BUILDING A DSL FOR SUPPORTING COMPUTATIONAL DESIGN SYNTHESIS OF CYBER-PHYSICAL SYSTEMS

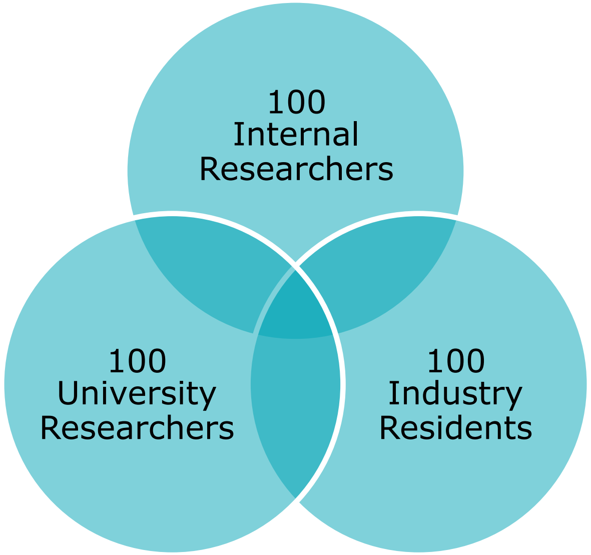
WHAT DOES IT TAKE?

KLAAS GADEYNE, FLANDERS MAKE

FLANDERS MAKE
MANUFACTURING INNOVATION NETWORK

Flanders Make - Organisation

- Research initiative of/for the Flemish industry
- Government + industry sponsored
- Domains:
  - Mechatronics, cyberphysical (machines + automotive)
  - Product development methods (f.i. MBE/MDD)
  - Advanced manufacturing technologies (f.i. additive manufacturing)
- Founded in 2003, merged in 2014

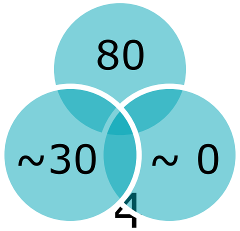


100
Internal
Researchers

100
University
Researchers

100
Industry
Residents

Current:



80

~30

~ 0

Actual industrial MEMBERS

CNH INDUSTRIAL

BEKAERT
better together

.Ocean
nautical innovations

SIEMENS

Continental

DANA

SPICER
DRIVETRAIN COMPONENTS

BOMBARDIER
the evolution of mobility

LayerWise
Your Partner in Additive Manufacturing

ZF

LVD

CENAT
ONTROL ENGINEERING & AUTOMATIO

Flexible Robotic
Solutions
- ahead in innovation -

Materialise
innovators you can count on

c-mec
creative mechanical solutions

Atlas Copco

pedeo

SL
SABCA LIMBURG NV

PICANOL GROUP

MAEX
PRECISION-PRODUCTION NV

XEIKON

NiniX
Technologies

Hansen®
industrial gearboxes

BLUWAYS

DEFAWES
GEARS AND GEARBOXES

VAN DE WIELE

GRAMMER

PHILIPS

RECTICEL
The passion for comfort

punch
powertrain
Gear up for the future

TENNECO

VOLVO
VOLVO TRUCKS

BEST
GETS EVERYTHING SORTED

LEUVEN
air bearings

œrlikon
balzers

televic
rail

vcst

INDUCT

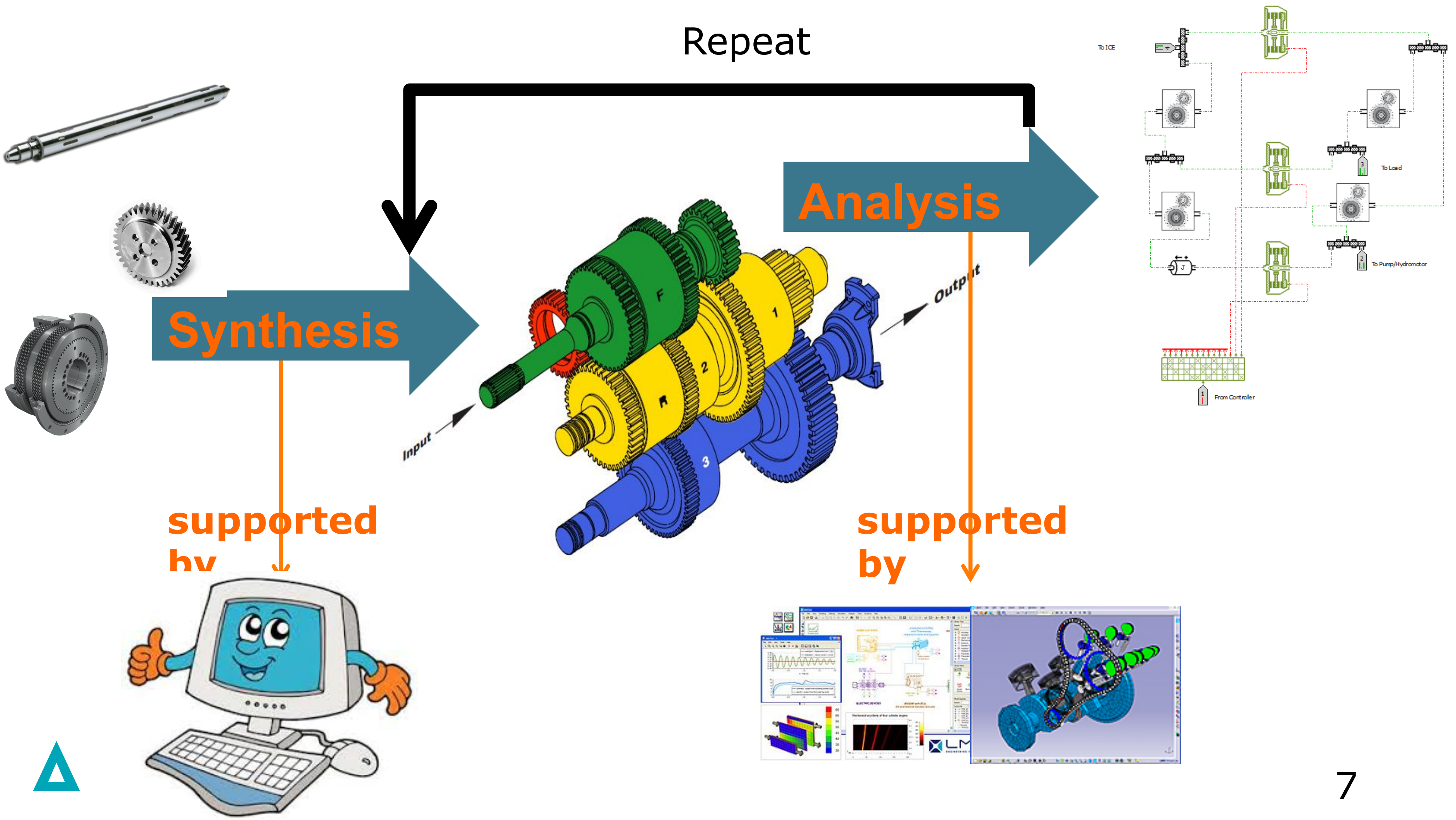
Gilbos

VOLVO

TERUMO®
We keep life flowing

VHA
Van Hoecke Automation nv

What is computational design synthesis?



Repeat

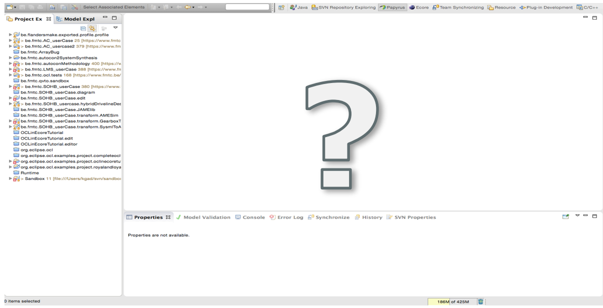
Synthesis

Analysis

supported by

supported by

What do you need for CDS?



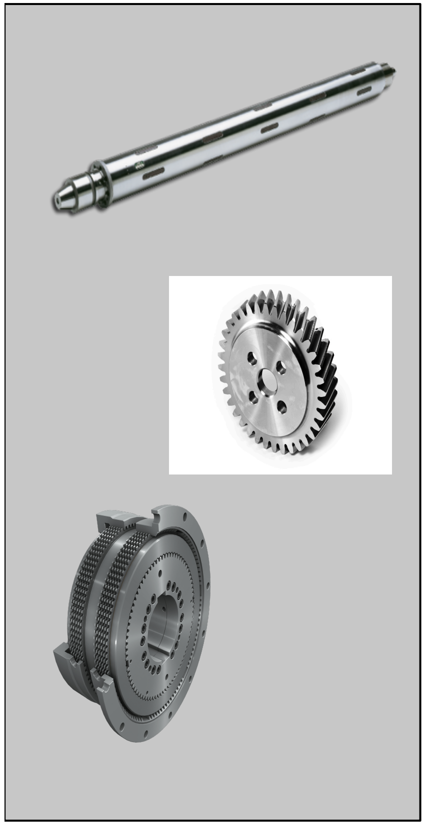
```
be.flandersmake.exported.profile.profile
> be.fmtc.AC_userCase 25 [https://www.fmtc
> be.fmtc.AC_usercase2 379 [https://www.fm
be.fmtc.ArrayBug
be.fmtc.autocon2SystemSynthesis
> be.fmtc.autoconMethodology 400 [https://w
> be.fmtc.LMS_userCase 388 [https://www.fn
> be.fmtc.ocl.tests 168 [https://www.fmtc.be/
be.fmtc.qvto.sandbox
> be.fmtc.SOHB_userCase 380 [https://www.
be.fmtc.SOHB_userCase.diagram
be.fmtc.SOHB_userCase.edit
> be.fmtc.SOHB_usercase.hybridDrivelineDes
be.fmtc.SOHB_userCase.JAMElib
be.fmtc.SOHB_userCase.transform.AMESim
be.fmtc.SOHB_userCase.transform.GearboxT
be.fmtc.SOHB_userCase.transform.SysmlToA
OCLinEcoreTutorial
OCLinEcoreTutorial.edit
OCLinEcoreTutorial.editor
org.eclipse.ocl
org.eclipse.ocl.examples.project.completeocl
org.eclipse.ocl.examples.project.oclinecoretu
org.eclipse.ocl.examples.project.royalandloya
Runtime
> Sandbox 11 [file:///Users/kgad/svn/sandbo
```

Properties are not available.

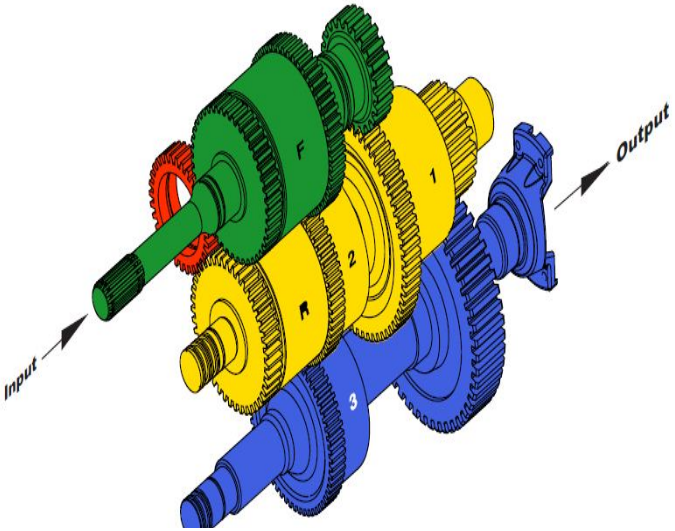
0 items selected

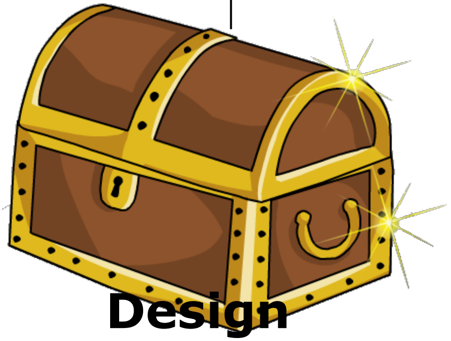
Requirements and objectives

Optimize weight
Max Bounding Box = …
Num Gears = …
Ratios = …



synthesis



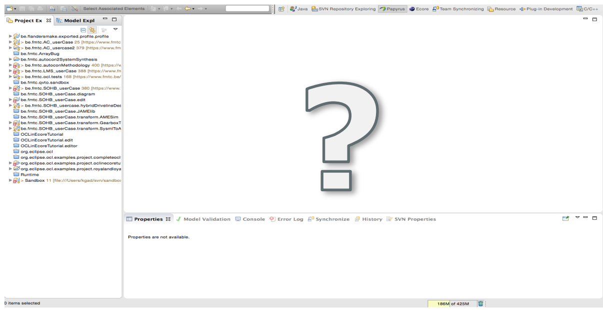


Design knowledge

***Valuable* System architecture : topology + key properties**

Component Types

A DSML for CDS



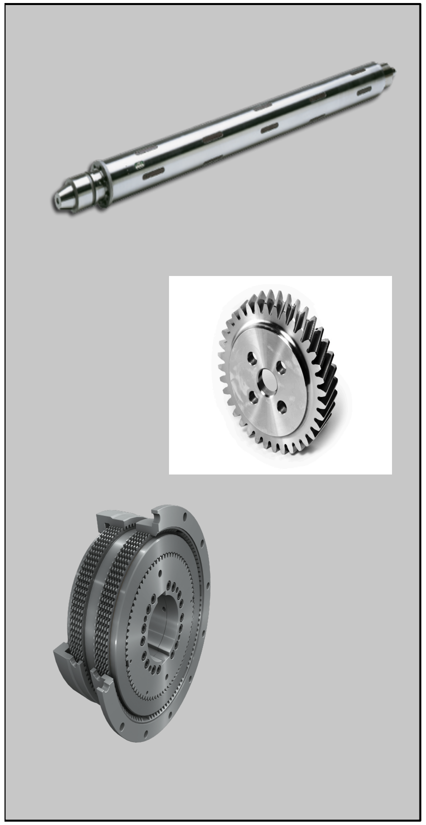
```
be.flandersmake.exported.profile.profile
> be.fmtc.AC_userCase 25 [https://www.fmtc
> be.fmtc.AC_usercase2 379 [https://www.fm
be.fmtc.ArrayBug
be.fmtc.autocon2SystemSynthesis
> be.fmtc.autoconMethodology 400 [https://w
> be.fmtc.LMS_userCase 388 [https://www.fn
> be.fmtc.ocl.tests 168 [https://www.fmtc.be/
be.fmtc.qvto.sandbox
> be.fmtc.SOHB_userCase 380 [https://www.
be.fmtc.SOHB_userCase.diagram
be.fmtc.SOHB_userCase.edit
> be.fmtc.SOHB_usercase.hybridDrivelineDes
be.fmtc.SOHB_userCase.JAMElib
be.fmtc.SOHB_userCase.transform.AMESim
be.fmtc.SOHB_userCase.transform.GearboxT
be.fmtc.SOHB_userCase.transform.SysmlToA
OCLinEcoreTutorial
OCLinEcoreTutorial.edit
OCLinEcoreTutorial.editor
org.eclipse.ocl
org.eclipse.ocl.examples.project.completeocl
org.eclipse.ocl.examples.project.oclinecoretu
org.eclipse.ocl.examples.project.royalandloya
Runtime
> Sandbox 11 [file:///Users/kgad/svn/sandbox
```

Properties are not available.

0 items selected

186M of 425M

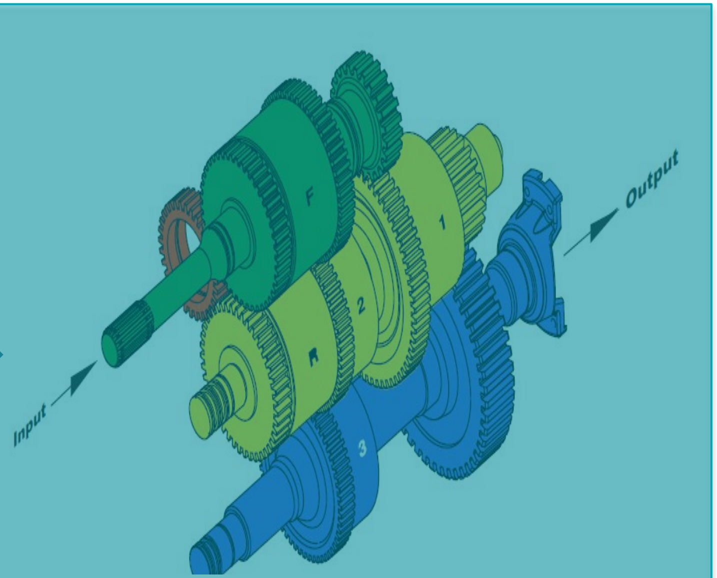
Requirements and objectives



Optimize weight
Max Bounding Box = …
Num Gears = …
Ratios = …

synthesis

Design knowledge

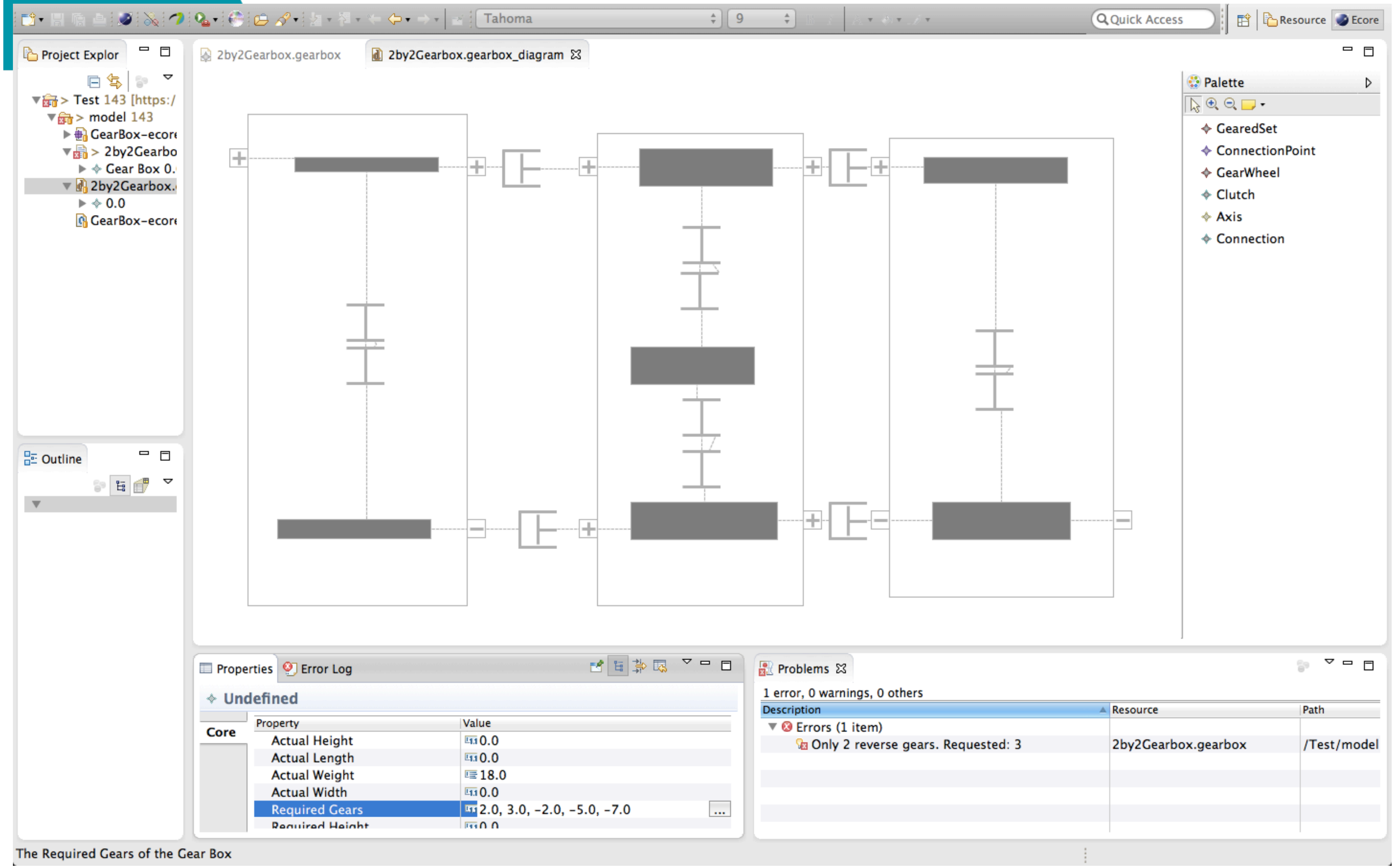


***Valuable* System architecture :**
topology + key properties

MOST DSMLs CREATED TODAY

Component Types

Example: A DSML for gearbox design



Undefined

Core

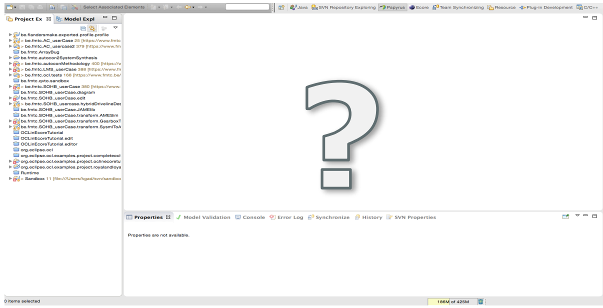
Property	Value
Actual Height	0.0
Actual Length	0.0
Actual Weight	18.0
Actual Width	0.0
Required Gears	2.0, 3.0, -2.0, -5.0, -7.0
Required Height	0.0

1 error, 0 warnings, 0 others

Description	Resource	Path
Errors (1 item)		
Only 2 reverse gears. Requested: 3	2by2Gearbox.gearbox	/Test/model

The Required Gears of the Gear Box

A DSML for CDS

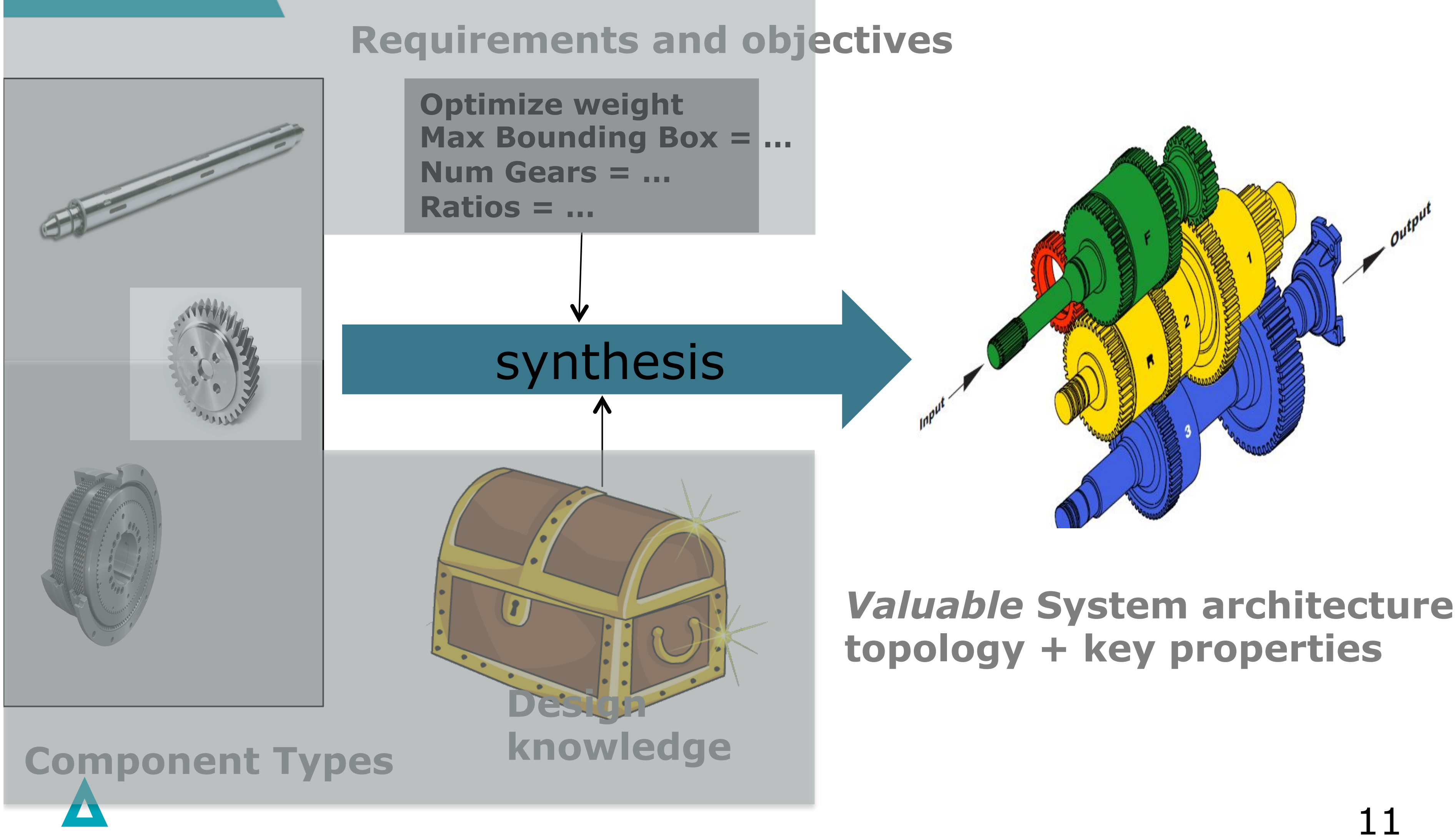


be.flandersmake.exported.profile.profile
> be.fmtc.AC_userCase 25 [https://www.fmtc
> be.fmtc.AC_usercase2 379 [https://www.fm
be.fmtc.ArrayBug
be.fmtc.autocon2SystemSynthesis
> be.fmtc.autoconMethodology 400 [https://w
> be.fmtc.LMS_userCase 388 [https://www.fn
> be.fmtc.ocl.tests 168 [https://www.fmtc.be/
be.fmtc.qvto.sandbox
> be.fmtc.SOHB_userCase 380 [https://www.
be.fmtc.SOHB_userCase.diagram
be.fmtc.SOHB_userCase.edit
> be.fmtc.SOHB_usercase.hybridDrivelineDes
be.fmtc.SOHB_userCase.JAMElib
be.fmtc.SOHB_userCase.transform.AMESim
be.fmtc.SOHB_userCase.transform.GearboxT
be.fmtc.SOHB_userCase.transform.SysmlToA
OCLinEcoreTutorial
OCLinEcoreTutorial.edit
OCLinEcoreTutorial.editor
org.eclipse.ocl
org.eclipse.ocl.examples.project.completeocl
org.eclipse.ocl.examples.project.oclinecoretu
org.eclipse.ocl.examples.project.royalandloya
Runtime
> Sandbox 11 [file:///Users/kgad/svn/sandbox

Properties are not available.

0 items selected

186M of 425M



Requirements and objectives

Optimize weight
Max Bounding Box = …
Num Gears = …
Ratios = …

synthesis

Component Types

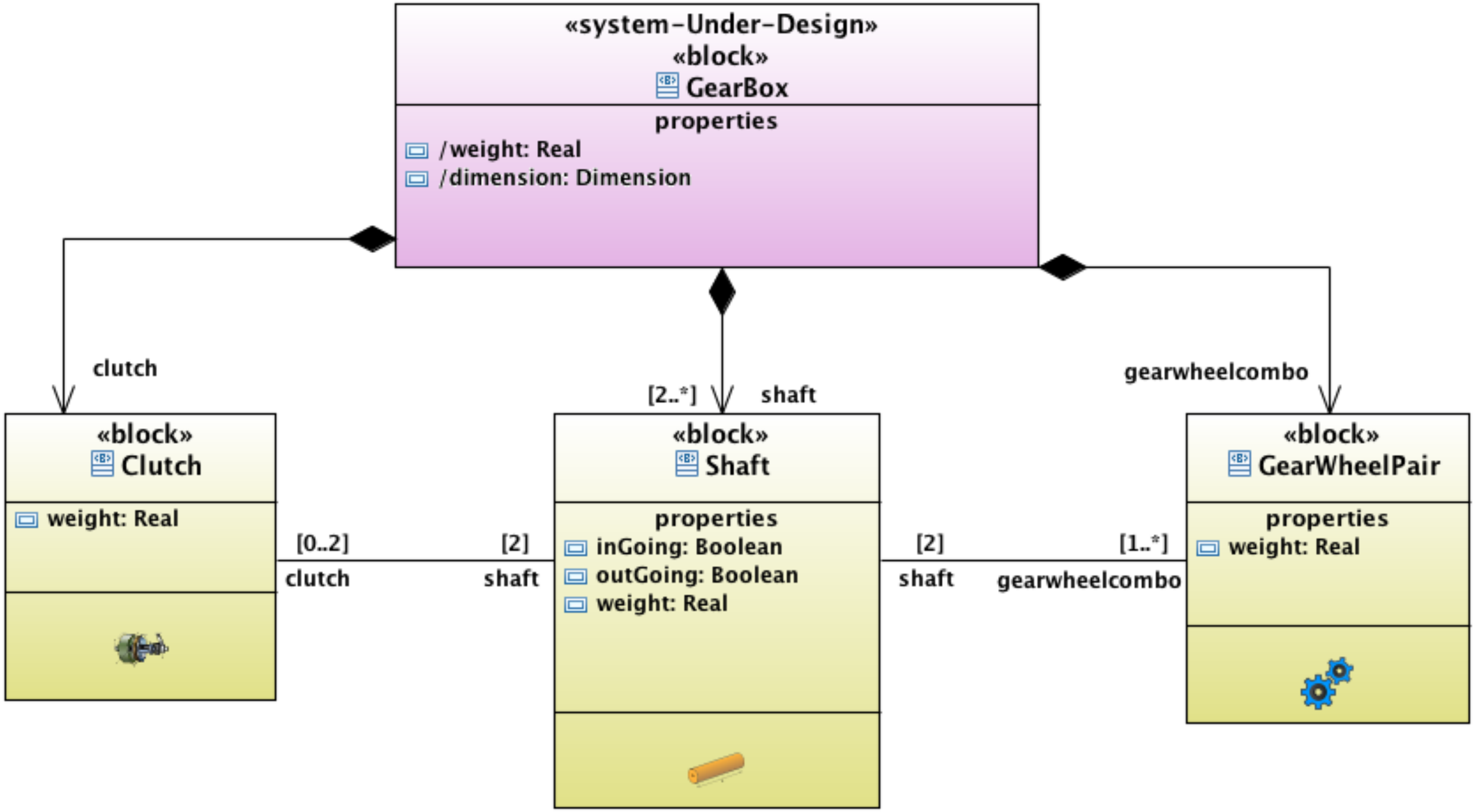
Design knowledge

Valuable System architecture : topology + key properties

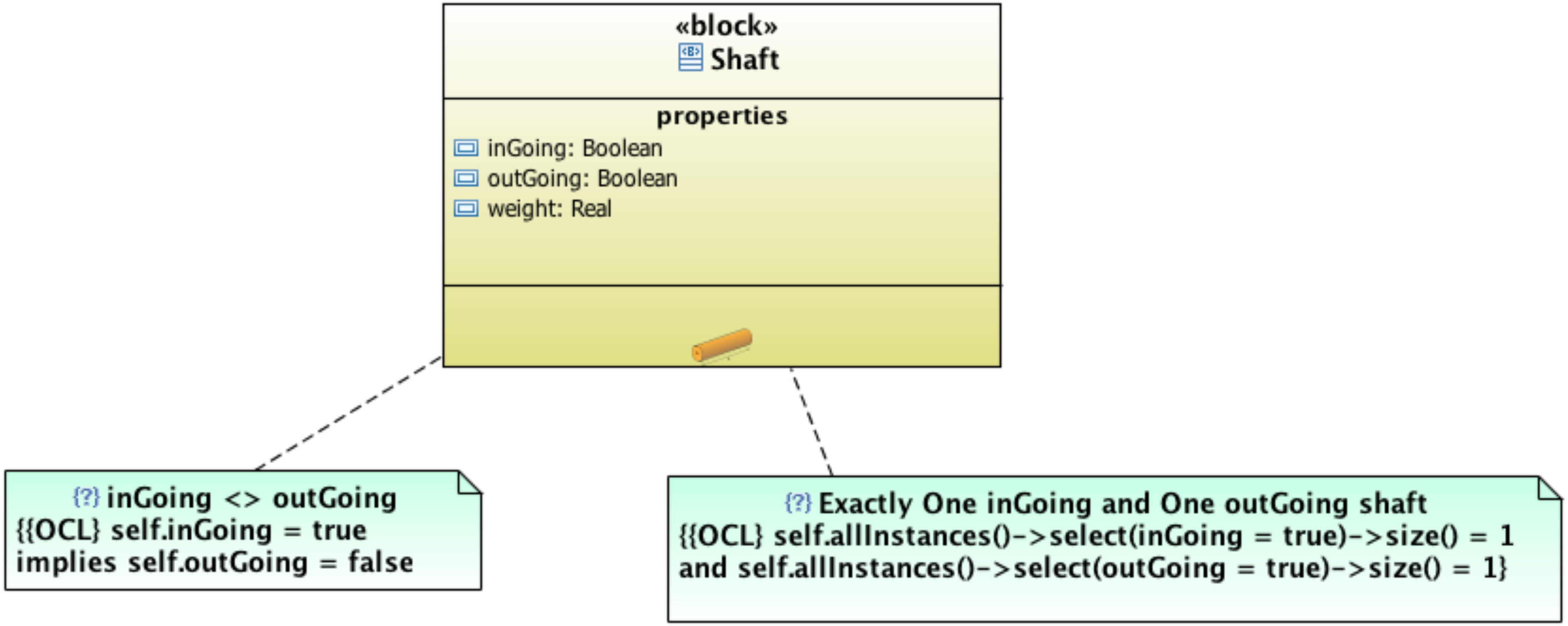
A DSML for supporting computational design synthesis of CPS

- The term ‘design space’ is often used to denote the Set of feasible CPS architectures that can satisfy the requirements
- A Synthesis algorithm then selects (and presents) valuable architectures to the designer
- The goal of a DSML that supports CDS is to **model this ‘design space’**

Design space = meta-model?



with (a lot of) OCL constraints?



«block»
Shaft

properties

inGoing: Boolean

outGoing: Boolean

weight: Real

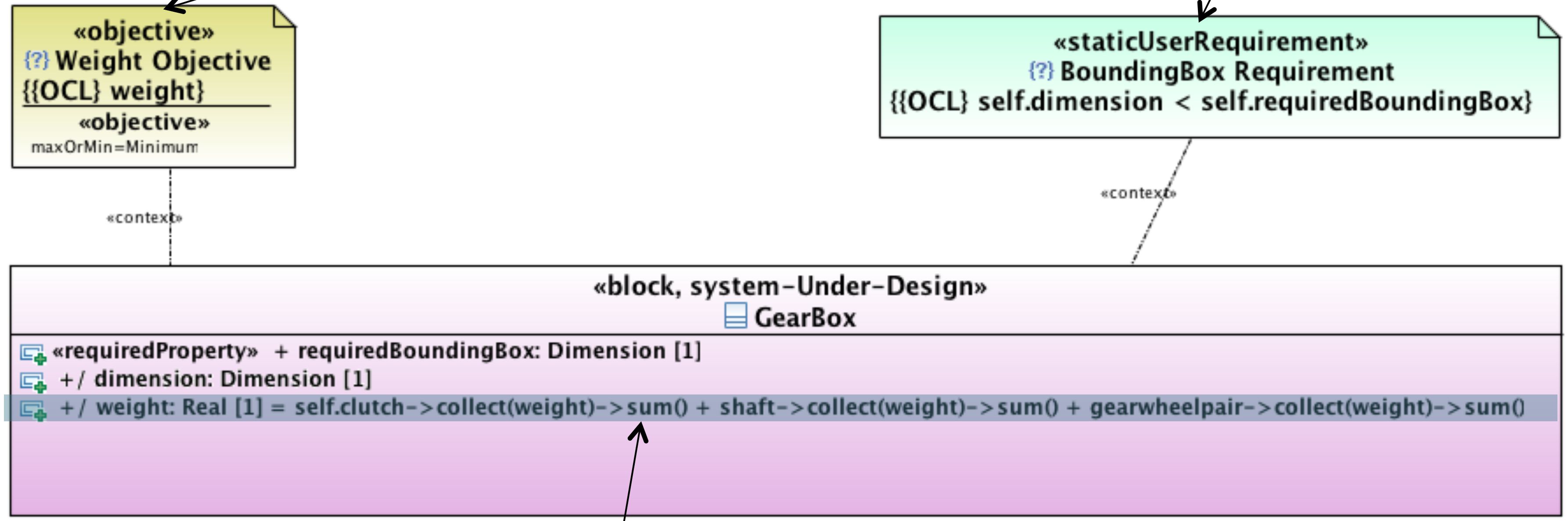
{?} inGoing <> outGoing
{{OCL} self.inGoing = true
implies self.outGoing = false

{?} Exactly One inGoing and One outGoing shaft
{{OCL} self.allInstances()->select(inGoing = true)->size() = 1
and self.allInstances()->select(outGoing = true)->size() = 1}

Design space = meta-model + requirements for the instances

objectives = minimal weight

bounding box requirement



«objective»
{?} Weight Objective
{{OCL} weight}
«objective»
maxOrMin=Minimum

«context»

«staticUserRequirement»
{?} BoundingBox Requirement
{{OCL} self.dimension < self.requiredBoundingBox}

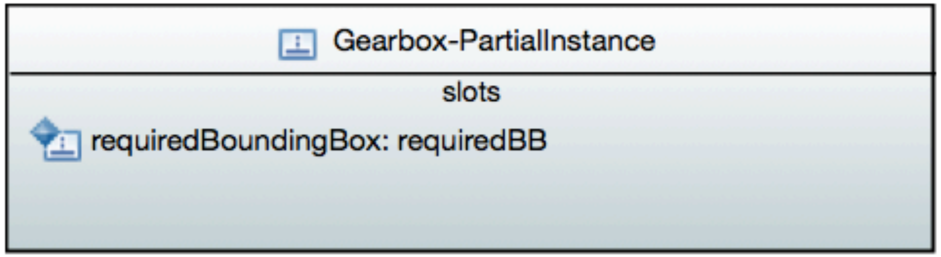
«context»

«block, system-Under-Design»
GearBox

«requiredProperty» + requiredBoundingBox: Dimension [1]
+/ dimension: Dimension [1]
+/ weight: Real [1] = self.clutch->collect(weight)->sum() + shaft->collect(weight)->sum() + gearwheelpair->collect(weight)->sum()

weight of the gearbox = sum of the weights of all its parts

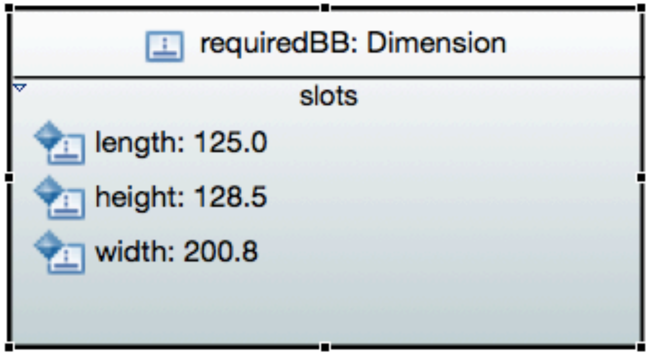
Design space = meta-model + requirements for the instances



Gearbox-PartialInstance

slots

requiredBoundingBox: requiredBB



requiredBB: Dimension

slots

length: 125.0

height: 128.5

width: 200.8

Going for a ‘lightweight extension’ approach

- SysML + OCL seem to be a good candidate to serve as a ‘Host language’ when creating a DSML for modeling the design space of CyberPhysical Systems,
  - and hence as a model that contains all necessary information for the Computational Design Synthesis ‘engine’
- Extended by a ConfigurationDesign profile
  - <<objective>>, <<requiredProperty>>, …

Accidental complexity

Remember req. #1 for creating DSLs…

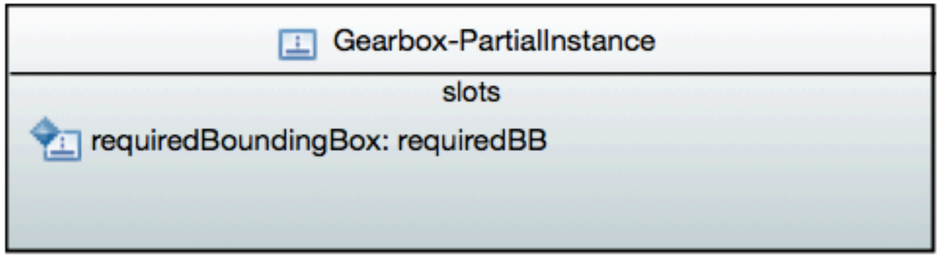
If one model conveys the same information as another model, but in a more concise way using less modeling elements and concepts, it is less complex

What could we do to further reduce the accidental complexity for modelers using the DSL4CDSofCPS?

- Brooks, F.P.: **No silver bullet: essence and accidents of software engineering.** Comput. Archive **20**(4), 10–19 (1987)
- Atkinson, C. and Kühne, T: **Reducing accidental complexity in domain models.** Softw Syst Model (2008) 7:345–359

‘Run-time instance creation’

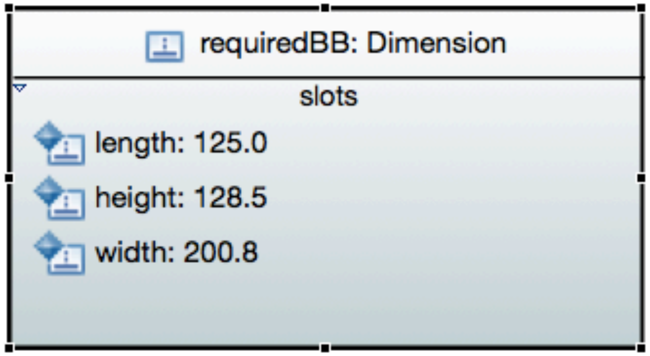
- Instances allow to
  - specify the requirements
  - specify partial architectures as ‘expert knowledge’ (*and validate them*!)
- UML _allows_ run-time instance creation
  - But papyrus doesn’t exploit it very well
  - Neither does it support graphical instances



Gearbox-PartialInstance

slots

requiredBoundingBox: requiredBB



requiredBB: Dimension

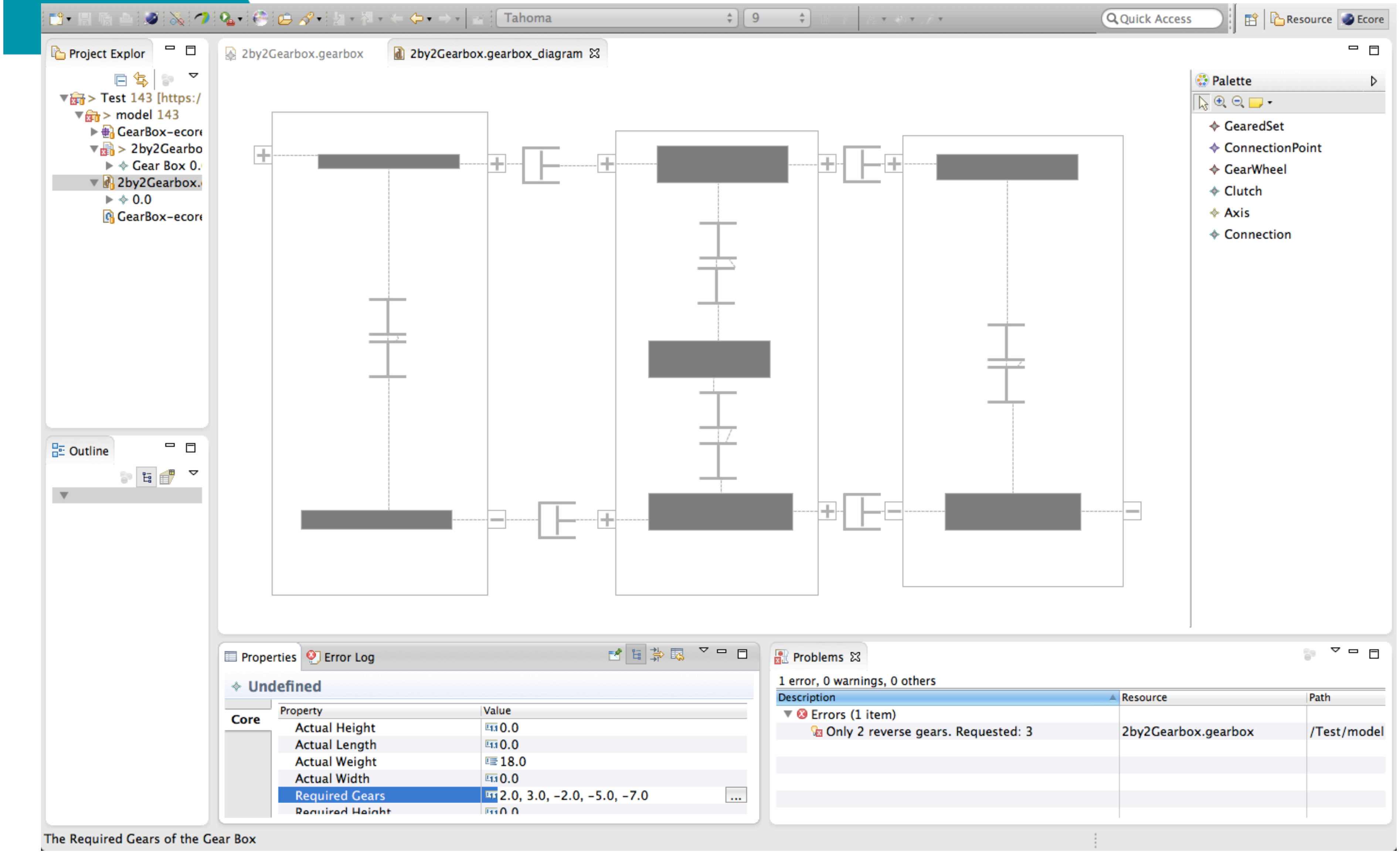
slots

length: 125.0

height: 128.5

width: 200.8

Instances with minimal accidental complexity



Undefined

Core

Property	Value
Actual Height	0.0
Actual Length	0.0
Actual Weight	18.0
Actual Width	0.0
Required Gears	2.0, 3.0, -2.0, -5.0, -7.0
Required Height	0.0

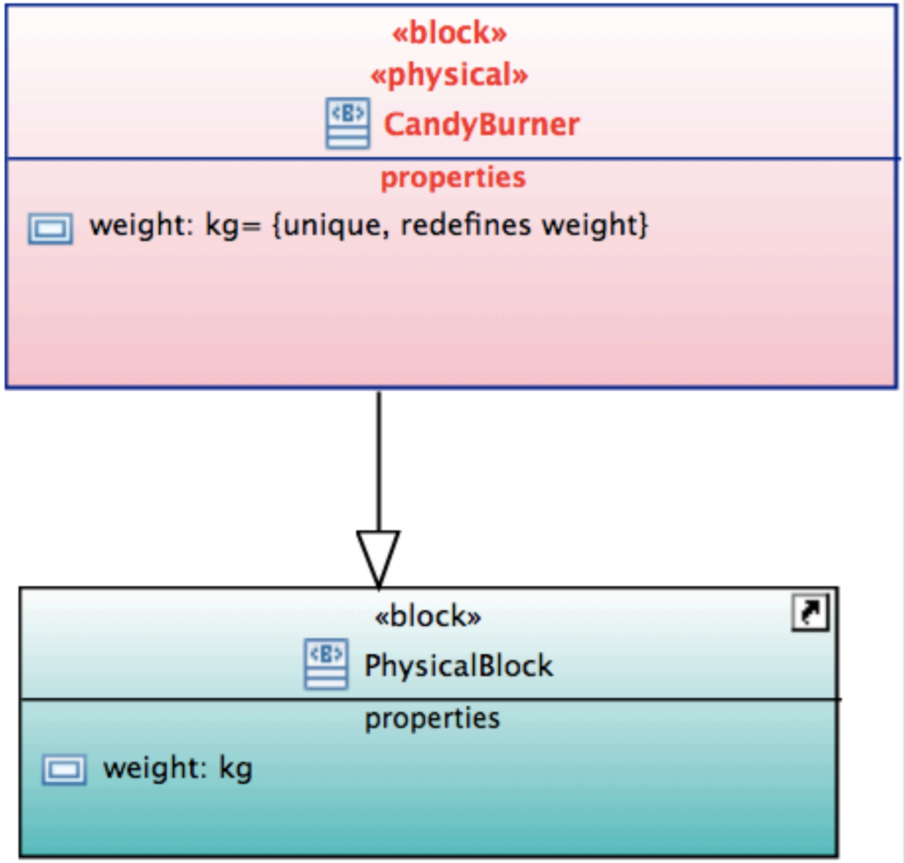
1 error, 0 warnings, 0 others

Description	Resource	Path
Errors (1 item)		
Only 2 reverse gears. Requested: 3	2by2Gearbox.gearbox	/Test/model

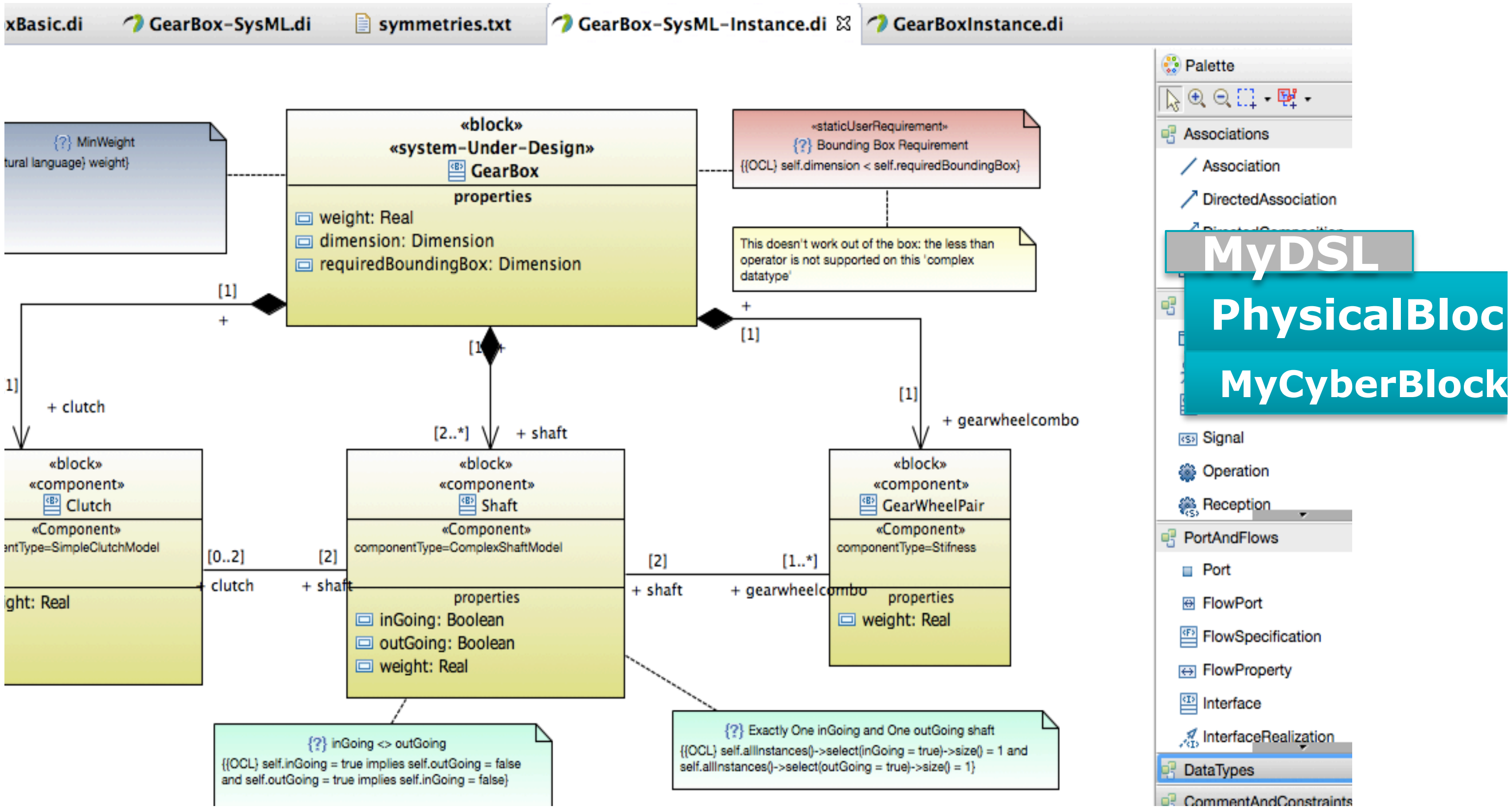
The Required Gears of the Gear Box

Sometimes, a profile is not enough to create a good DSML

Support for easily extending *model libraries*



Better support for model libraries (2)



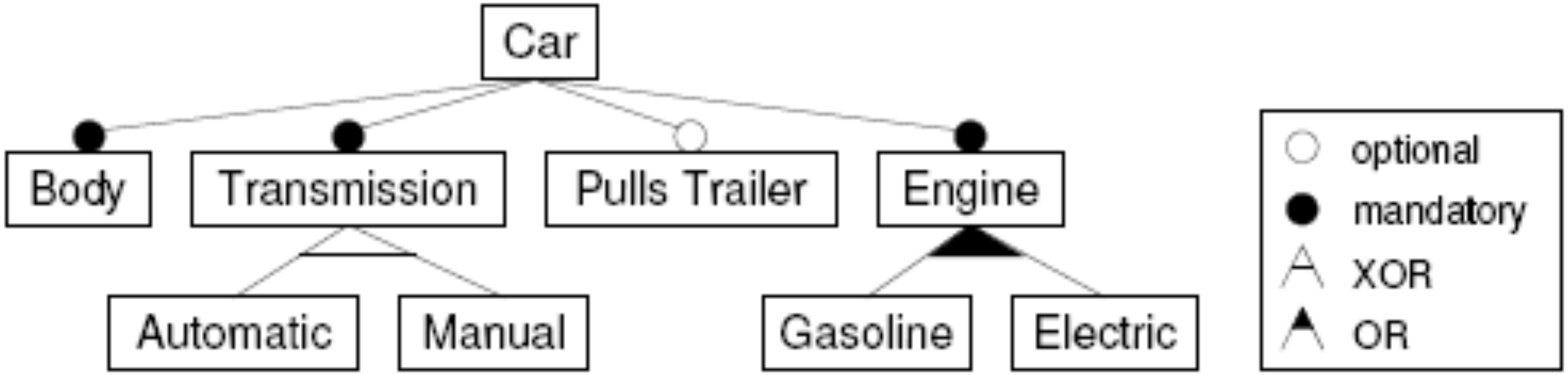
{{OCL} self.dimension < self.requiredBoundingBox}

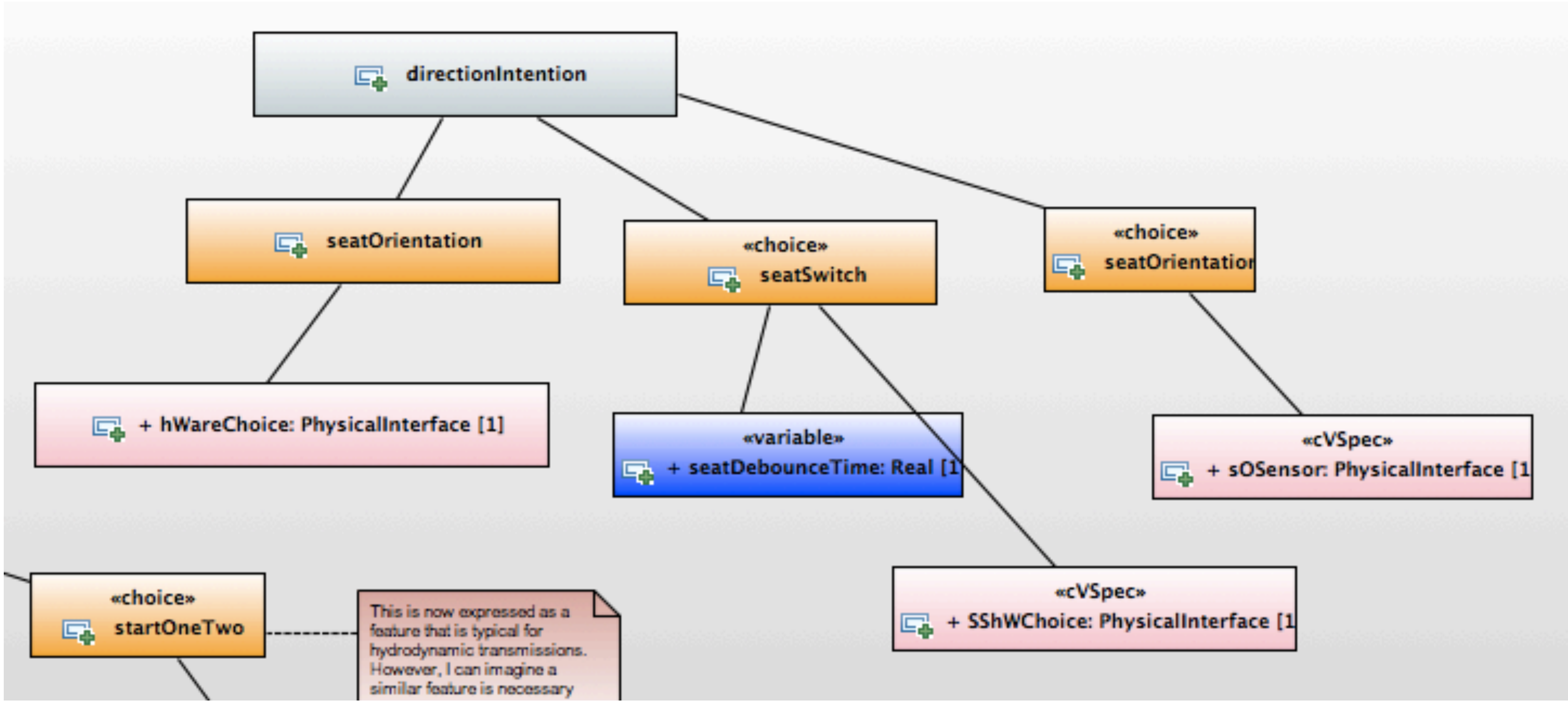
This doesn't work out of the box: the less than operator is not supported on this 'complex datatype'

{{OCL} self.inGoing = true implies self.outGoing = false and self.outGoing = true implies self.inGoing = false}

{{OCL} self.allInstances()->select(inGoing = true)->size() = 1 and self.allInstances()->select(outGoing = true)->size() = 1}

Hey, where's the Cyber?




directionIntention
seatOrientation
«choice»
seatSwitch
«choice»
seatOrientatio
+ hWareChoice: PhysicalInterface [1]
«variable»
+ seatDebounceTime: Real [1
«cVSpec»
+ sOSensor: PhysicalInterface [1
«choice»
startOneTwo
«cVSpec»
+ SShWChoice: PhysicalInterface [1

This is now expressed as a feature that is typical for hydrodynamic transmissions. However, I can imagine a similar feature is necessary

CDS4CPS is not the only DSL

- “Cross”-tables
  - Example: SysML allocation Matrices
- Auto-layouting capabilities
  - and in a second step: generated views
- Support for derived attributes
  - in stereotypes
  - in model instances
- Navigation: Show all diagrams in which a semantic element is used
- Filtering the read-only property views to avoid exposing the user to the UML meta-model

Deploying profile based DSLs

- Using heavyweight approaches (such as Sirius), ‘deploying’ a DSL is quite easily. In papyrus (luna), the customization ‘code’ seems to be more fragmented and a developer needs to know quite a lot in order to deploy his/her DSL
  - css customization for a particular type of diagram
  - shipping property view customizations and palette customization
  - Model explorer extensions

Conclusions: What could papyrus do to create an even better DSL for the CDS of CPS?

- Using UML as a host language does offer interesting capabilities for creating a DSL for the CDS of CPS
- Profiles are typically not enough to create good DSL, better support for model libraries would lower the accidental complexity
- Advanced graphical customization possibilities for stereotypes and UML instance specifications
- Semantic lay-out capabilities for modeling physical systems

abstract syntax 2 concrete syntax

[Not in presentation, since not a papyrus problem as such]

- The step from meta-model towards a profile for any non-trivial domain model is still a _very_ difficult exercise
  - Finding a match on the abstract syntax level?
    - Could this be automated using some ‘pattern recognition techniques’?
  - And additionally on the concrete syntax level…