likely to require to deliver.	
21	Vehicle Delivery Feature	First Availability Date	Feature Attribute [Property]	The targeted introduction date by which the configured version of the system must be available for production delivery.	
22	Vehicle Performance Feature	Load	Feature Attribute [Property]	The amount of weight that the vehicle is capable of transporting, not including people.	
23	Maintainability Feature	Maintenance Capability	Feature Primary Key [Property]	An internal management control capability (active or passive) provided by the system, in the area of management (prevention or correction) of system faults. More than one capability may be instantiated.	Engine Diagnostics: Inspection and Routine Servicing: Maintenance and Fault Data Management: Tire Pressure Diagnostics: Transmission Diagnostics:
24	Military Vehicle Application Feature Group	Military Application Type	Feature Primary Key [Property]	The type of military application(s) the vehicle is configured to perform. More than one value may be populated.	Armored fuel transport: Armored personnel transport: Exterior Camouflage: Gun Mount--7.62 mm: Local Delivery: Low Radar Signature:
25	Navigation Feature	Navigation Capability	Feature Primary Key [Property]	A type of automatic navigation capability. More than one type may be instantiated.	Central Mission Route Download: GPS-based Location Sensing: Map Location Display: Trip and Mission Route Display and Directions:
26	Cost of Operation Feature	Operating Cost	Feature Attribute [Property]	The cost of operating the vehicle in its intended use.	
27	Operability Feature	Operations Capability	Feature Primary Key [Property]	An internal management control capability (active or passive) provided by the system, in the area of management of its main purpose performance outcomes. More than one capability may be instantiated.	Automatic Performance Data Logging: Automatic Performance Data Measurement and Display: Automatic Performance Threshold Detection and Reporting: Maneuverability: Manual Performance Data Logging: Manual Performance Data Management: Operations Procedures: Visibility:
28	Personal Vehicle Application Feature Group	Personal Application Type	Feature Primary Key [Property]	The type of personal application(s) the vehicle is configured to perform. More than one value may be populated.	Basic Personal Transportation: Compact City Car: Luxury Car:
29	Reliability & Availability Feature	Reliability	Feature Attribute [Property]	The likelihood that the system will operate without fault during its operational service life.	
30	Remote Management Access Feature	Remote Access Capability	Feature Primary Key [Property]		
31	Remote-Autonomous Operation Feat...	Remote Operations Capability	Feature Primary Key [Property]		
32	Safety Feature Group	Safety Rating	Feature Attribute [Property]	The rating granted to vehicles of this type by an authoritative agency, concerning its ability to be operated without undue risk to people or property.	

Feature configuration space: Bigger than expected

Like the Tardis: Bigger on the Inside!



A perhaps surprising thing about Features is that they model a lot more than might be thought of at first when considering "value":

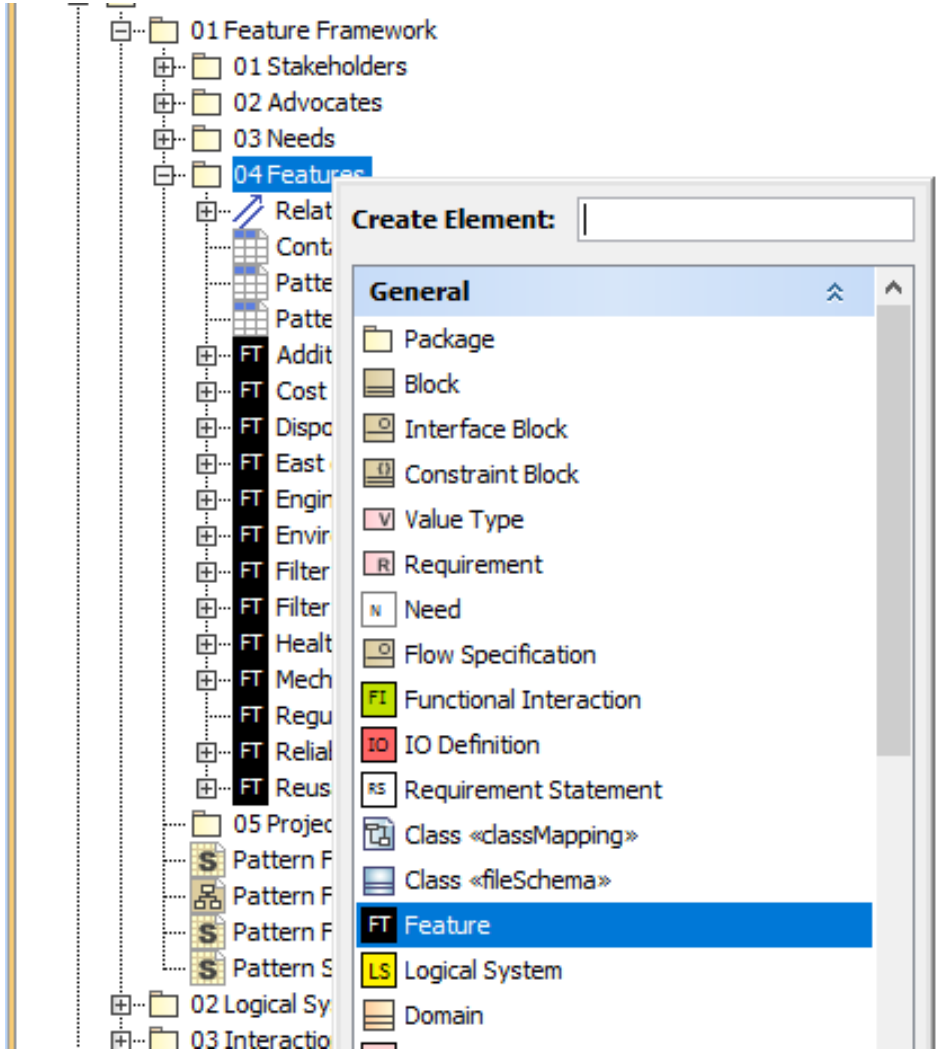
- Features offer good examples of models that are both "too small" and "too large" (redundant and conflicting) at the same time.

1. Features model the trade-space for optimization and trades—that one is not too surprising, but serves as a reminder to include the full range of stakeholder issues, not just end Customer Features—who are all the stakeholders? The resulting Pareto Frontiers are in Feature Configuration Space.
2. All *purpose*, even when discovered by emergence and agile pivots, is in Feature Space.
3. *All risk is risk to Stakeholder Features*. So, the whole outcomes side of any Risk model should terminate in Feature space. (More on this when we cover model-based FMEAs.)
4. *All Effects (the "E" part in FMEA analyses) are effects in Feature Space*. Not realizing this, they are often described completely separately—a redundancy that costs a lot when not used to reinforce and improve both the positive and negative sides of models. (More on this when we cover model-based FMEAs.) This also applies to Consequences described in Safety and Cyber analyses.
5. *All product line segmentation / selection is describe in Feature Space*. (More on this when we cover S*Patterns and pattern-based methods.)

Simple exercise: Creating Features

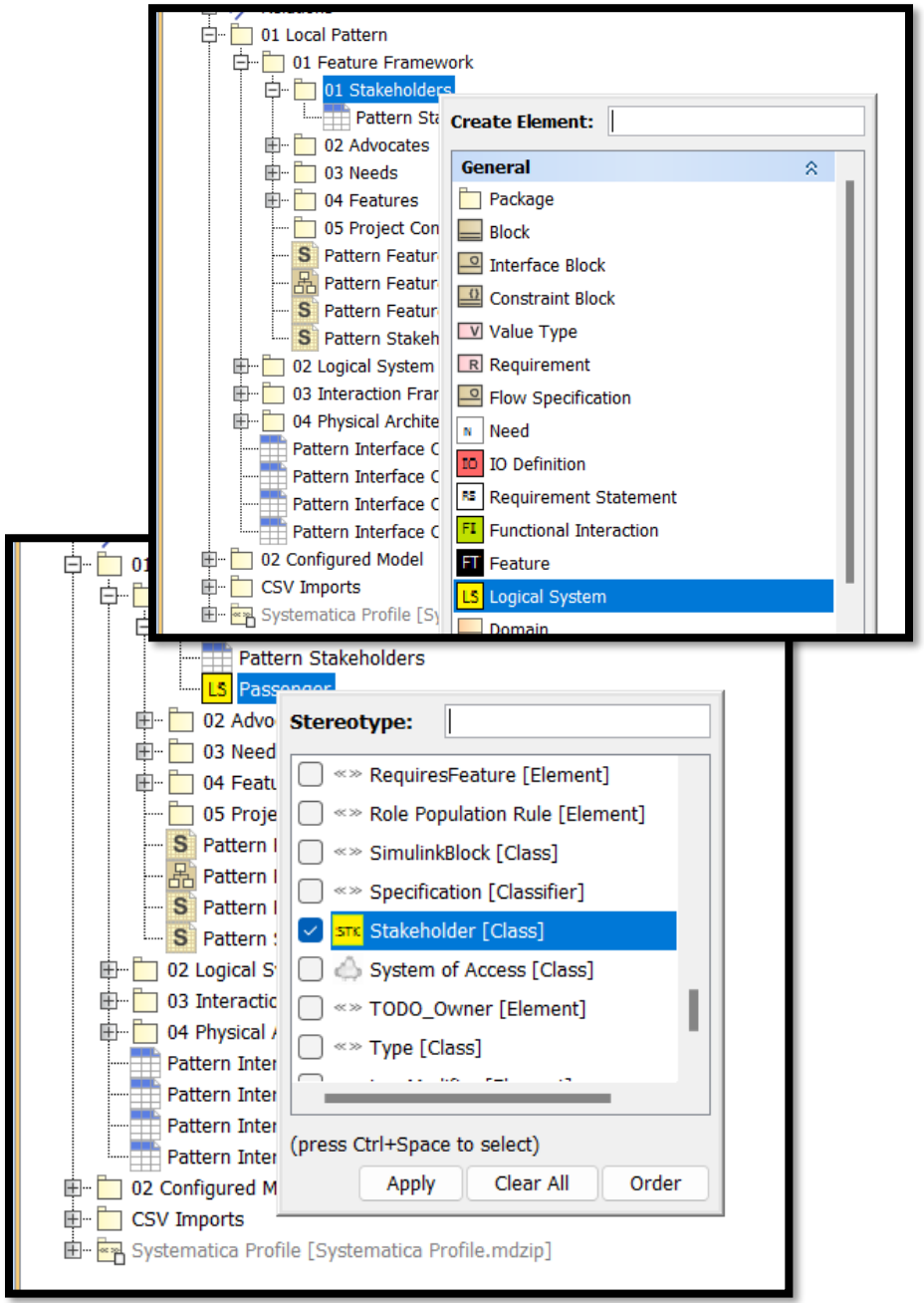
- Try inserting a Stakeholder and a Stakeholder Feature into your Cameo project space, in the Local Pattern's Features Package.
- Pattern Features are created in package 01 Local Pattern::01 Feature Framework::04 Features
- Use ~Create Element and choose Feature to create an S* Feature
- Name the Feature (use of stakeholder-familiar language is recommended) and verify it has been added to the model.

(We will learn to add Feature Relationships and Attributes later.)



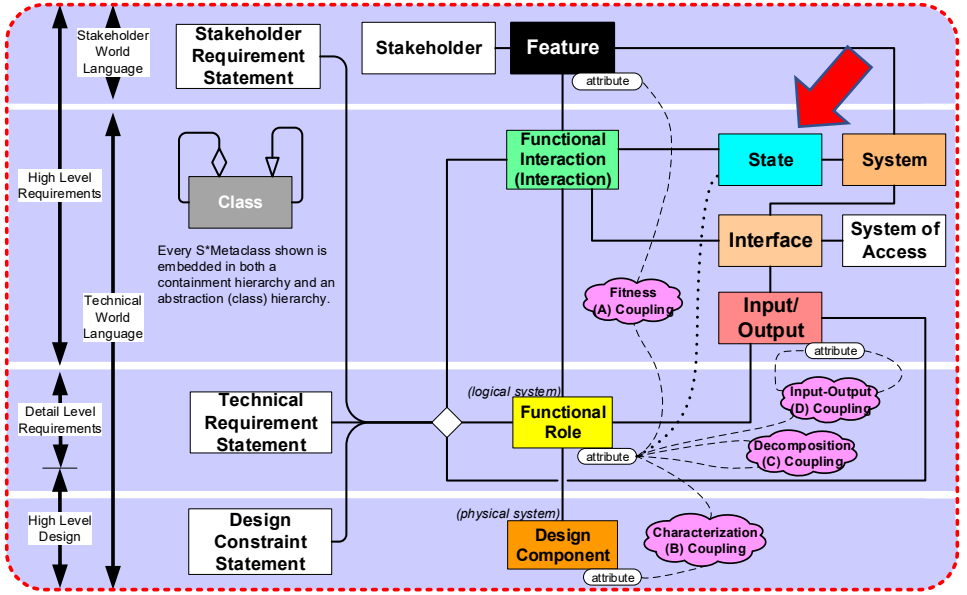
Feature Framework Modeling: Creating Stakeholders

- Pattern Stakeholders are created in package 01 Local Pattern::01 Feature Framework::01 Stakeholders
- Use ~Create Element and choose Logical System, because Stakeholder is a type of Logical System.
- Right-click on the newly created Logical System; select Stereotype and change the Stereotype to Stakeholder and remove the Logical System Stereotype.
- Name the stakeholder (noun recommended) and verify it has been added to the model.

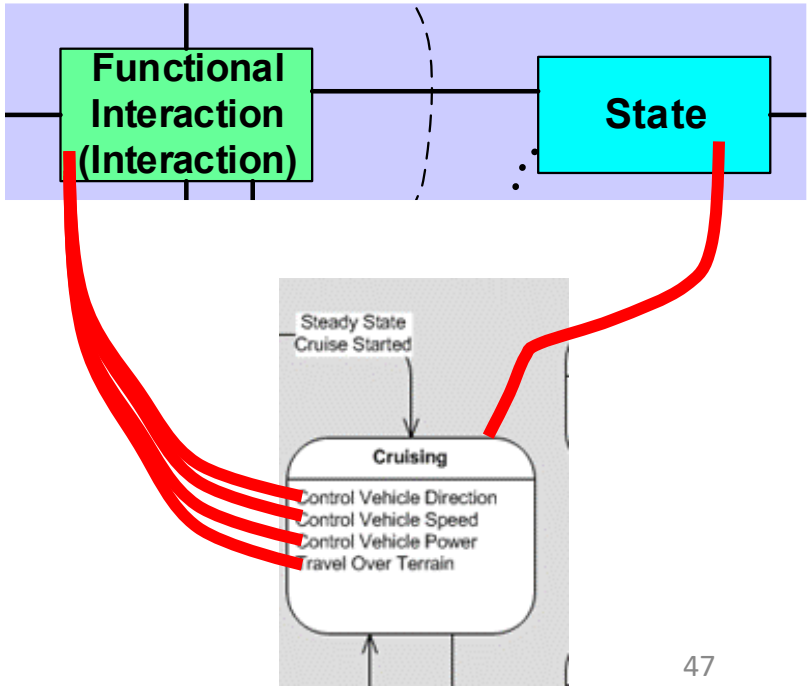


States, State Variables

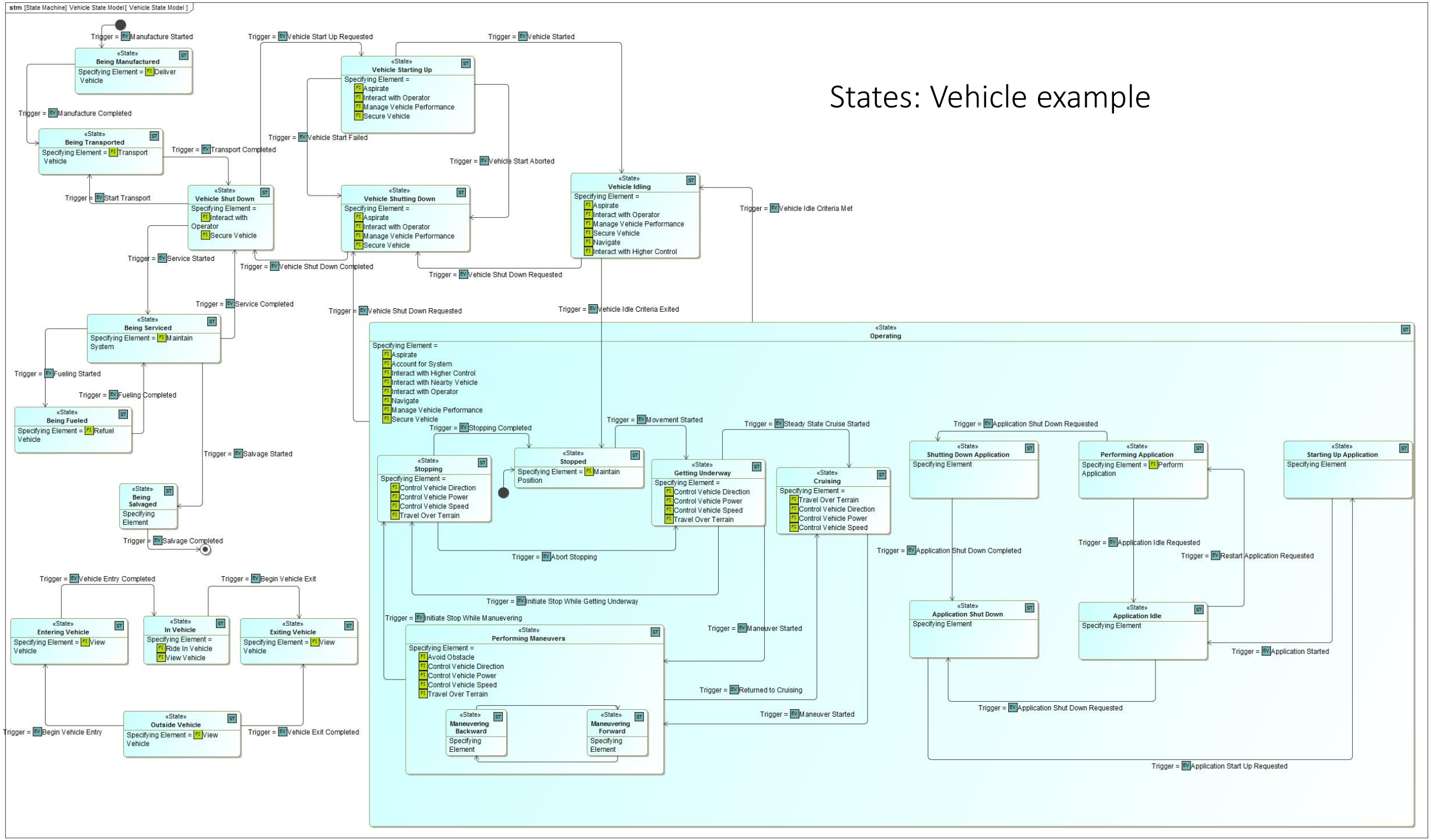
- In general, a State is a condition of a system, described by its State Variable(s) (e.g., position, velocity acceleration, temperature, pressure, etc.):
  - The state of a system component may determine its input-output behavior (even if statistical) during Interactions in which it participates.
- For the important special case of model-based Finite State Machines (FSMs; finite automata), a State is a single value of the related state variable, represented by one block of an FSM diagram, . . .
  - representing a condition, mode, or situation, persisting for a period of time,
  - during which the system exhibits behavior described by associated Interaction.
  - We may model "State Transitions" from one finite State to another (typically instantaneous).
  - Those transitions may be caused by modeled State Transition Trigger Events.



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

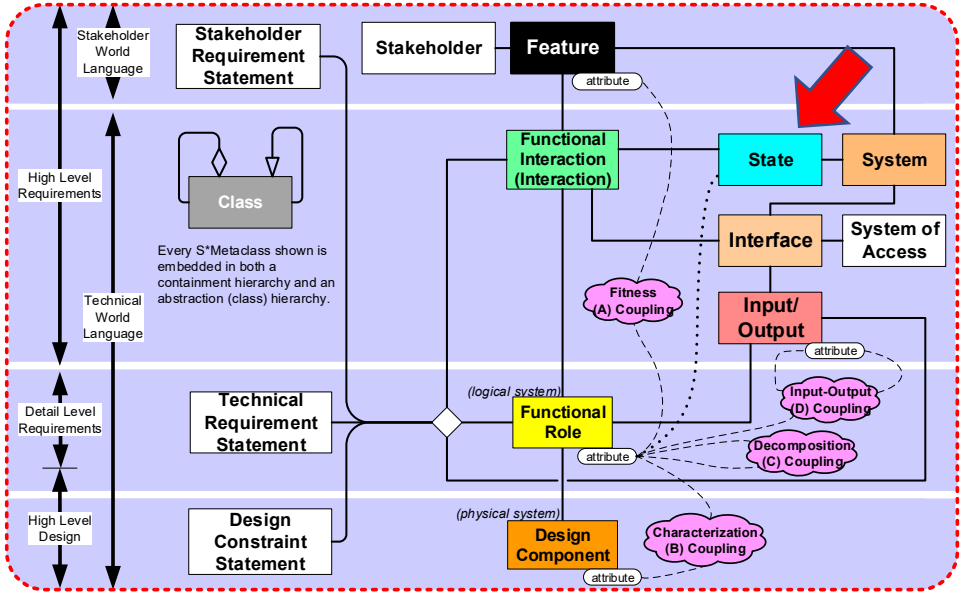


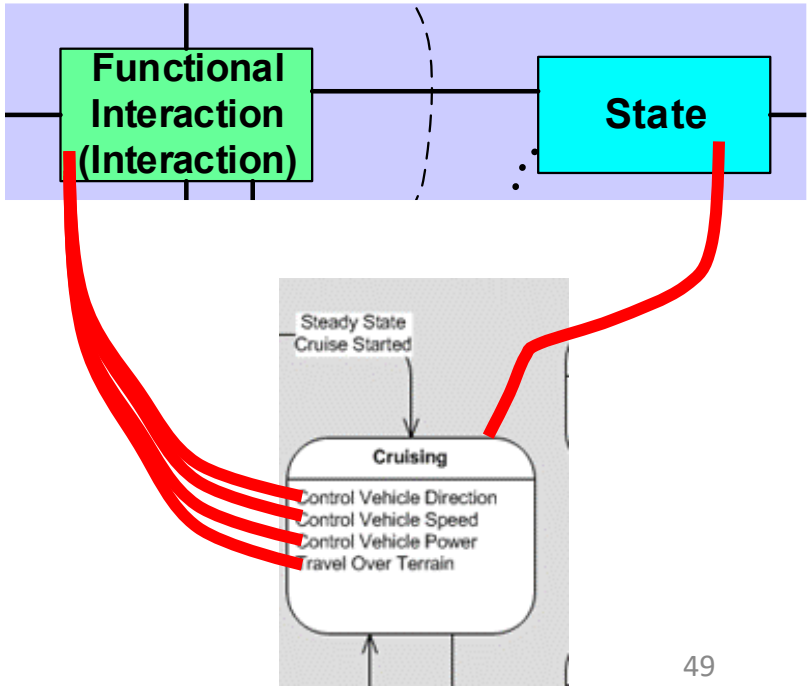
States: Vehicle example



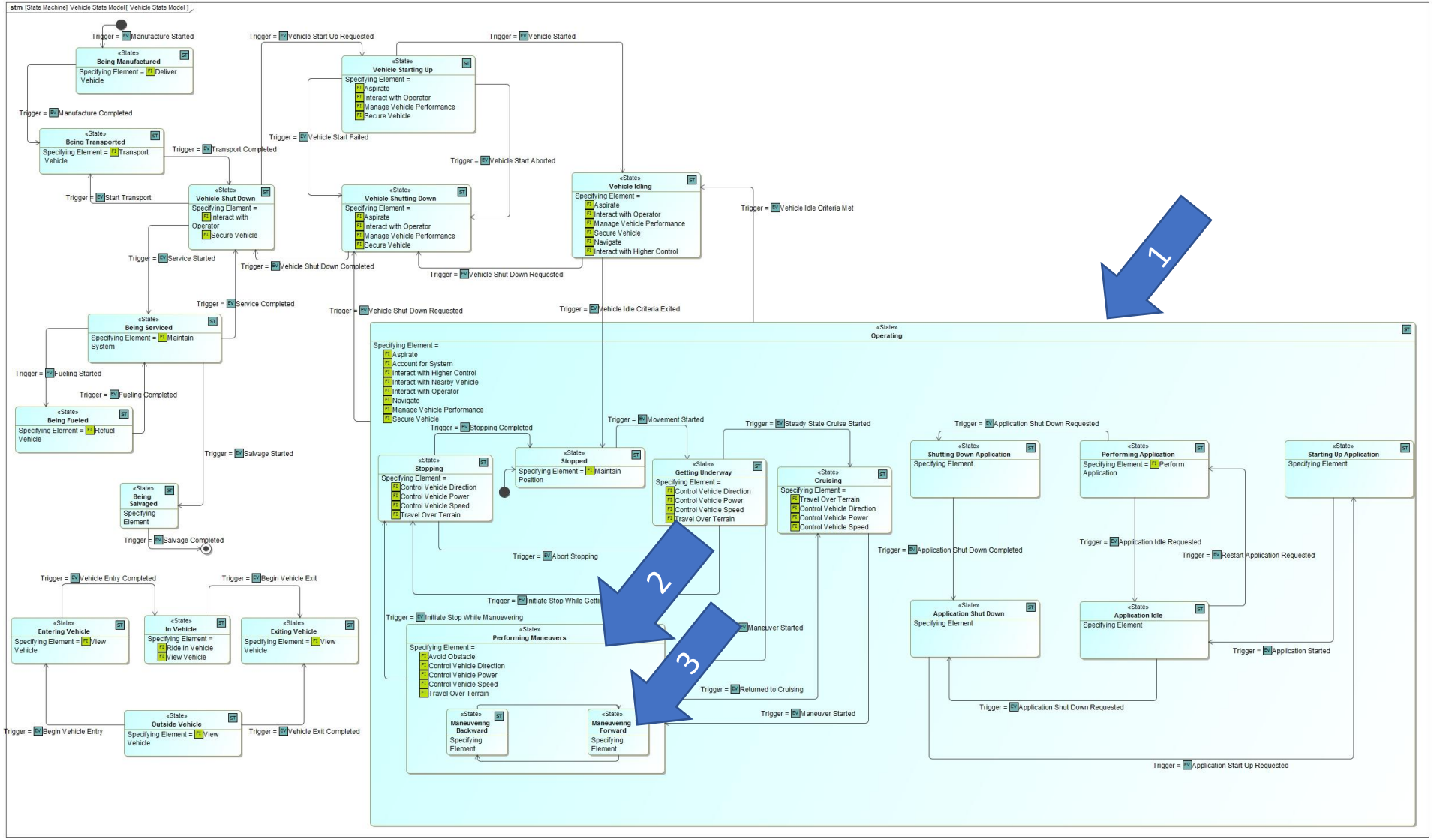
States, State Variables

- For the finite state machine case, it can be helpful (not essential) to name the states with “gerund” names (a verb form ending in “-ing”, treated as a noun)
  - The name the summarizes what is occurring during that state.
- Some states machines describe operational conditions (in operational state models); others describe life cycle stages (e.g., being manufactured, being distributed, being installed, being serviced, etc.).
- There is a natural mapping between State/Interaction sets and Use Cases—they both describe a “situation” and what happens during that situation:
  - See also “Scenarios” used in validation of concepts.
- We can “unwind” a “single thread” path through a state machine, as a linear thread of activity:
  - For stakeholder validation, this is often more understandable (concrete) as a “story” of an instance path, compared to the more general (but therefore more abstract) state machine view, which represents all the paths together.

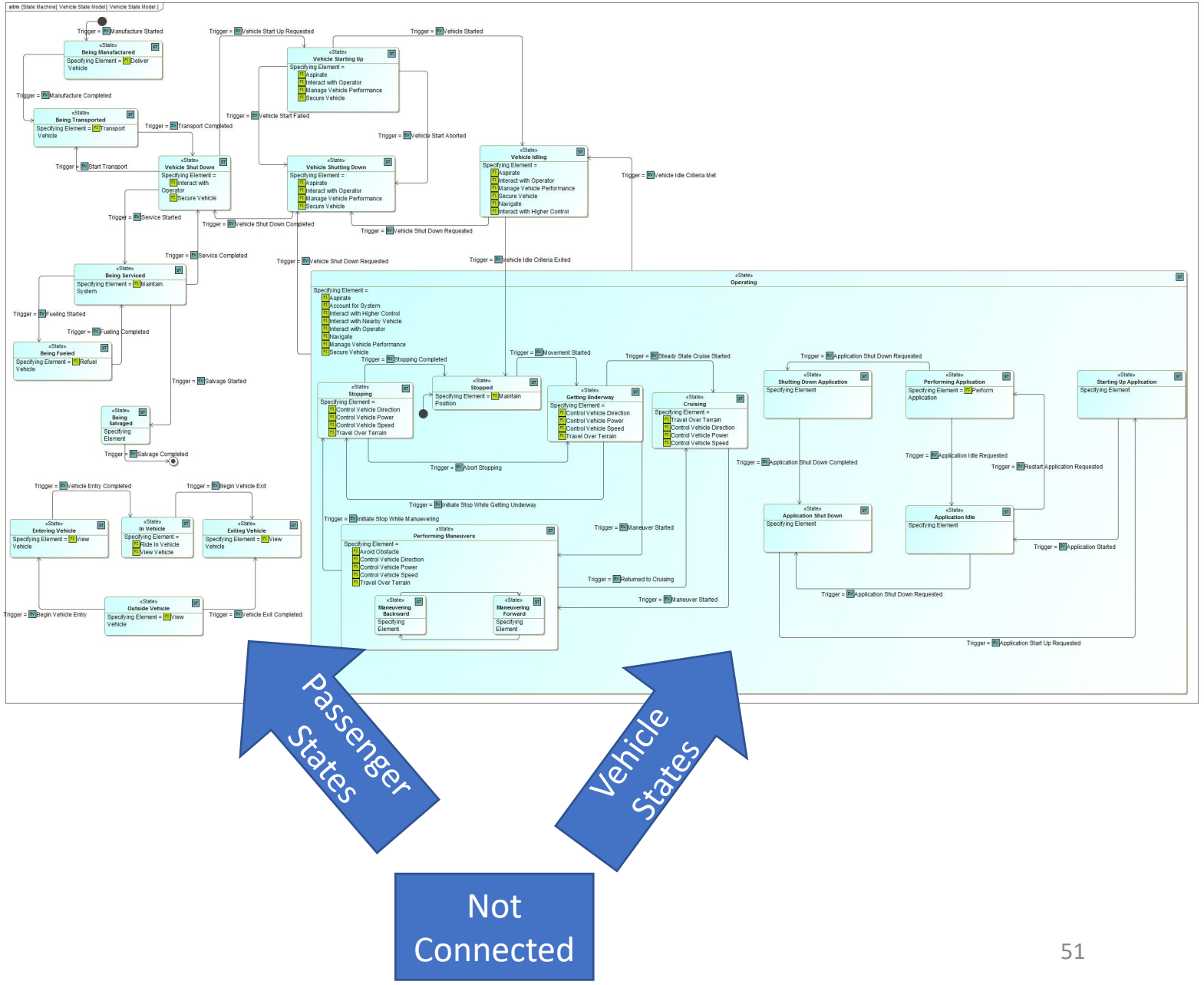




- Substates:
  - When a state is drawn inside another state, this means that inside state (called a “substate”) occurs during the larger parent state. Note the three levels shown here--



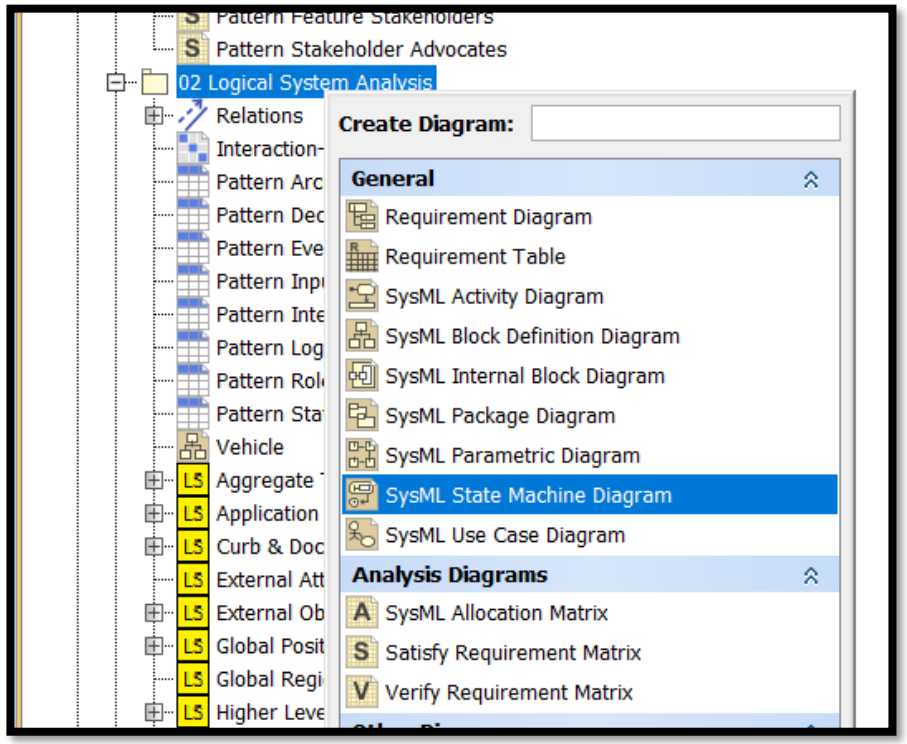
- Concurrent states:
  - Within a “connected” subsection of a state machine, only one state of that section can be the currently “live” state at any one time—the connected section describes one state variable.
  - When there are no connecting transitions between two sections of the state model, the two sections describe different state variables, and a state in one section can be “live” at the same time as a state in the other section. They are called “concurrent state machines”.



Simple exercise: Creating State Machines and States

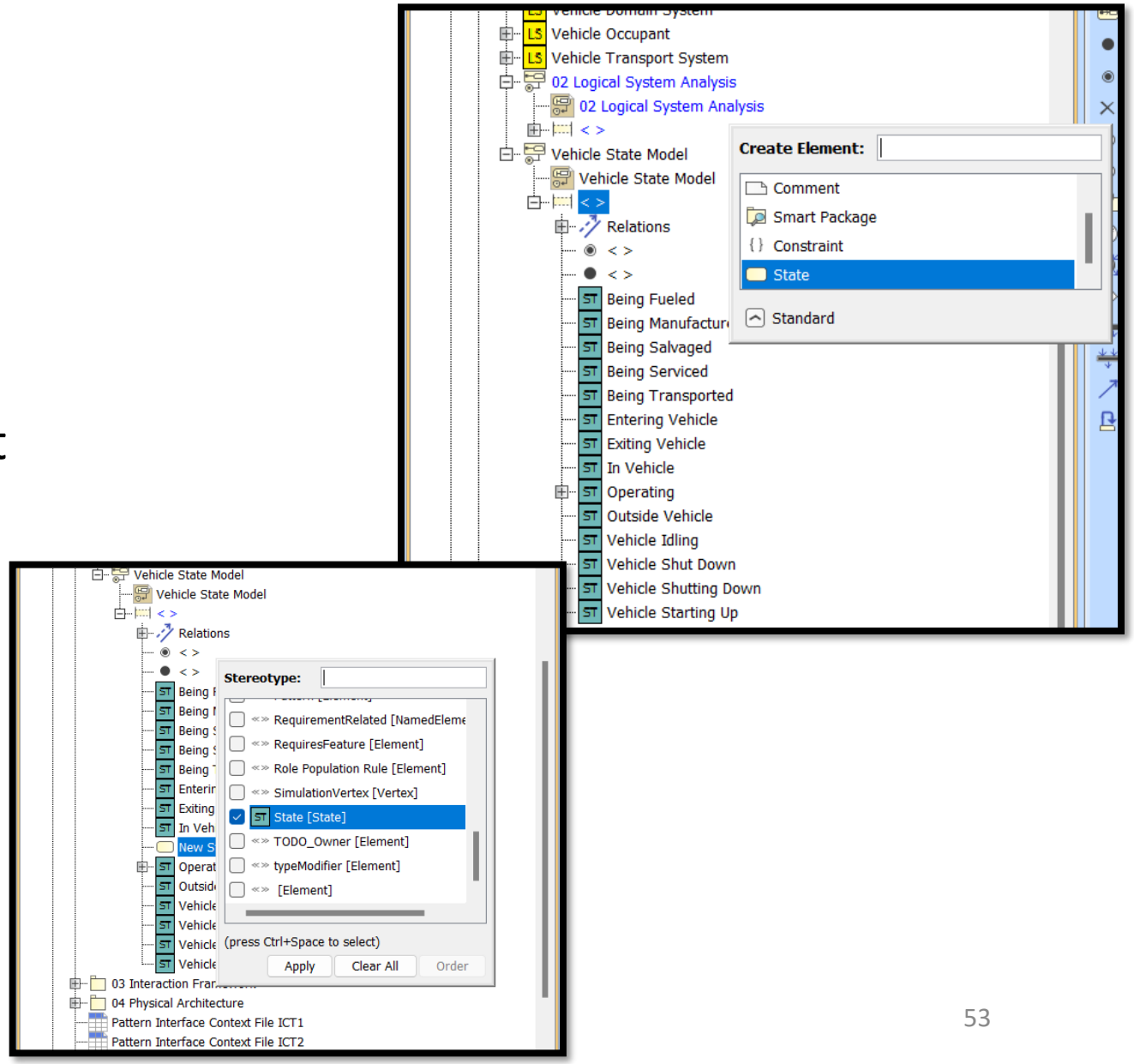
- A State Machine must be created or exist before adding States. Your Project should already have an existing State Machine, so you can continue to the next slide. Follow the steps below if you need to create a new State Machine.
- Try inserting a State Machine and State into your Cameo project space.
- Pattern State Diagram and States are created in package 01 Local Pattern::02 Logical System Analysis
- Use ~Create Diagram and choose SysML State Machine Diagram

(We'll learn more about creating State Transitions, Events, and relationships later.)



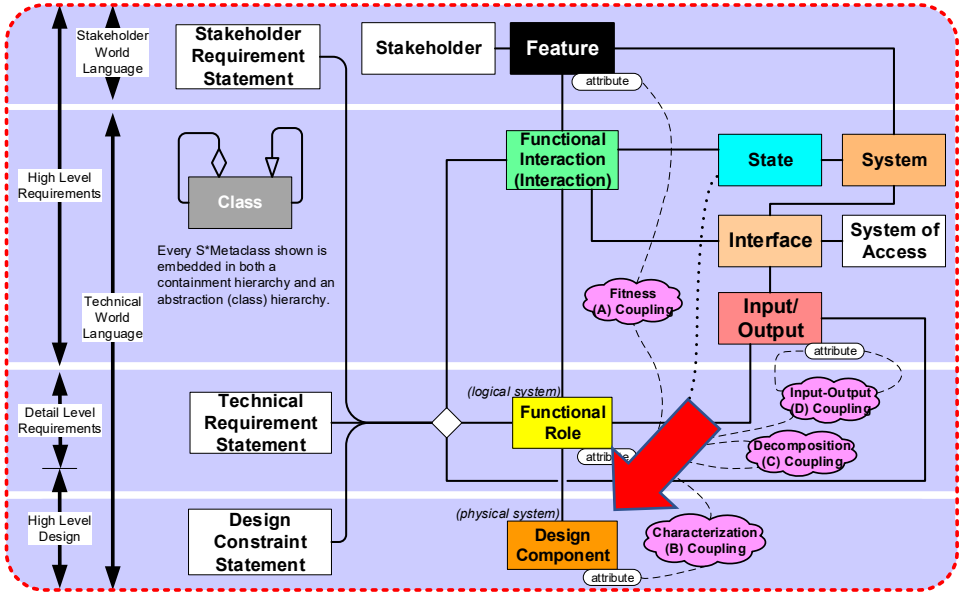
State Machine Modeling: Creating State Machine and States

- To create a State for the State Machine, right-click the automatically created Region under the State Model and use ~Create Element and choose State. Make sure the menu is expanded to Expert.
- Right-click the newly created State, select Stereotype, and change it to State as shown below.
- Recommended names for states include static or dynamic conditions (e.g.,. “Stopped”) and “-ing” names (gerunds; e.g., “Operating”, “Taxiing”).



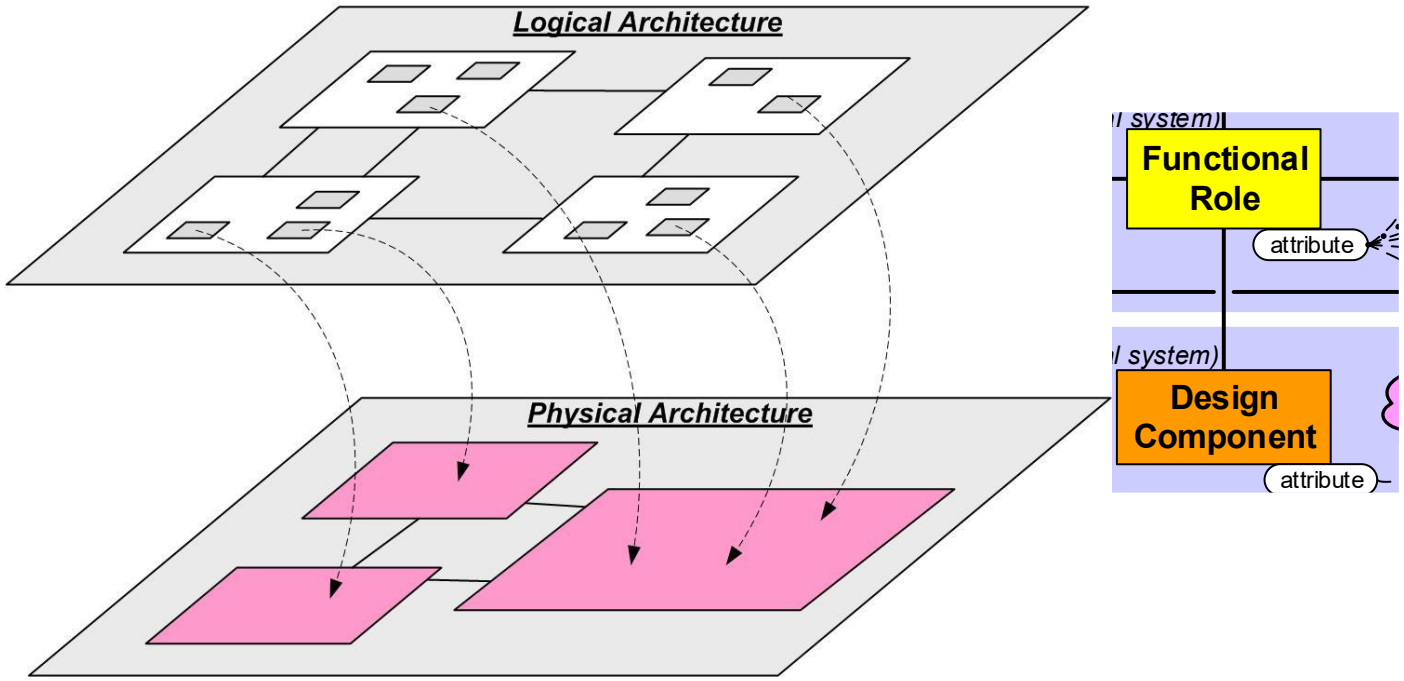
Design Components: Clarifying SE views of allocation, logical, physical

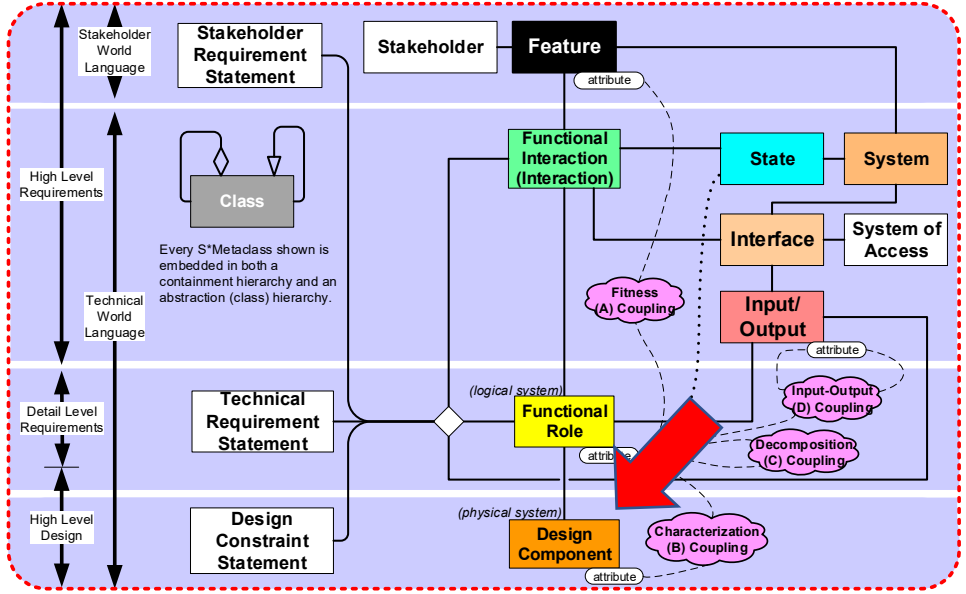
- A Design Component is what Roles are allocated to—an entity responsible to deliver (have or perform) the behaviors that the Roles describe.
- A Design Component is described entirely by its identity, not behavior, so its attributes are only about identity—the behavior is all in the Roles allocated to it.
- Examples:
  - 2N404 Transistor; 741 Op Amp
  - Dell Precision 5200 Computer
  - Executable Program Module A7552
  - Zoeller M53 Pump
  - Bob Snodgrass, Jr.
  - Aircraft Carrier Carl Vinson
  - Cranston Machine Shop



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

Design Components



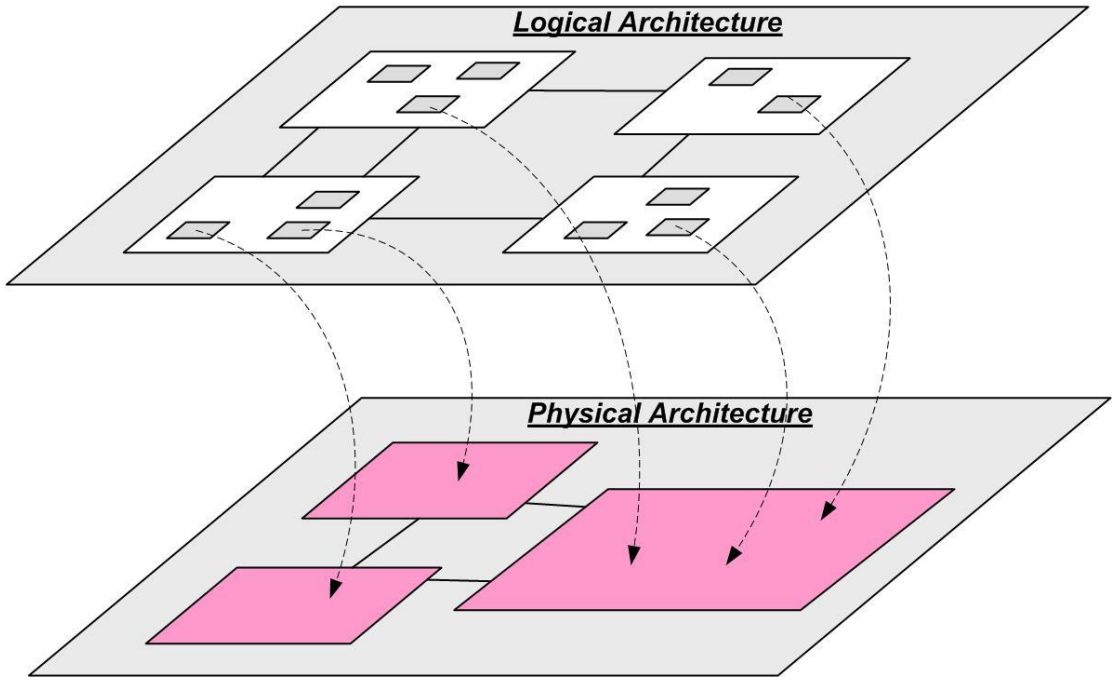


S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

Traditional (good) practice is to:

1. separate the organization of interacting logical Roles (the Logical Architecture) from the organization of connected "Physical Components" (the Physical Architecture), and …
2. Then allocate the logical roles onto the physical architecture.

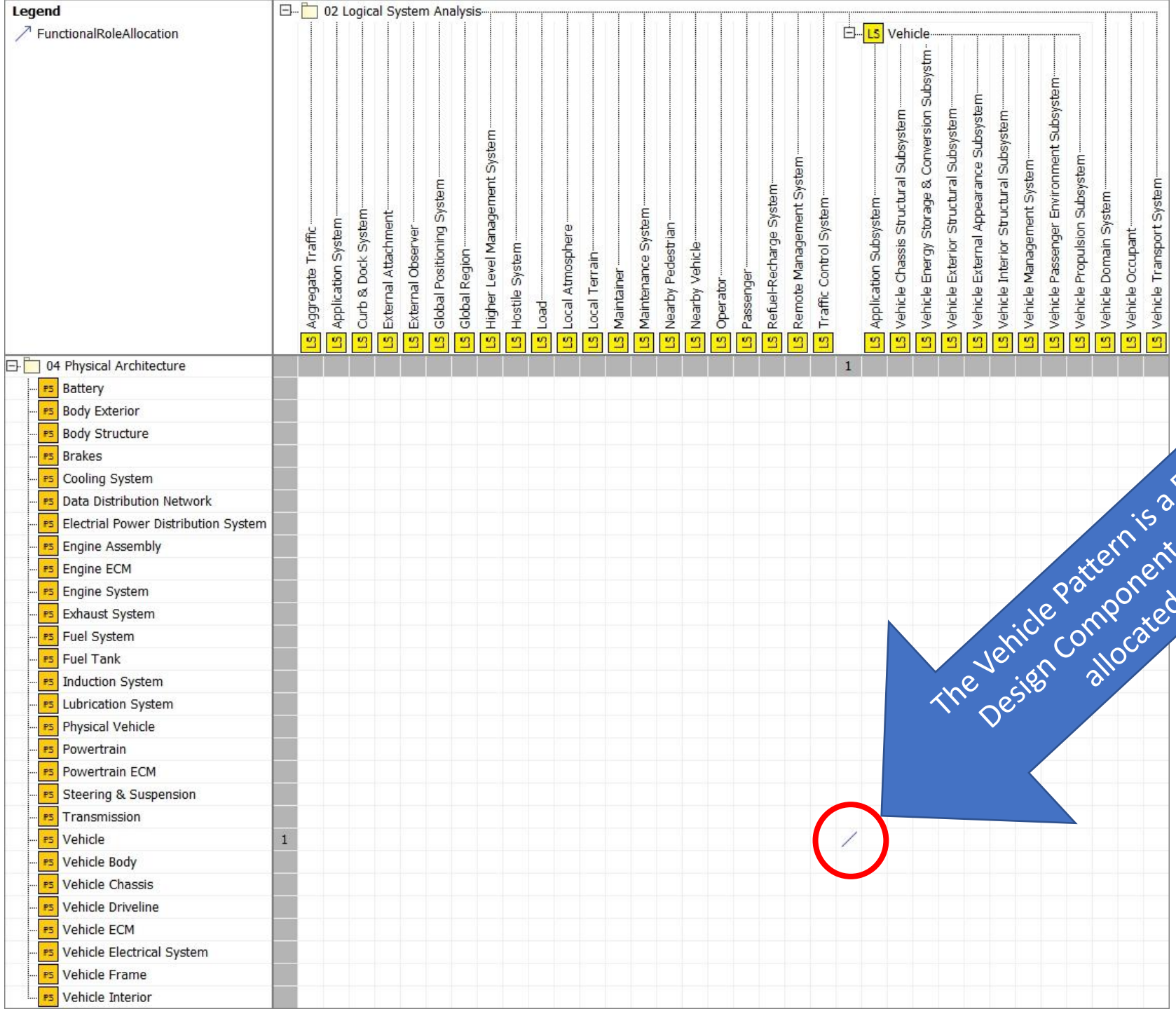
Design Components



Many to one, but not one to many!

Key issue in avoiding ambiguous requirement allocation:

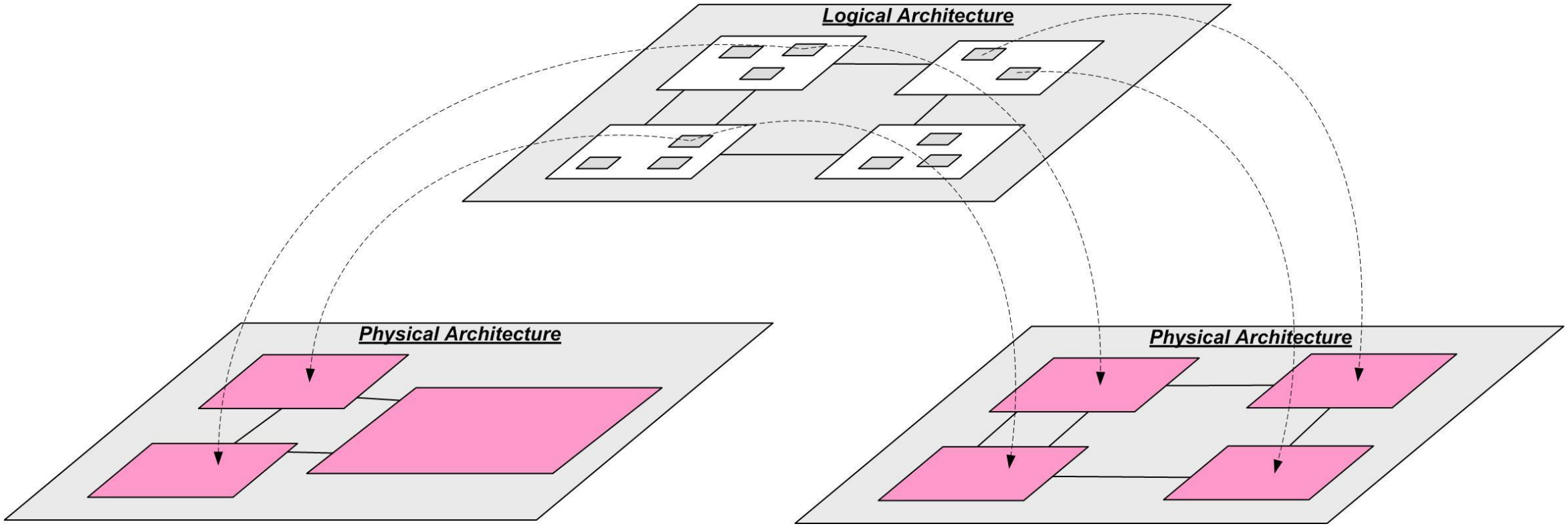
- It is okay and very common to allocate multiple Logical Roles to a single Design Component, but . . .
- It is not okay to allocate a single Logical Role to multiple Design Components!
- The reason is that this would be ambiguous as to the responsibilities of each Design Component—a common mistake.
- So, this tells us to keep decomposing Roles until they are “small enough” to be unambiguously allocated to single Design Components.



The Vehicle Pattern is a Black Box Requirements Pattern—so the only Design Component allocation is the whole Vehicle Logical System allocated to the whole Vehicle Design Component

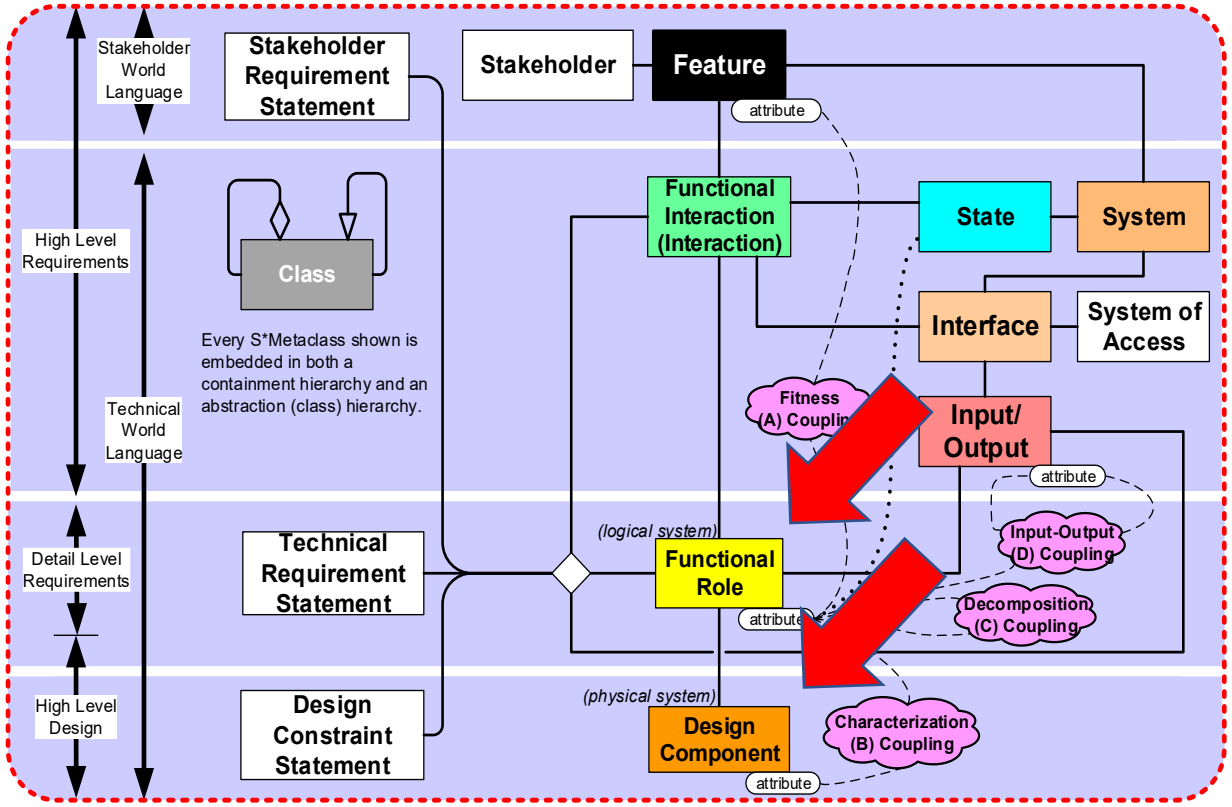
Design Components

Another powerful aspect of this area is to recognize that a single generic or configurable Logical Architecture can be allocated to different “Physical Architectures”, for different markets, configurations, or applications:



Shifting Terms and Minds

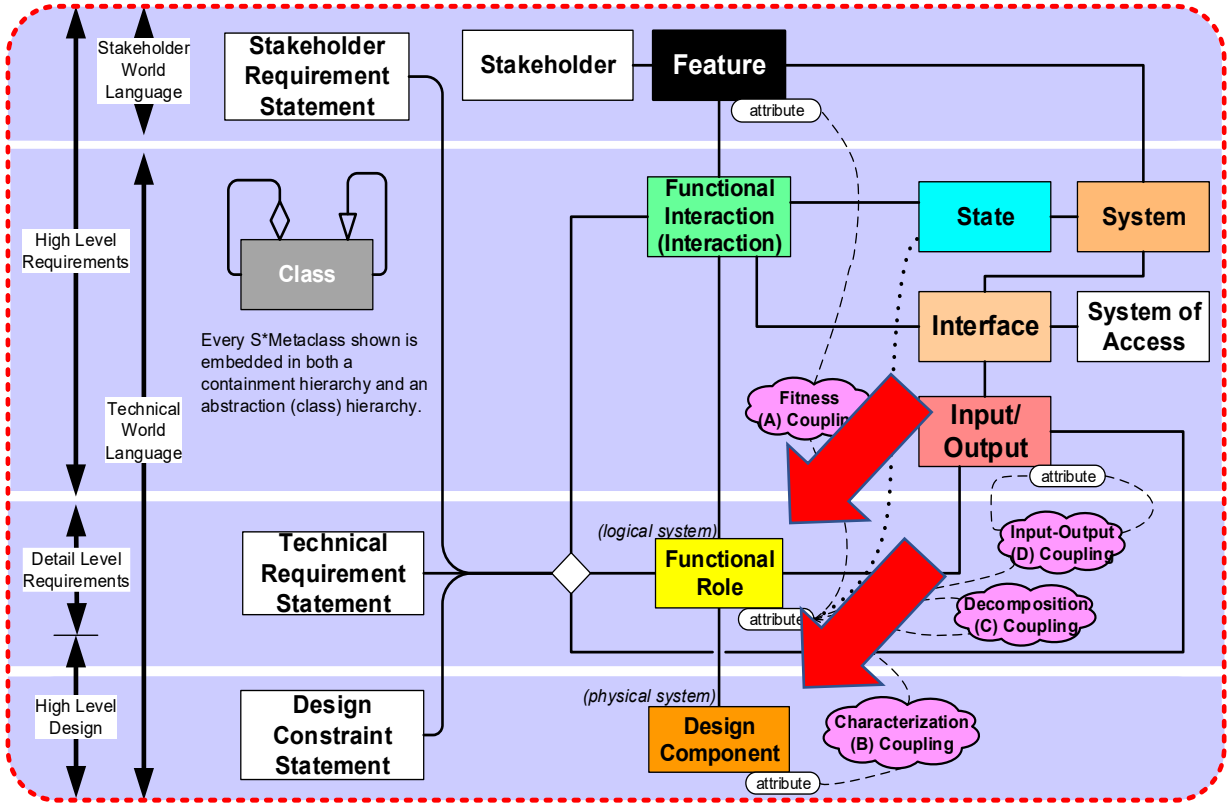
- Now we will show how those traditional perspectives, while containing some very good aspects we must preserve, also contains one confusing aspect that we need to move beyond . . .
- The confusing part is the traditional practice of referring to “logical systems” (that is the Functional Role) and “physical systems” (that is the Design Component).
  - The widely-used acronym “R-F-L-P” comes to mind.
- Notice that the diagram shows we have referred to the bottom modeled block as a “Design Component” instead of a “Physical Component”—a past name that we have eventually deprecated.
- But why? . . . .



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

Shifting Terms and Minds

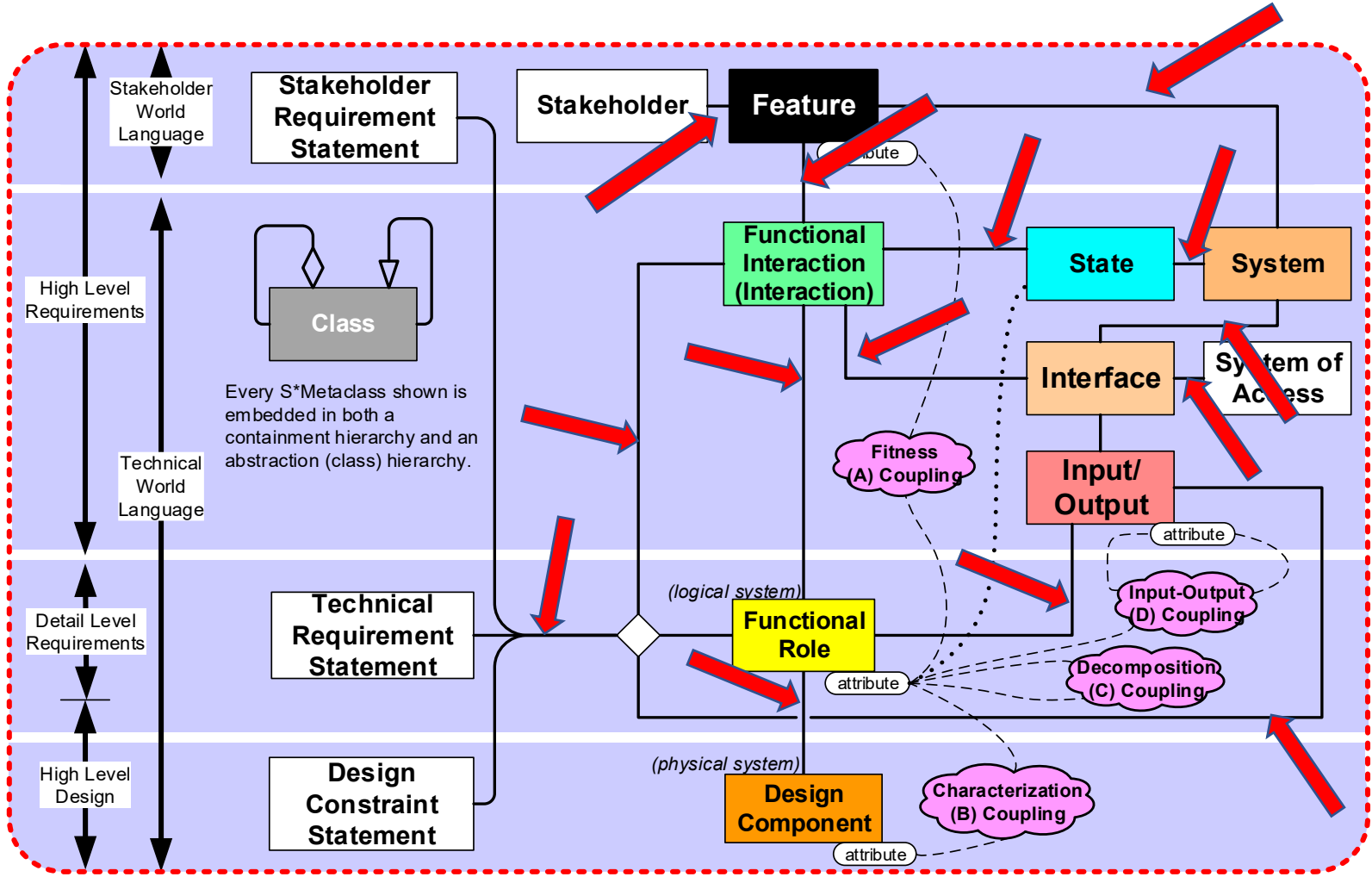
- The reason for this deprecation of terminology is the following:
- The use of the contrasting terms “Logical System” and “Physical System” together conveys the unstated impression that the “Logical System” is not physical.
- What does that mean?
- Bear in mind that we use the “Logical System” to describe all the behavior:
  - Input-output forces, energy conversion, frictional losses, etc. etc.
- Try “pounding” on a real, concrete, table top: Everything that you feel and hear is behavior!
- Realizing that “logical systems” are there to model “physical behavior” is an important way to step up to the full responsibility of behavioral modeling.
- “Abstraction” (generalization) is not an excuse for not modeling physical behavior, even if general.
- But it requires recognizing confusion in the current traditional terminology.



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

S*Relationships Between Different S*Metaclasses

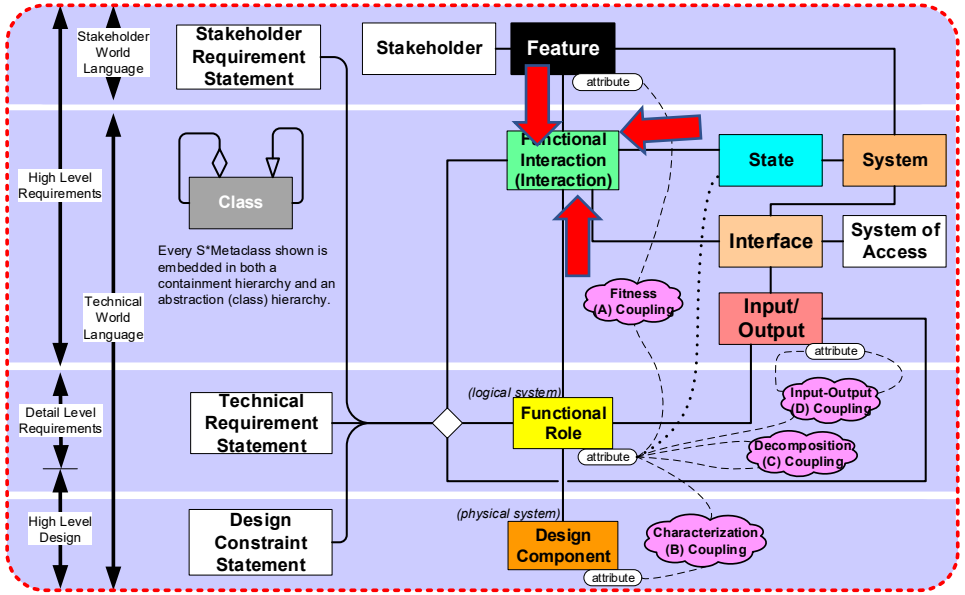
- S*Relationships between different S*Metaclasses express numerous statements about combinations of those classes. For example:
  - A particular Functional Interaction may require certain Functional Roles.
  - A particular Functional Role may be allocated to a certain type of Design Component.
- The S*Metamodel informal "pedagogical diagram" summarizes some of the more prominent S*Relationships as unlabeled connecting lines.
- The S*Metamodel formal document lists the larger complete set of S*Relationships, and shows that each of them has a labeled relationship name.



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

Relationships: Three paths to finding all the Interactions

- "Discovering all the Interactions" that need to be modeled is very important:
  - For example, Session 2 will show how this can greatly help us "find all the Requirements" for a system.
  - Key: Relationships between different S*Metaclasses
- So, the following is provided as a powerful way to "find all the Interactions":
  - Following relationships, there are three orthogonal "paths" to Interactions in the S*Metamodel:
  1. Feature-Interaction pairs tell us "why" an Interaction occurs (for what Feature).
  2. State-Interaction pairs tell us "when" an interaction occurs (during what State).
  3. Actor/Interface – Interaction pairs tell us "who or what" engages in interaction (what Roles interact).

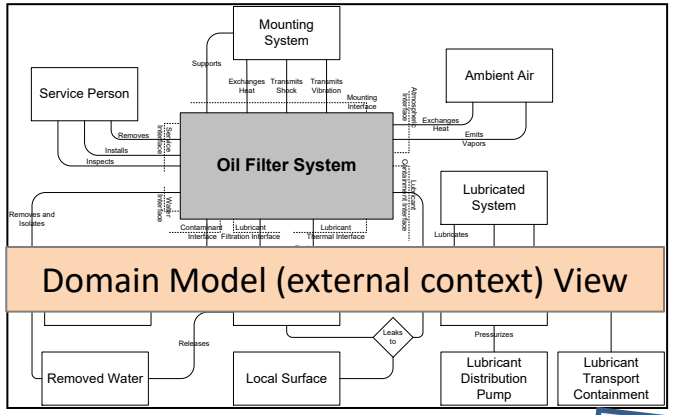


S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

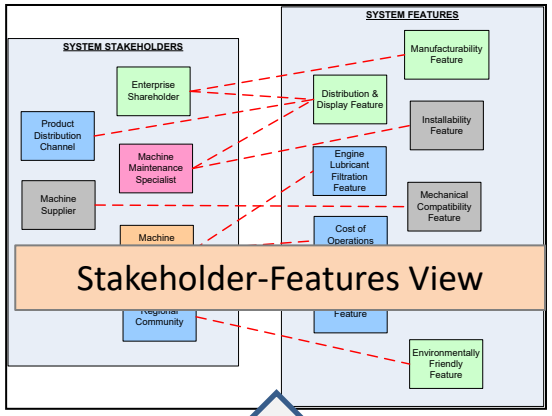
- Each interaction should appear in all three lists of pairings!
- During model construction, it is common to discover, for one of these three different perspectives:
  - missing interactions or pairings that need to be added.
  - Missing Features, States, or Actors that need to be added.

A key point about model completeness

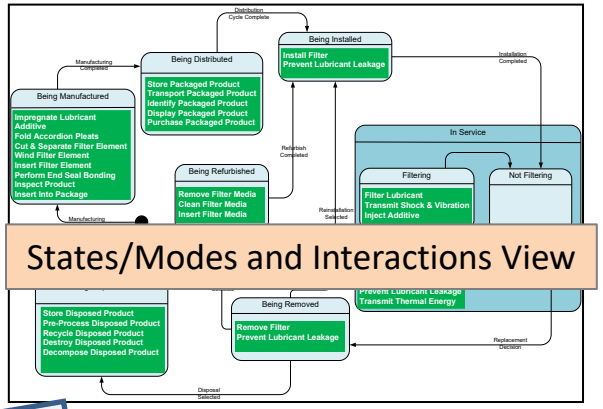
- The above means that a system model is less likely to be complete (or we should be less confident in its completeness) if it does not include:
  - Some form of domain model, showing all external actors/external interfaces.
  - Some form of state model, showing all possible system black box states.
  - Some form of stakeholder feature model, showing the stakeholders' value space.
- A listing of all the external interactions of the system of interest:
  - Mapped to (paired with) its external actors/external interfaces
  - Mapped to (paired with) its features model
  - Mapped to (paired with) its state model
  - That "matches" the nature of each actor, feature, and state
  - That "exactly covers" all four lists, neither larger nor smaller than each list.



Domain Model (external context) View

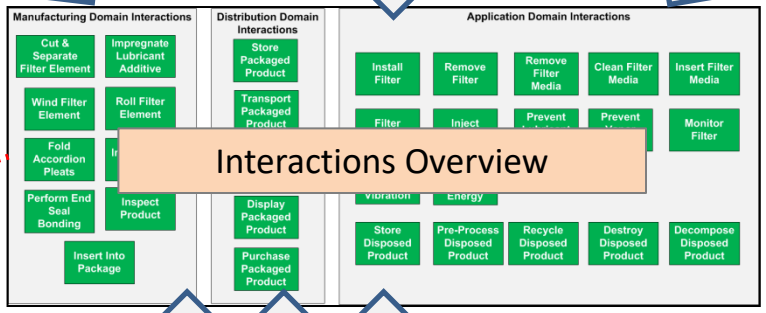


Stakeholder-Features View



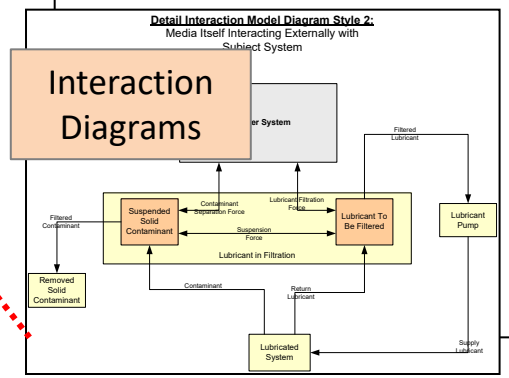
States/Modes and Interactions View

Discovering all the External Interactions



Interactions Overview

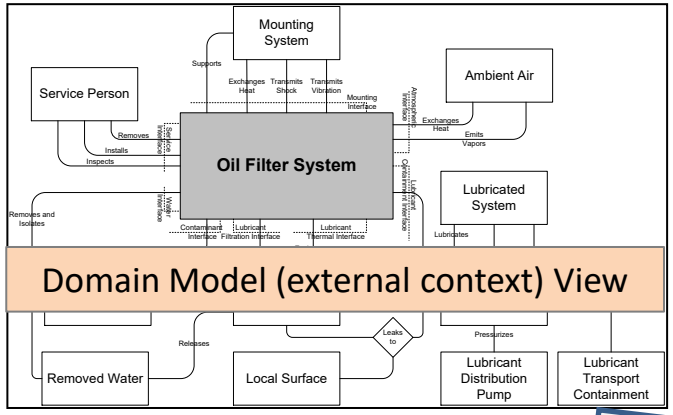
Discovering all the Technical Requirements

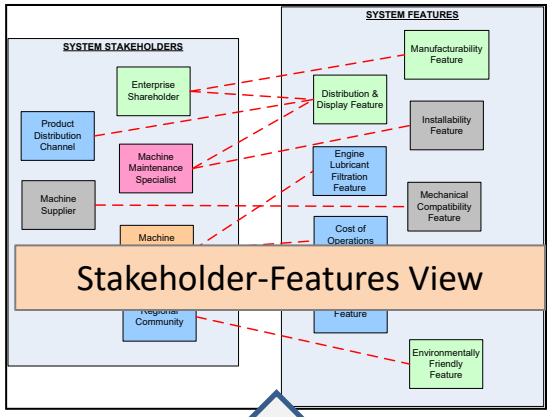


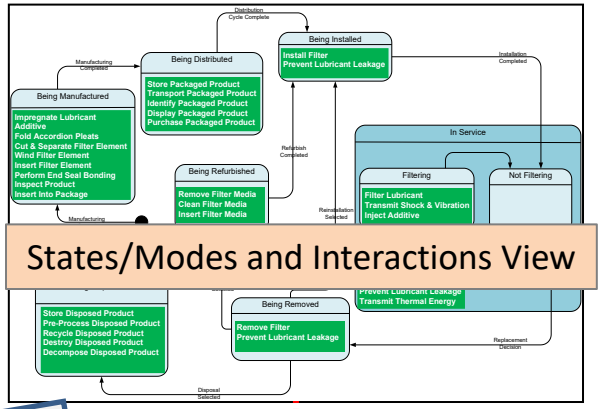
Interaction Diagrams

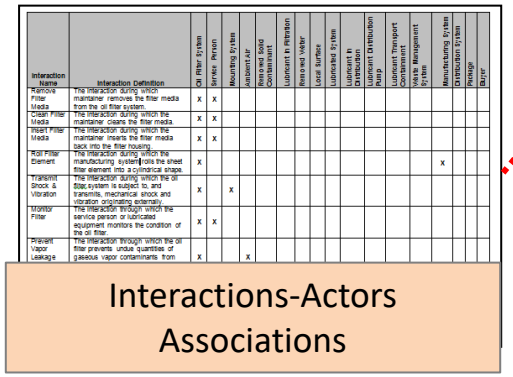
Interaction	Role	Requirement (Required or Assumed Behavior)
Filter Lubricant	Oil Filter System	For a Return Lubricant stream of [Lubricant Viscosity Range] and [Lubricant Pressure Range], the Oil Filter shall separate Filtered [illegible]icant output stream, [illegible] Distribution Profile].
Filter Lubricant	[illegible]	[illegible] viscosity within the
Filter Lubricant	[illegible] Pump	[illegible] within the [Lubricant
Install Filter	Oil Filter System	The Oil Filter shall be manually installable in ten minutes or less, using only a screwdriver.
Install Filter	Oil Filter System	The Oil Filter shall have installation instructions printed on its exterior surface, in English
Install Filter	Service Person	The Service Person shall have the visual acuity and hand strength of an average 40 year old adult.
Install Filter	Service Person	The Service Person shall be capable of reading English at the tenth grade level.

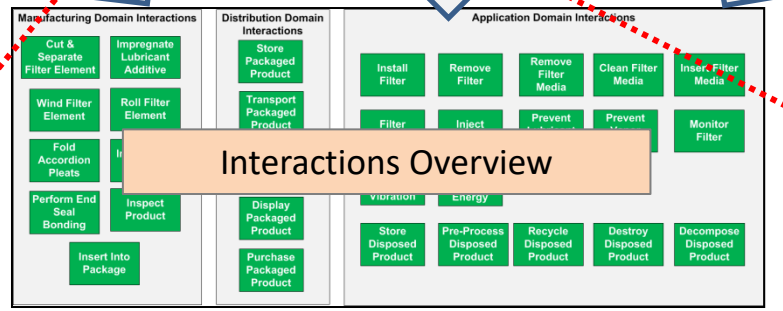
Technical Requirements

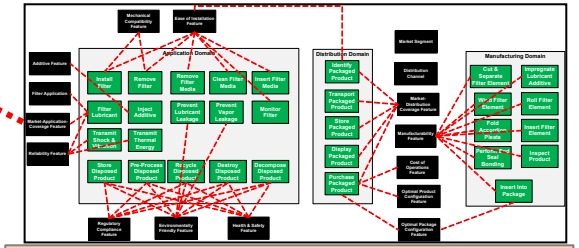


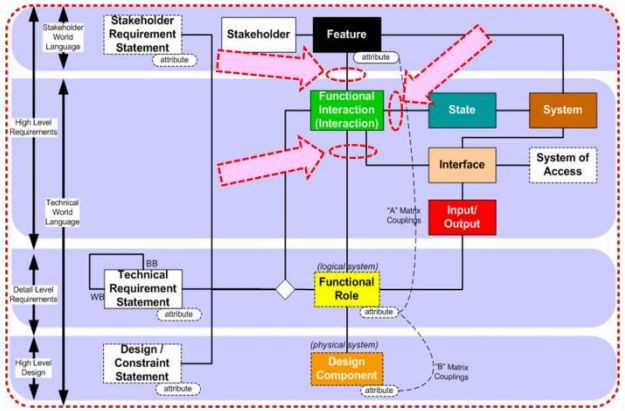






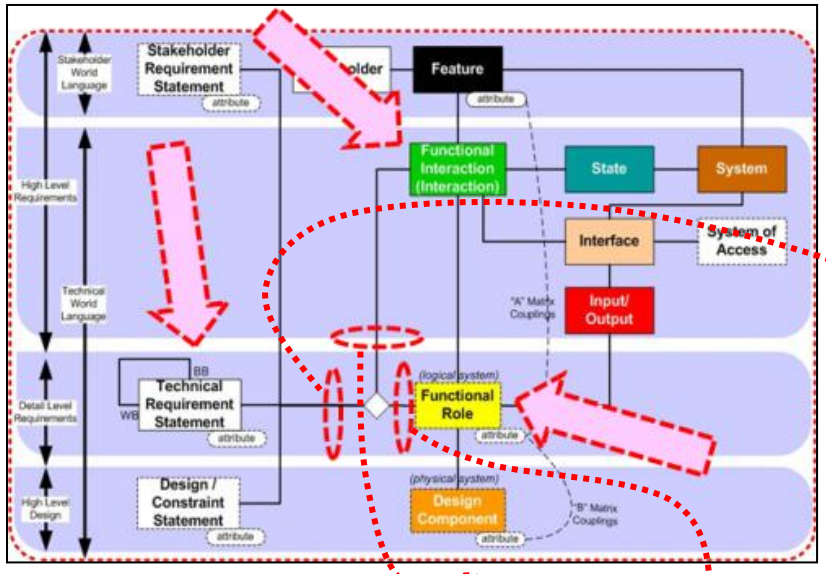






Inherent Relational Checks of <u>High</u> Level Model Completeness / Consistency (Model Metrics)

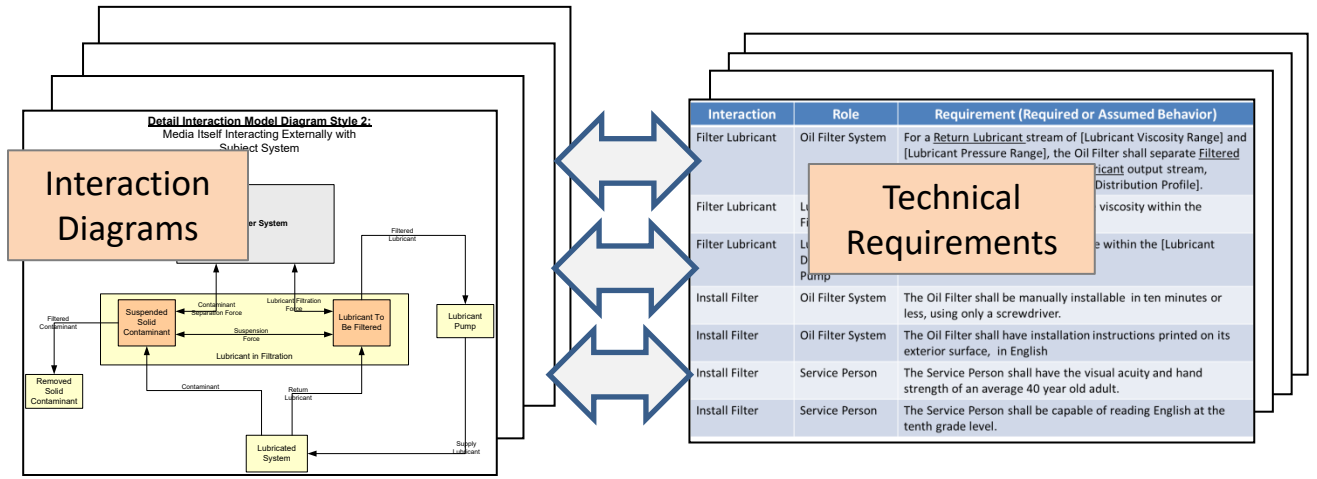
Three paths to the same Interactions



Inherent Relational Checks of Detail Level Model Completeness / Consistency (Model Metrics)

Requirements Statements are Transfer Functions

Interaction	Role	Requirement (Required or Assumed Behavior)
Filter Lubricant	Oil Filter System	For a Return Lubricant stream of [Lubricant Viscosity Range] and [Lubricant Pressure Range], the Oil Filter shall separate Filtered Contaminant particles from the Lubricant output stream, according to the [Filter Particle Size Distribution Profile].
Filter Lubricant	Lubricant in Filtration	The Lubricant in Filtration shall have viscosity within the [Lubricant Viscosity Range].
Filter Lubricant	Lubricant Distribution	The Pump shall maintain oil pressure within the [Lubricant Pressure Range]



Homework assignment 1

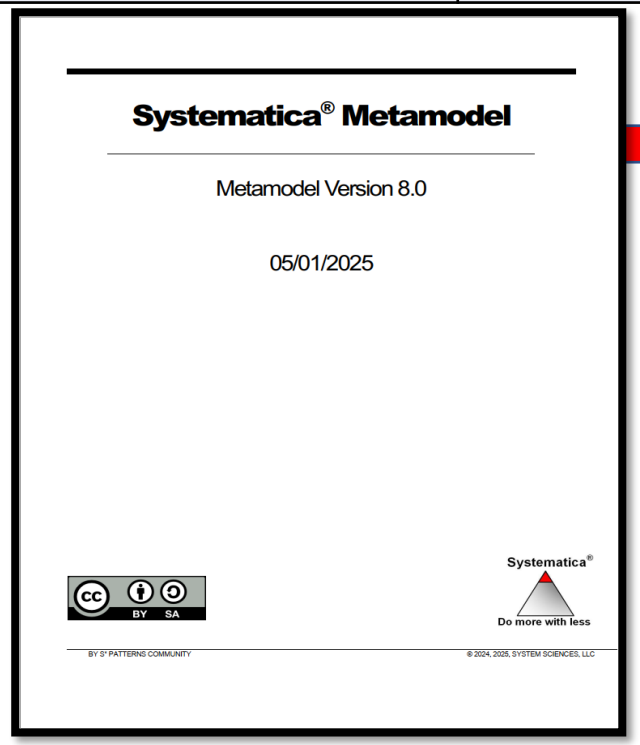
Save results for use in the next class HW

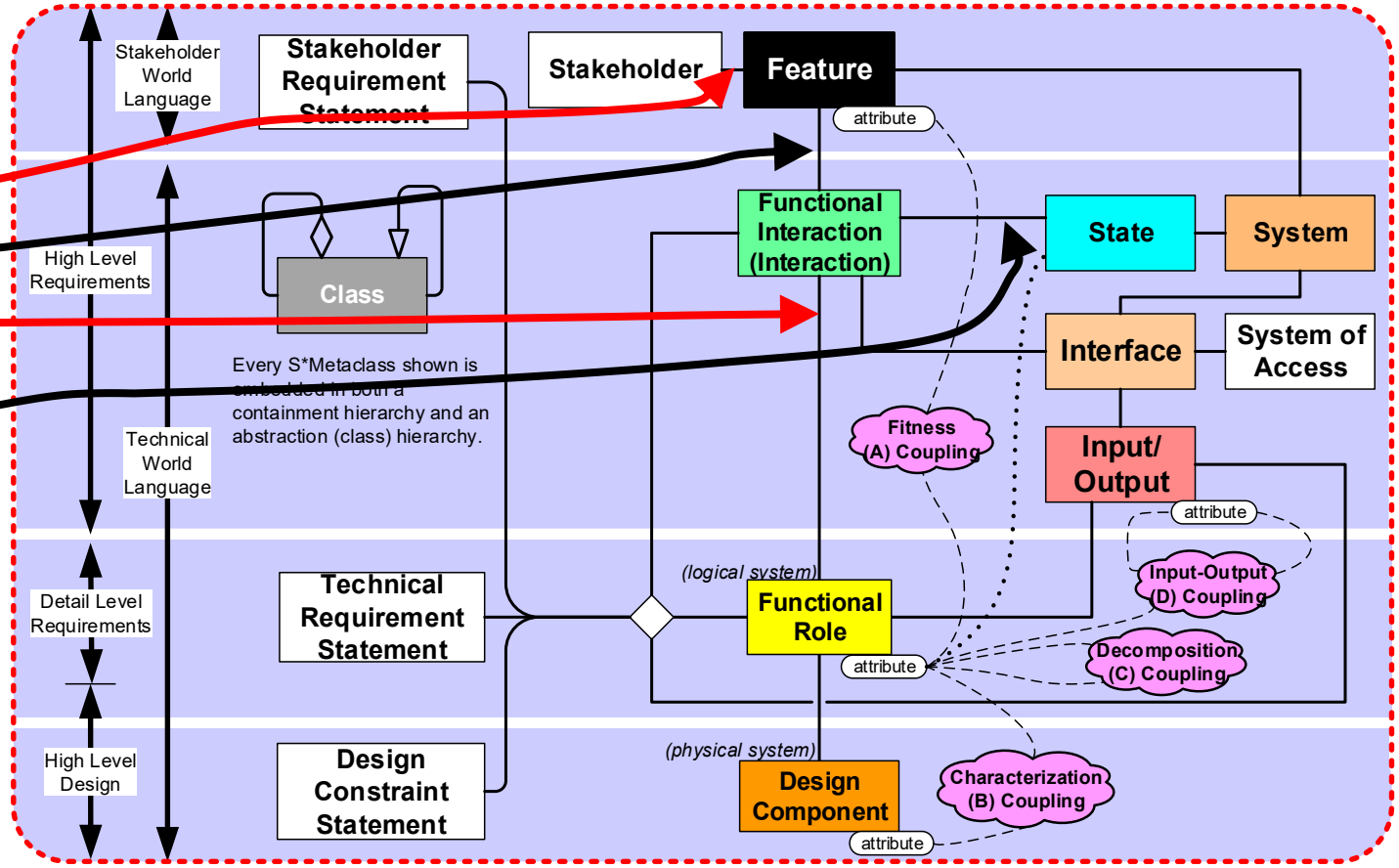
- Locate the pattern (shown below) of an engineered system of interest to <<company/org. name>> or its Customers, or the example teaching pattern:
  - Since we will be studying models of the system of engineering itself as a modeled system in later units, for now an “engineered system” will be used for modelling.
- In a Local Pattern Package, *containing the existing Pattern*, augment the high-level S*Pattern for the above, to include:
  - Domain System: Add more external Domain Actors
  - Add more Stakeholders and Stakeholder Features
  - Add more Interactions (external Black Box interactions only) and the external Actors they include (as logical systems)
  - Add more States
- Create Additional External Interaction relationships with the States, Features, and external Actors that you added:
  - Determine the completeness of your external Interactions set by their common coverage of States, Features, and External Actors
  - Are all the Features “satisfied” by their Interactions?
  - Are all the Actors “satisfied” by their Interactions?
  - Are all the States (think about Use Cases) “satisfied by their Interactions?

International Power Converter	International Power Converter Example Pattern xx May 2025.mdzip	<<Pattern/Model File Location>> Training materials

Homework assignment 1: The four types of S*Metarelationships that you will need to populate

Related S*Metaclass Pair	S*Relationship Formal Type
Feature → Stakeholder	Benefits
Feature → Interaction	Uses_Functional_Interaction
Interaction → Role	Has_Role
State → Interaction	Requires



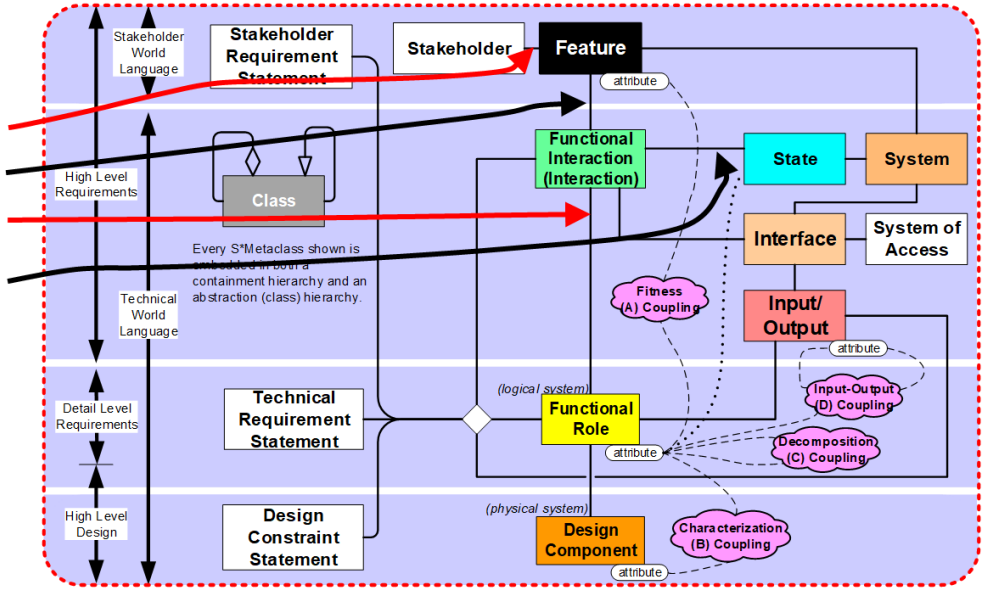


Informal S*Metamodel Diagram: Does <u>not</u> show the formal Types of S*Relationships it pictures.

←**<u>Formal S*Metamodel</u>**: Shows formal *<u>Types</u>* for all S*Metarelationships.

Homework assignment 1: Each tool provides various ways to populate, query, and view relationships.

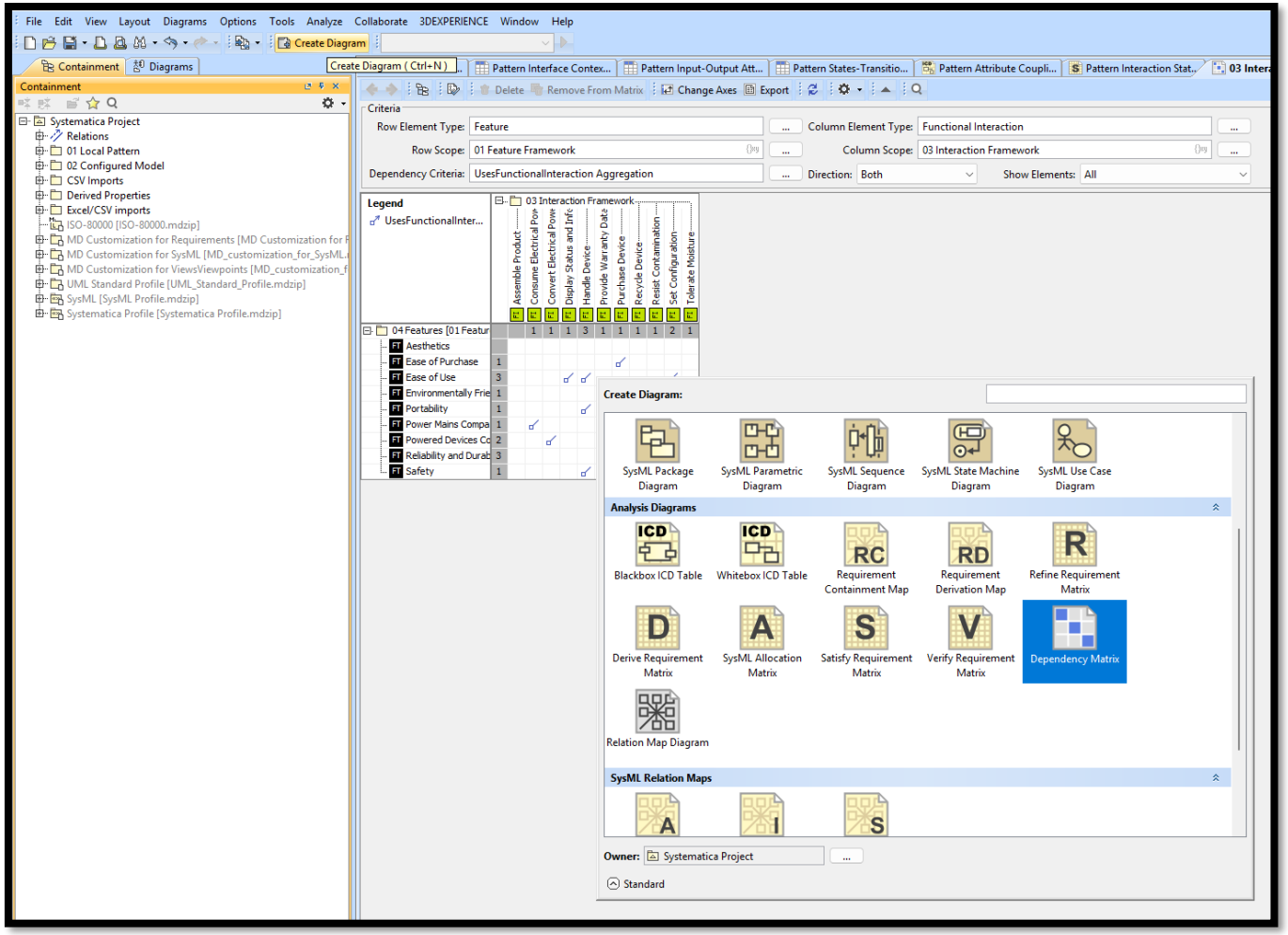
- Sometimes by (relational) diagrams.
- Sometimes by tables.
- Sometimes by browser.
- But in all these cases, we are talking about links (of various *types*) between classes.
- The *type of each relationship* connecting the classes is important to subsequent uses:
  - For queries, views, configuring patterns, etc.
  - Defined by the formal S*Metamodel document.
  - Sometimes resulting in SysML Stereotypes of the same name.



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

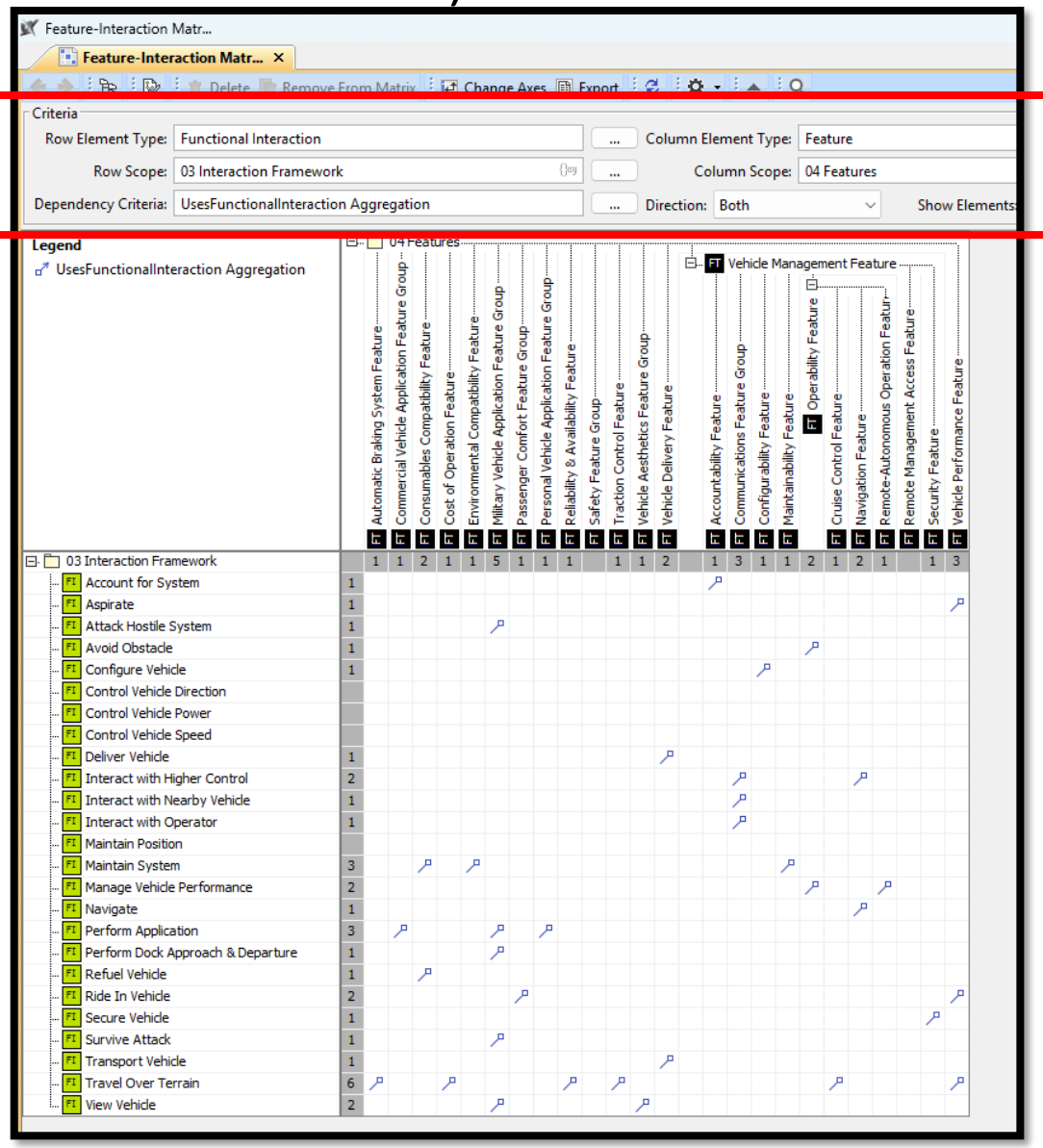
Homework assignment 1: Using Dependency Matrices to create Metarelationships

- The simplest way to create relationships is using a matrix.
- Select the Create Diagram option as shown.
- Choose a Dependency Matrix.
- Row Type, Row Scope, Column Type, Column Scope, and Dependency Criteria for the matrices are shown on subsequent slides.



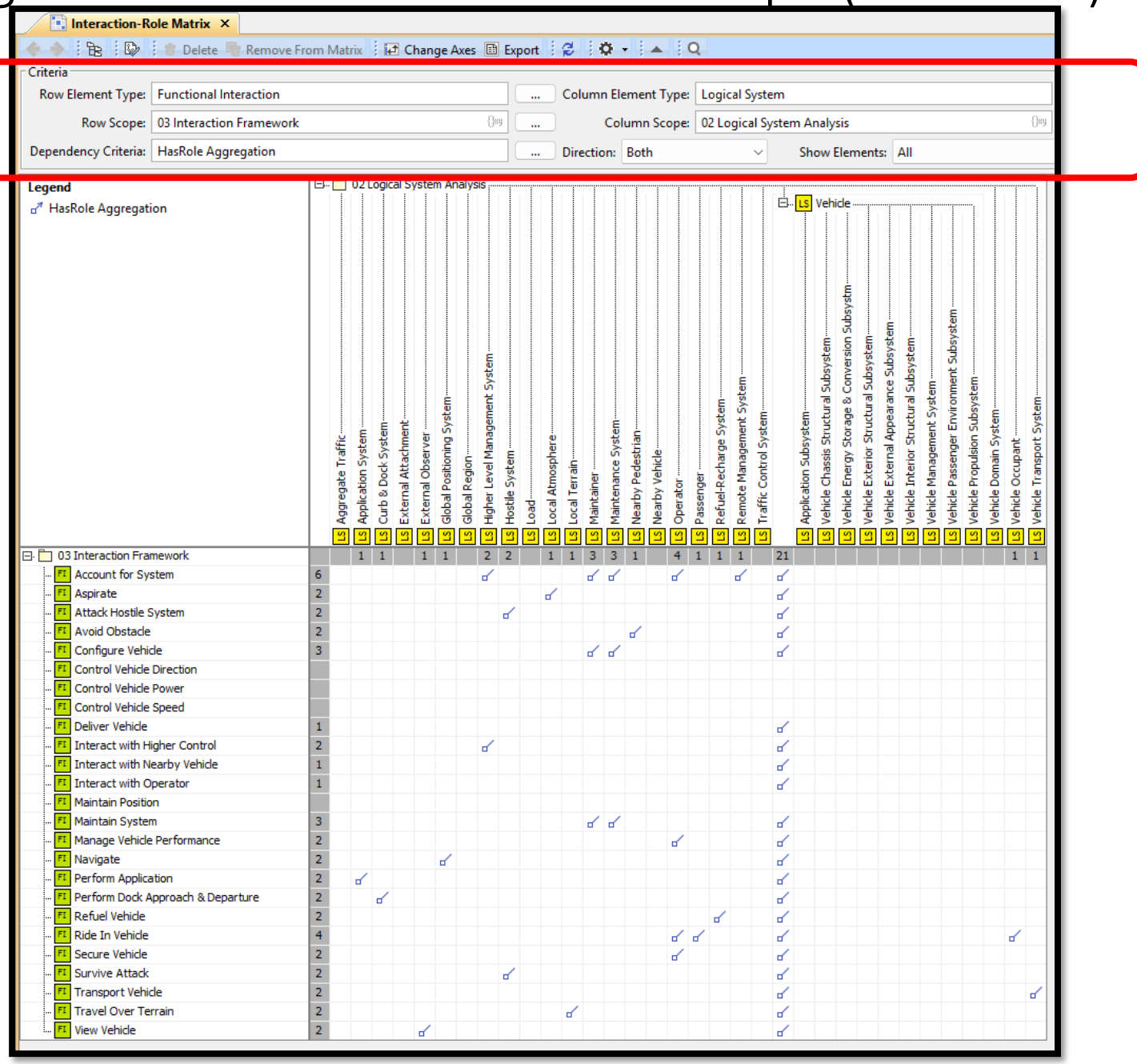
Homework assignment 1: Creating Feature-Interaction Relationships (Relationship type: Uses Functional Interaction)

- A simple way to create relationships is using a matrix: The tool auto-populates the row and column headings, and you use the intersection cells for relationships.
- Create a Dependency Matrix with Row Type, Row Scope, Column Type, Column Scope, and Dependency Criteria as shown.
- Right-click in a row-column intersection cell to create a new Feature-Interaction relationship.



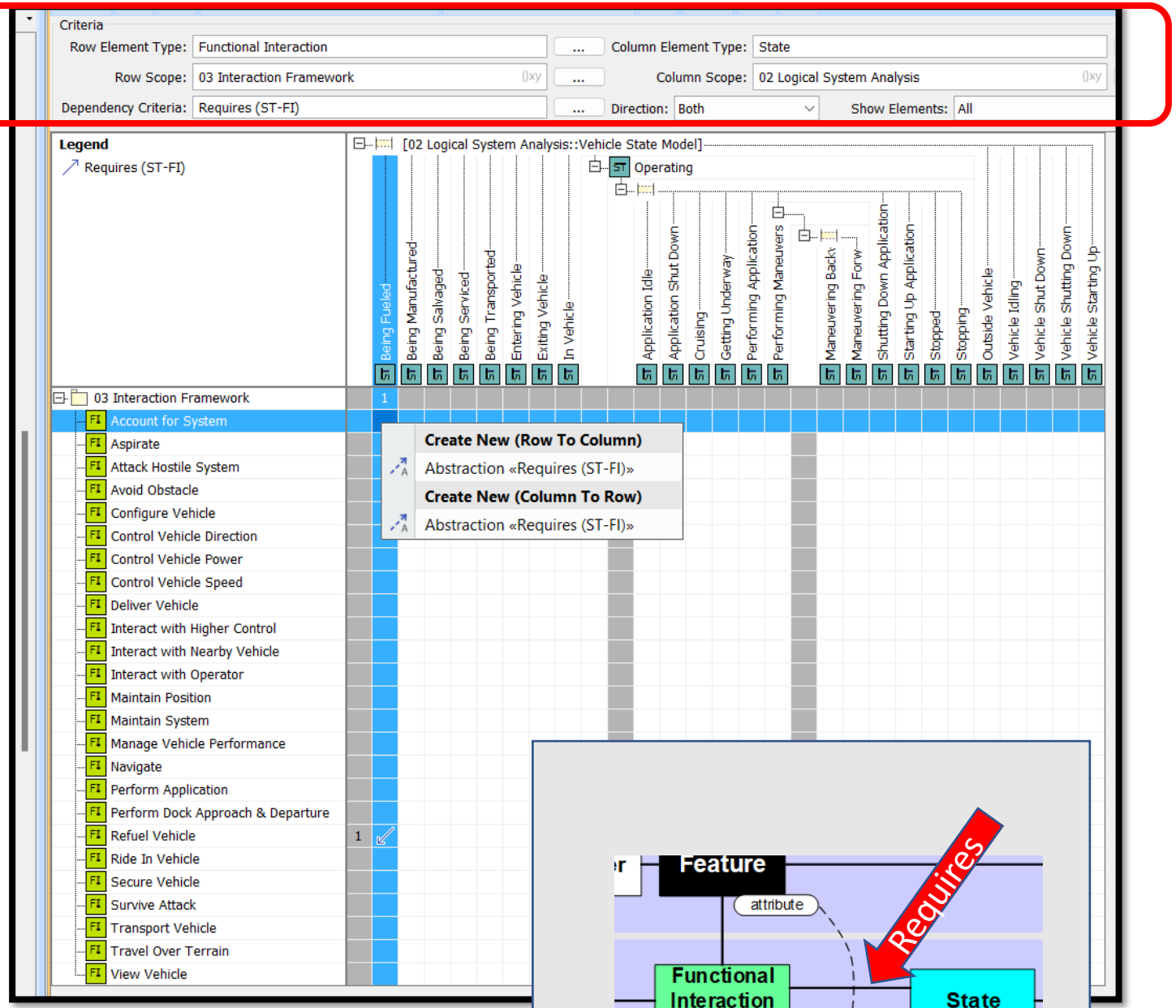
Homework assignment 1: Creating Interaction-Role Relationships (Has Role)

- A simple way to create relationships is using a matrix: The tool auto-populates the row and column headings, and you use the intersection cells for relationships.
- Create a Dependency Matrix with Row Type, Row Scope, Column Type, Column Scope, and Dependency Criteria as shown.
- Right-click in a row-column intersection cell to create a new Interaction-Role relationship.

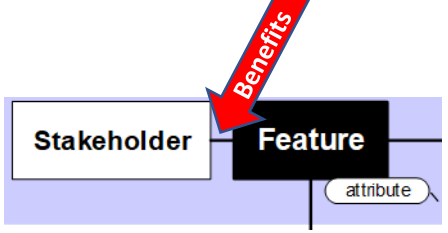


Homework assignment 1: Creating State-Interaction Relationships (Requires)

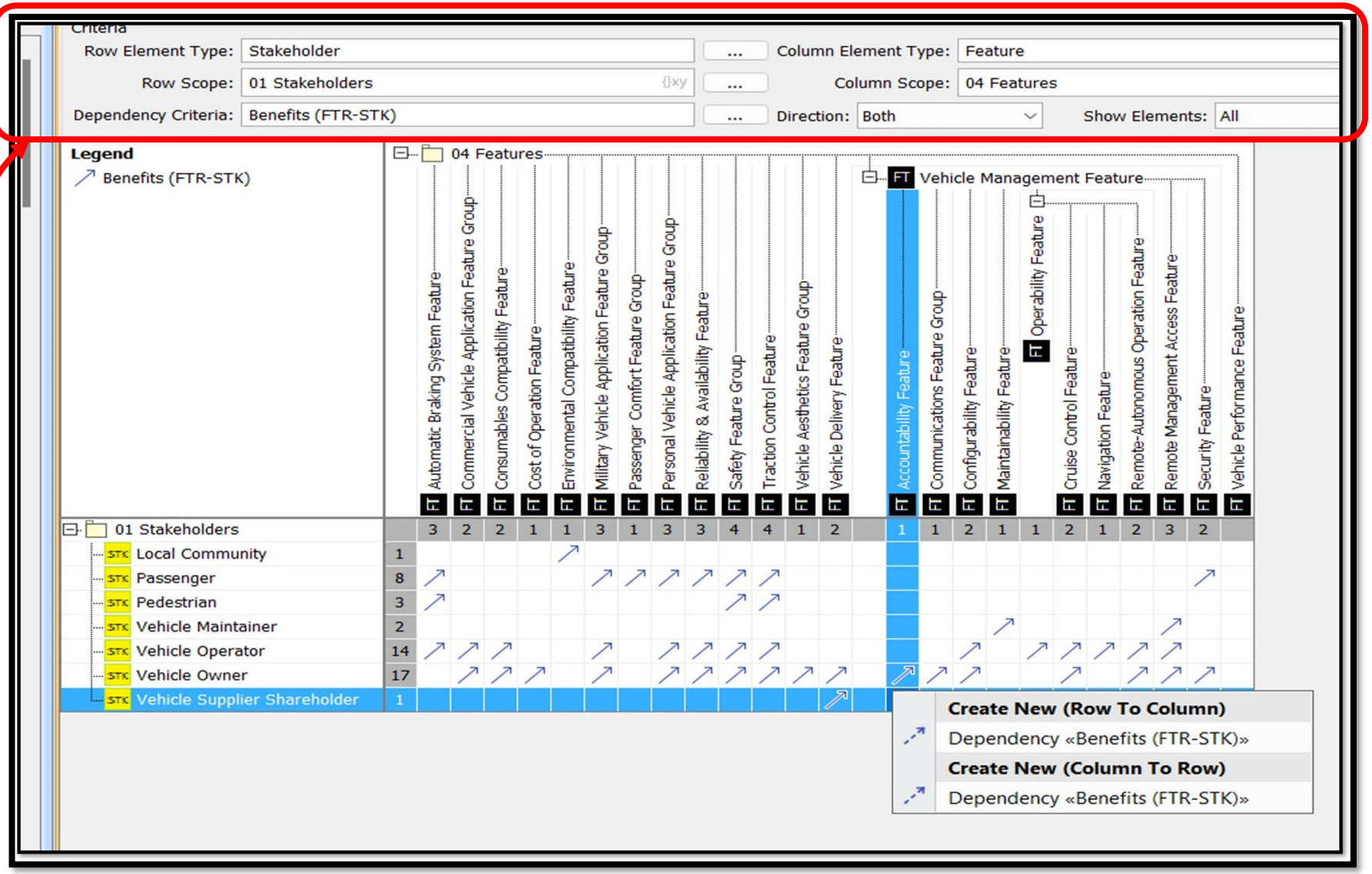
- A simple way to create relationships is using a matrix: The tool auto-populates the row and column headings, and you use the intersection cells for relationships.
- But that depends on whether the tool supports matrix use for the type of relationship needed.
- Cameo changes that support between V19 (which supports Dependency) and V202x (which adds support for Aggregation);
- Create a Dependency Matrix with Row Type, Row Scope, Column Type, Column Scope, and Dependency Criteria as shown.
- Right-click in a row-column intersection cell to create a new State-Interaction relationship.



Homework assignment 1: Creating Stakeholder-Feature Relationships (Benefits)



- A simple way to create relationships is using a matrix.
- Create a Dependency Matrix with Row Type, Row Scope, Column Type, Column Scope, and Dependency Criteria as shown.
- Right-click on row-column intersections to create new Stakeholder-Feature relationships.



Discussion, Q&A.

-
-
-
-
-
-



Additional references

1. System Interactions: Making the Heart of Systems More Visible: https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:system_interactions--making_the_heart_of_systems_more_visible_v1.2.2.pdf
2. Systematica Metamodel: https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:systematica_8_metamodel_v8.0.61.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. A Systems Engineering Insight from Ancient Navigators: Maps vs. Itineraries: https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:maps_vs._itineraries_v1.3.1.pdf
5. Guide to Systematica Patterns in Cameo Systems Modeler: https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:guide_to_systematica_patterns_in_cameo_systems_modeler_v1.4.6.pdf