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

WHAT DOES IT TAKE?

KLAAS GADEYNE, FLANDERS MAKE
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
Actual industrial MEMBERS
What is computational design synthesis?
What do you need for CDS?

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 CDS

Requirements and objectives

**Design knowledge**

**synthesis**

- Optimize weight
- Max Bounding Box = ...
- Num Gears = ...
- Ratios = ...

**Valuable** System architecture:

- topology + key properties

MOST DSMLs CREATED TODAY
Example: A DSML for gearbox design
A DSML for CDS

Requirements and objectives

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

synthesis

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?

- «system-Under-Design»
  - «block»
    - GearBox
      - properties
        - /weight: Real
        - /dimension: Dimension
  - clutch
  - [0..2] clutch
  - [2] shaft
  - shaft
  - [2] gearwheelcombo
  - properties
    - InGoing: Boolean
    - outGoing: Boolean
    - weight: Real

- «block»
  - Clutch
    - weight: Real

- «block»
  - Shaft
    - properties
      - clutch
      - shaft
      - gearwheelcombo
      - [2..+] weight: Real
with (a lot of) OCL constraints?

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

```oclm
{?} 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

weight of the gearbox = sum of the weights of all its parts
Design space = meta-model + requirements for the instances
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?


‘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
Instances with minimal accidental complexity
Sometimes, a profile is not enough to create a good DSML

Support for easily extending model libraries
Better support for model libraries (2)
Hey, where’s the Cyber?
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...