

Model-Based Product Line Engineering

Enabling Product Families with Variants

Matthew Hause & Jim Hummell

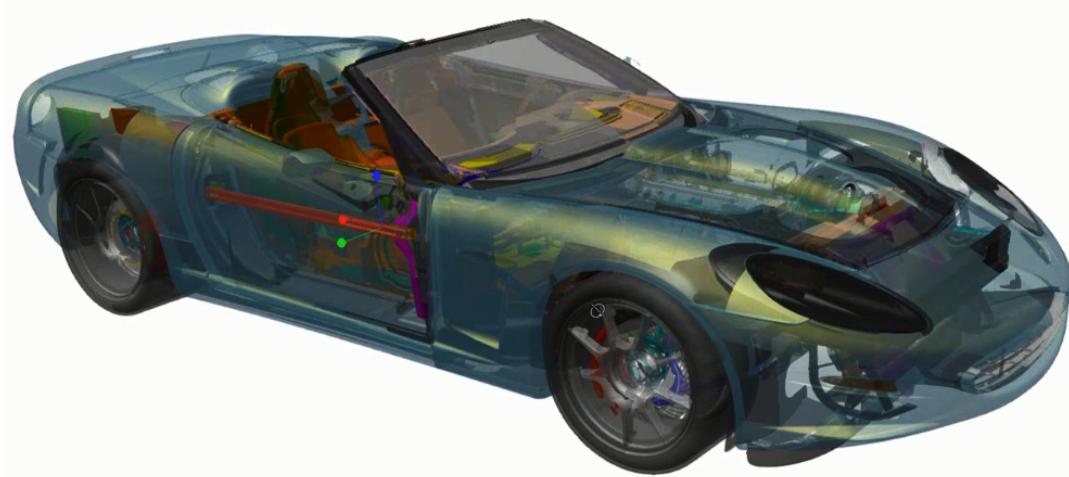
Technical Fellow & Principal Engineer, MBSE

July 2015



Smart connected systems & products

- **Growing complexity & functionality of systems & software**
 - Larger share of a products cost & capability is software
 - System & sub-system Integration
 - Customer, certification, regulation & standards compliance needs
- **Larger, more distributed & distinct discipline teams**
 - Communication language barriers & collaboration
 - Implementing common, architected Goals
- **Increasing time pressures**
 - Shorter development cycles
 - Delivering on schedule
- **Quality assurance**
 - Risk of building the wrong system
 - Increased costs of later stage errors
- **Cost reduction demands**
 - Total development cost
 - Risks & costs of delays or cancellation



Design before you build

- Standard based graphical modelling

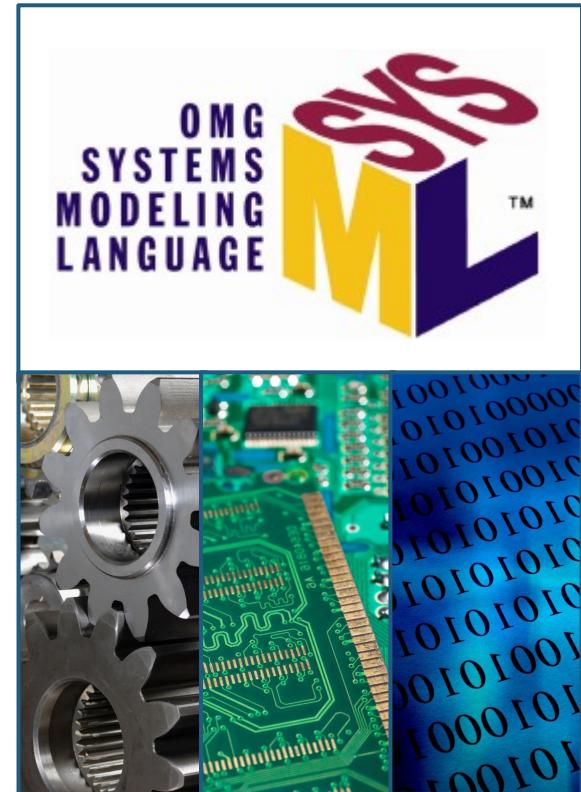
- Common language
 - Improves understanding
 - Facilitates collaboration
 - Achieves stakeholder buy in
- Problem abstraction, to see the 'wood from the trees'

- Systems engineering process automation

- Tools enable a more efficient systems engineering process
- Tangible designs to review, finding problems earlier
- Traceability from requirements through models to system
- Enables Rapid Prototyping, Simulation & Trade Studies

- Reduces the total cost of systems engineering

- Reduce learning curve & cost with an industry standard language
- Capture system design IP to reduce risks & retain value
- Optimized allocation to mechanical, electrical & software engineering
- Design & build the right systems, right



Product line explosion

- Increasing number of product families
- Increasing number of products in families
- Understanding product similarity
- Maximizing reuse
- Understanding product variations
- Deciding between options
- Development cycle time
- Commercial product needs
 - Customize existing capabilities to suit client requirements
 - Redeploy common systems & software to the Market
 - Time from requirements to cash



Designing a single system platform rather than as creating a multitude of products

- **MBSE + Modular Design + Variation**

- Common language improves
 - Communication
 - Collaboration
 - Stakeholder buy in
- Architected modular design & reuse
- System product lines designed up front

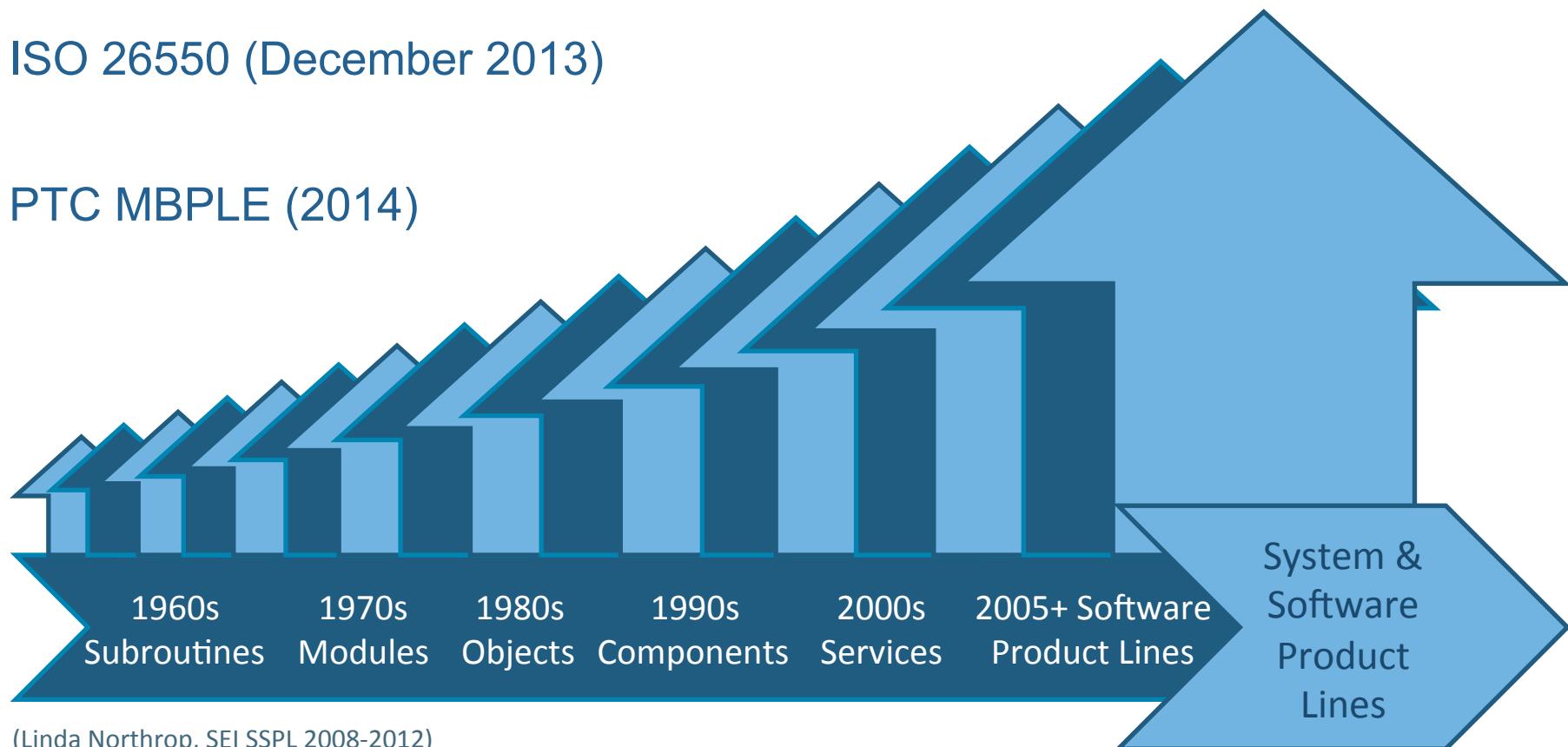
- **Maximum commonality & minimal variation**

- Less duplicated effort with optimized reuse
- Parallel working through 'design by contract'
- More commonality between designs and implementations
- Managed product line complexity



Model-based Systems & Software Engineering (2006) +
System & Software Product Line Engineering (2001-2008)

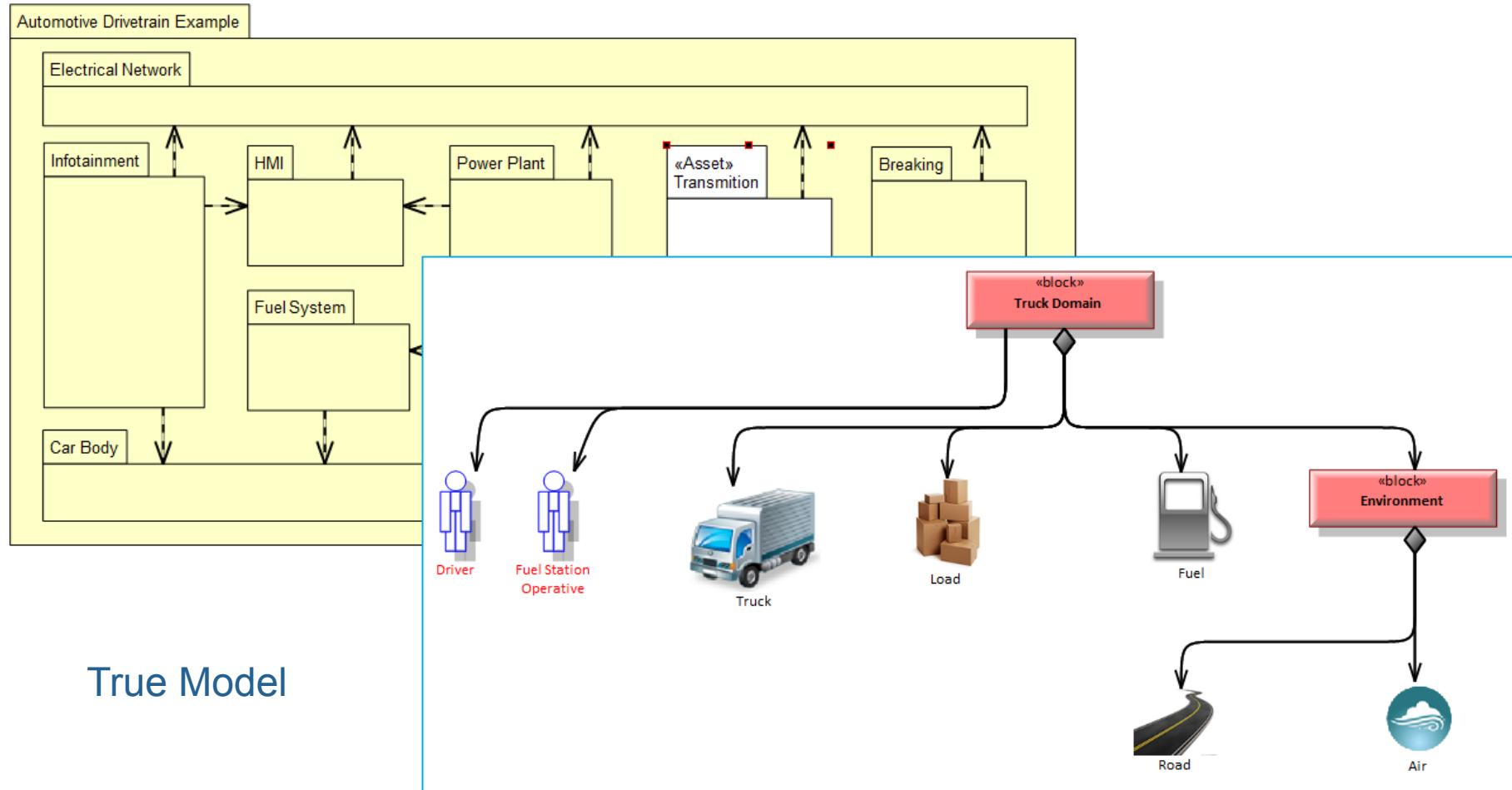
- ISO 26550 (December 2013)
- PTC MBPLE (2014)



Model-Based Systems Engineering

System Domain View - SysML

- Scope & Stakeholder Identification for Requirements Elicitation



Stakeholder & Environmental Requirements Elicitation & Analysis

Outline

- Document
 - 1 - Heading
 - 1.1 - Functional Requirement
 - 1.2 - Heading
 - 1.2.1 - Functional Requirement
 - 1.2.2 - Functional Requirement
 - 1.2.3 - Functional Requirement
 - 1.3 - Heading
 - 1.3.1 - Functional Requirement
 - 1.3.2 - Functional Requirement
 - 1.3.3 - Functional Requirement
 - 2 - Heading
 - 2.1 - Functional Requirement
 - 2.2 - Functional Requirement
 - 2.3 - Functional Requirement
 - 2.4 - Heading
 - 2.4.1 - Functional Requirement
 - 2.5 - Heading
 - 2.5.1 - Functional Requirement
 - 3 - Heading
 - 3.1 - Functional Requirement
 - 3.2 - Functional Requirement
 - 4 - Heading
 - 4.1 - System Requirement
 - 4.2 - System Requirement
 - 5 - Functional Requirement

Show items containing where

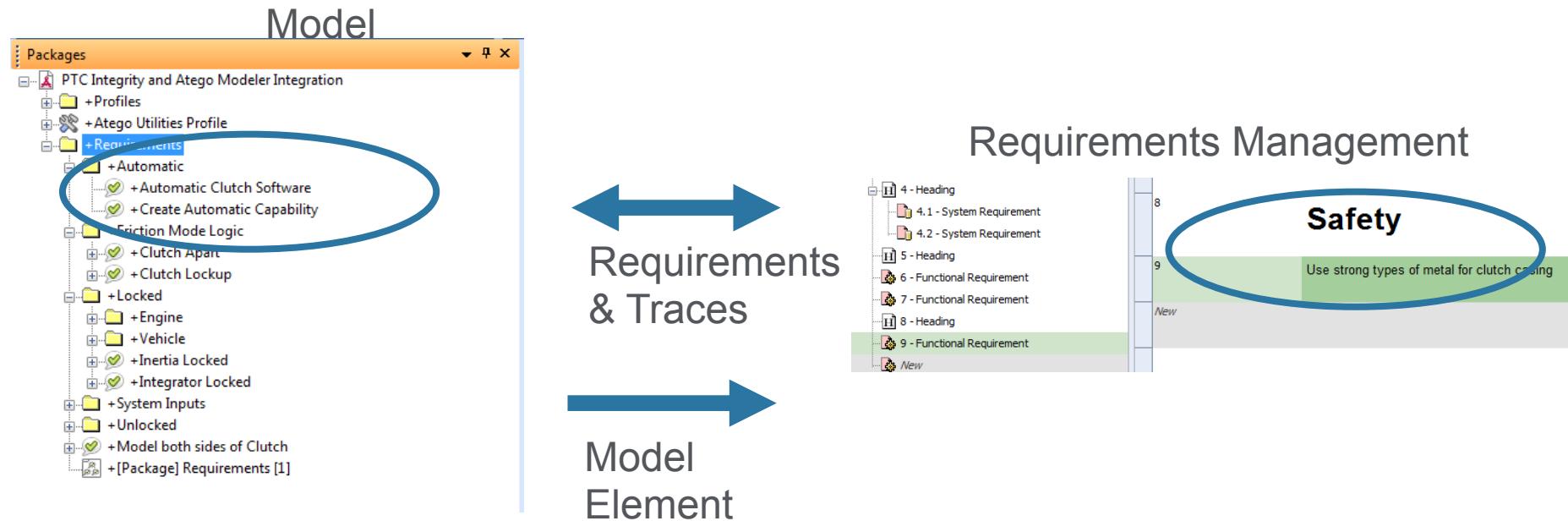
Section	Text	Category	Priority	ID
1	Unlocked	Heading	4	3
1.1	The system must allow the clutch to be unlocked	Functional Requirement	6	3
1.2		Heading	8	3
1.2.1	Model Engine Integrator in Unlocked State	Functional Requirement	10	3
1.2.2	Model Engine Damping in Unlocked state	Functional Requirement	12	3
1.2.3	Model Engine Inertia in Unlocked state	Functional Requirement	14	3
1.3		Heading	16	3
1.3.1	Model Vehicle Integrator in Unlocked state	Functional Requirement	18	3
1.3.2	Model Vehicle Damping in Unlocked state	Functional Requirement	20	3
1.3.3	Model Vehicle Inertia in Unlocked state	Functional Requirement	22	3
2		Heading	24	3
2.1		Functional Requirement	26	3
2.2		Functional Requirement	28	3
2.3		Functional Requirement	30	3
2.4		Heading	32	3
	Locked			
	The System must allow the clutch to be locked	Functional Requirement		
	Model Engine and Vehicle Inertia in Locked state	Functional Requirement		
	Model Engine and Vehicle Integrator in Locked state	Functional Requirement		
	Engine	Heading		

Synchronizing Requirements with PTC Integrity

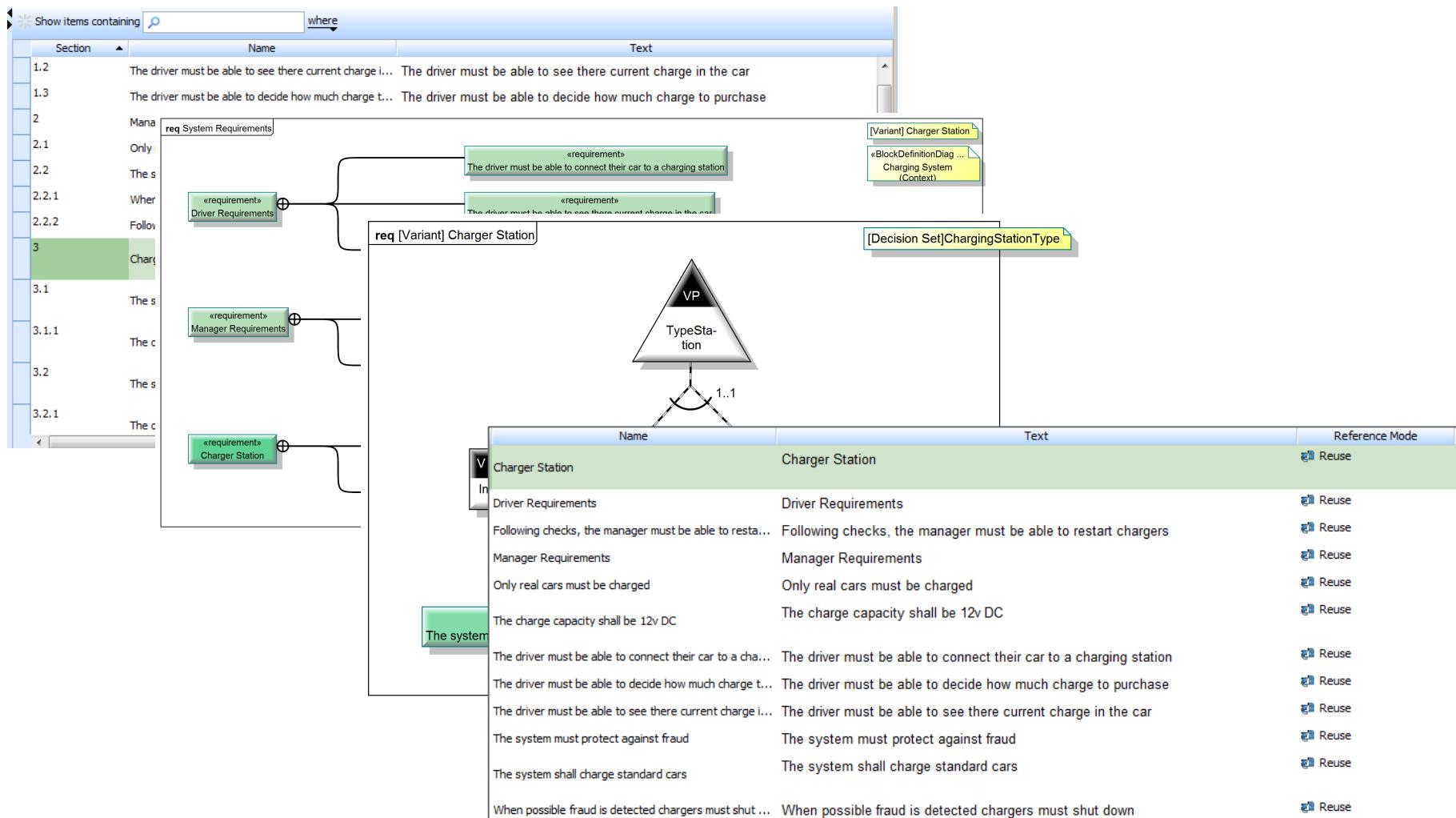
PTC®

Round tripping with PTC Integrity Modeler – Integration for Lifecycle Manager

- Updates can be made from either Modeler or PTC Integrity Lifecycle Manager
- Bi-directional synchronization to update both the model and requirement document in tandem

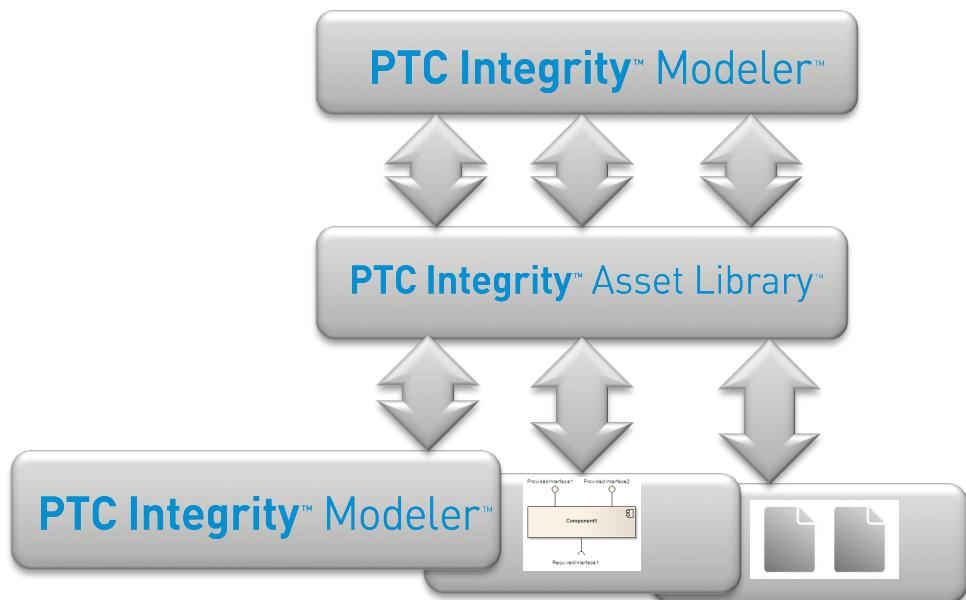


Product Line Branching of Requirements

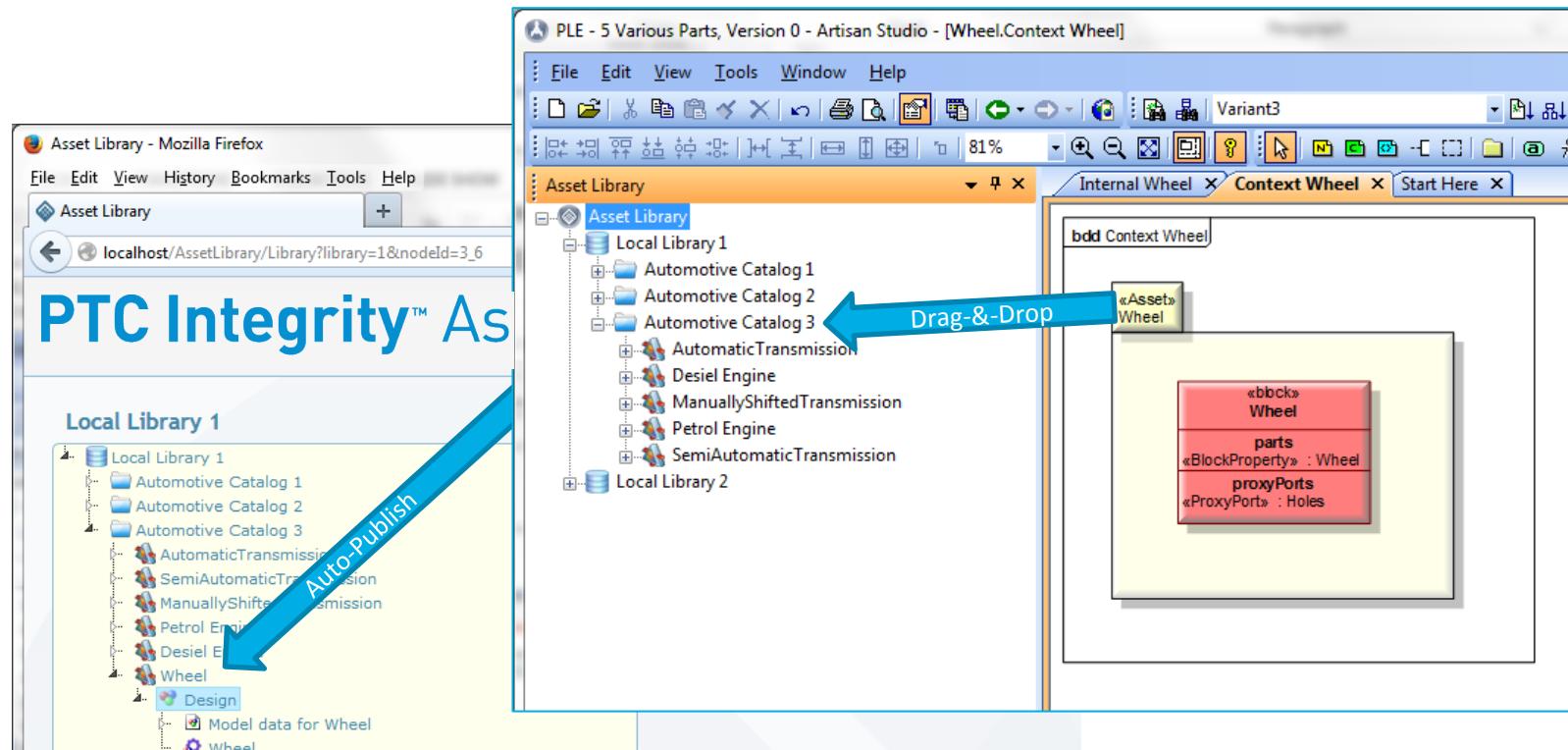


Asset Based Modular Design

- Design the same way you Build
 - Construct Systems of Sub-Systems (SoS)
 - Use Services to build your Application (SOA)
 - Plug Components together (CBD)
- Modular Design
 - Top-Down, Architected
 - Specification (& Requirements) Driven
 - Parallel Working
 - Separation of Concerns
 - Bottom-Up, Asset Mining
 - Un-modeled Assets
 - Other Modeling Tools
 - Legacy Integration
 - Published Interfaces (e.g. IDL)



- Publish from Sub-system model into PTC Integrity Asset Library
 - Auto-creates Trace Links



- Search PTC Integrity Asset Library for Sub-systems

The screenshot shows the PTC Integrity Asset Library interface in a Mozilla Firefox browser. The title bar reads "Asset Library - Mozilla Firefox". The address bar shows the URL "localhost/AssetLibrary/Library?library=1&nodeld=3_6". The main content area displays the "PTC Integrity™ Asset Library" logo and a "Local Library 1" tree view. The tree view includes "Local Library 1" with "Automotive Catalog 1" and "Automotive Catalog 2" expanded, showing "Automotive Catalog 3" which further contains "AutomaticTransmission", "SemiAutomaticTransmission", "ManuallyShiftedTransmission", "Petrol Engine", "Diesel Engine", and "Wheel". The "Wheel" node is selected. Below the tree view is an "Asset" form with the following data:

Name	Wheel
Short Description	
Description	
Date	18/11/2013 12:01:37
Variation points	
Configured variants	
Keywords	
Development Effort (days)	0
Estimated Reuse Effort (days)	0
Estimated Savings (days)	0
State	
Version	0

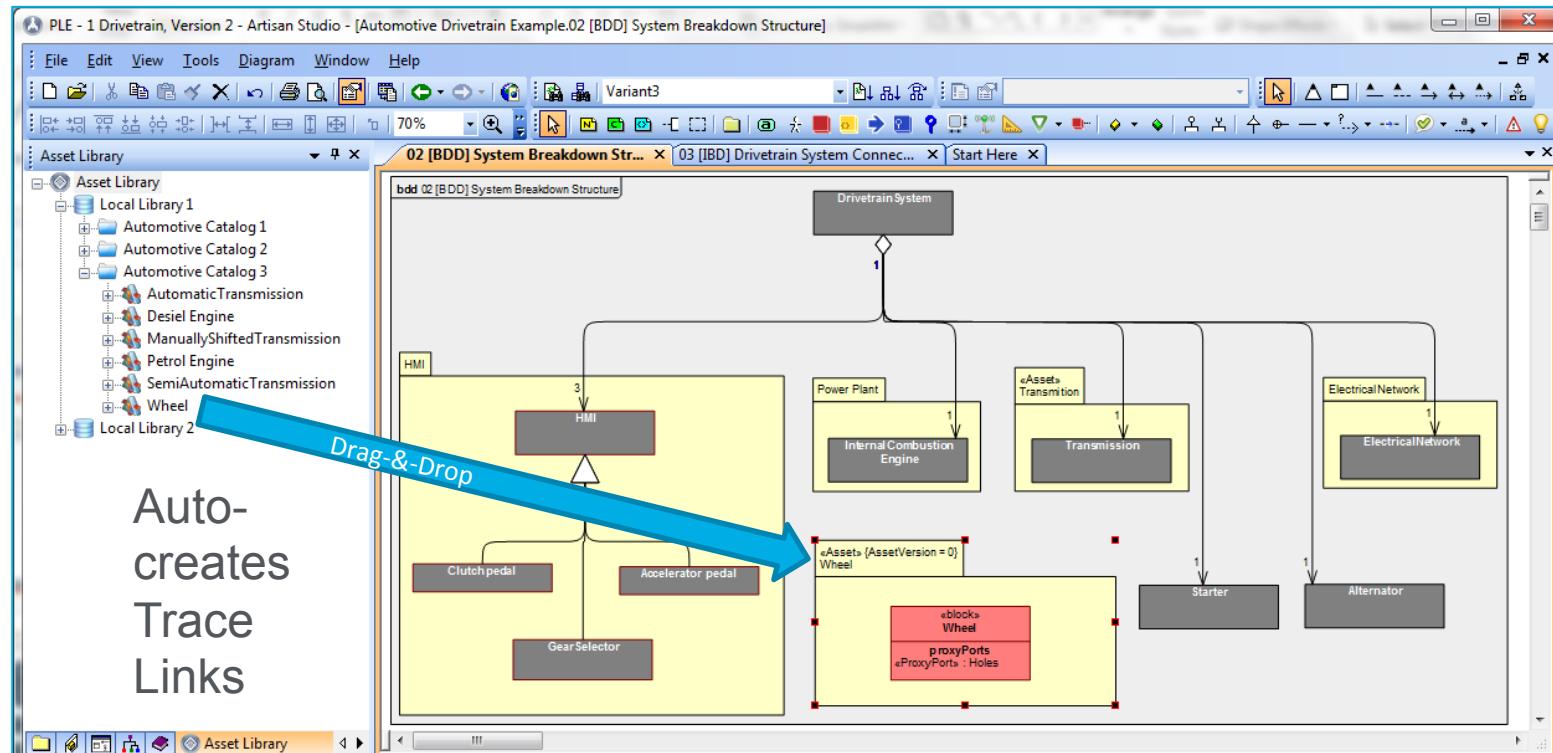
On the right side of the interface, there is a search panel with the following fields:

- Text:
- Keyword:
- State:

Buttons in the search panel include "Search" (red), "Publish" (green), and "Search" (black). Below the search panel is a list of search results:

- AlloyWheel
- SteelWheel
- SteelWheel
- Wheel
- AlloyWheel
- SteelWheel
- SteelWheel
- Wheel

- Use Sub-system from PTC Integrity Asset Library in Super-system Model (BDD)

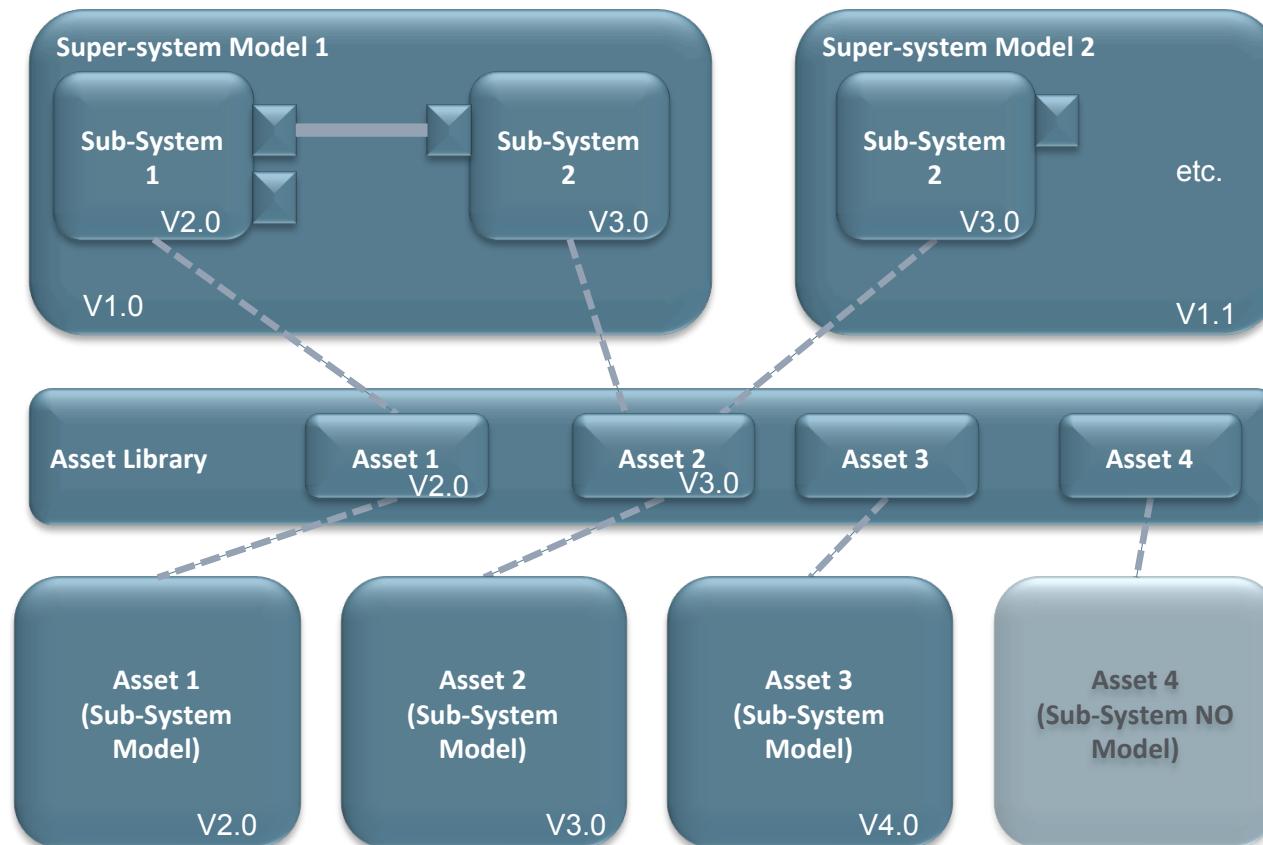


- Super-system Model = Configuration of Versioned Sub-systems

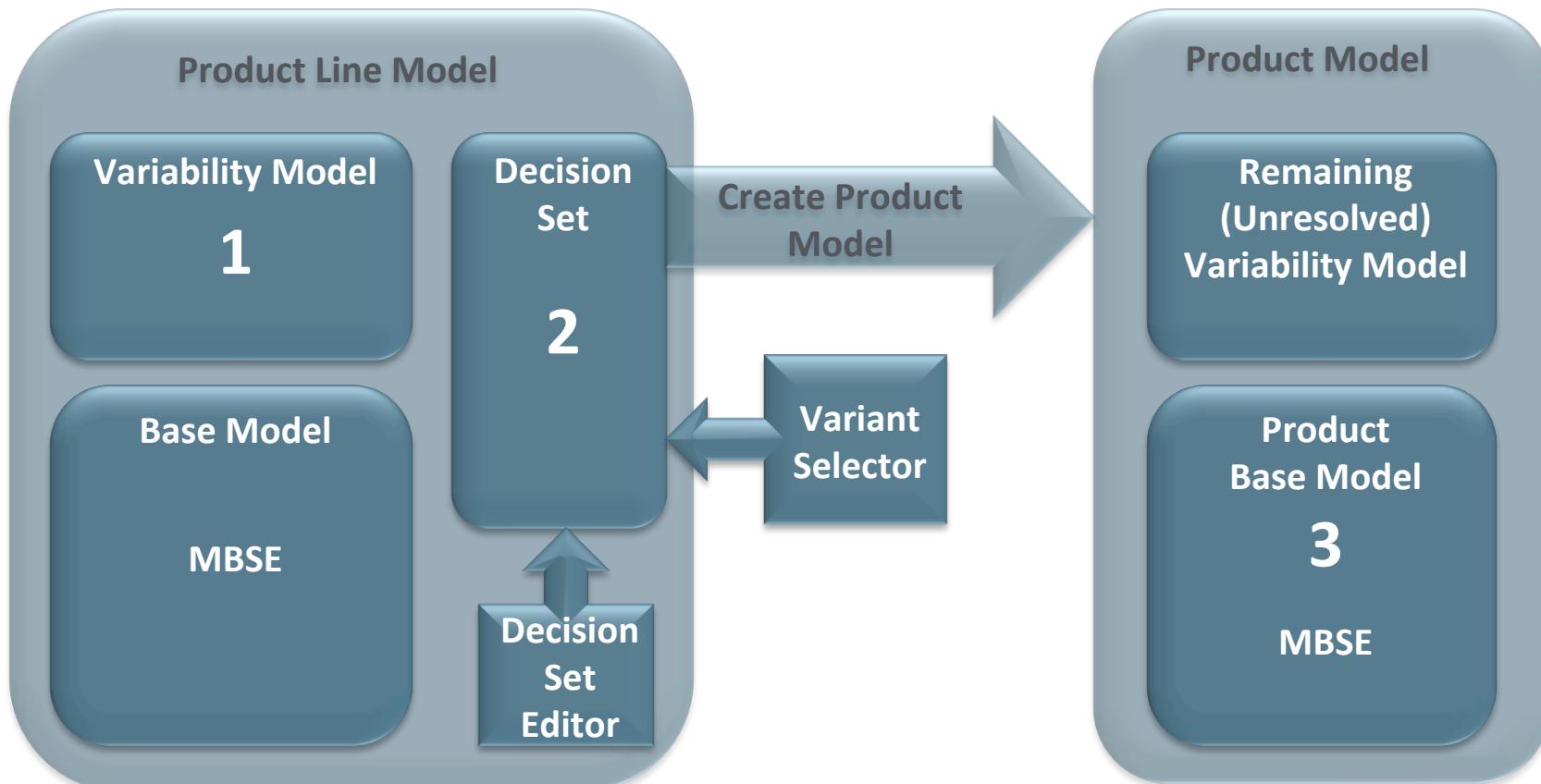
Super-system
Models

Links via
Assets

Sub-system
Models



Model-Based Product Line Engineering



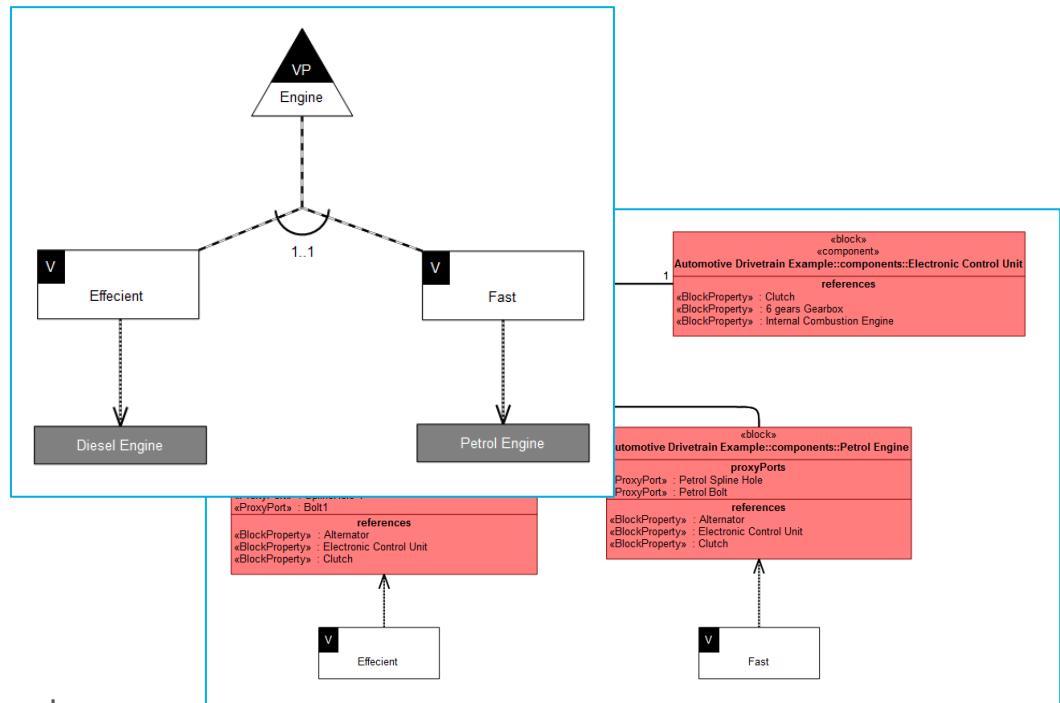
- Variant Diagram
- Variation on all Diagrams
- Simple Notation



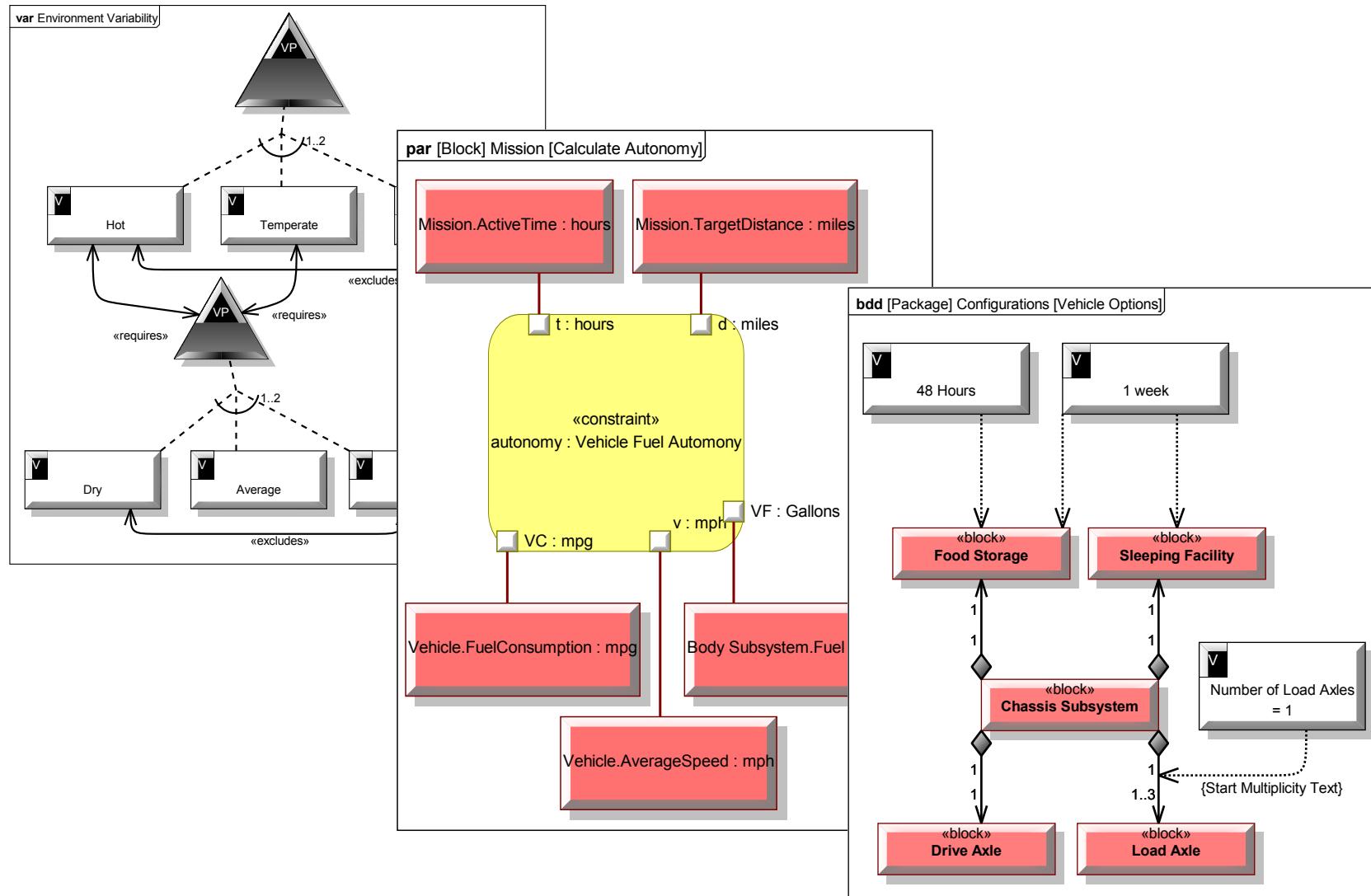
- Variation Point
- Variant
- Variability Dependency
 - Mandatory/Optional
- Requires Dependency
- Excludes Dependency
- Artifact Dependency
- Alternate Choice

• OVM

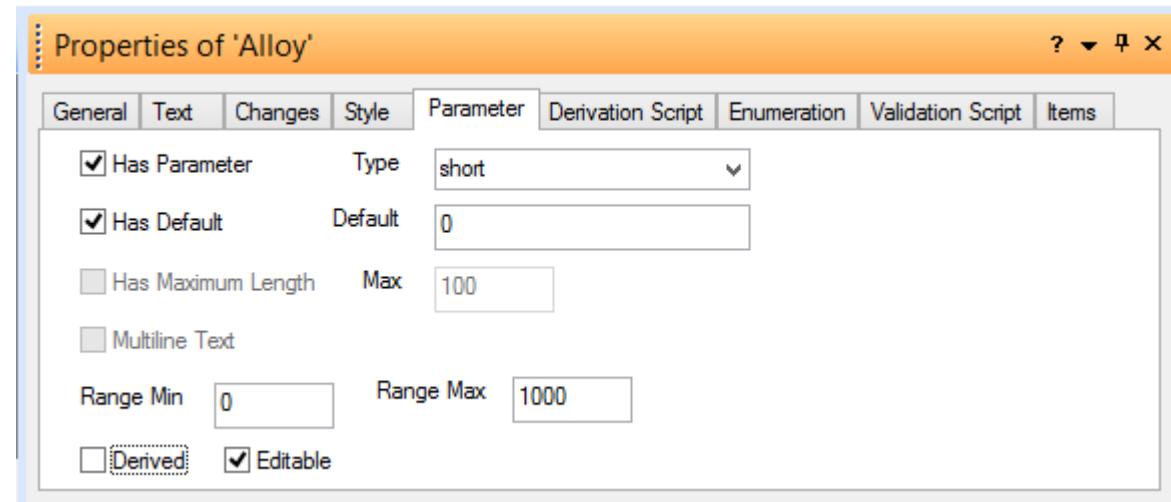
PALUNO, The Ruhr Institute of Software Technology
 Software Product Line Engineering (Pohl et al - Springer 2005)



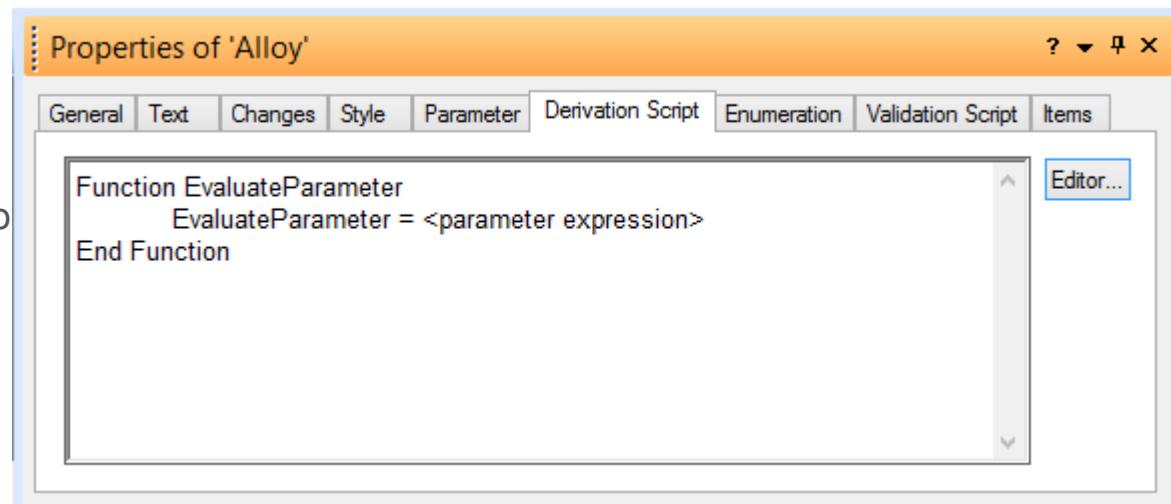
Parameters can be used that tied into values in the model itself to allow or not allow you to select a Variant



- There are additional rules you can add.
- Variant can have a parameter that can have some values.
- Based on the value you will be able to select it during the Decision or not.

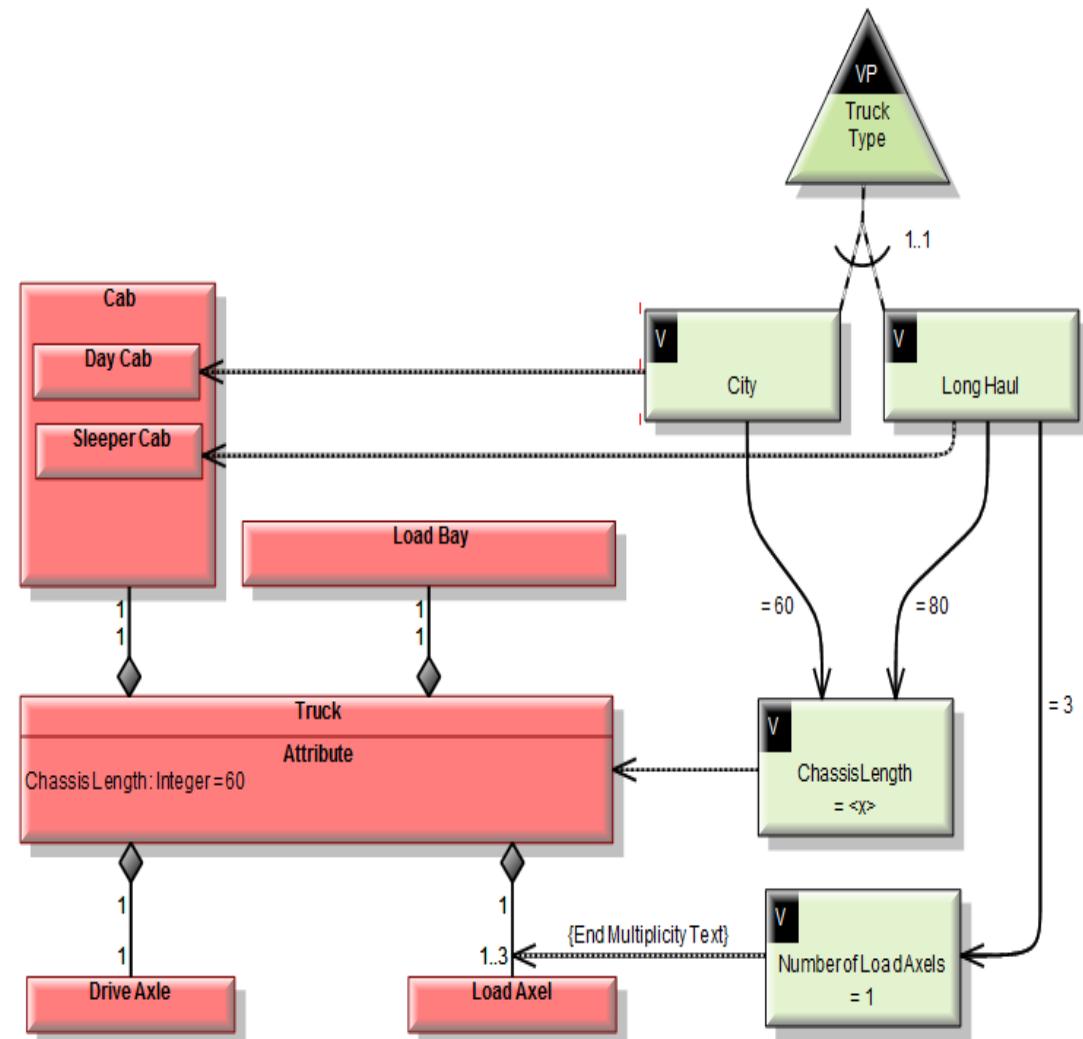


- Other additions are the use of Derivation Scripts that allow the users to have full control over when a variant is allowed or not allowed to be selected.
- Based on other modeling values and or rules the user defines.



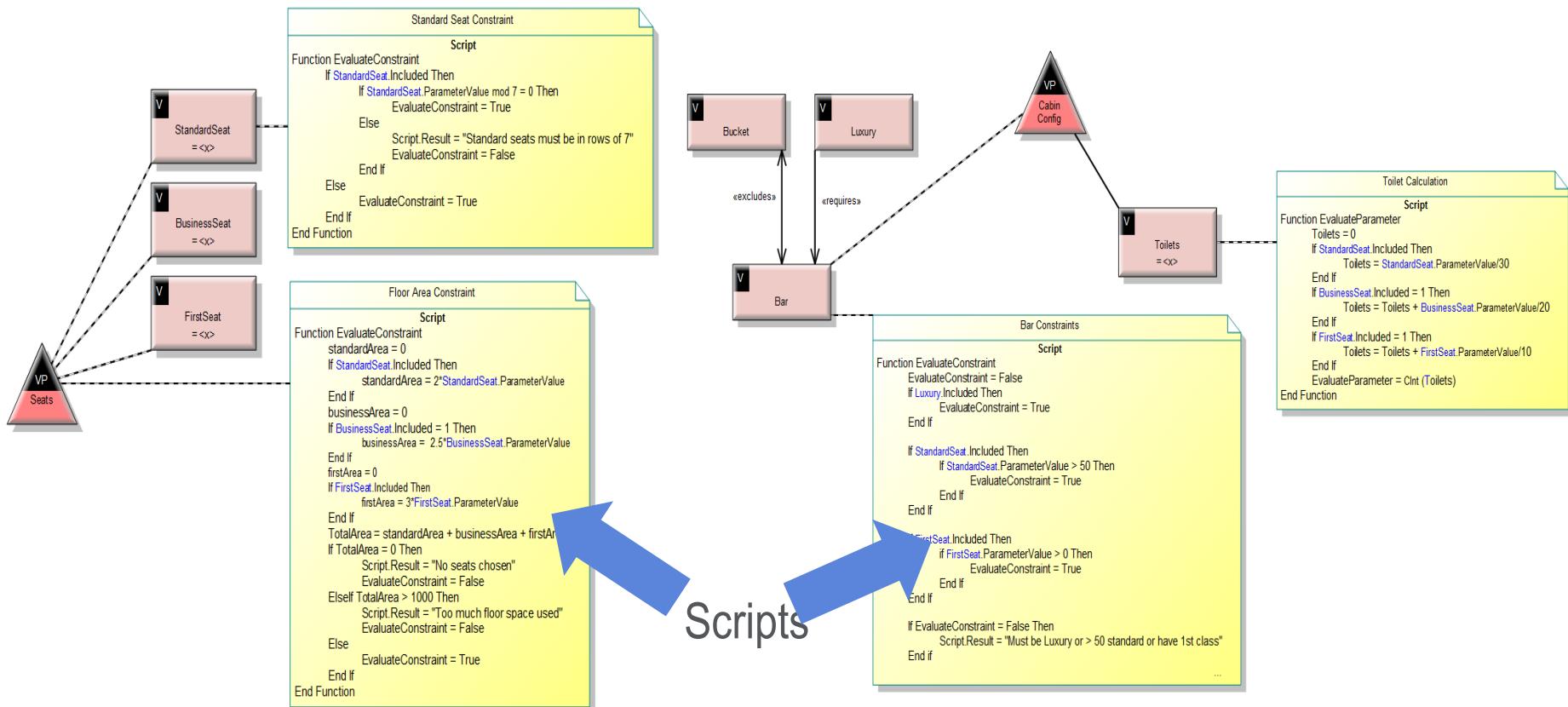
Variable Parameters

- Variable Parameter Passing



Variable Parameters

- Derivation & Validation Scripts



2 - Variant Selection

PTC®

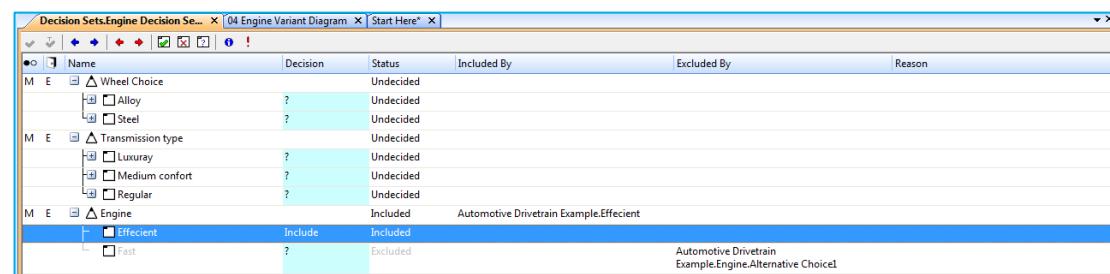
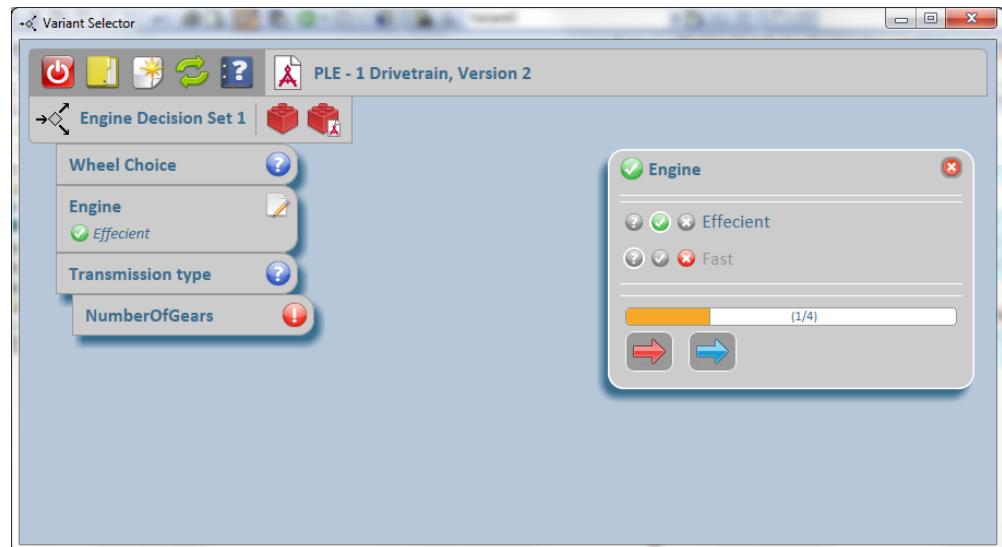
- Variant Selector

- Browser User Interface
- External Variation Points Only
- Jump to Next Decision/Problem
- Progress Bar

- Decision Set Editor

- Variant Debug
- External & Internal Variation Points
- Jump to Next Decision/Problem

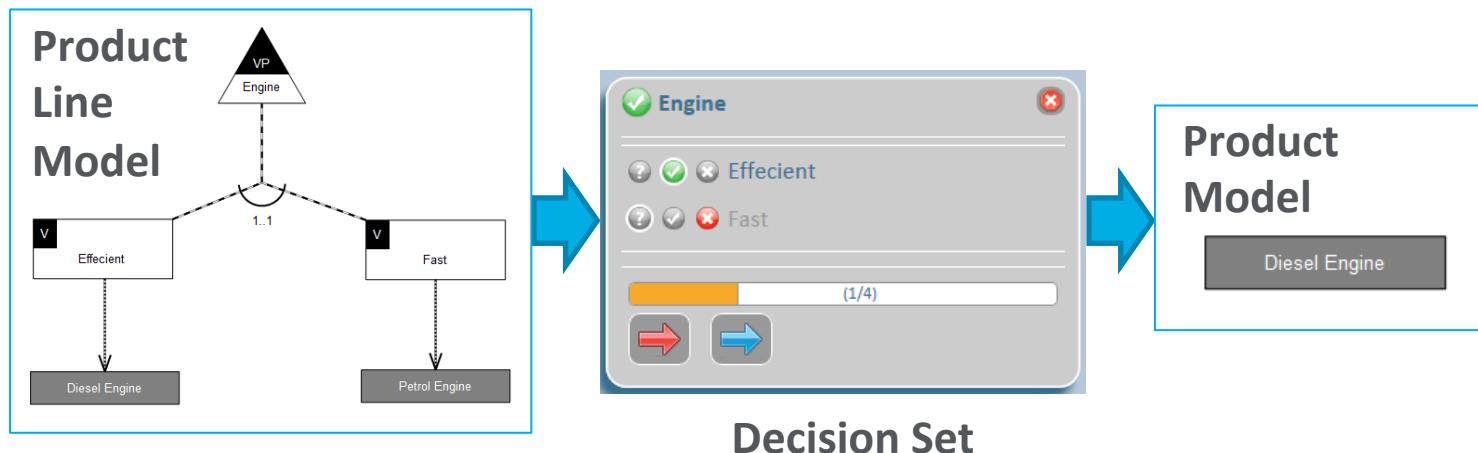
- Both Edit the Same Decision Sets



3 - Product Model Creation

- Auto-Create Product Models

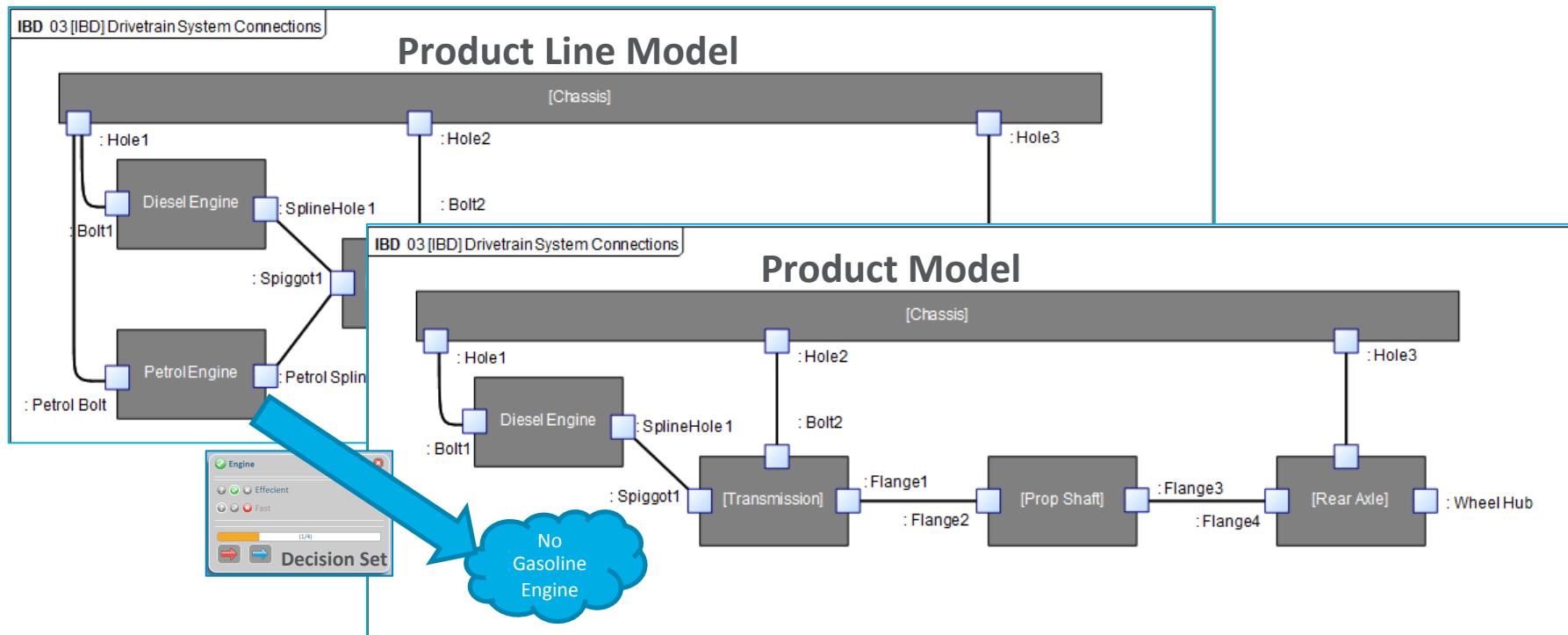
- Applies Variability Decisions
- Unnecessary Variation Points, Variants & Base Model Artefacts Removed



- Creates New Product Model Branch, Original Product Line Model Retained
- Product Model suitable for Trade Studies, Simulation, Documentation & Generation

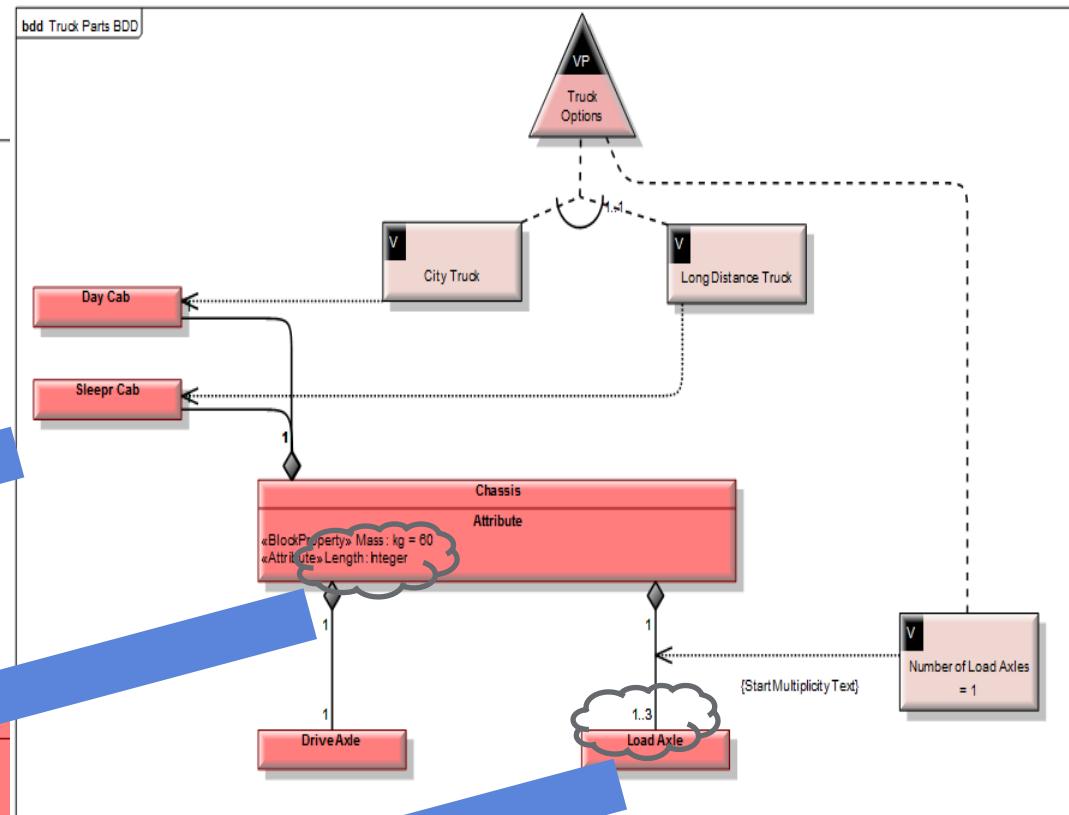
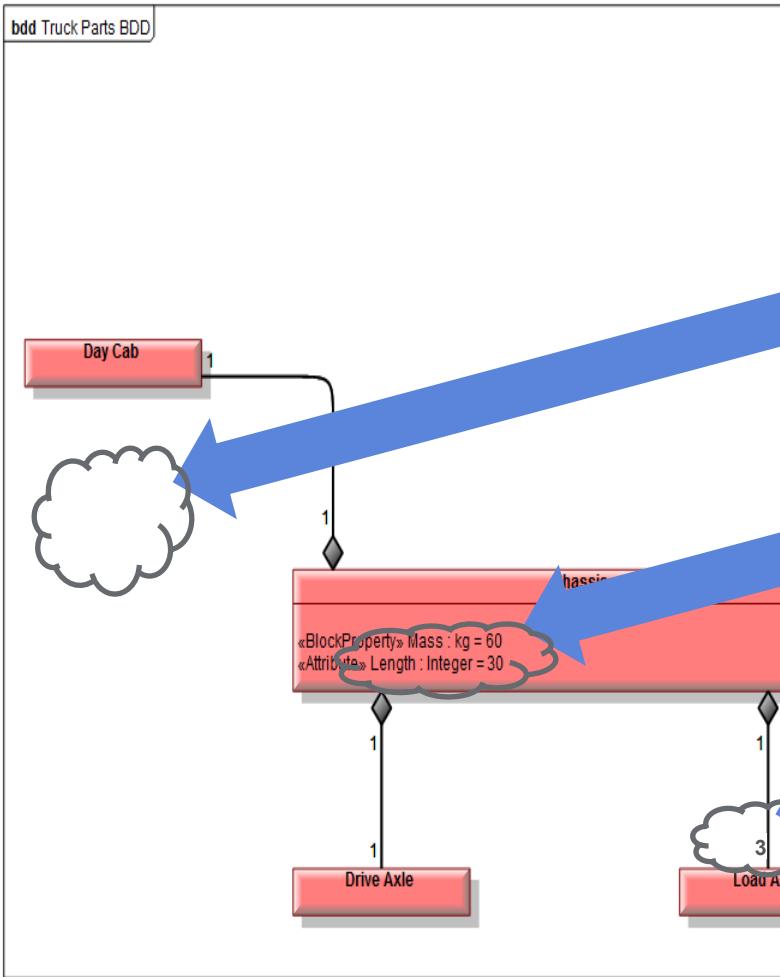
3 - Product Model Creation

- Auto-Create Product Models (IBD)



Variable Parameters

- Generated Product Model

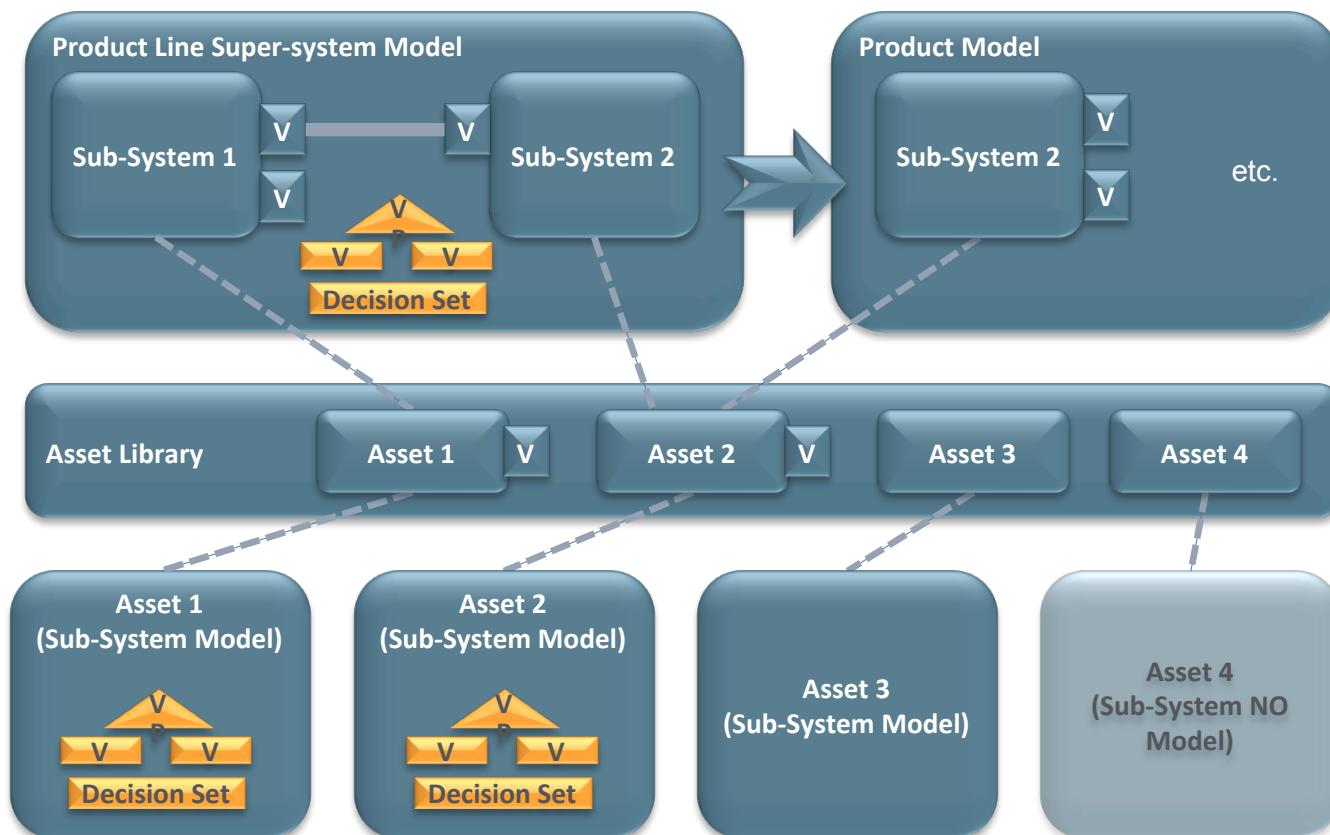


- Integrated MBSE, Modular Design & Variability Modeling = Model-Based Product Line Engineering

Product Line
Models

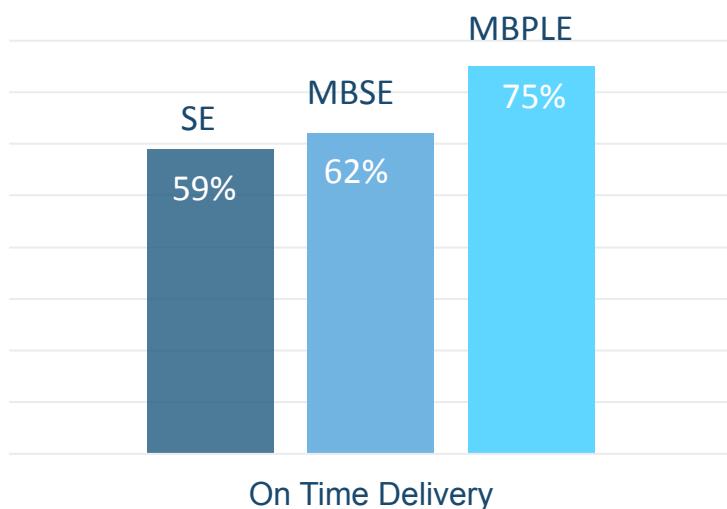
Links via
Assets

Sub-system
Product Line
Models

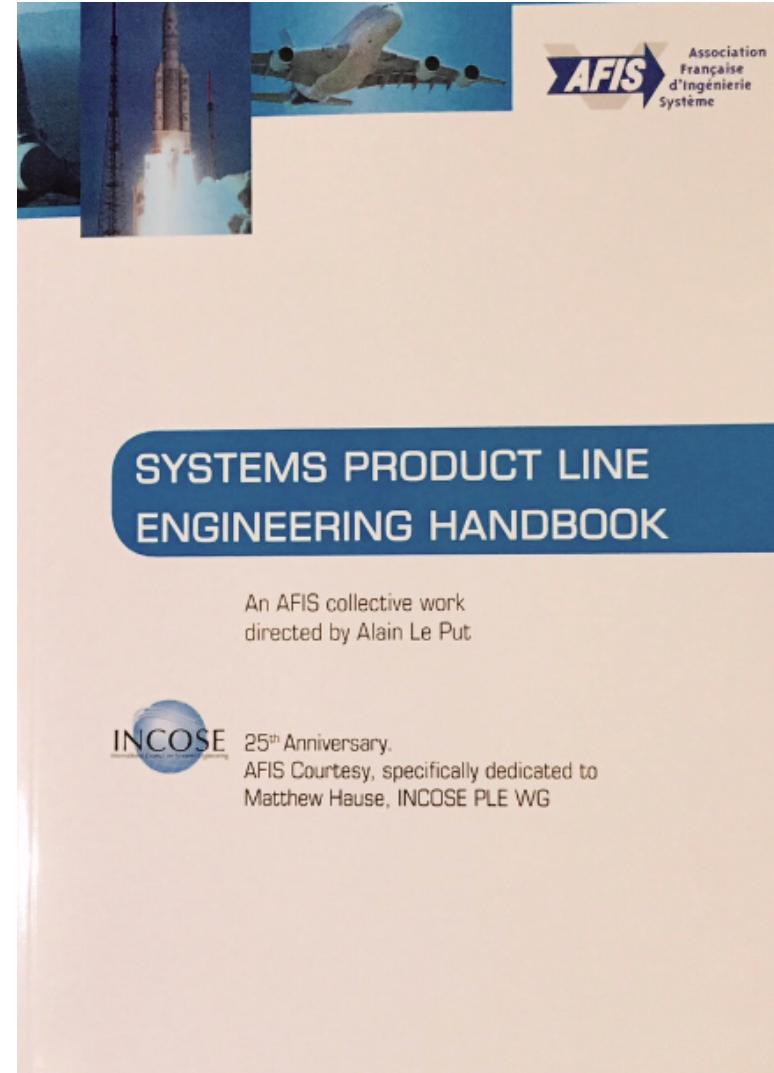


- **SE (Non-Modelled Systems Engineering)**
 - 59% of Projects Delivered on Time
- **MBSE (Model Based Systems Engineering)**
 - 62% of Projects Delivered on Time
 - Compared to SE
 - 55% Reduction in Total Development Cost per Project
- **MB-PLE (Model Based Product Line Engineering)**
 - 75% of Projects Delivered on Time
 - Compared to SE
 - 62% Reduction in Total Development Cost per Project

(EMF 2013 Independent Survey Results from 667 Systems engineering respondents)



These books are the foundation of the INCOSE Systems Product Line Engineering Handbook





Speaker

Thanks for your attention!

PTC® PRODUCT & SERVICE
ADVANTAGE®