32nd Annual INCOSE international symposium

hybrid event

Detroit, MI, USA
June 25 - 30, 2022

You Can't Touch This!: Logical Architectures in MBSE and the UAF

Presenters

- Matthew Hause,
  Principal
  SSI – System Services Innovation

- Lars-Olof Kihlström,
  Principal Consultant
  Syntell AB

Agenda

- What is a Logical Architecture?
- Definition and Process
- Example Model
- Benefits
- Known Resources
- Conclusion

Abstract

- Logical or abstract architectures are an essential concept in systems engineering. They are included in the systems engineering handbook, the OOSEM process, the SEBOK, several modeling languages, and the ISO 15288 process definition. A logical architecture is a solution-independent model of the problem domain used to understand “what” needs to be done, while avoiding defining “how” it will be done. The logical architecture includes all the related logical elements without constraining the architecture to a particular technology or environment. It traces to the physical architecture which defines how to implement the architecture using specific technologies. Logical architectures can be defined using MBSE languages such as the systems modeling language (SysML) and is implicit in architecture frameworks such as DoDAF, MODAF, NAF and their implementation in UAF using SysML. DoDAF and MODAF call this the Operational set of views. NAF has recently changed the title of the Operational views to Logical views to further emphasize the purpose of the views. This paper will define the benefits of using a logical architecture and provide guidance on how it can be implemented.

What is a Logical Architecture?

- The logical architecture is a model that is used to provide a detailed description of the system without defining the system technology or environment.
- Logical architecture does not contain solution specific elements
- Rule of Thumb: “If you can hit it with a hammer, it should not be in the logical model.”
- Selection of the contents of a logical architecture is a balancing act that needs careful consideration

INCOSE Systems Engineering Handbook

- “Logical models, also referred to as conceptual models represent logical relationships about the system such as whole-part relationship, an interconnection relationship between parts, or a precedence relationship between activities to name a few.”
- UAF provides Strategic/Capability, Operational and Services Views that provide a solution independent expression of stakeholder requirements and solution independent services specifications.
- “The models to be used are those that best address key stakeholder concerns. Logical models may include the functional, behavioral, or temporal models.” (INCOSE, 2015)

Systems Engineering Body of Knowledge (SEBOK)

- “The logical architecture defines system boundary and functions, from which more detailed system requirements can be derived.
- The starting point for this process may be to identify functional requirements from the stakeholder requirements and to use this to start the architectural definition, or to begin with a high-level functional architecture view and use this as the basis for structuring system requirements.
- The exact approach taken will often depend on whether the system is an evolution of an already understood product or service, or a new and unprecedented solution.
- However, when the process is initiated, it is important that the stakeholder requirements, system requirements, and logical architecture are all complete, consistent with each other, and assessed together at the appropriate points in the systems life cycle model.” (INCOSE, 2021)

OOSEM

- The Object-Oriented Systems Engineering Methodology (OOSEM) defines logical architecture definition as follows:
- “This activity is part of the system architecture design that includes decomposing the system into logical components that interact to satisfy system requirements.
- The logical components are abstractions of components that implement the system, which perform the system functionality without imposing implementation constraints.
- An example of a logical component is a user interface that may be realized by a web browser or display console or an entry/exit sensor that may be realized by an optical sensor.
- The logical architecture serves as an intermediate level of abstraction between the system requirements and the physical architecture that can reduce the impact of both requirements and technology changes on the physical design.”

Benefits

- The Logical architectures provide the following benefits:
  - To ensure that the system design meets the stakeholder requirements
  - Providing a bridge from requirements to the solution
  - Preventing “solutioneering” rather than engineering
  - Capturing the main concepts of the architecture prior to defining a solution
  - Reflect the true location of the system of interest (SOI) in relation to stakeholders and other systems
  - Define an objective set of concepts and measures for trade-off analysis of solutions
  - Providing business objects for service definition
  - Bridging the gap between capabilities and implementation
  - Facilitating impact analysis and traceability
  - Minimizing the effect of changes in the architecture.
  - Spurs innovation by forcing a rethink of the problem

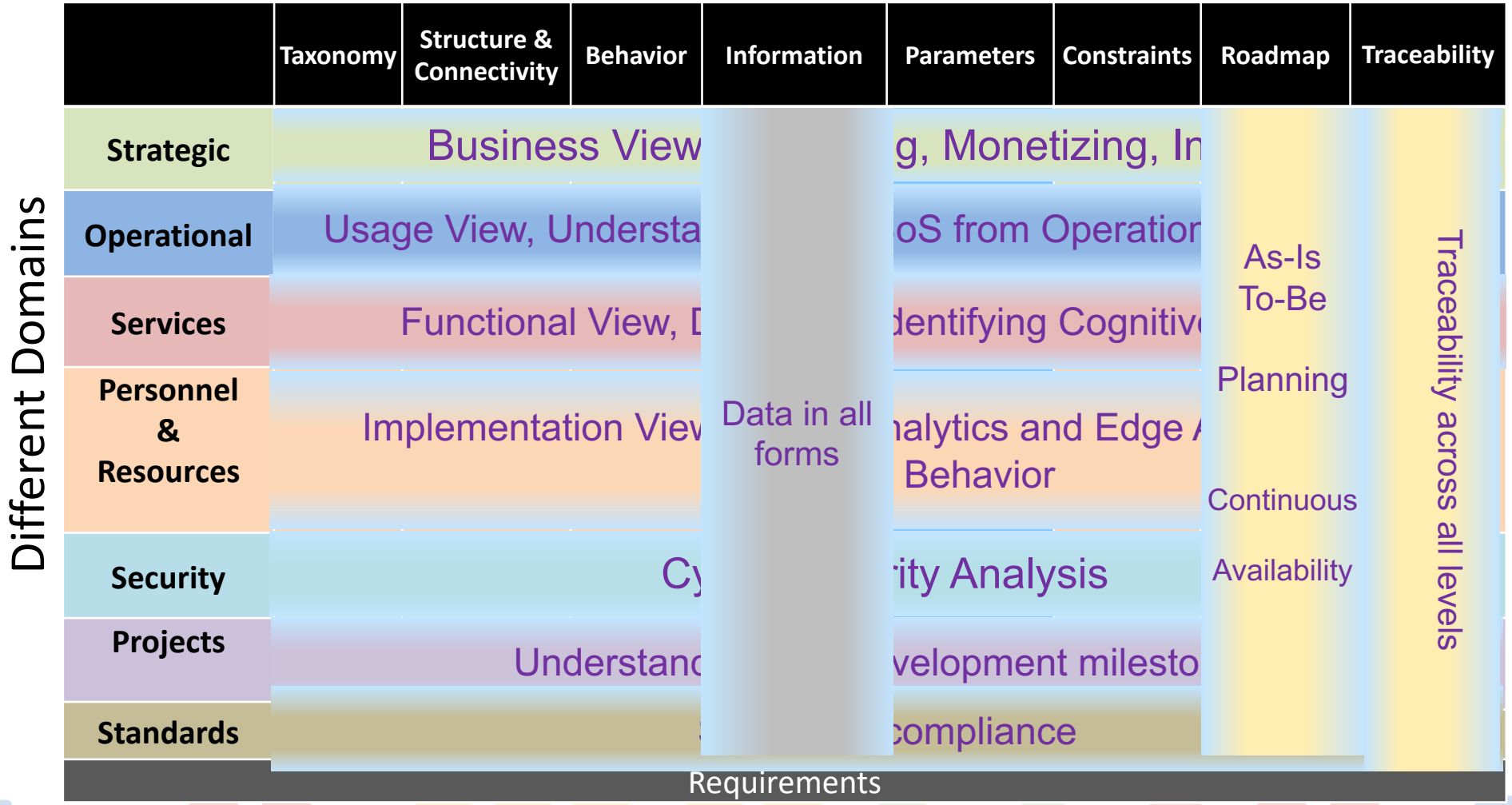
Unified Architecture Framework (UAF)

- The UAF is used for defining system architectures and system of systems architectures
- It is focused on the scope, needs, strategy, expectations, stakeholders, and long-term plans
- It is built on SysML, so has built-in traceability to system development in SysML.

Great for large organizations to figure out what they are doing and why.

The Unified Architecture Framework Grid

Standard means of expression – model kinds



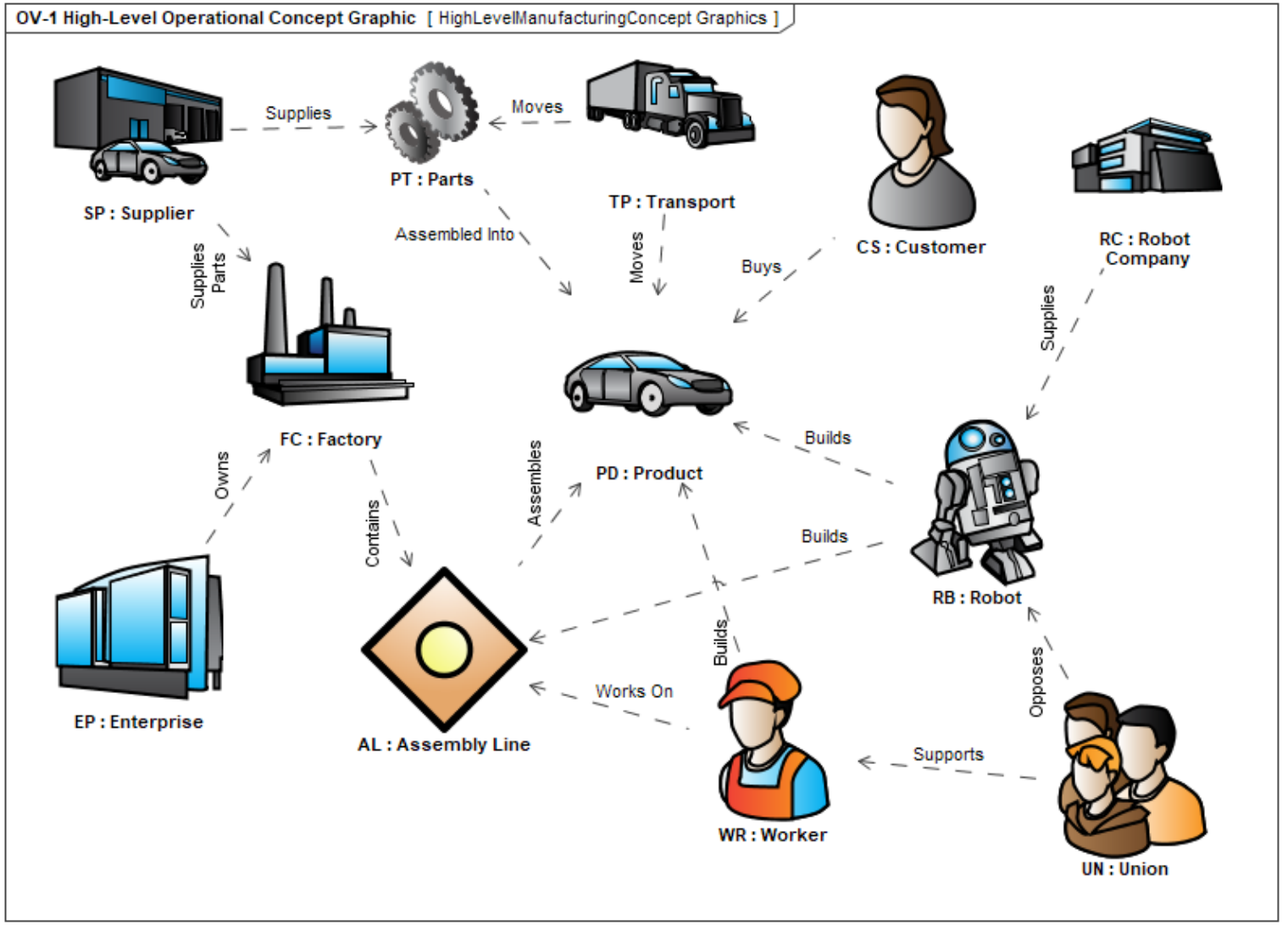
	Taxonomy Tx	Structure Sr	Connectivity Cn	Processes Pr	States St	Interaction Scenarios Is	Information If	Parameters Pm	Constraints Ct	Roadmap Rm	Traceability Tr
Metadata Md	Metadata Taxonomy Md-Tx	Architecture Viewpoints [a] Md-Sr	Metadata Connectivity Md-Cn	Metadata Processes [a] Md-Pr	-	-			Metadata Constraints [a] Md-Ct		Metadata Traceability Md-Tr
Strategic St	Strategic Taxonomy St-Tx	Strategic Structure St-Sr	Strategic Connectivity St-Cn	-	Strategic States St-St	-			Strategic Constraints St-Ct	Strategic Deployment, St-Rm Stategic Phasing St-Rm	Strategic Traceability St-Tr
Operational 0p	Operational Taxonomy Op-Tx	Operational Structure Op-Sr	Operational Connectivity Op-Cn	Operational Processes Op-Pr	Operational States Op-St	Operational Interaction Scenarios Op-Is			Operational Constraints Op-Ct	-	-
Services Sv	Service Taxonomy Sv-Tx	Service Structure Sv-Sr	Service Connectivity Sv-Cn	Service Processes Sv-Pr	Service States Sv-St	Service Interaction Scenarios Sv-Is	Conceptual Data Model,	Environment Pm-En	Service Constraints Sv-Ct	Service Roadmap Sv-Rm	Service Traceability Sv-Tr
Personnel Pr	Personnel Taxonomy Pr-Tx	Personnel Structure Pr-Sr	Personnel Connectivity Pr-Cn	Personnel Processes Pr-Pr	Personnel States Pr-St	Personnel Interaction Scenarios Pr-Is	Logical Data Model,		Competence, Drivers, Performance Pr-Ct	Personnel Availability, Personnel Evolution, Personnel Forecast Pr-Rm	Personnel Traceability Pr-Tr
Resources Rs	Resource Taxonomy Rs-Tx	Resource Structure Rs-Sr	Resource Connectivity Rs-Cn	Resource Processes Rs-Pr	Resource States Rs-St	Resource Interaction Scenarios Rs-Is	Physical schema, real world results		Resource Constraints Rs-Ct	Resource evolution, Resource forecast Rs-Rm	Resource Traceability Rs-Tr
Security Sc	Security Taxonomy Sc-Tx	Security Structure Sc-Sr	Security Connectivity Sc-Cn	Security Processes Sc-Pr	-	-		Measurements Pm-Me	Security Constraints Sc-Ct	-	-
Projects Pj	Project Taxonomy Pj-Tx	Project Structure Pj-Sr	Project Connectivity Pj-Cn	-	-	-			-	Project Roadmap Pj-Rm	Project Traceability Pj-Tr
Standards Sd	Standard Taxonomy Sd-Tx	Standards Structure Sd-Sr	-	-	-	-			-	Standards Roadmap Sr-Rm	Standards Traceability Sr-Tr
Actuals Resources Ar		Actual Resources Structure, Ar-Sr	Actual Resources Connectivity, Ar-Cn	Simulation [b]					Parametric Execution/ Evaluation [b]	-	-
Dictionary * Dc											
Summary & Overview SmOv											
Requirements Rq											

Example Automotive Factory Model

- Problem Statement: Powerhouse Engines (PE Inc.) is an automotive supply company providing internal combustion engines. PE Inc. finds that it has gradually become less competitive over the years largely due to their outdated technology and largely manual processes. Foreign and domestic competitors have started to cut into their business and the stakeholders are concerned that the company's loss of market share will accelerate and that they will eventually become insolvent. To combat this, the shareholders have proposed an investigation into strategies and technologies such as Augmented reality, Robotic assembly systems, 5G, AI, Additive manufacturing, outsourcing of select manufacturing and IT systems, Battery technology, Data analytics, Hybrid/electric engines, etc. These technologies will be rolled out over a 3-phase technology deployment plan.

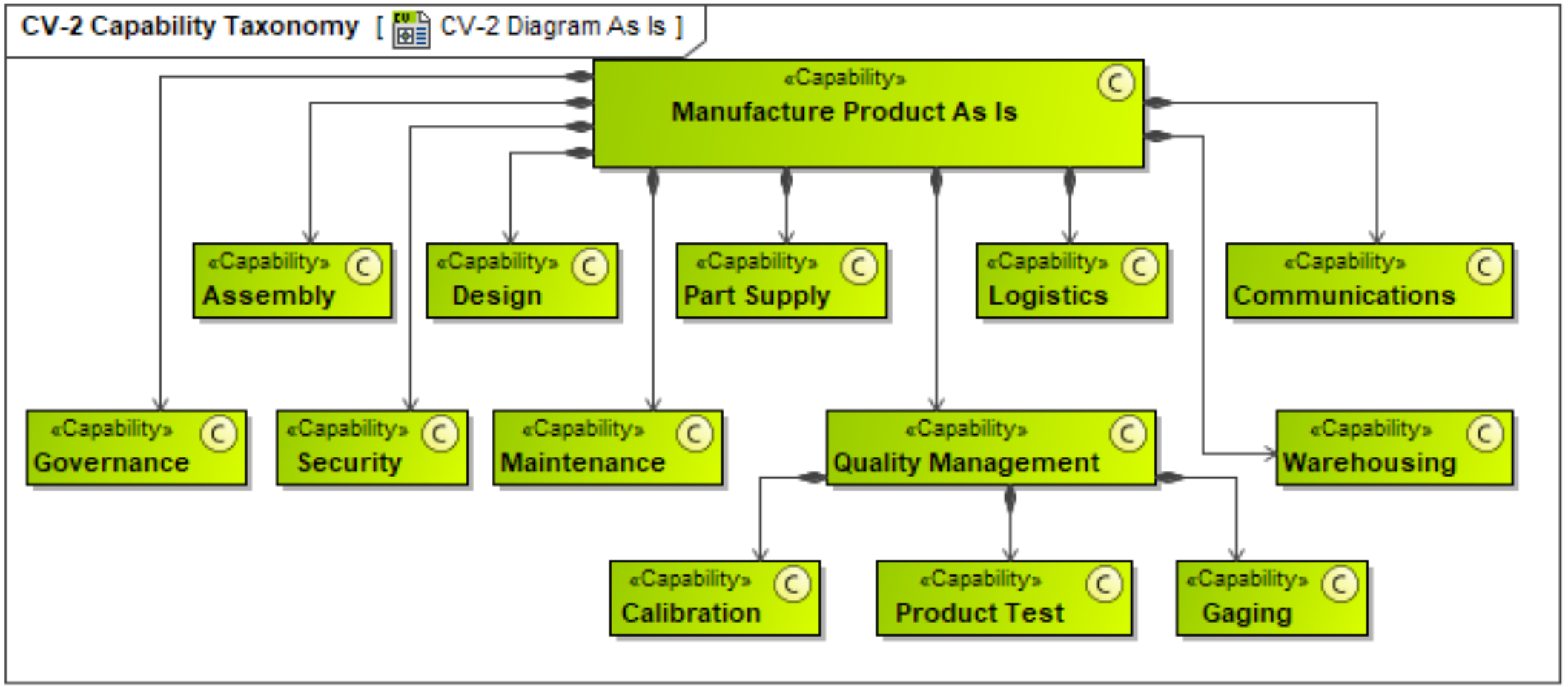
High Level Manufacturing Concept for Powerhouse Engines

- Solution independent concepts in the architecture
- The part supplier could be an external company, an internal casting department, or an in-house 3D printer.
- All supply parts, and each has advantages and disadvantages regarding supply chain delays, cost, flexibility, etc.
- All 3 will be deployed over the 3 phases of technology introduction.



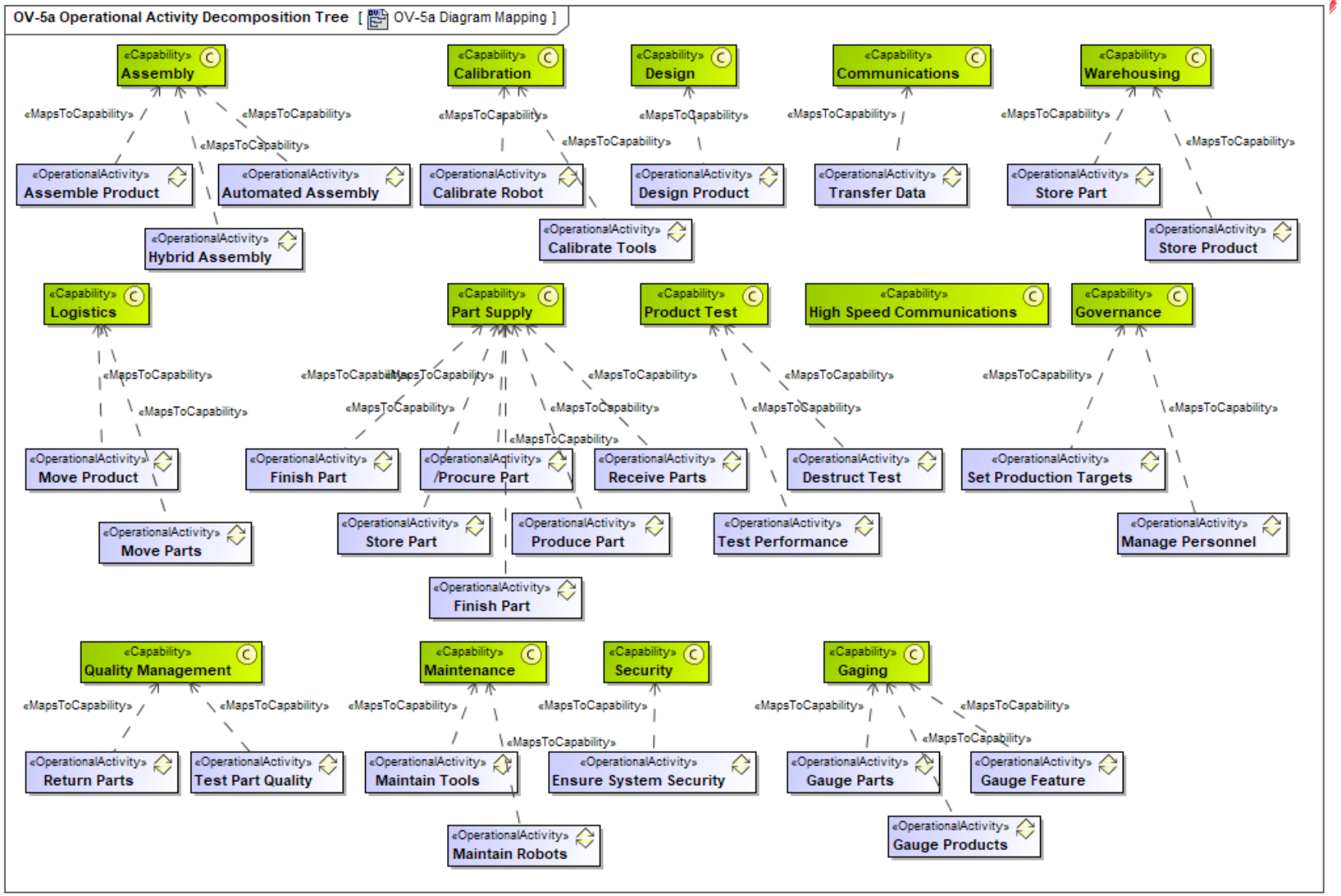
Powerhouse Engines Enterprise Capabilities

- Defines what the enterprise can do, not how it does it.
- Linked to effects that implementing systems accomplish



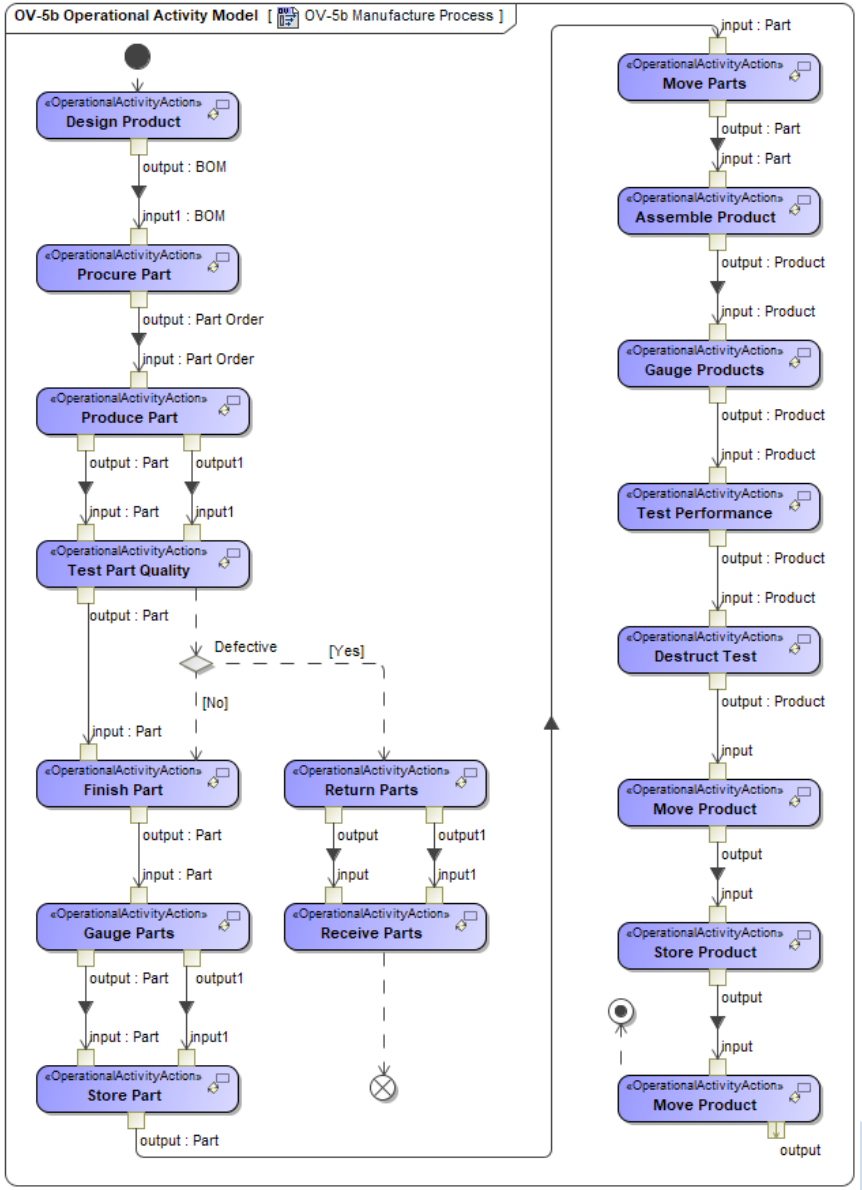
Capability Mapping to Operational Activities

- Operational activities are solution-independent behaviors that realize the capability
- These are further defined in activity diagrams



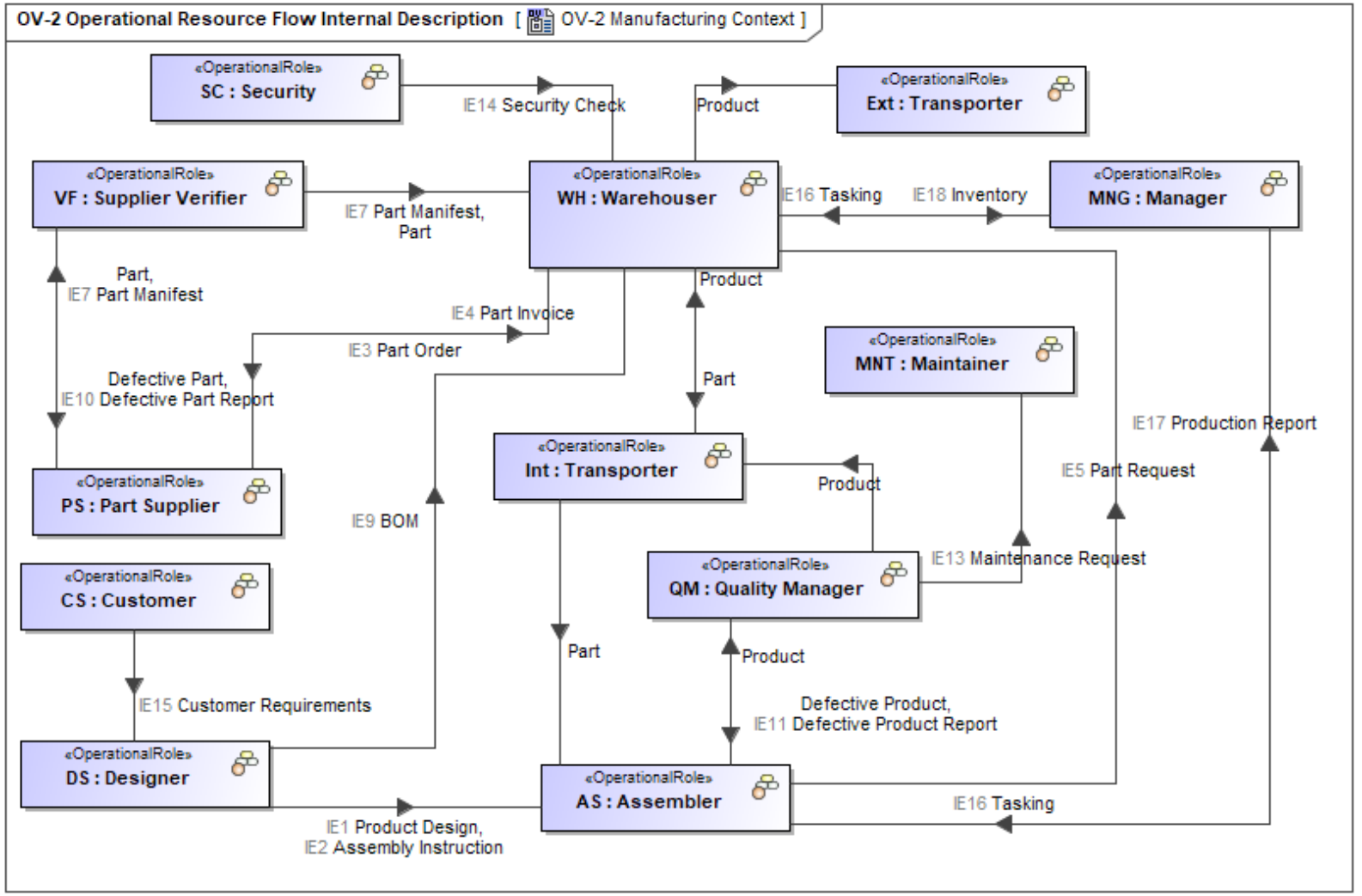
Manufacture Process Operational Activity Diagram

- Operational activities are further decomposed into steps.
- They are then placed in order
- Inputs & outputs for each activity are defined.
- Logical controls, (decision, fork, merger, join) are added.
- Signals and timing can also be defined.
- Swimlanes are added after the performers have been defined

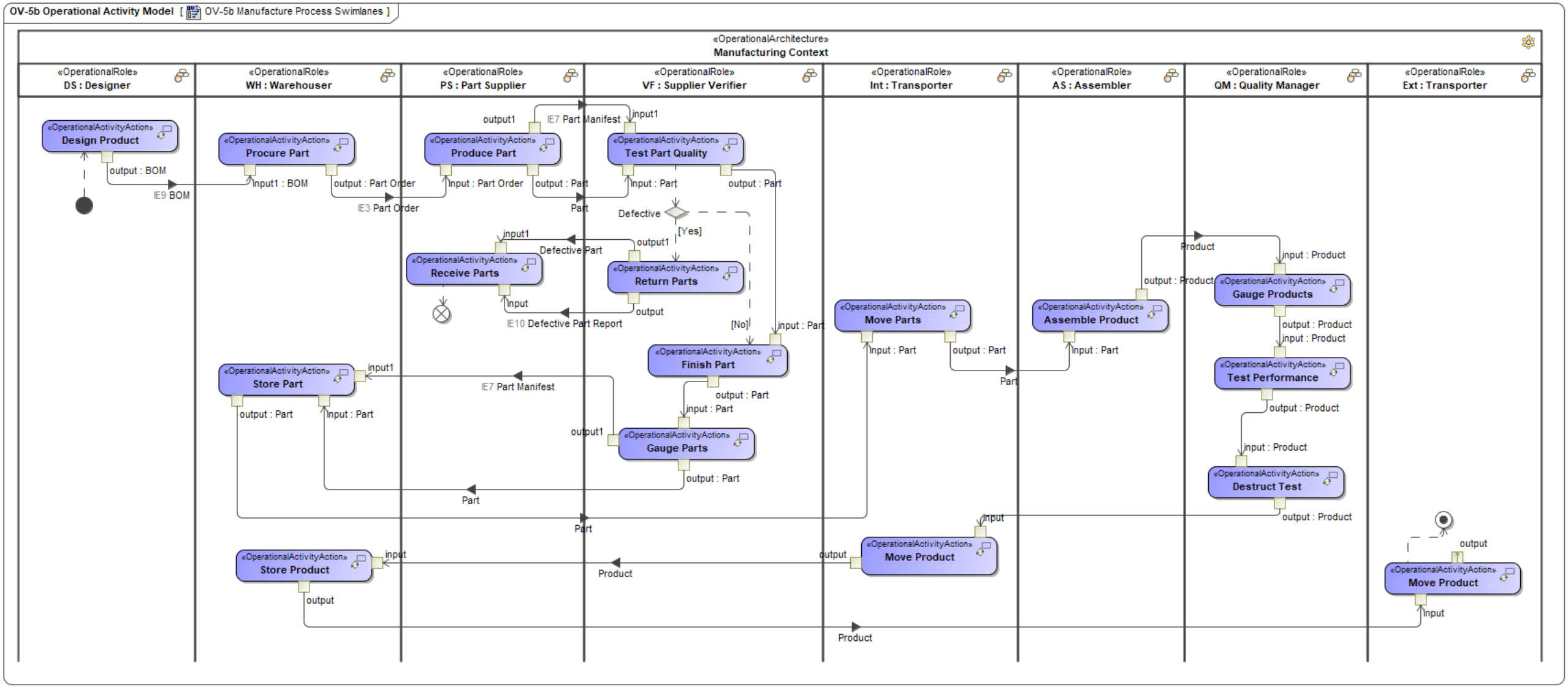


Manufacturing Logical Performers

- Operational activities are grouped together to define operational performers
- Deriving performers from their activities concentrates on behavior before structure
- Helps prevent “Solutioneering”



Manufacture Process with Swimlanes

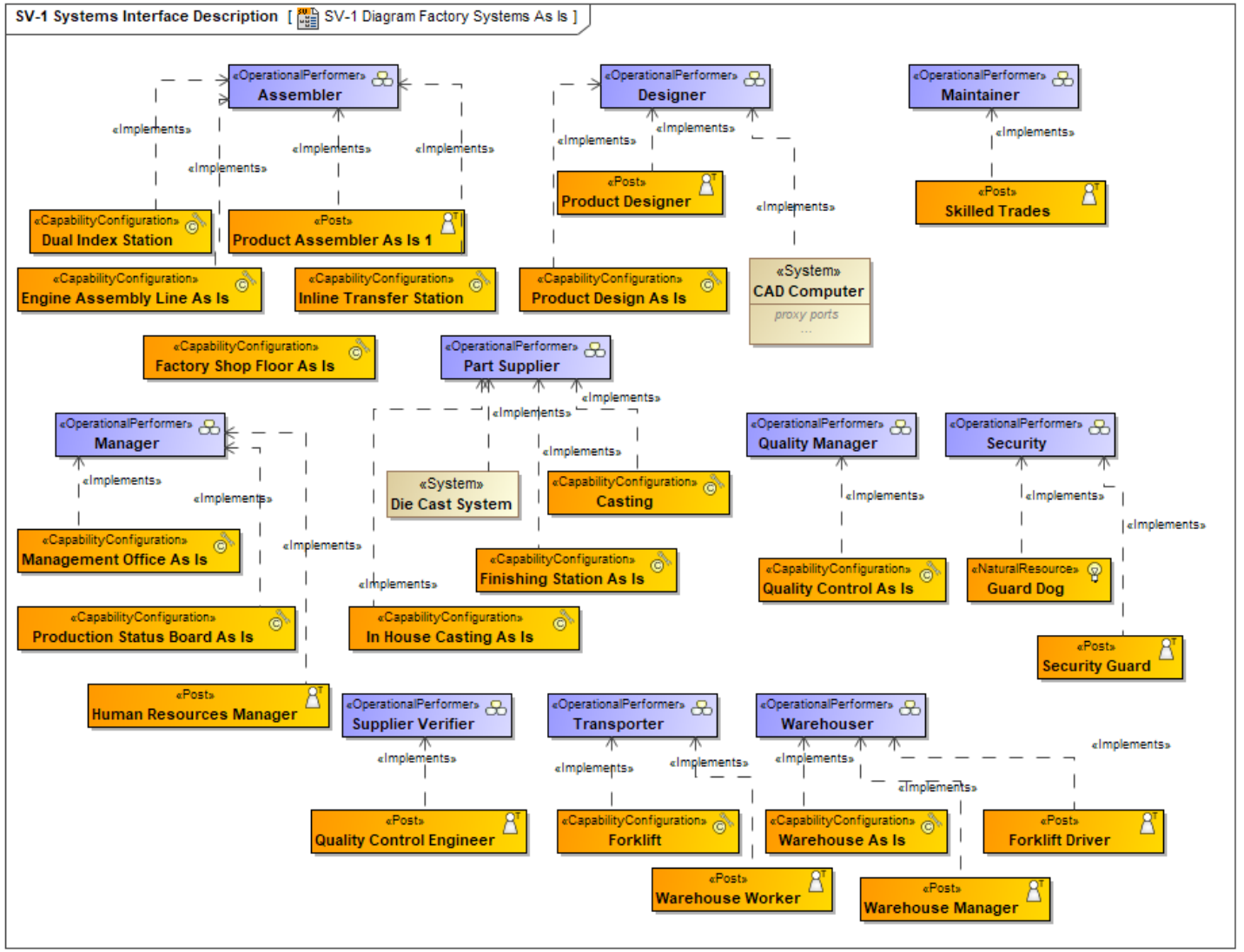


Solution Independent

- The exchanges in the previous slides are non-specific.
  - A product is assembled using parts rather than a specific type of engine made of specific parts.
  - Product design and assembly instructions formats are not specified.
  - This provides flexibility for the solution architecture.
  - The logical architecture is valid for all 3 solution phases
    - The As-Is solution manufactures internal combustion engines mostly using manual processes.
    - Phase I adds high speed wireless comms, digital data exchanges, instructions, outsourced services.
    - Phase II manufactures electric engines using robotic systems, AI, and parts created using 3D printing.
- Behaviors are also solution-independent
  - Test part, procure part, assemble part, move part, design product, etc.
  - These are implements by solution-specific function corresponding to the implementing systems.

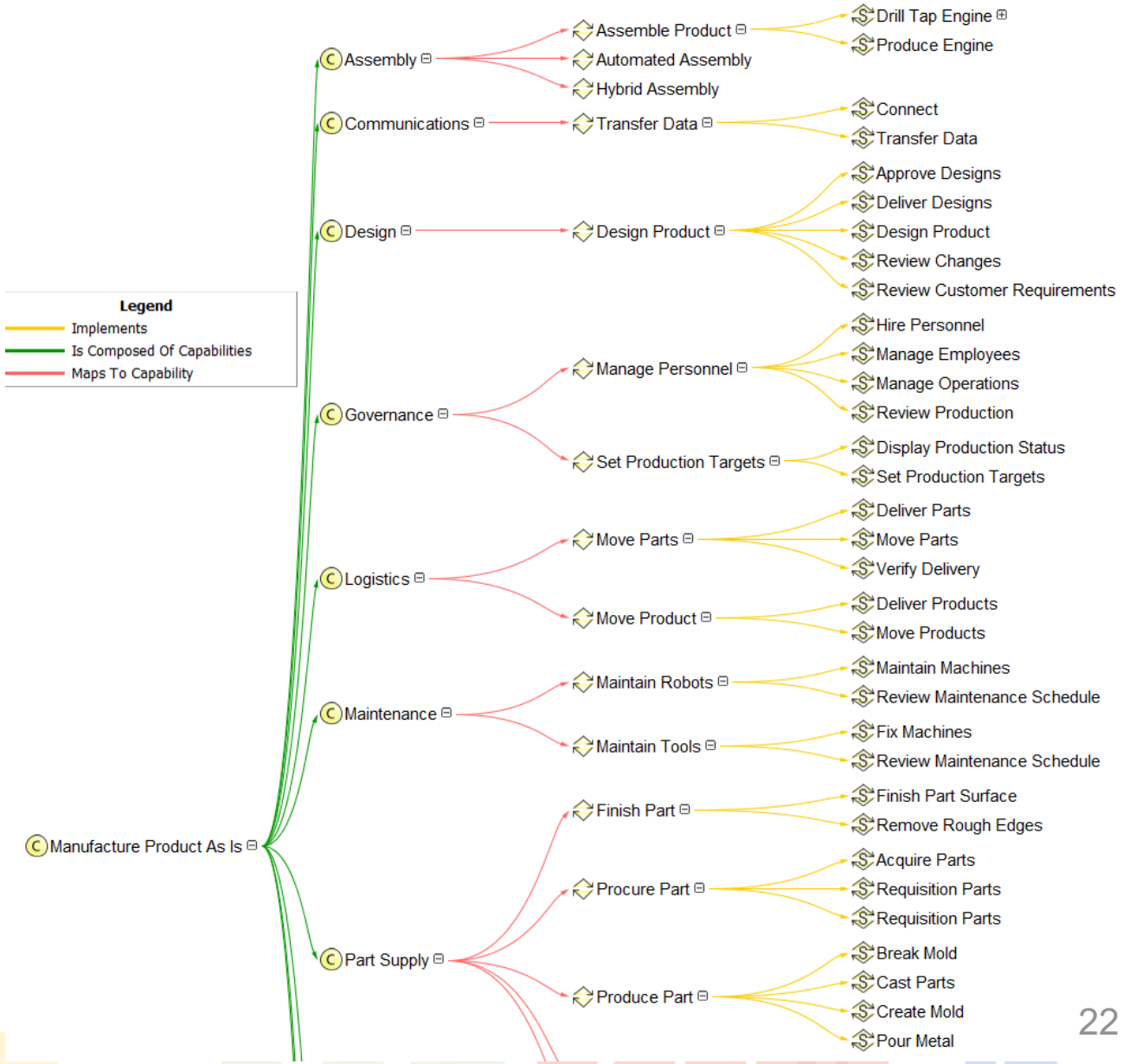
Mapping Between Operational & Resource Elements

- "Implements" relationship shows implementing systems and behavior
- Can be added between behavior, structure, data, interactions, etc.
- Demonstrates that the abstract is made concrete
- Mapping tables and diagrams can be generated.



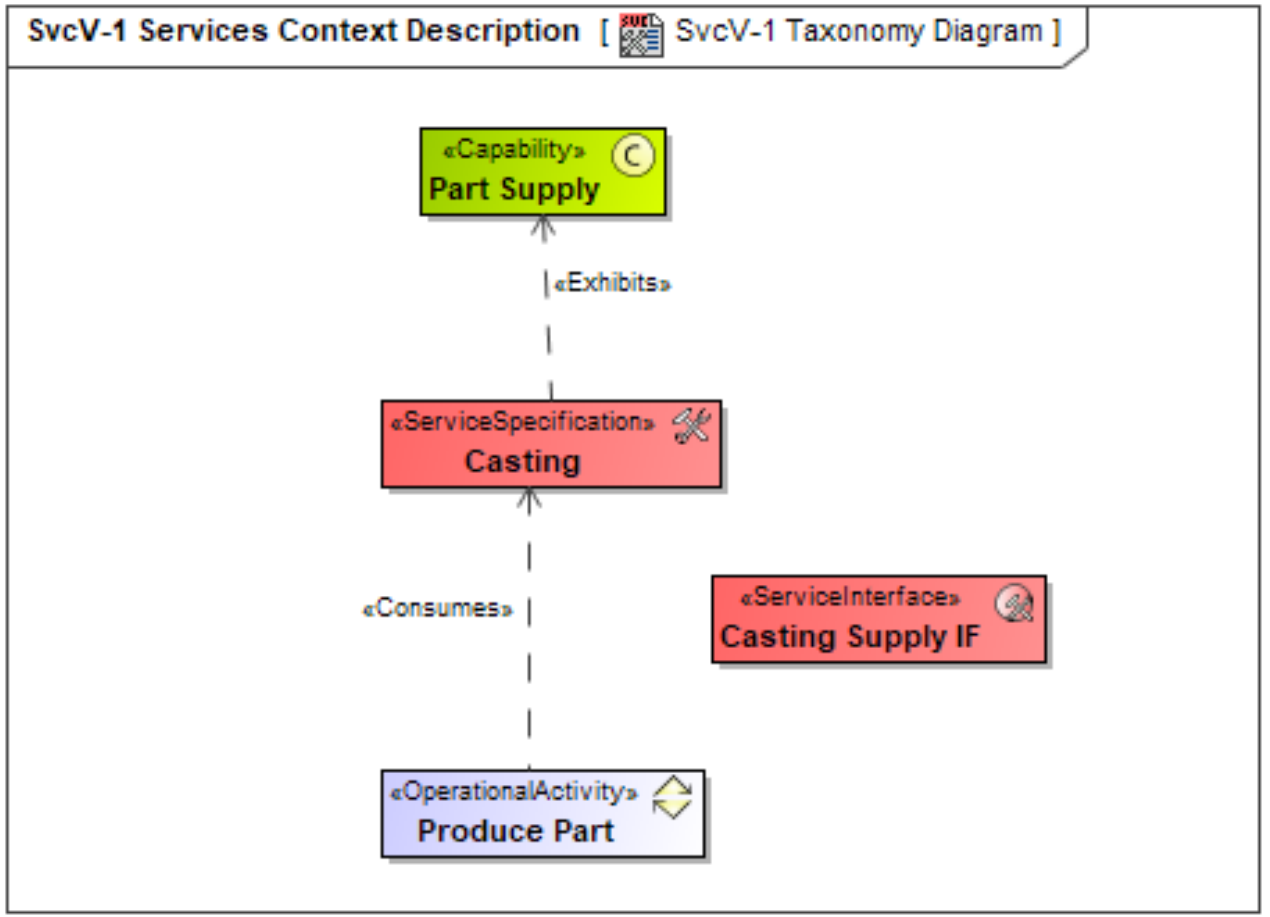
Enterprise Behavioral Mapping

- Auto generated traceability diagram from capability to owned capability to operational activity to resource function
- Identifies “orphan” elements that are not implemented
  - Automated and hybrid assembly



Service Mapping Between Capabilities and Operational Elements

- Service elements are a lower-level solution independent view
- Services allow the operational layer to be developed without impacting on the resource layer by well-defined service interfaces.
- The resource layer can also develop on its own without impacting on the operational layer provided that the service interfaces are untouched.
- Services define KPIs and other performance attributes.



A Bridge from Requirements to Solution

- Systems Engineers progress from system requirements through an intermediate model of logical architecture to allocate the elements of the logical architecture model to candidate physical architecture models.
- System requirements and logical architecture models share many characteristics, as they are both organized on functional lines, independently of the implementation.
- Models can also be made of the requirements
- Design decisions and technological solutions are selected according to performance criteria and non-functional requirements.
- Creating logical architecture models facilitates the validation of functional, behavioral, and temporal properties of the system against the system requirements.

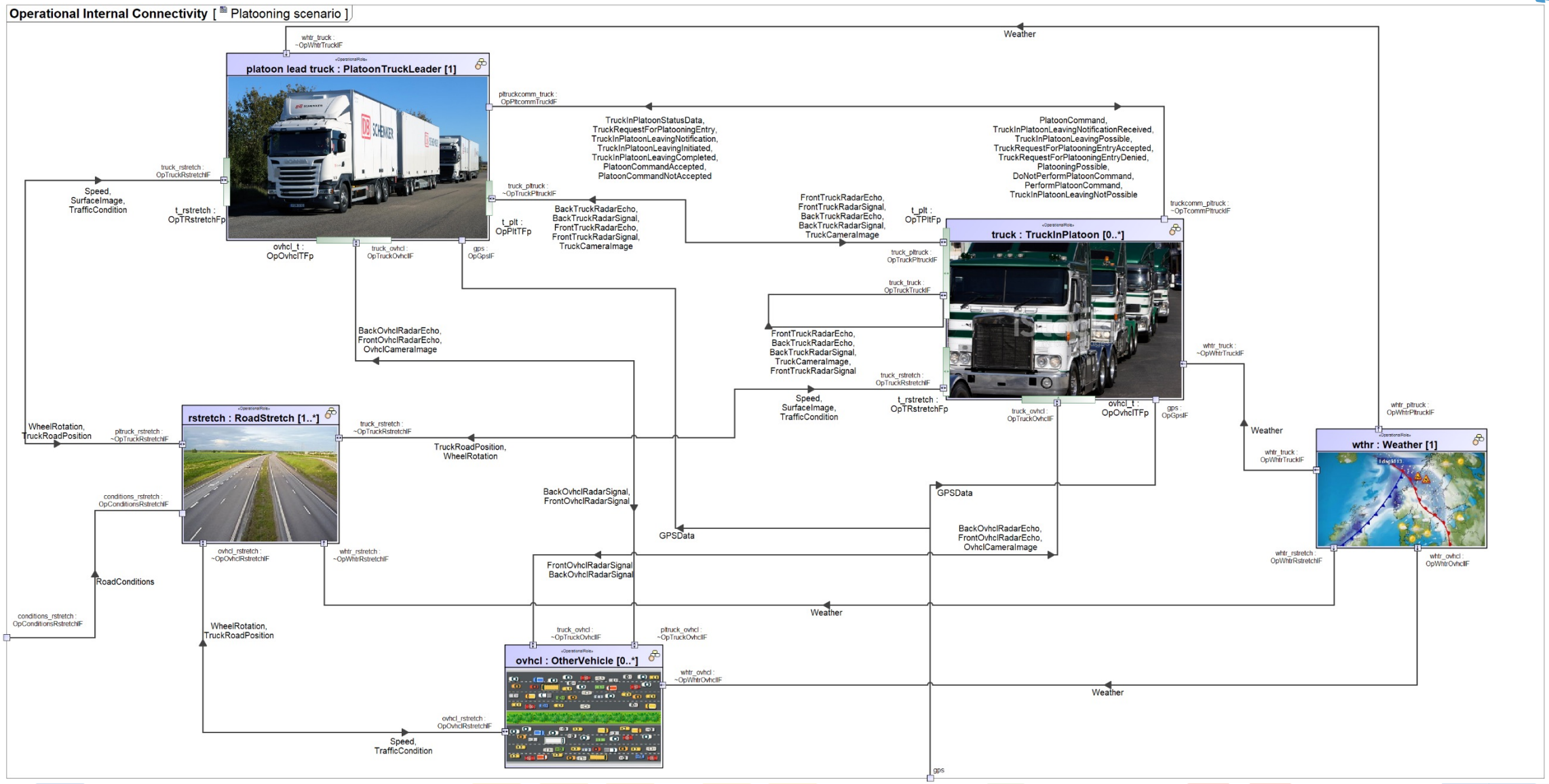
Known Resources in the Logical Architecture

- Known systems that must be part of the eventual architecture are identified early on to ensure that the system will meet the stakeholder requirements.
- Essentially boundary conditions, i.e., systems in the context that must be accommodated.
  - The known resource's behavior & interfaces are already defined.
- Making the system too abstract or "too logical" and ignoring these systems can result in an artificially unconstrained system
- By integrating these systems into the logical architecture, issues can be identified early and resolved.
- If the constraints imposed by a known resource prevent the logical architecture from satisfying stakeholder needs and requirements, then changes to that known resource will also be required.

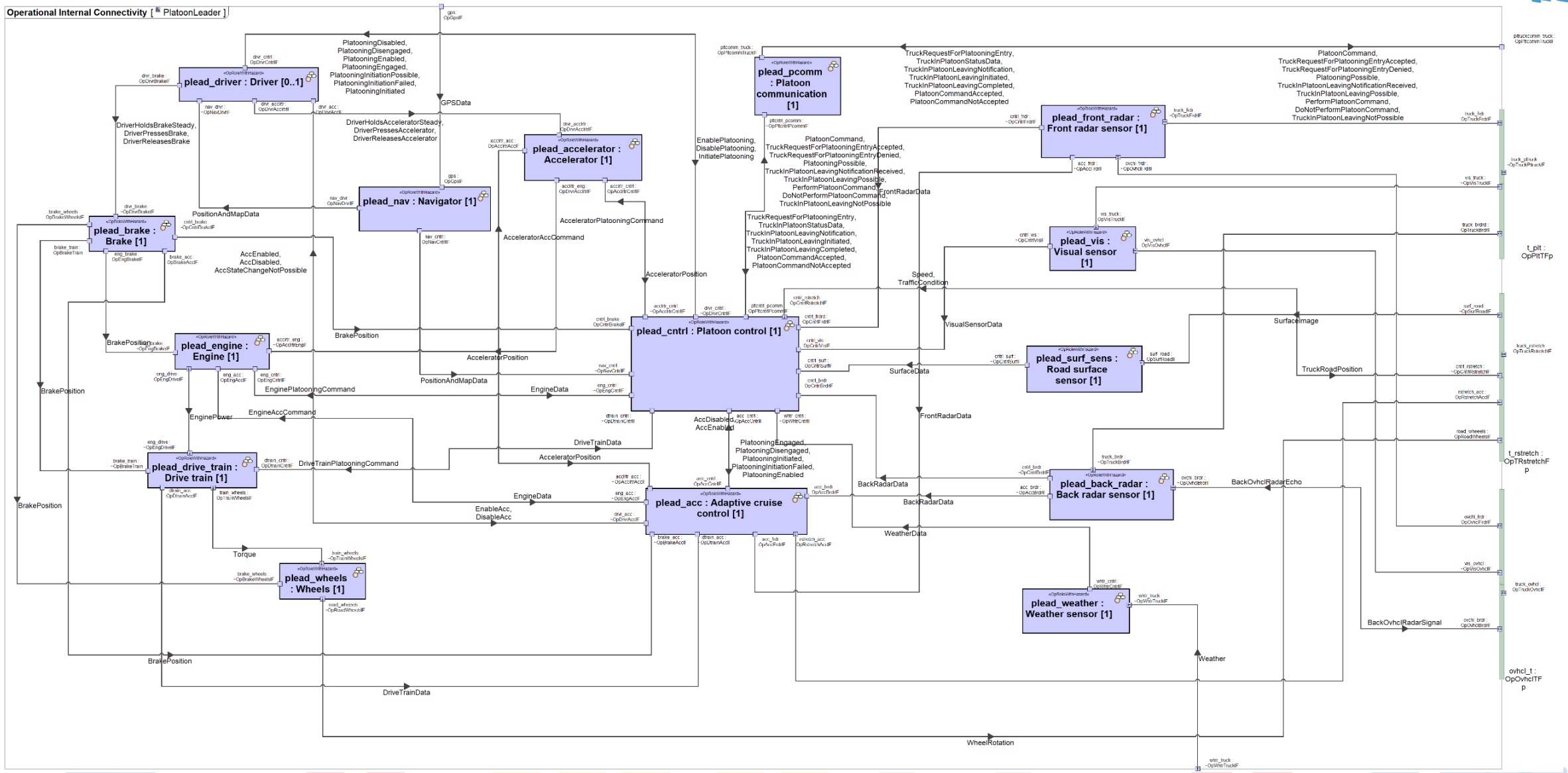
Known Resource Example

- Autonomous vehicles will pick up and deliver parts
- The Platooning example describes a set of trucks in a convoy driving closely to mitigate congestion and save energy
- The trucks & their control behavior, interfaces & constraints can be abstracted to allow for definition of hazards and safety.
- Modeling each truck, its behavior and interactions allows for evaluation of components and solutions

Logical Model of Truck, Platoon & Environment

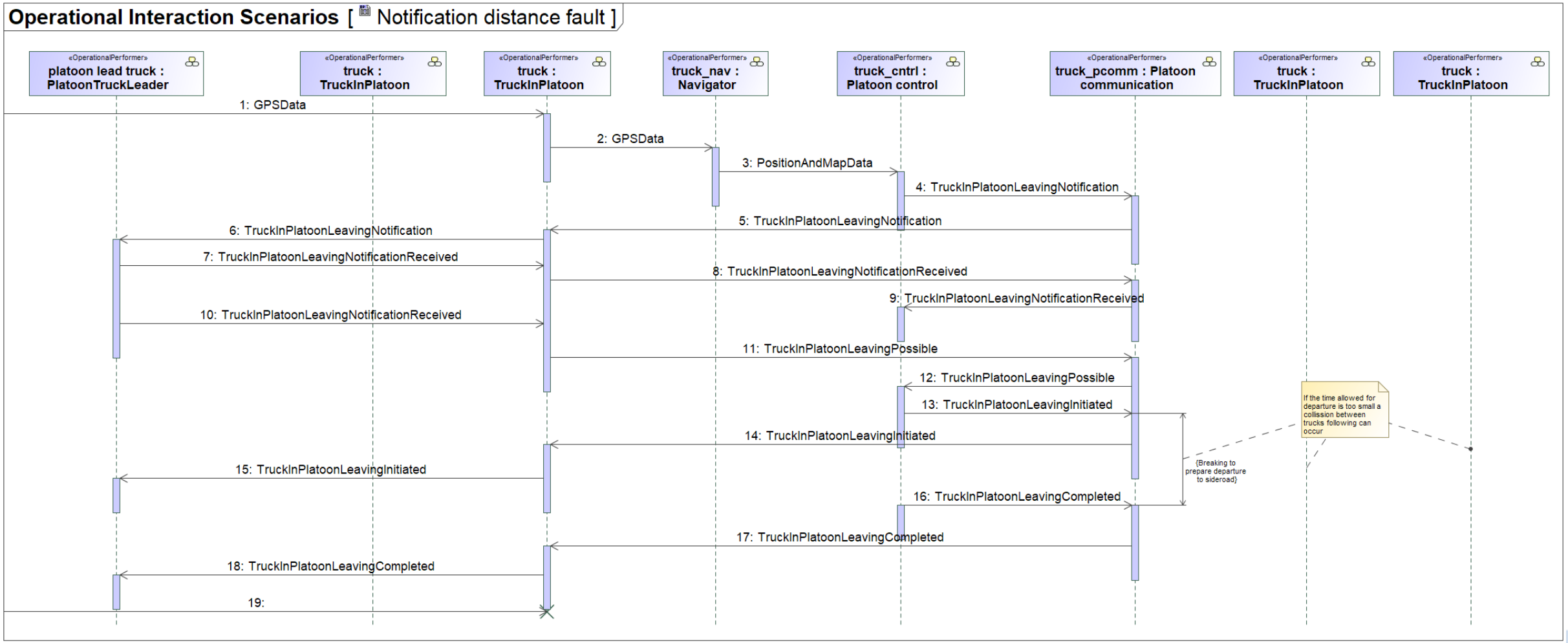


Logical Breakdown of Truck Components



Truck Platooning Hazard Scenario

- Shows a truck leaving the platoon, communicating with the others, speed control & platoon behavior



To SysML, PLM, CAD and Manufacturing

LOGICAL V. PHYSICAL MODELING WITH IBDS

- IBDs can be used to capture both a logical model of parts, connections and flows, and a physical model

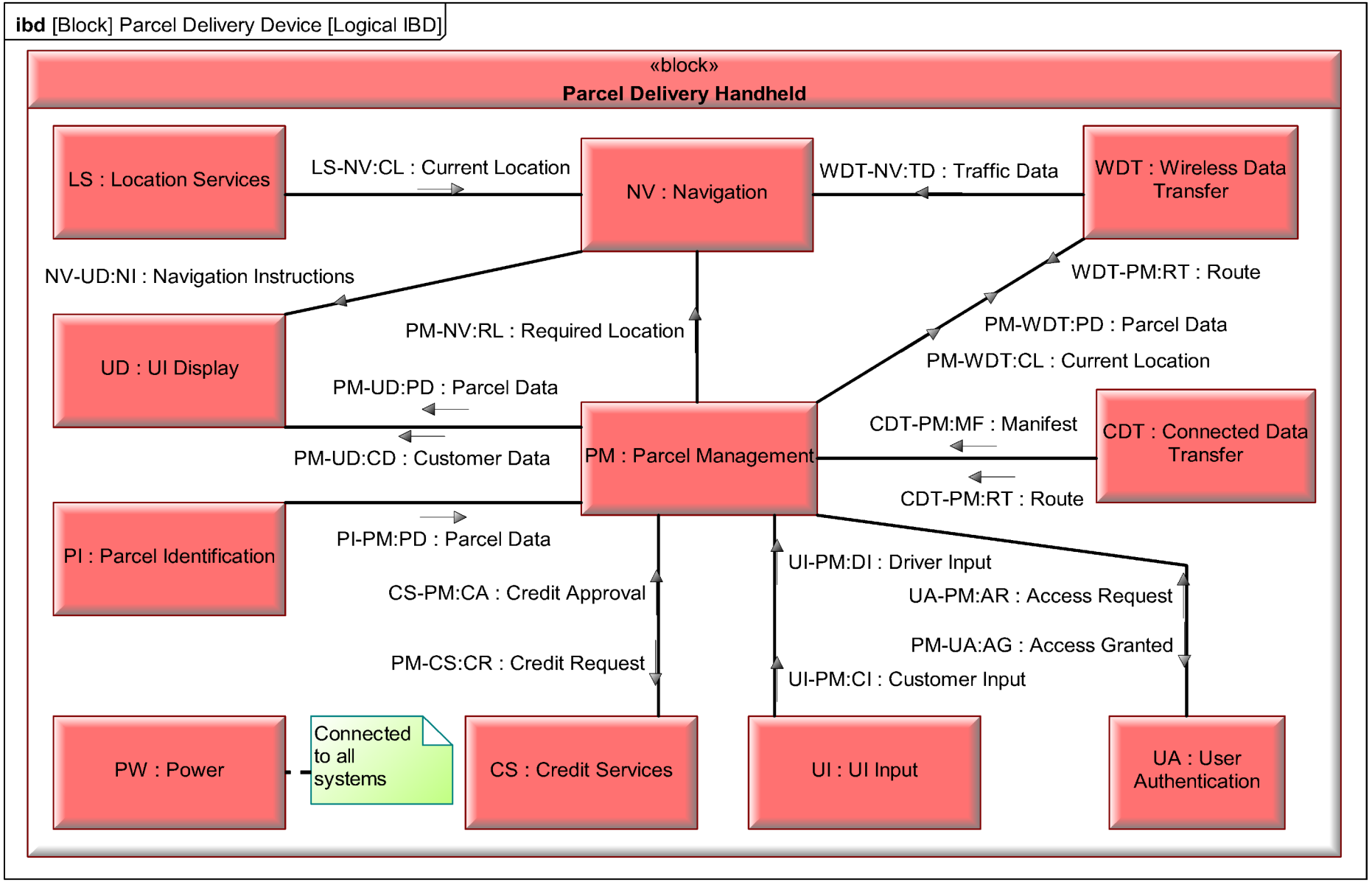
- Logical model focuses on logical parts and flows and may not show ports or types (unless logical types defined)
  - Based on specification rather than implementation ('what' not 'how')
  - Abstract types (if any)

- Physical model focuses on physical parts and flows and normally shows ports and physical (implementation) types
  - Normally follows logical modeling
  - May be many physical models for one logical model
  - Real-world types

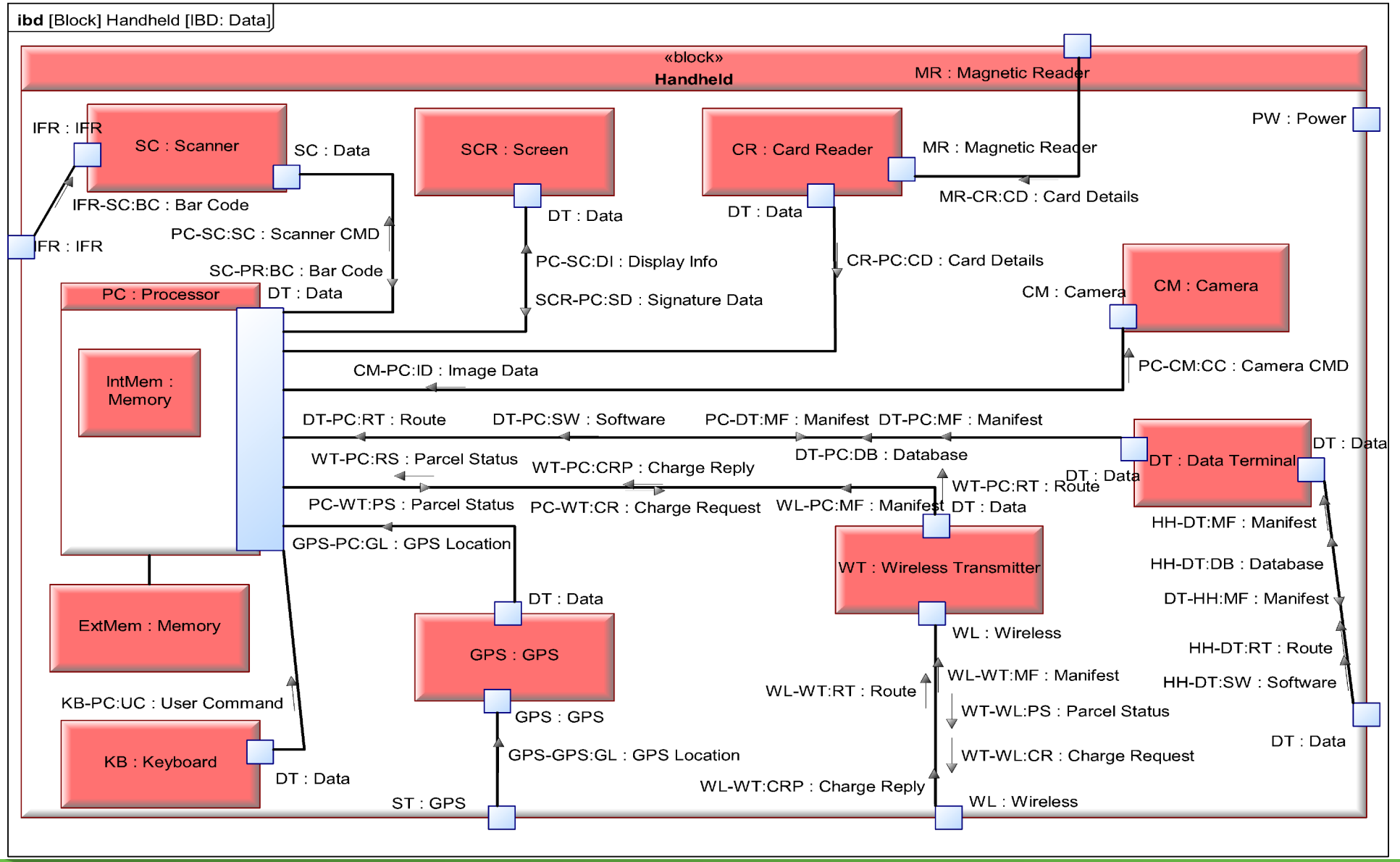
- May affect package structure
  - Logical package contains logical types
  - Physical package contains physical types

- Can link logical model items to physical model items via Allocation

EXAMPLE IBD - LOGICAL MODEL

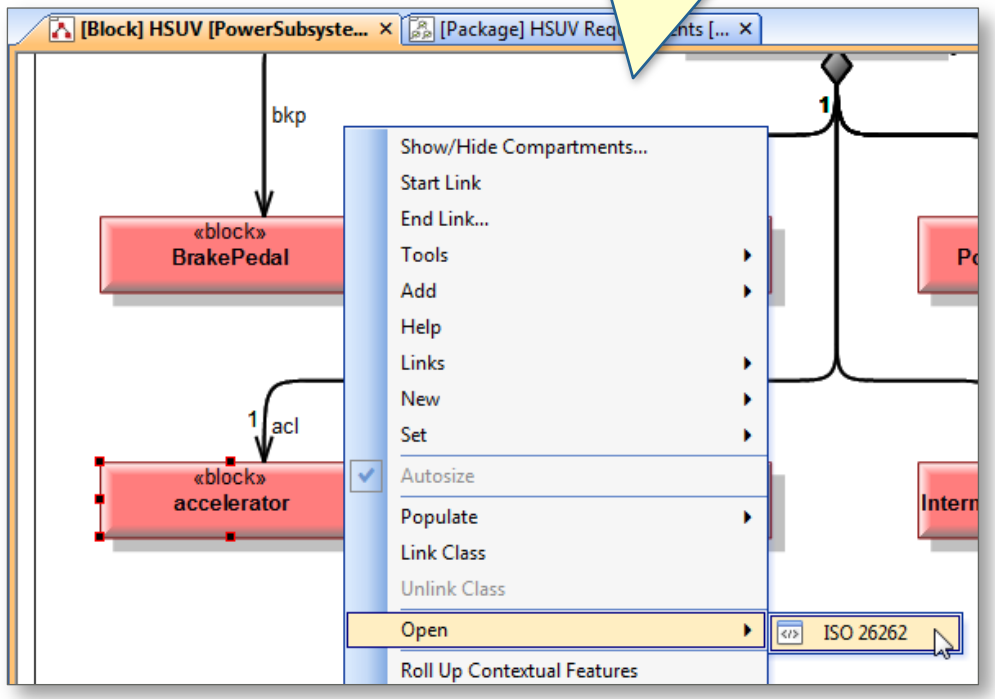


EXAMPLE IBD – PHYSICAL MODEL

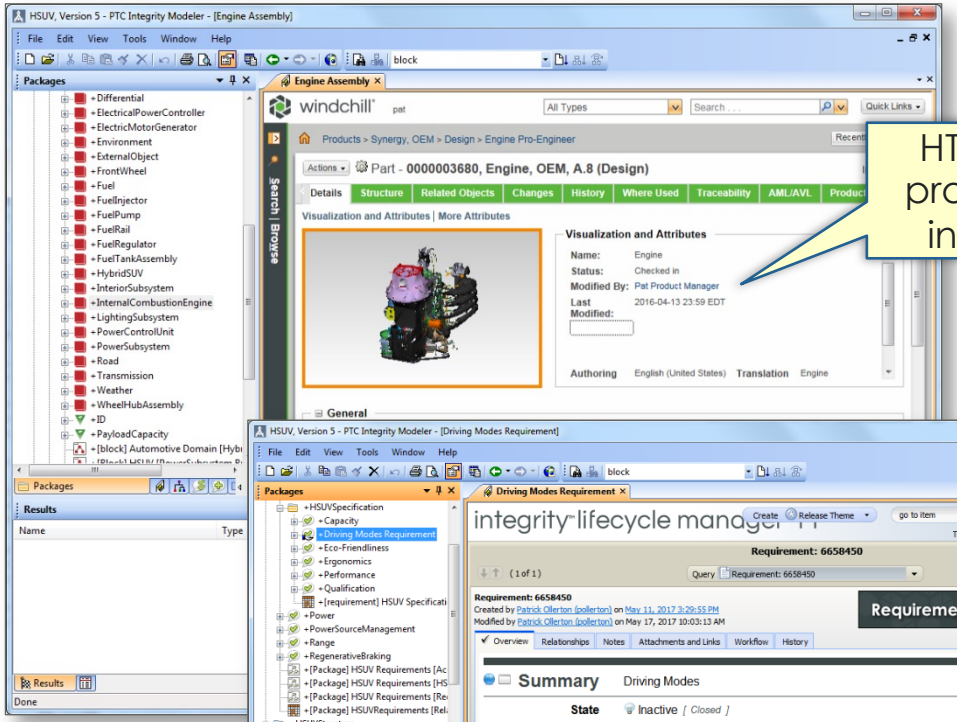


TRACING FROM REQUIREMENTS TO SYSML TO CAD

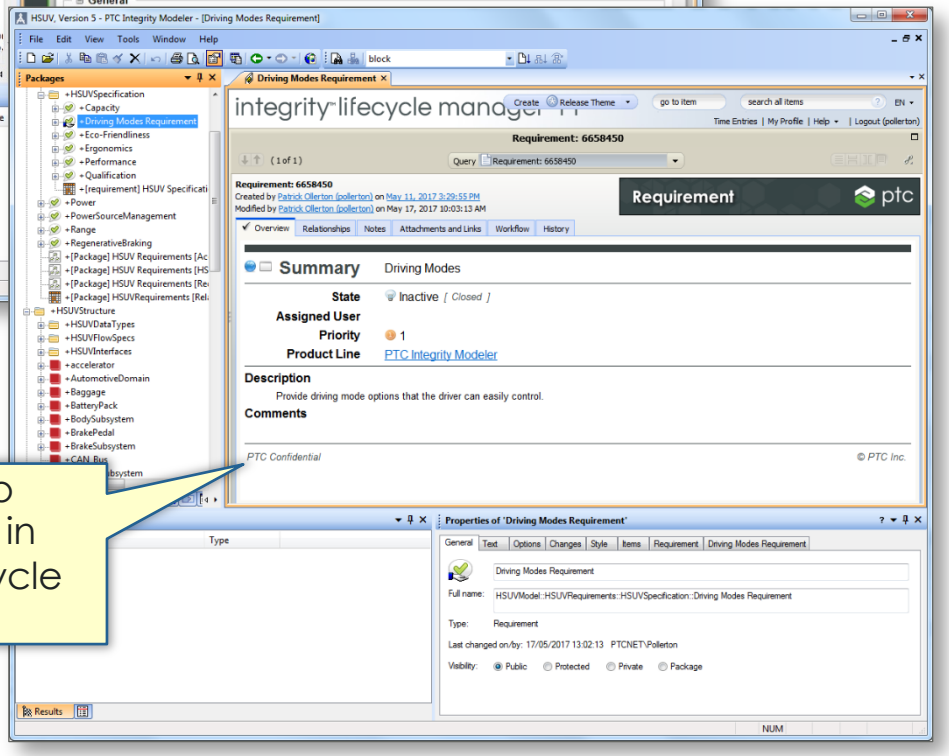
Right-click on items in browsers or on diagrams to open HTML Links and Surrogates



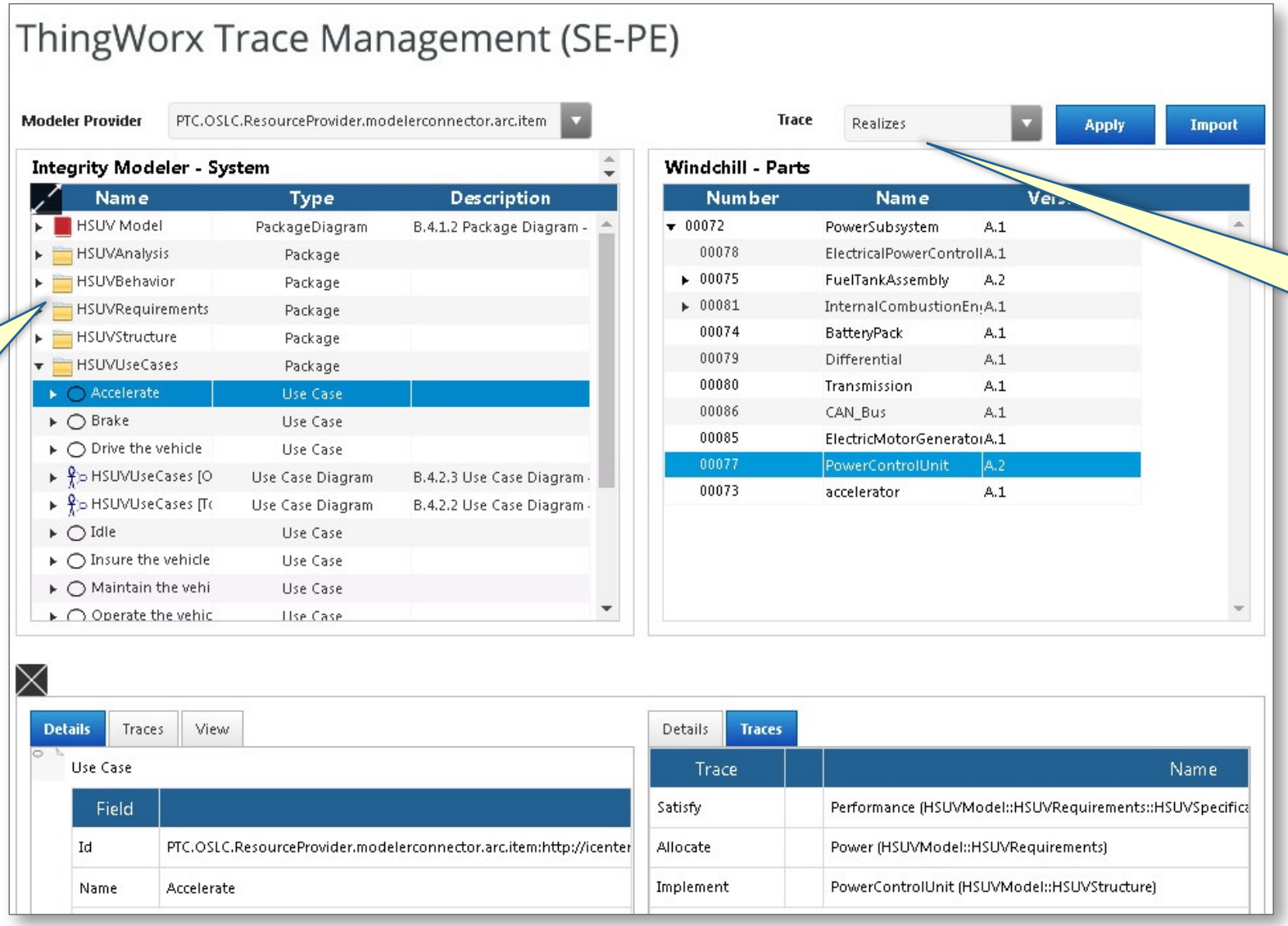
HTML Link to product data in Windchill



HTML Link to requirement in Integrity Lifecycle Manager



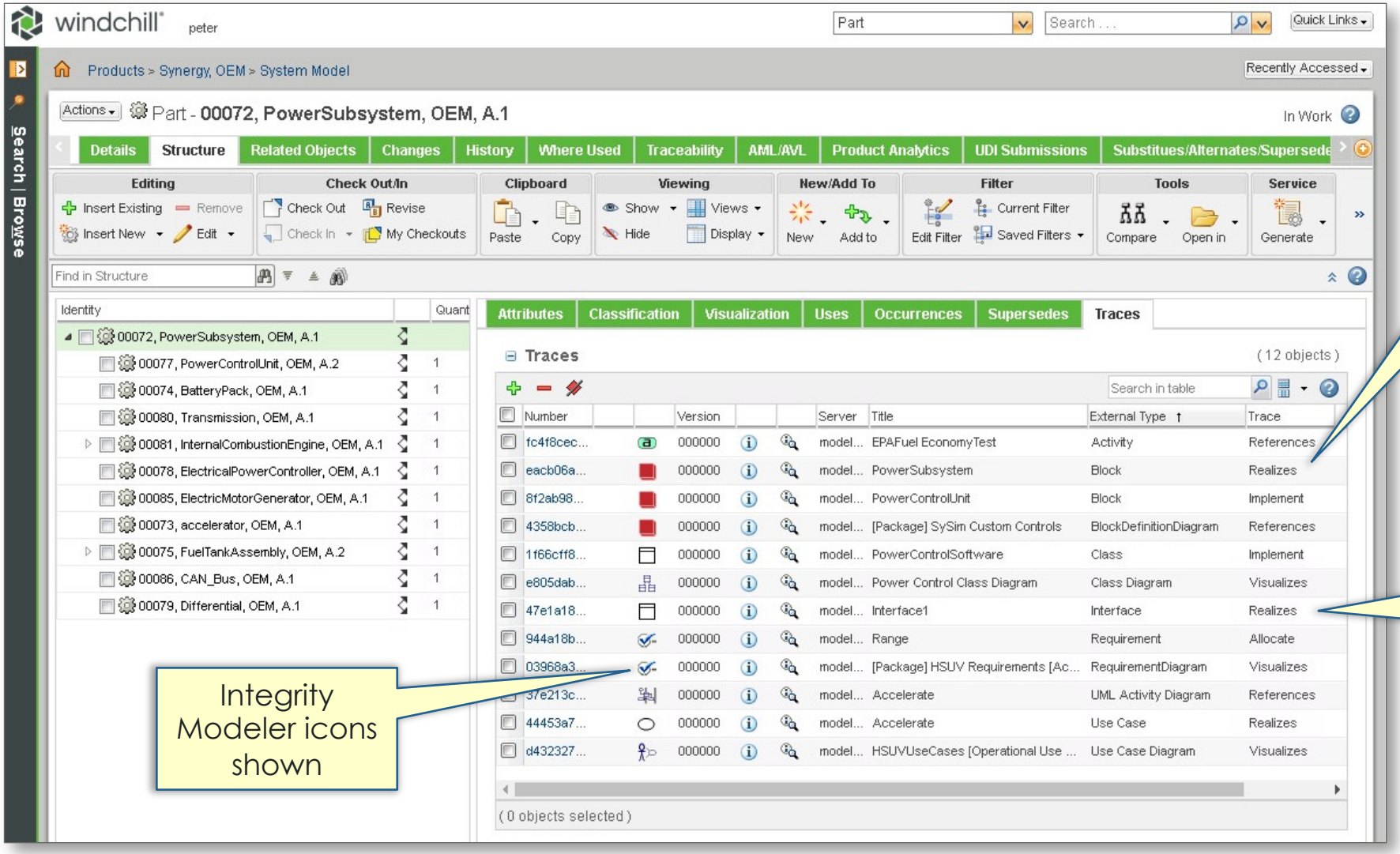
THINGWORX TRACE MANAGEMENT (SE-PE) DISPLAY



You define the Integrity Modeler types that are available in the ThingWorx Trace Management app

You define the valid link types for your organization

WINDCHILL LINKS TO INTEGRITY MODELER



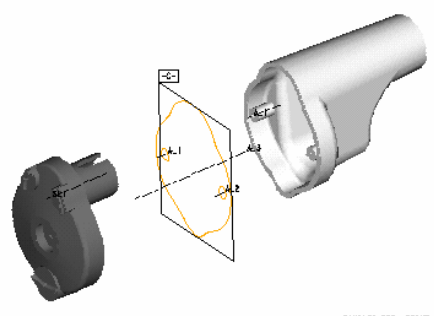
PHYSICAL INTERFACES

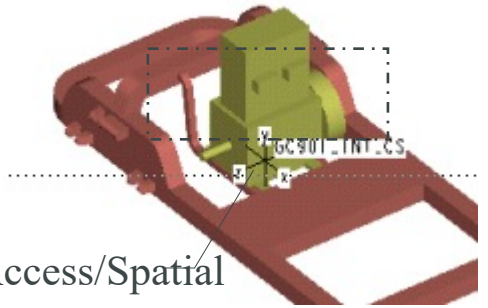
Interfaces are controlled boundaries between modules, components or parts

Types include:

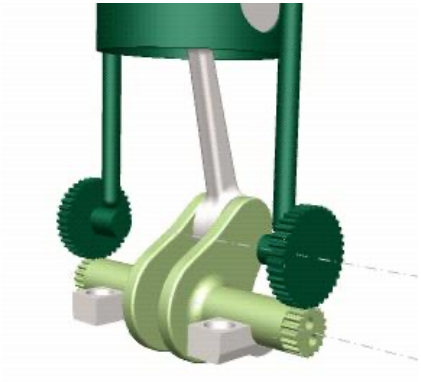
- Attachment, Spatial (envelope)
- Transfer (e.g. power)
- Communication
- User Interface

Direct/Attachment





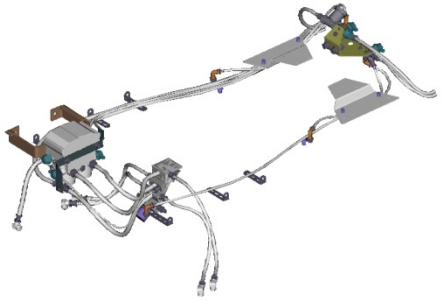
Access/Spatial



Transfer of Power

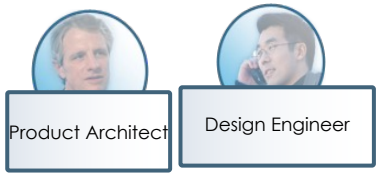
User Interface

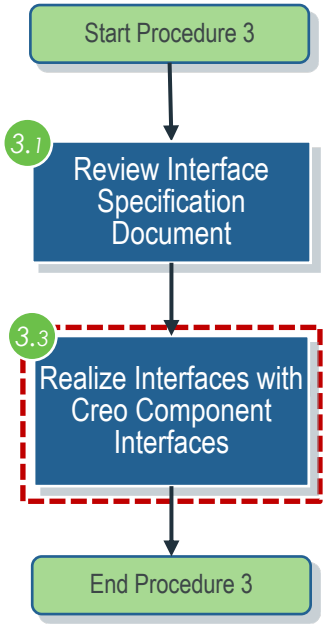
Communication

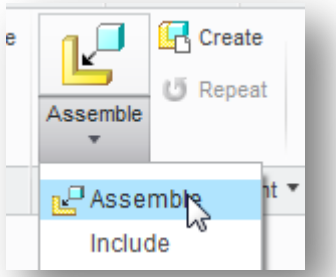


REALIZING INTERFACES

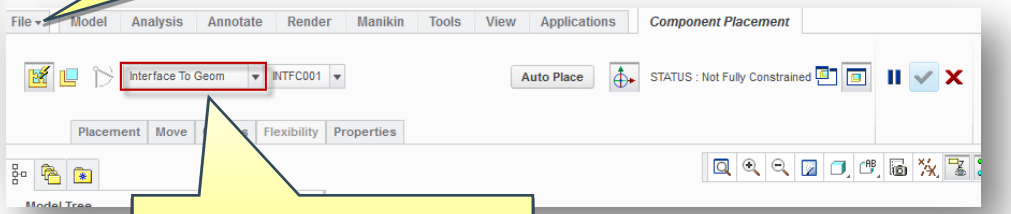
- Develop and Propagate Interfaces



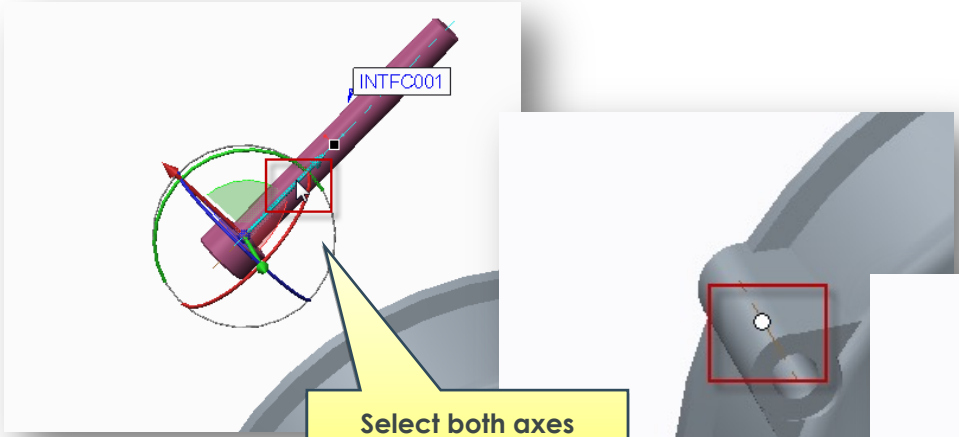




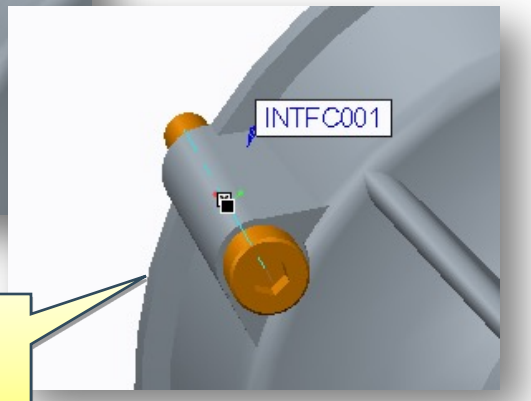
In Assembly mode, add the housing then assemble select the placing component



Select Interface to Geom



Select both axes

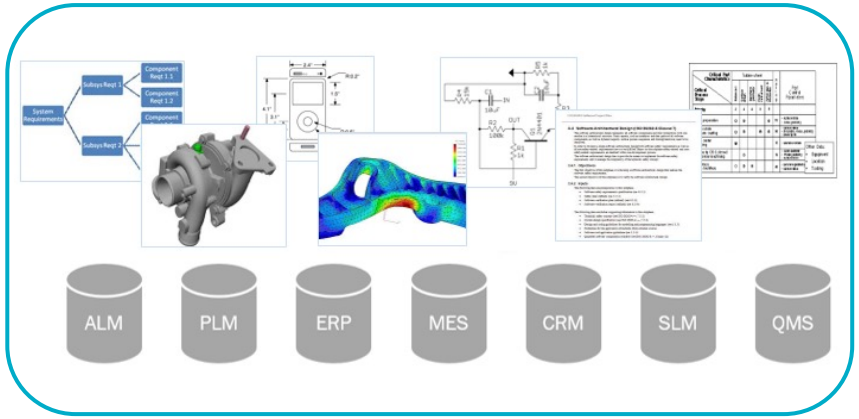


Confirm that component has been placed correctly and repeat as necessary

COLLABORATIVE AR/VR DESIGN

A Few Simple Steps from CAD to AR/VR

Collaborate Globally



Effortlessly Collect all Relevant Information

Closed-Loop Change Management

Logical Architectures Facilitate Innovation

- Systems engineering involves translating customer needs into viable systems that meet those needs.
- All systems become dated, competitors introduce faster, better, and cheaper products, technologies evolve, system environments change.
- The iPhone: several devices in our pockets combined to provide a completely new product.
- In the 1950s and 1960s containerized shipping used reusable, standardized, containers and special ships for carrying them – increased shipping capacity and reduced costs.
- The Tesla electric car changed the configuration of the engine from a single engine and drive shaft, to individual motors for each wheel.
  - Improves handling, reduces cost, reduces vehicle weight and removes links in the chain from power source to the target device.
- Only possible by looking at the functionality and purpose of a system and its elements in a solution independent way and imagining the number of ways in which they could be realized.
- Leads to large leaps of innovation rather than incremental improvements.

Conclusion

- The logical architecture avoids “Solutioneering” or starting with too many preconceived solutions.
- Limiting the initial models and analysis to the functions, allows the innovative engineer to explore multiple means of achieving this functionality.
- This is essential in complex systems of systems as well as competitive environments where multiple solutions could exist.
- Eliminates the “We have always done it this way” attitude
- Frees the creative engineering process and enables innovation.

Questions?

32nd Annual INCOSE international symposium

hybrid event

Detroit, MI, USA

June 25 - 30, 2022

www.incose.org/symp2022