Embedded Systems Innovation BY TNO
A CANON COMPANY

# The Next Challenge in Model-Based Development:

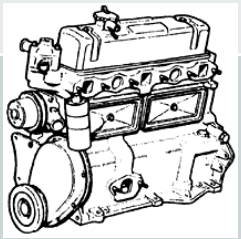
# Model (Life-Cycle) Management

Bart Theelen

16 June 2017

# Product Lines share Components




| BMC | Minor | Sprite | Mini | 1100/1300 |
|---|---|---|---|---|
| Engine | A-series | | | |
|  | in-line | | transversal | |
| | 803cc, 948cc, 1098cc | 948cc, 1098cc | 848cc, 970cc, 997cc, 998cc, 1071cc, 1098cc, 1275cc | 1098cc, 1275cc |
| Gearbox | behind engine (RWD) | | below engine (FWD) | |
| Suspension | torque bars + spring leaves | wishbones + spring leaves | rubber cones or hydrolastic | hydrolastic |

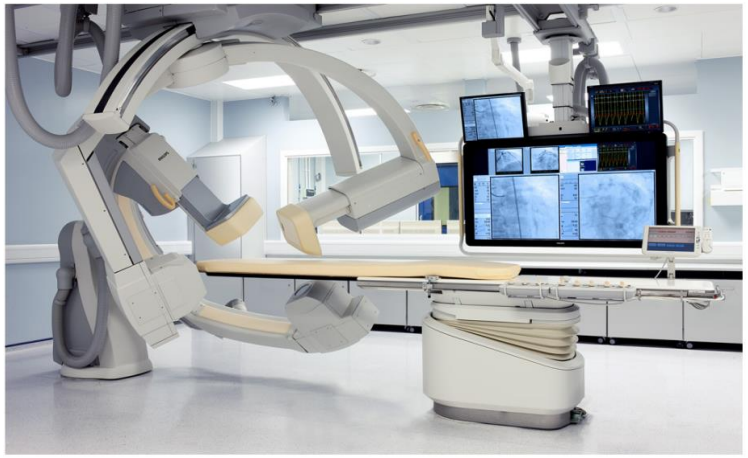
↓

aftermarket series of one customization

# What has changed?

- Not so much from system architecting perspective, but
- (More) software → series of one customization at build
- (More) model-based development

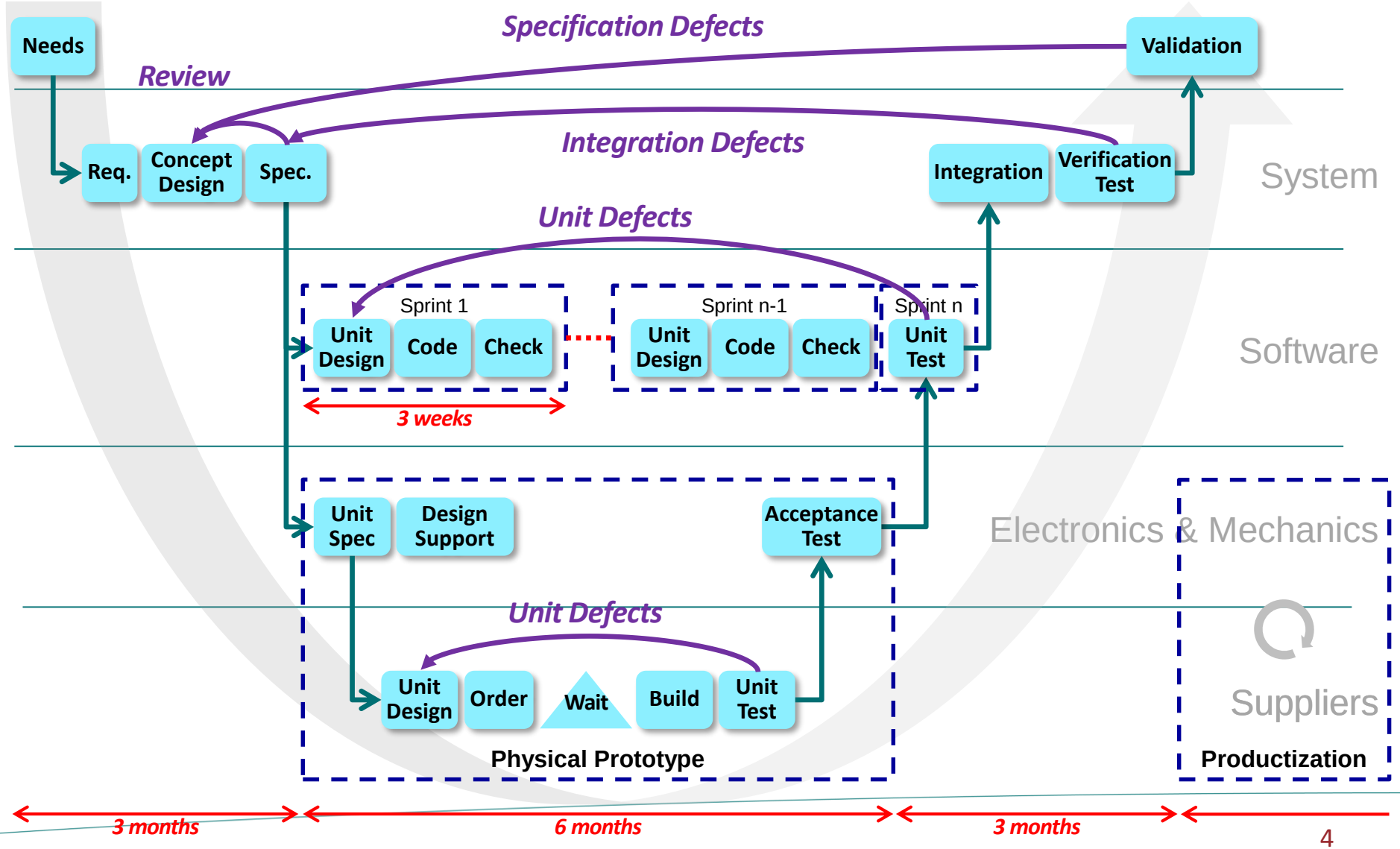
*Virtual prototyping → let users experience the system before building it*



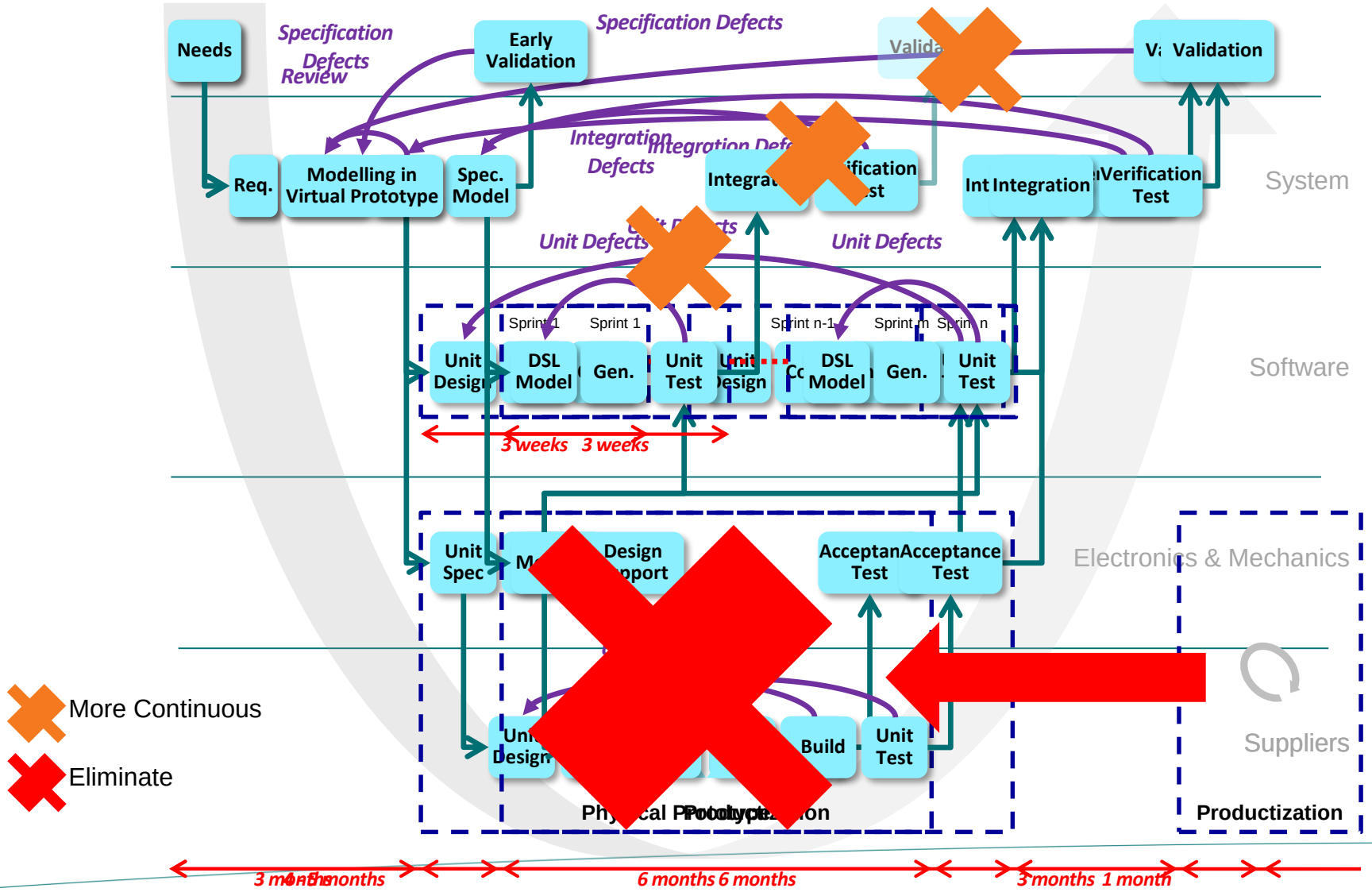
Coherent execution of real components together with co-simulated modelled components using existing interfaces

mechanics, mechatronics, electronics, X-ray, medical staff, patient, medical equipment

# Development Process



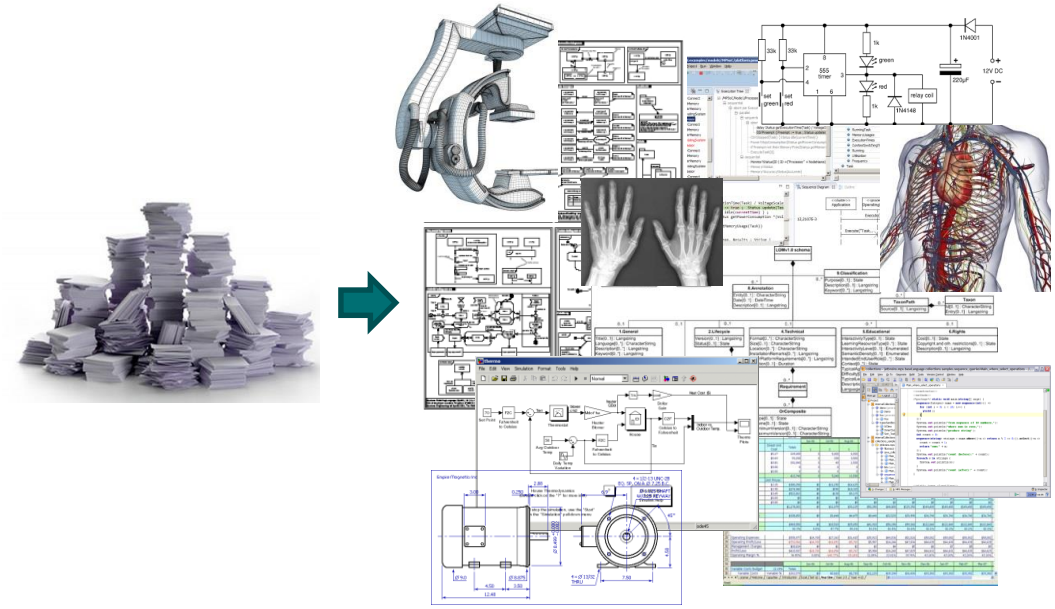
# Development Process



MBSE = Model-Based System Engineering
MBSA = Model-Based System Architecting

# Moving to Model-Based Development

*Industrial Practice* *State of the Art*



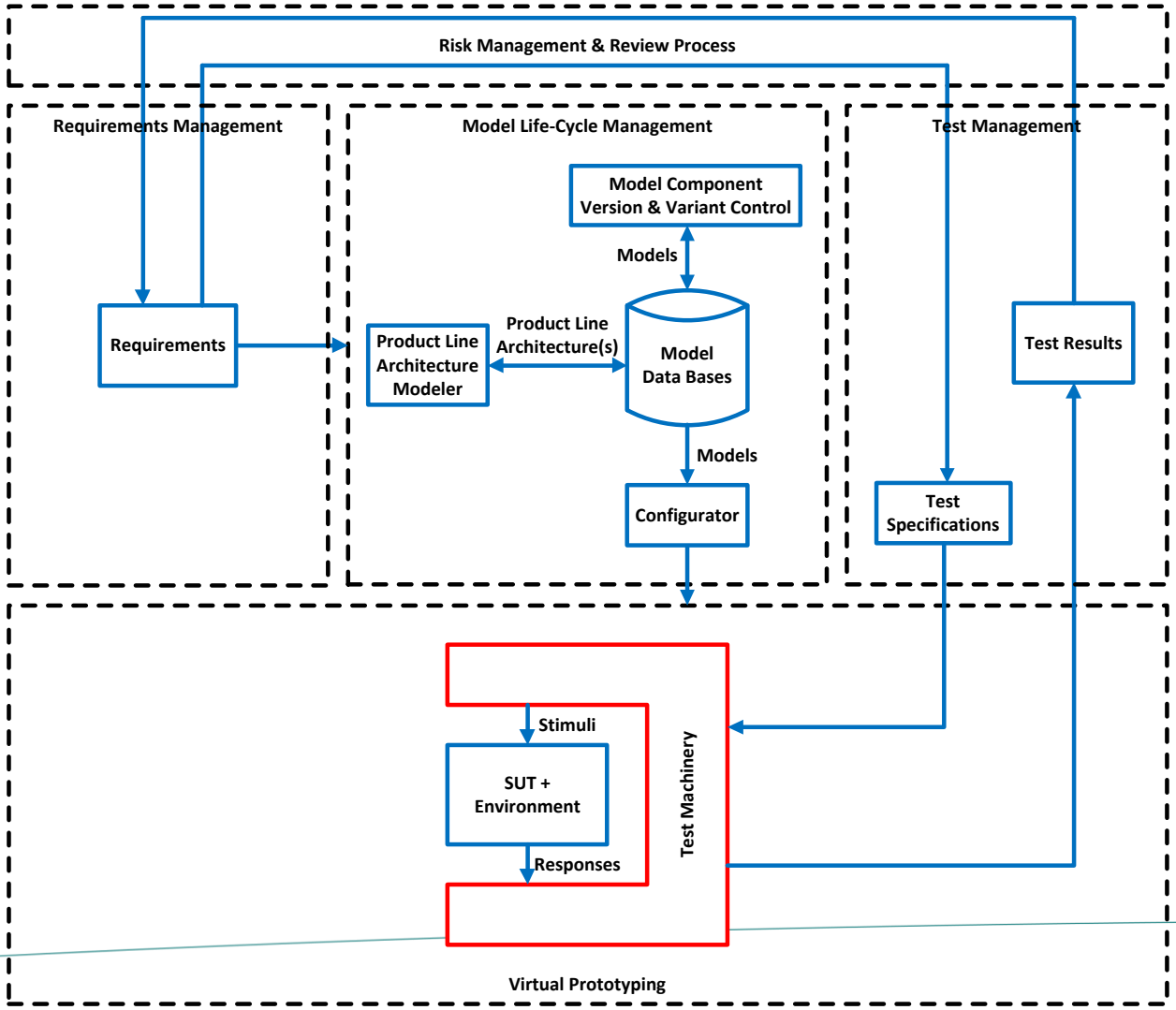
Key is Development Efficiency

Variety in languages, models & tools exceeds variety in document types

- How to get the complexity out of the development process using MBSE/MBSA?
- How to embed MBSE/MBSA in development process?
- What model to use when and for what purpose?
- How to ensure consistency of models (of same and different domains) throughout development process?
- Getting MBSE/MBSA tooling approved for evidence testing (in particular in view of safety requirements)!

# Model (Life-Cycle) Management

***dealing with diversity and vast amount of models (and their tools) to enable supporting MBSE/MBSA for complete life-cycle of system variants in business processes***



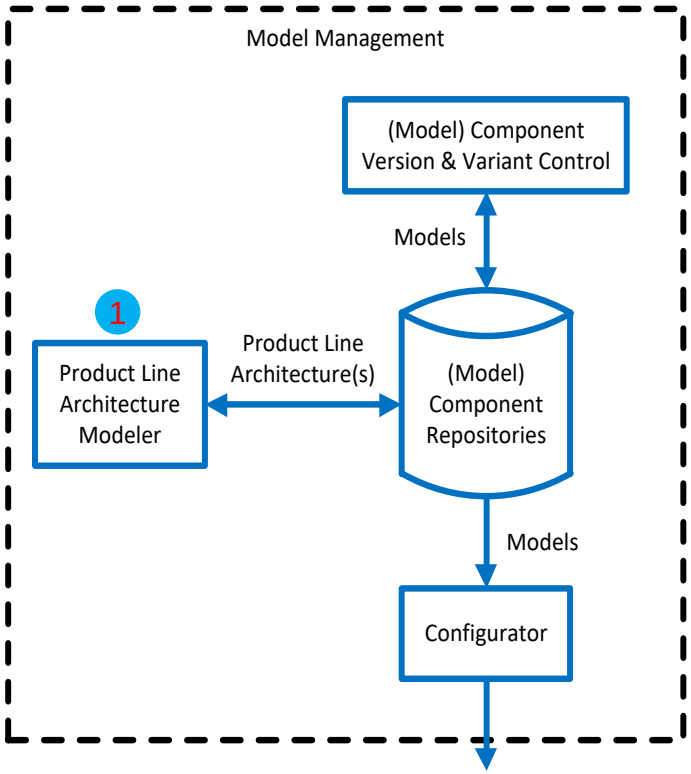
# Requirements for Model Management

- Evolving *product line architectures* as foundation
  - Architecture template for (multiple) product lines
  - Component composition rules
  - Component interface & test interface specifications
  - Model component reuse across product line architectures

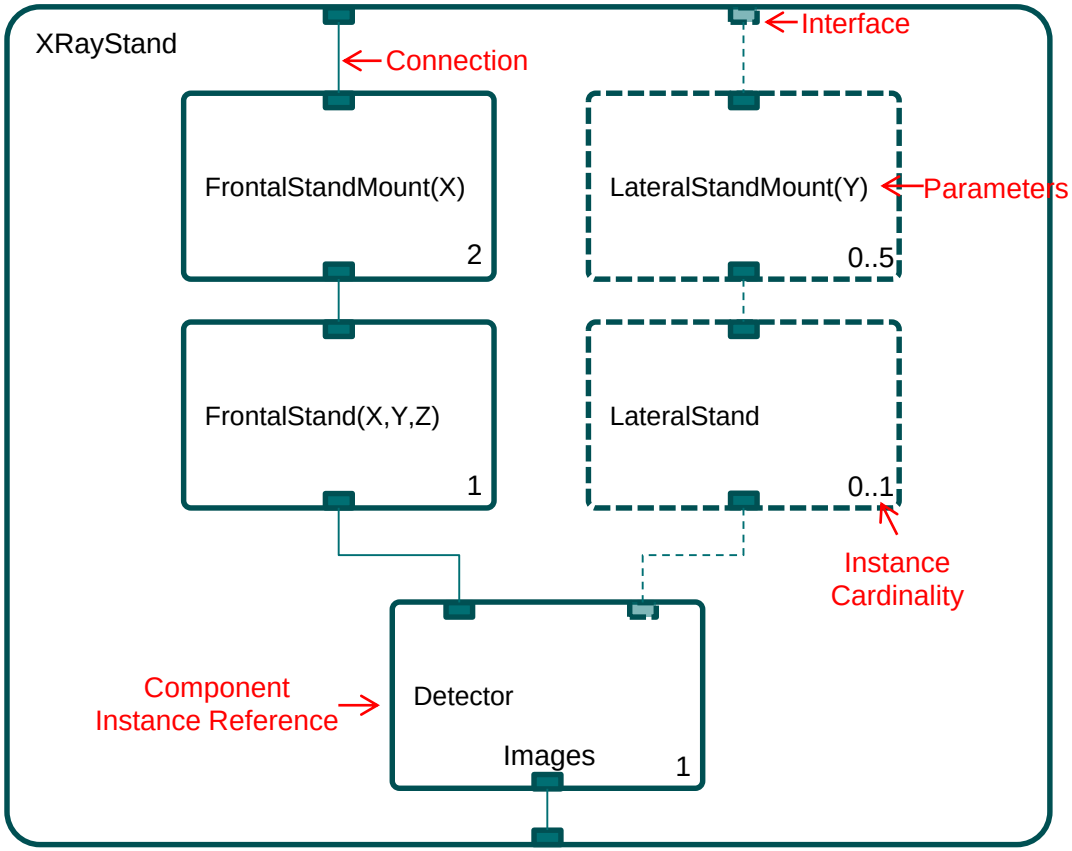
1

Product Line Architecture is a Designers' View

(CA**FC**R)



# Product Line Architecture Modelling




- Hierarchical Composition
- Interface Specifications
- Component Connections
- Composition Constraints
- Parameters (Constrained)
- Cardinalities (Constrained)
- Model Types (Associated with each component, incl. composites)

Dashed = Optional

Expansion Operator

Composition Constraints

LateralStand ⇔ LateralStandMount

Detector.FD12 ∨ Detector.FD15 ∨ Detector.FD20 ⇒ ¬ LateralStand ∧ ¬ LateralStandMount

Dectector.FD12 ∨ Detector.FD1212 ⇔ FrontalStand.PolyG2

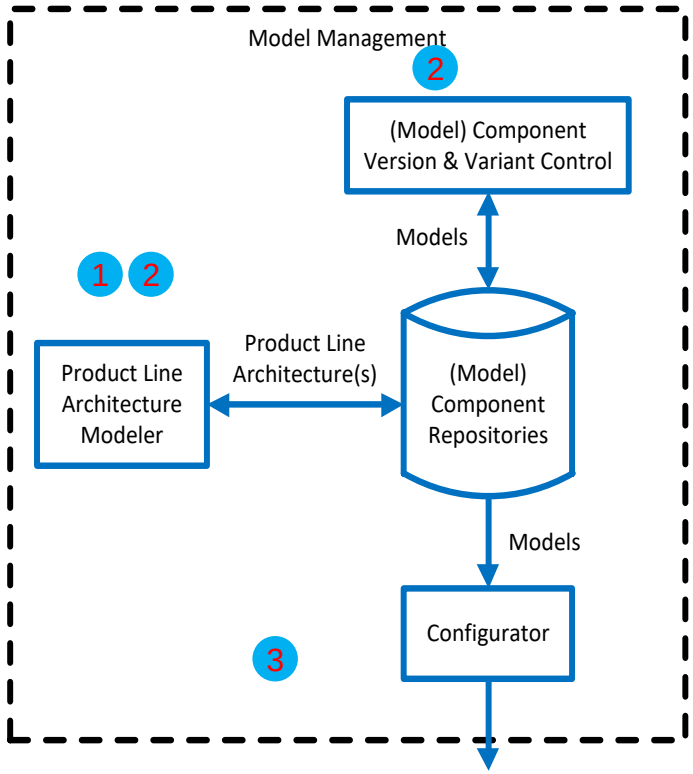
Dectector.FD12 ∨ Detector.FD20 ∨ Detector.FD2012 ∨ Detector.FD2015 ⇔ FrontalStand.Clea

...

Also: Constraints on Parameter Values & Cardinalities

# Requirements for Model Management

- Evolving *product line architectures* as foundation
  - Architecture template for (multiple) product lines
  - Component composition rules
  - Component interface & test interface specifications
  - Model component reuse across product line architectures
- Model *consistency* tracking & enforcement
  - Model types (depending on validation & verification purpose)
  - Model governance by product line architecture & interface compliance checks
  - Cross discipline / department notification of changes
- Easy instantiation / configuration of *virtual prototypes* considering
  - System variant and/or *features* of interest
  - Desired validation / verification procedure
  - Available real components / model components / model types
  - Available model execution equipment



# Features

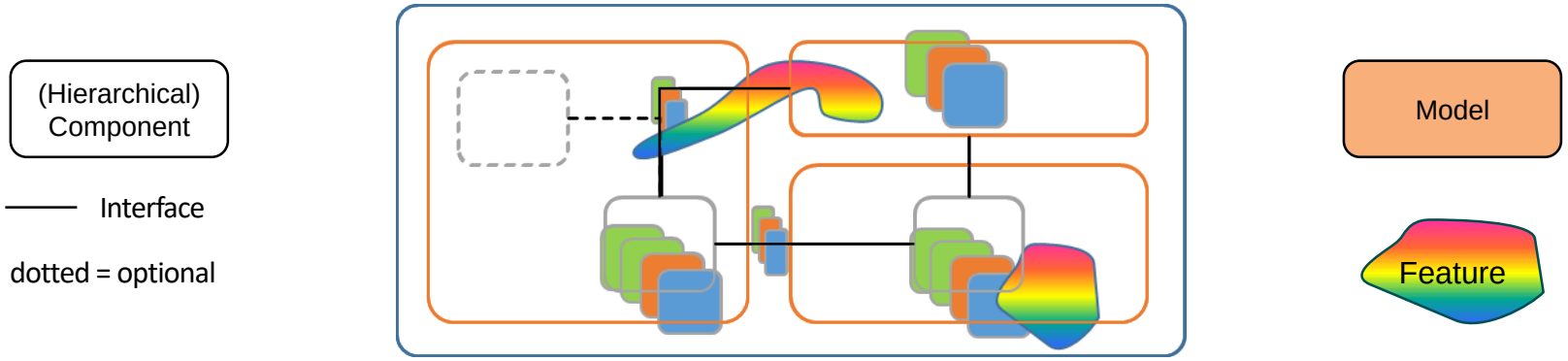


| | Basic | Cooper S |
|---|---|---|
| Engine | 848cc | 1275cc |
| Carburation | Single 1.25” | Twin 1.5” |
| Tank | 24l | 2x 24l |
| Wheels | 3.5x10 | 4.5x10 |

Features are a Customers’ View

(**CA**FCR)

# Product Line Architecture ↔ Features

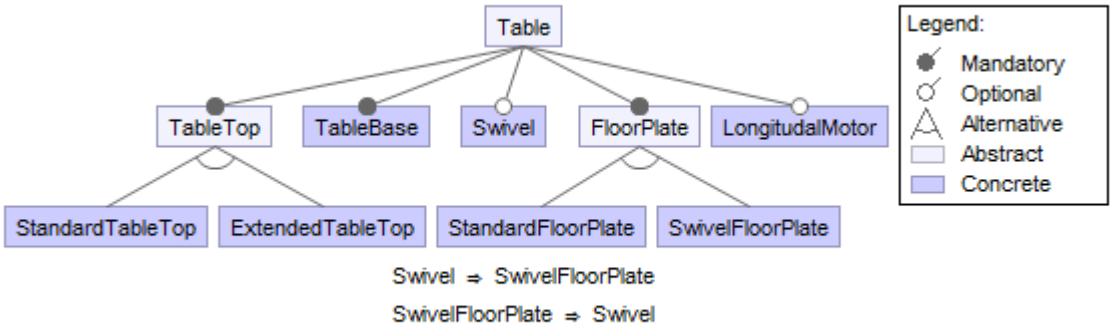



Features can relate to not necessarily (directly) related components & interfaces

How to keep product line architecture and feature views consistent when things evolve?

# Feature Modelling

- Capture variability in system's capabilities "catalog" view
  - Can be at solution level (like Dell website)
    - Compared with product line architecture model, this misses the interfaces

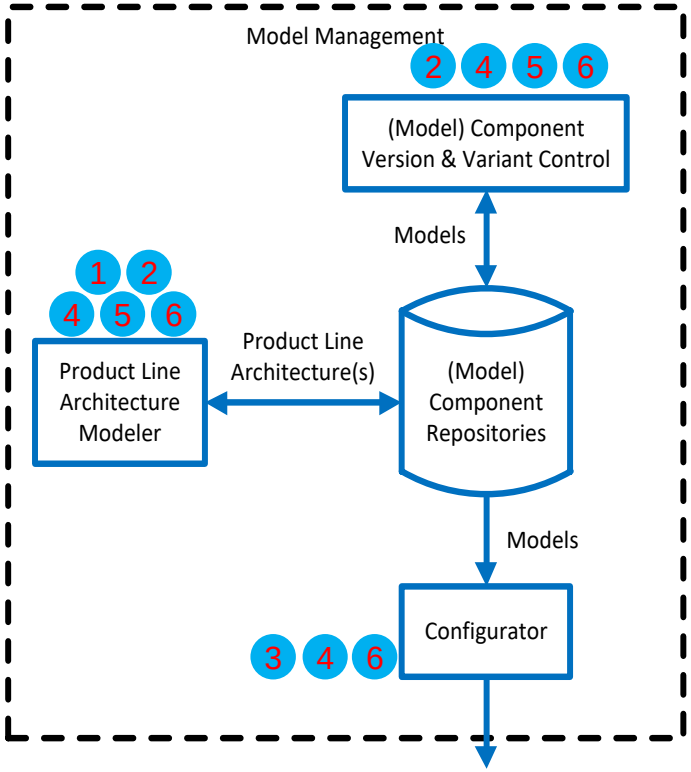



Possible extensions to existing feature model formalism that we may need include parameters and cardinalities

  - Can be at user-need level
    - What a user can do with the system
    - Like: Neuro or Cardio with optional Radial Access

# Requirements for Model Management

- Evolving *product line architectures* as foundation
  - Architecture template for (multiple) product lines
  - Component composition rules
  - Component interface & test interface specifications
  - Model component reuse across product line architectures
- Model *consistency* tracking & enforcement
  - Model types (depending on validation & verification purpose)
  - Model governance by product line architecture & interface compliance checks
  - Cross discipline / department notification of changes
- Easy instantiation / configuration of *virtual prototypes* considering
  - System variant and/or *features* of interest
  - Desired validation / verification procedure
  - Available real components / model components / model types
  - Available model execution equipment
- Status monitoring and *access control* for real components / model components to match (existing) business processes (ALM/PLM)
- Version control supporting *co-evolution* of model components (using variety of existing repositories) including design rationale tracking
- *Unobtrusive,* effective and *scalable* engineering *tools*
  - Customizable existing (industrial) (ALM/PLM) tools (also for validation/verification/test results)




MBSE/MBSA Tool Evolution (Maintenance & Support)

# What's Available?

- Complete tool with multi-disciplinary virtual prototyping & model management
  - PREEVision for automotive → unsuitable for any other domain
- Example ALM/PLM tools allowing SysML-based modelling
  - IBM Rhapsody Design Manager
  - PTC Integrity MBSE

  } Code generation possible
  No proper virtual prototyping
  Domain specific customization via UML profiles (not via DSLs)
- Example architecting tools
  - Capella → No link to ALM/PLM
- Example software product line tools
  - Pure::Variant
  - Supermod (Academic)

  } Software not system
- Version control tools for software development are insufficient

Embedded Systems Innovation BY TNO

# What's Available?

*rough first impression based on 'spec' sheets → your help is appreciated to get a better overview*

| | PREEVision | Design Manager | Integrity MBSE | Capella | Pure::Variant | Supermod | Traditional Version Control |
|---|---|---|---|---|---|---|---|
| Product Line Architectures | ✓ | SysML | SysML | ✓ | ✓ | ✓ | ✗ |
| Features | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ |
| Consistency Tracking & Enforcement | ✓ | Possible but Difficult | ? | ✓ | ? | ? | ✗ |
| Virtual Prototyping | ✓ | Software Build Only | Software Build Only | Depends on V&V Goals | Software Build Only | Software Build Only | Software Build Only |
| Access Control | ? | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ |
| Consistent Co-Evolution | ✓ | ? | ? | ? | ? | ? | ✓ |
| Component Model Governance | ✓ | ✗ | ✗ | ? | ? | ? | ✗ |
| Scalable | ✓ | ✓ | ✓ | ? | ? | ? | ✓ |
| Domain Independent Customization | ✗ | UML Profiles | UML Profiles | UML Profiles + DSLs | ? | ? | ✗ |
| Y-Chart Based | ✓ | Depends on SysML use | Depends on SysML use | ✓ | ? | ? | ✗ |