



# Combining ontologies with DSLs

---

## Training in software services 2010

Krzysztof Miksa, Paweł Sabina, Marek Kasztelnik

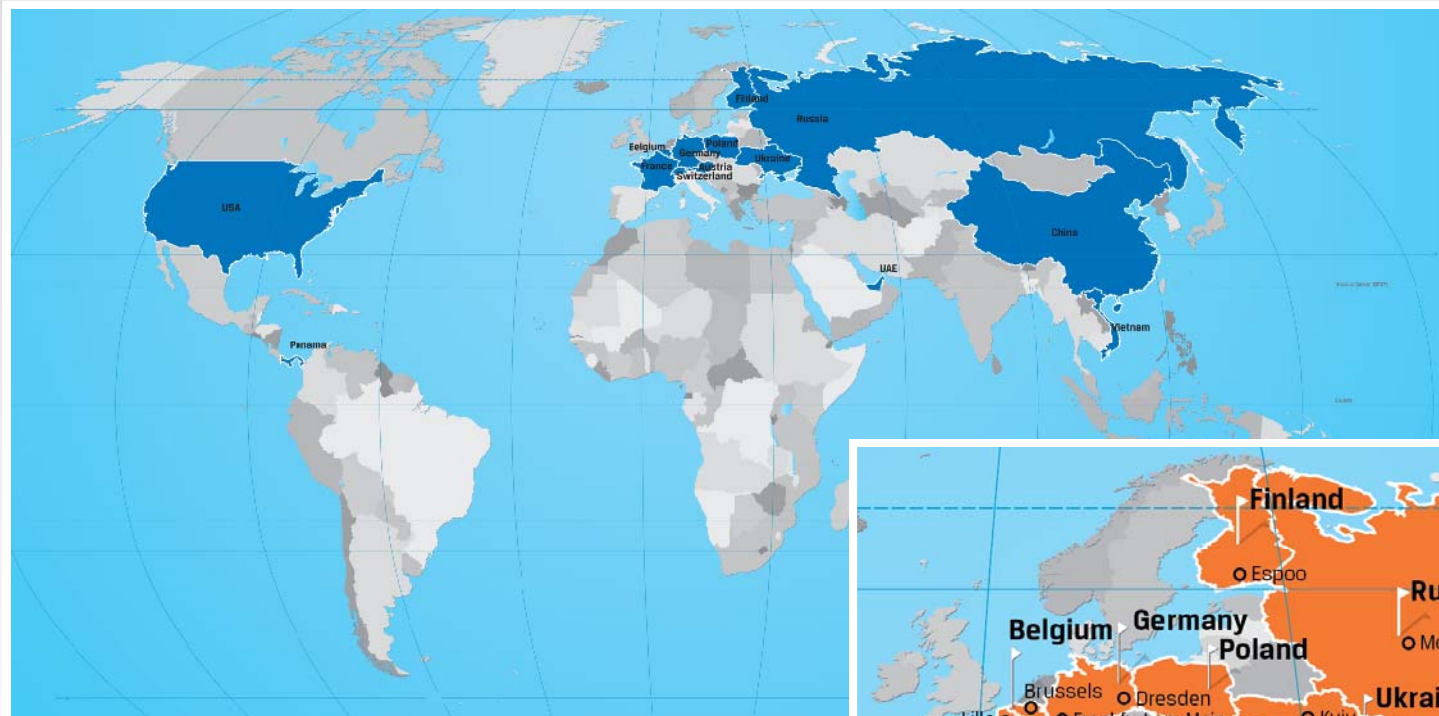
**Comarch SA**

Timișoara

10<sup>th</sup> December 2010

**MOST - Marrying Ontology and Software Technology**

# Comarch



Established in 1993  
**Headquarters** – Krakow, Poland  
**Over 3000 employees worldwide**

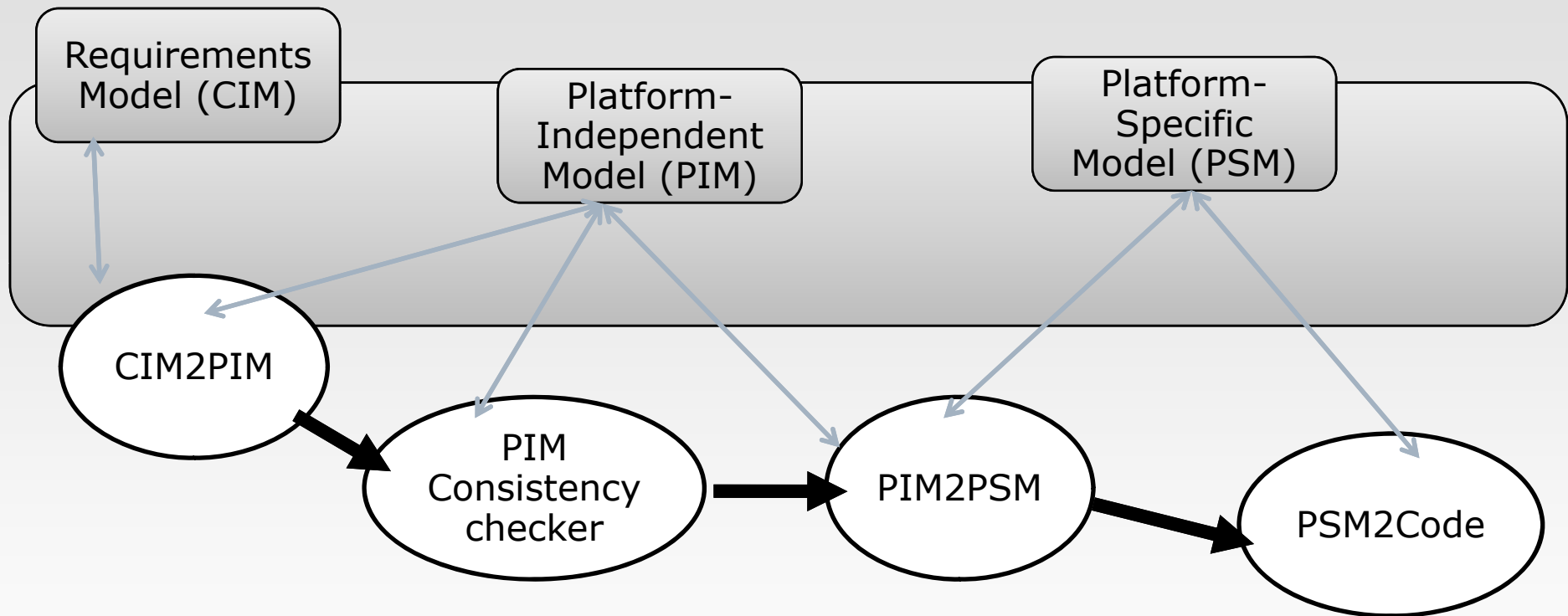
# Agenda

---

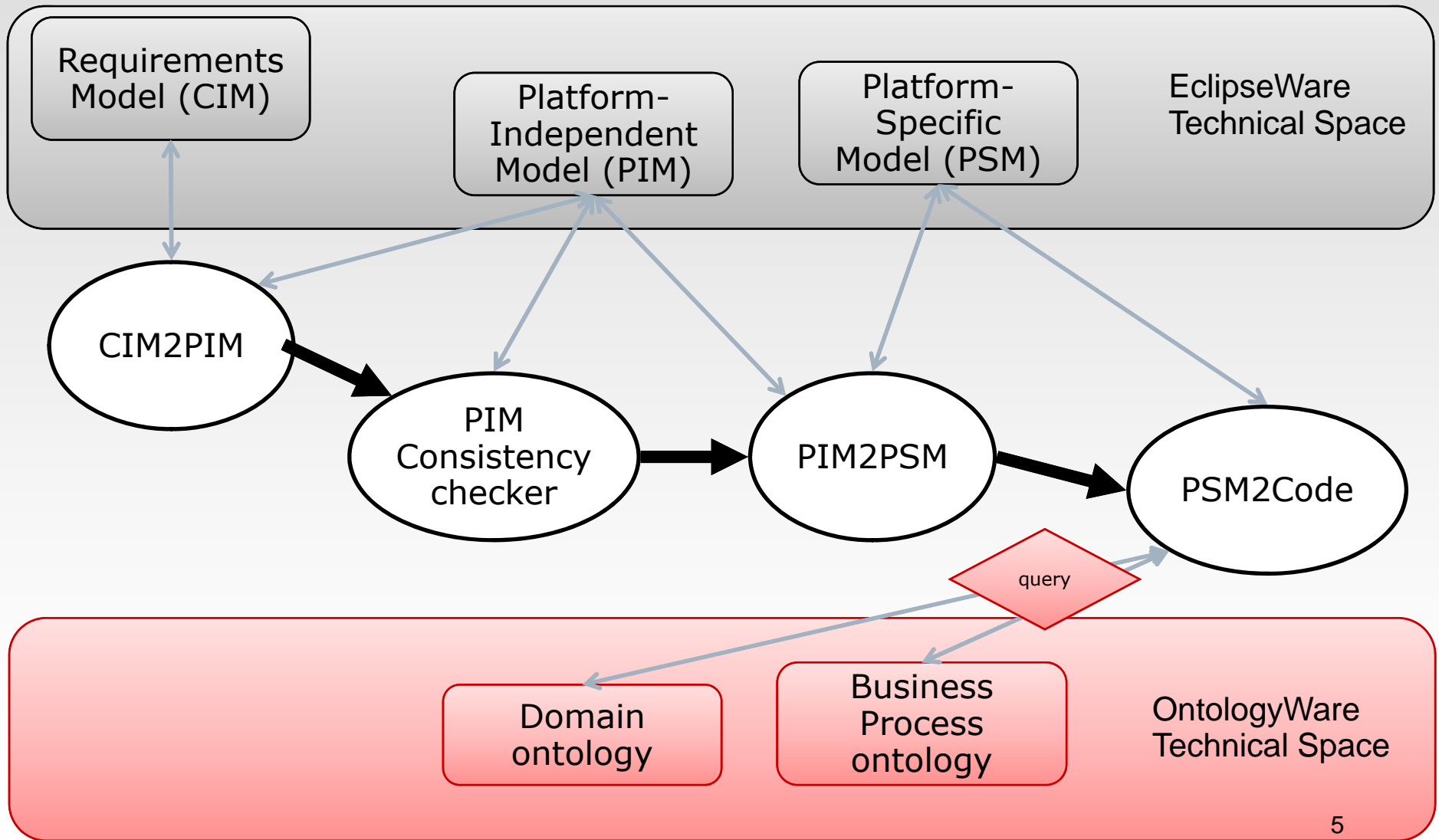


- MOST project overview**
- MDE & DSLs
- Case study
  - Motivation
  - 2D metamodelling
  - Physical Devices Ontology
  - DSL and OWL integration
  - Implementation of guidance services
- Evaluation

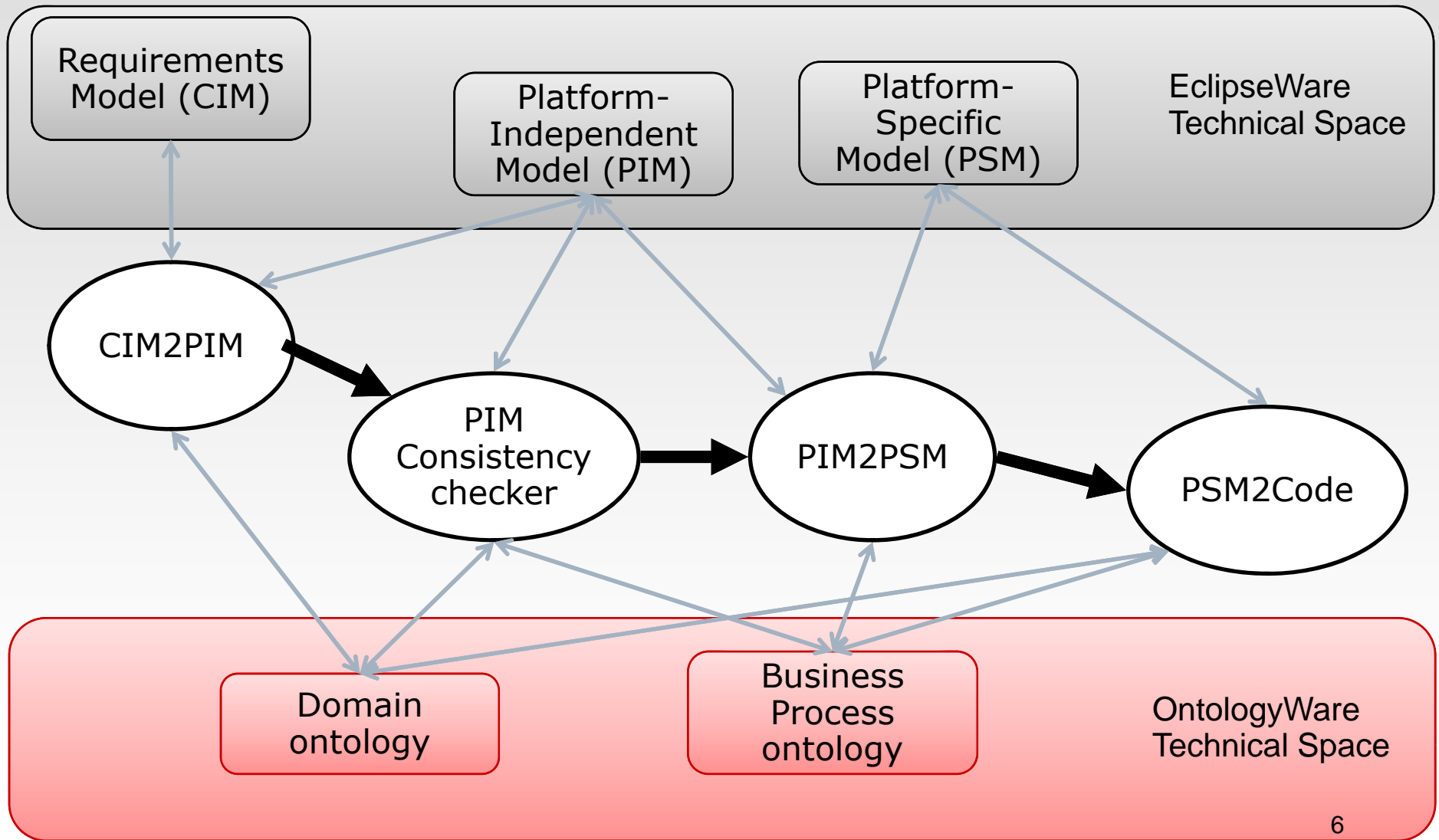
# State of the Art: Model-Driven Software Development (MDSO)



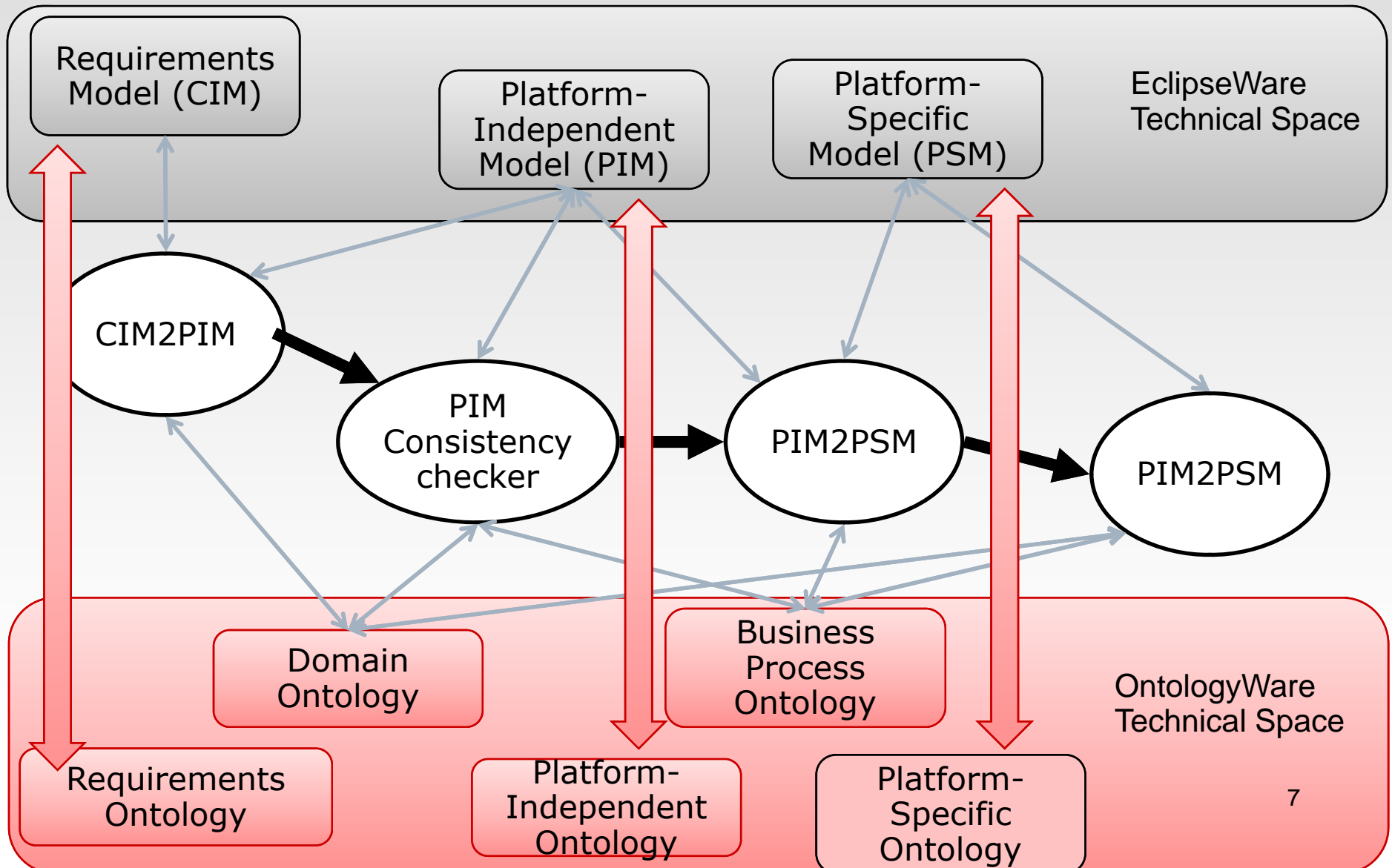
# State of the Art: Ontology Querying



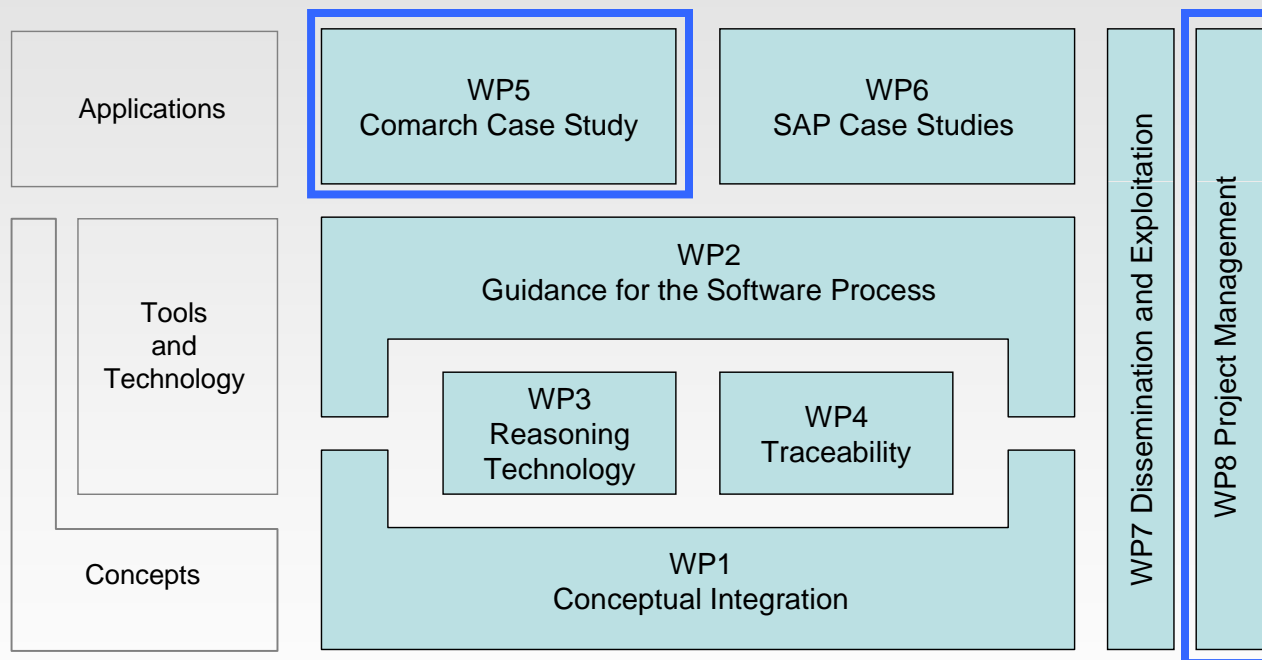
# Ontology Supported Modelling in MDSD



# How to reason on models? Bridges



# Comarch role in MOST



# Agenda

---

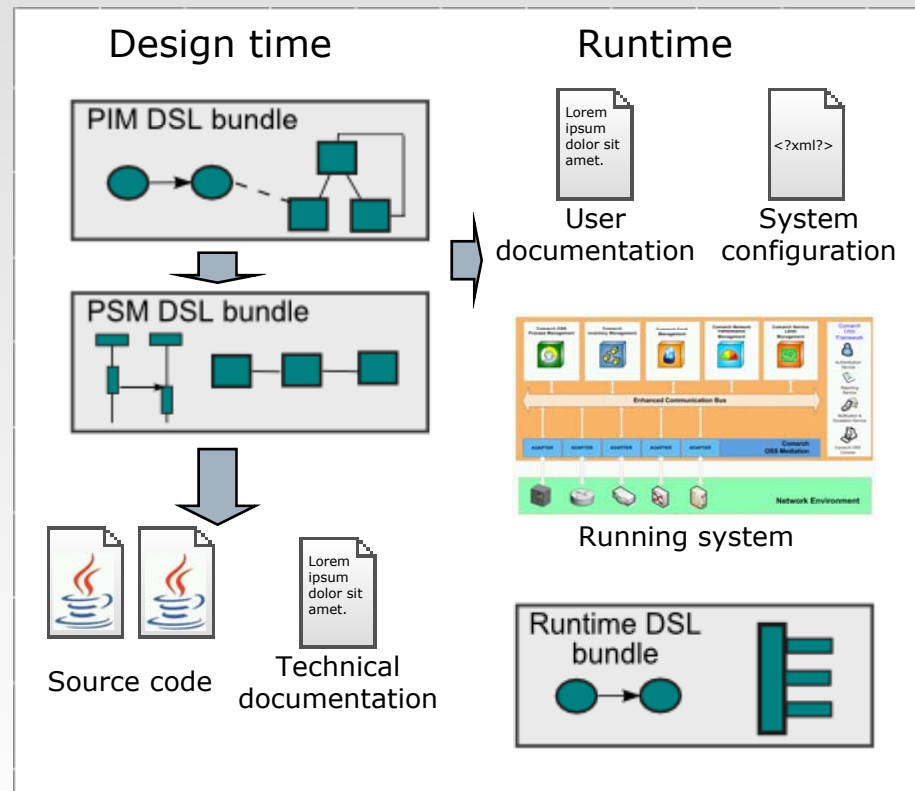


- MOST project overview
- MDE & DSLs**
- Case study
  - Motivation
  - 2D metamodelling
  - Physical Devices Ontology
  - DSL and OWL integration
  - Implementation of guidance services
- Evaluation

# Comarch OSS Model Driven Engineering



- Domain Specific Languages
  - Productivity
  - Quality
- Platform abstraction
- Eclipse Modeling Framework
- Generation of code
  - ...and other things



## Shortcoming of DSL engineering

---



- Tools interoperating with DSL should have access to its unambiguous semantics
  - Otherwise: the interpretation of DSL constructs is hidden within the tools
  - ... or left out to the user
  - Important also for interoperability of DSLs
- Solution
  - *Provide formal semantics along language definition*

# Agenda

---



- MOST project overview
- MDE & DSLs
- Case study**
  - Motivation
  - 2D metamodelling
  - Physical Devices Ontology
  - DSL and OWL integration
  - Implementation of guidance services
- Evaluation

# Problem description

---



- The domain of the case study:  
*Managing physical devices inventory within OSS system*
  - OSS = *Operations Support Systems*
    - Computer systems used by telecommunications service providers
    - Aid the network operators in their daily work
    - Abstract from telecom business (e.g. billing) as opposed to Business Support Systems
  - Inventory Management
    - Core part of OSS
    - Management of telecom assets: sites, equipment, network links
  - Physical devices management
    - Important part of Inventory Management
    - Switches, routers, splitters, etc.
-

# Why this is complicated?



- Equipment is modular
  - Devices have slots
  - Different cards can be plugged into the slots
  - Compatibility issues
  - Card constraints
- State-of-the-art in OSS
  - Easy access to extensive databases storing attributes of component types
  - No simultaneous support to the user by answering questions that involve sophisticated constraints
  - Custom solutions, hard coded constraints



# Use cases

---



- Questions:
  - Are the instances consistent with their types?
    - If not why?
  - Is the definition of the device type consistent?
  - What can I put into this slot?
    - Which cards?
    - Which types of cards?
  - Which device can I put this card in?
    - In which slot?
  - How can I repair my invalid device?
  
- *If the semantics is within the model*
  - *We can make use of semantic reasoning*
  - *These questions will remain simple*

# 2D metamodelling

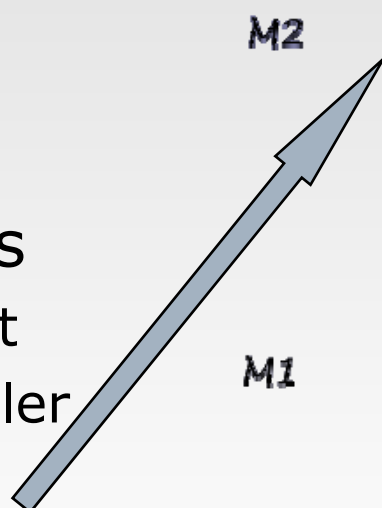


- Physical Devices modelling

- Device types
- Instances of device types

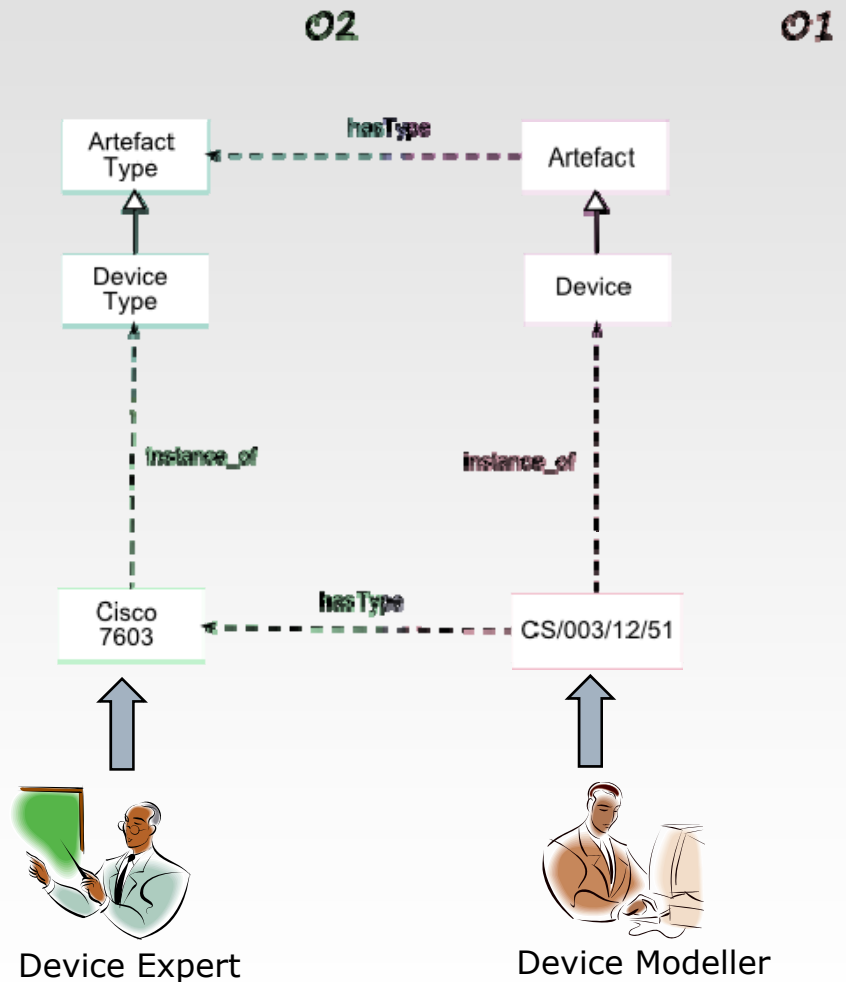
- Two user roles

- Device Expert
- Device Modeller

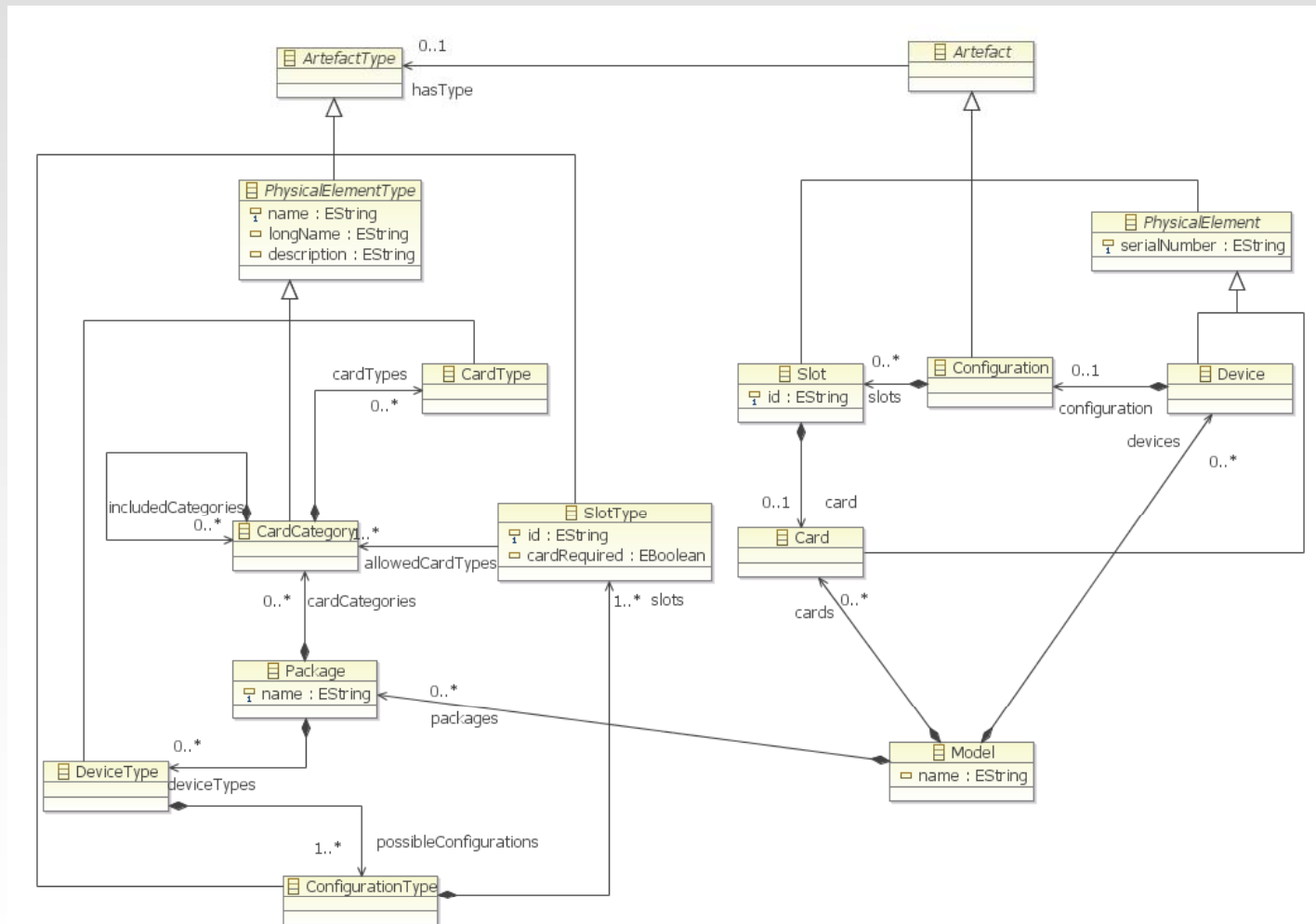


M2

M1



# PDDSL abstract syntax



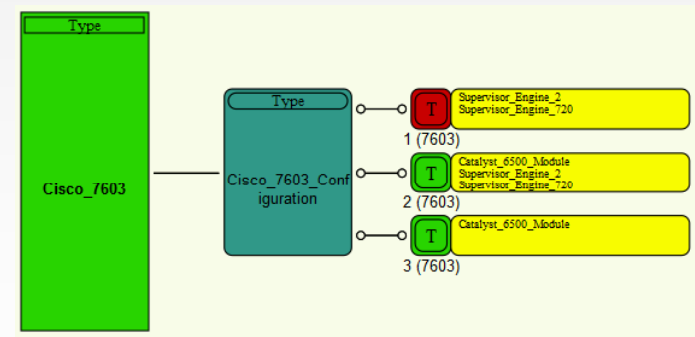
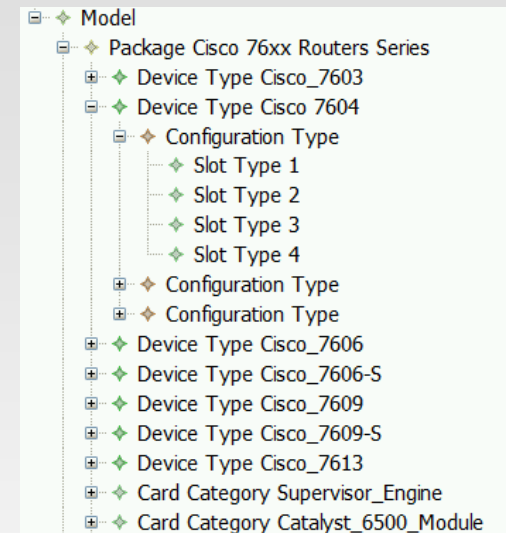
# PDDSL concrete syntax



## □ Demo

- Tree based editor
- Graphical syntax
- Textual syntax

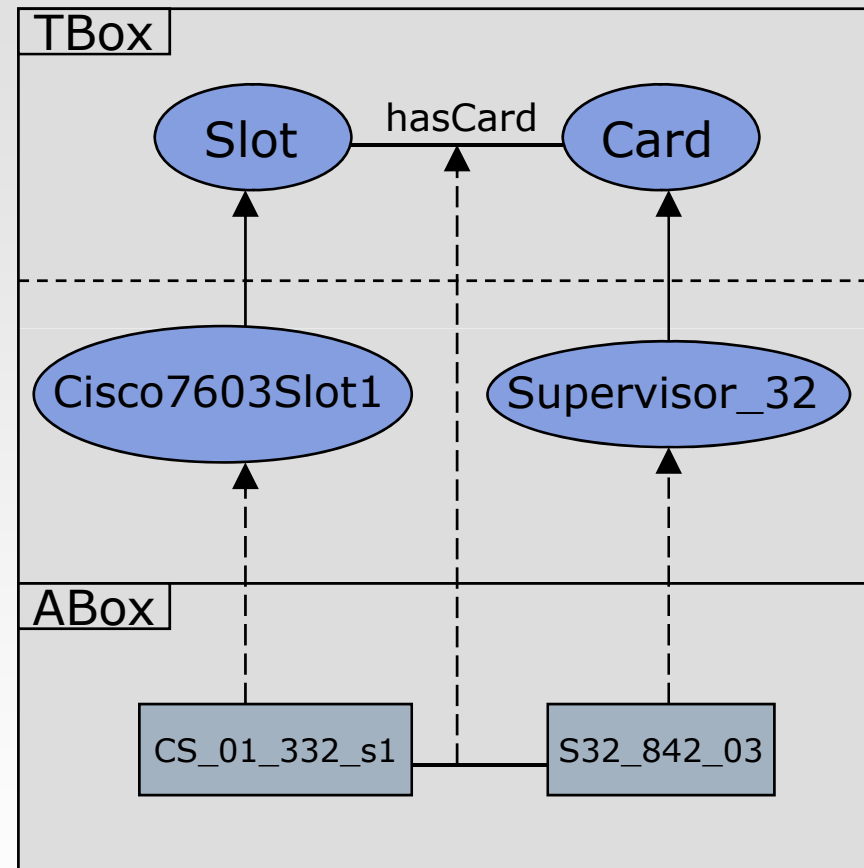
```
DeviceType "Cisco_7604" longName : "Cisco 7604 Chassis"
  allowed : {
    PossibleConfiguration "Cisco_7604_Configuration_1" {
      Slot "1" allowed : "Supervisor_Engine_32" required : true
      Slot "2" allowed : "Supervisor_Engine_32" "Catalyst_6500_Module" required : false
      Slot "3" allowed : "Catalyst_6500_Module" required : false
      Slot "4" allowed : "Catalyst_6500_Module" required : false
    }
    PossibleConfiguration "Cisco_7604_Configuration_2" {
      Slot "1" allowed : "Supervisor_Engine_720" required : true
      Slot "2" allowed : "Catalyst_6500_Module" "Supervisor_Engine_720" required : false
      Slot "3" allowed : "Catalyst_6500_Module" required : false
      Slot "4" allowed : "Catalyst_6500_Module" required : false
    }
    PossibleConfiguration "Cisco_7604_Configuration_3" {
      Slot "1" allowed : "Route_Switch_Processor_720" required : true
      Slot "2" allowed : "Catalyst_6500_Module" "Route_Switch_Processor_720" required : false
      Slot "3" allowed : "Catalyst_6500_Module" required : false
      Slot "4" allowed : "Catalyst_6500_Module" required : false
    }
  }
}
```



# PD Ontology



- TBox
  - Core concepts
    - Classes: Device, Slot, Card, ...
    - Properties: hasCard, hasSlot, ...
  - Device types
    - Cisco 7603, ...
- ABox
  - Instances
- Domain Closure
  - Depending on the use case
- Unique Names Assumption



# PD Ontology



## □ Demo

```
Class: Cisco7603Configuration
  EquivalentTo: Configuration and
  # cardinality restriction on slots:
  hasSlot exactly 3 Slot and
  # required cards restriction:
  (hasSlot some (hasCard some Supervisors and id value 1)) and
  #optional card restriction:
  (hasSlot only (((hasCard some Supervisors and id value 1)) or
    ((hasCard some Supervisors and id value 2) or
      (hasCard some Hot_Swappable_OSM and id value 2)) or
    ((hasCard some Hot_Swappable_OSM and id value 3) or
      (hasCard some SPA_interface_processors and id value 3))))))
```

```
Namespace: pd <http://www.comarch.com/oss/pd.owl#>
Ontology: <http://www.comarch.com/oss/pd-ext.owl>
```

```
Class: pd:Cisco7603Configuration
  SubClassOf:
    ((pd:containsCard some pd:Hot_Swappable_OSM)
    and (pd:containsCard some pd:Supervisor_engine_720))
    or (pd:containsCard only (not (pd:Hot_Swappable_OSM)))
```

# Integrating DLS and OWL



- Goal
  - Take profit from the **advantages of ontologies**, but...
  - still use the **productive DSL**
- Solution: integrate DSL and OWL
  - within modelling environment
- Integration process

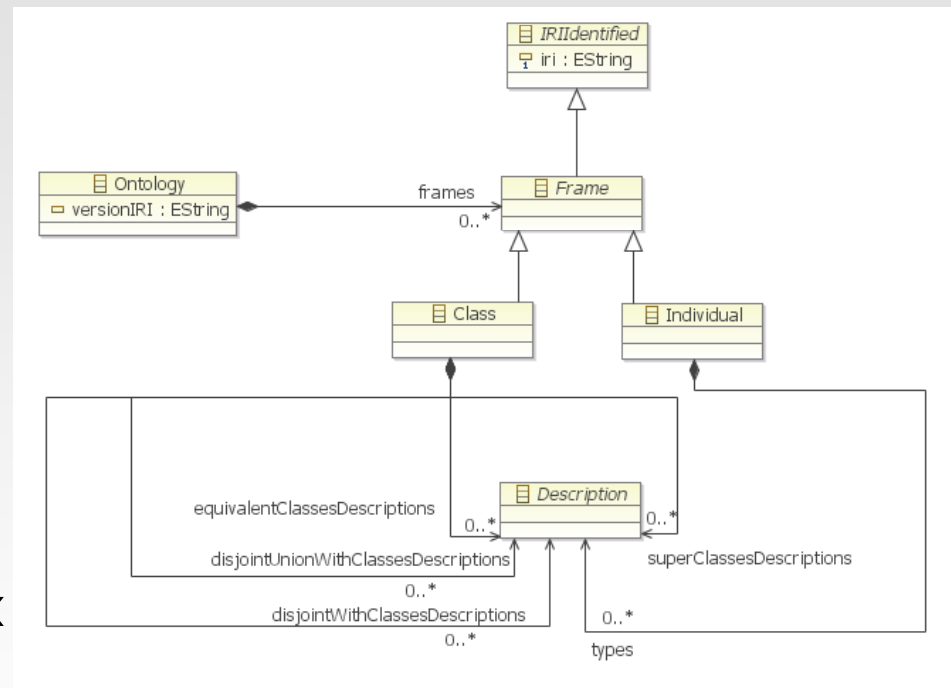


# Represent OWL2 in the metamodelling technical space



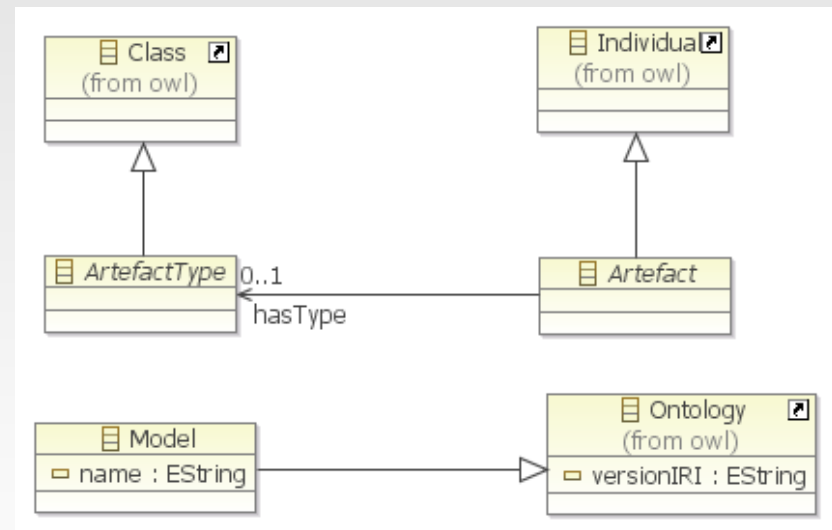
## □ OWL2 Manchester Syntax

- Easy to use by the software developers
- Used in Protégé
- Already available from *emftext.org*
- Metamodel
- Concrete syntax
- Concrete syntax can be directly parsed by OWL API



## Integrate the abstract syntaxes

- ❑ PDDSL ArtefactType *is an* OWL Class
- ❑ PDDSL Artefact *is an* OWL Individual
- ❑ PDDSL Model *is an* OWL Ontology



# Integrate the concrete syntaxes



```
CardCategory ::= !"CardCategory" iri[","]
(
  (annotations !1)
  | ("SubClassOf:" superClassDescriptions
    ("," superClassDescriptions)* !1)
  | ("EquivalentTo:" equivalentClassesDescriptions
    ("," equivalentClassesDescriptions)* !1)
  | ("DisjointWith:" disjointWithClassesDescriptions
    ("," disjointWithClassesDescriptions)* !1)
  | ("DisjointUnionOf:" disjointUnionWithClassesDescriptions
    ("," disjointUnionWithClassesDescriptions)* !1)
)*
("longName" ":" longName[","])? ("description" ":" description[","])?
(!"cardTypes" ":" "{" cardTypes* !0"}")?
(!"includedCategories" ":" "{" includedCategories* !0"}")? ;
```

# Integrated Language



- PhysicalDevice DSL (PDDSL)
  - Structural modelling
- OWL2-DL
  - Semantic constraints which reference PDDSL elements

---

```
DeviceType "Cisco_7603"  
  SubClassOf: pd_hasConfiguration some ( pd_hasSlot some ( pd_hasCard some Cisco_7600_SIP ) )  
  longName : "CISCO 7603 CHASSIS" description : "The Cisco® 7603 Router is a high-performance ..." allowed : {  
  PossibleConfiguration "Cisco_7603_Configuration" {  
    Slot "1" allowed : "Supervisor_Engine_2" "Supervisor_Engine_720" required : true  
    Slot "2" allowed : "Supervisor_Engine_2" "Supervisor_Engine_720" "Catalyst_6500_Module"  
    required : false  
    Slot "3" allowed : "Catalyst_6500_Module" required : false
```

---

```
Device serialNumber : "cisco_7603" hasType : "Cisco_7603"  
configuration :  
{  
  Slot id : "1" : Card serialNumber : "supervisor_2_5" hasType : "WS-X6K-S2U-MSFC2"  
  Slot id : "2" :  
  Slot id : "3" :  
}
```

# Concrete syntaxes

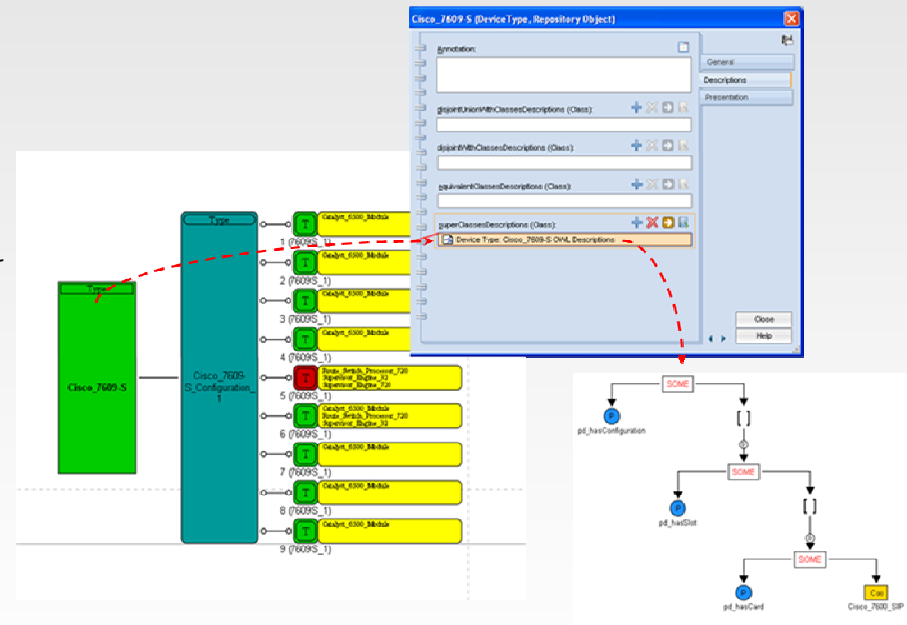


- Demo
  - Integrated textual syntax
  - Integrated graphical syntax

```
SubClassOf: pd_hasConfiguration some
  ( pd_hasSlot some
    ( pd_hasCard some Cisco_7600_SIP ) )
```

```
longName : "CISCO 7609-S CHASSIS", allowed : {
  PossibleConfiguration "Cisco_7609-S_Configuration_1" {
    Slot "1" allowed : "Catalyst_6500_Module"
    required : false
    Slot "2" allowed : "Catalyst_6500_Module"
    required : false
```

```
...
}
}
```



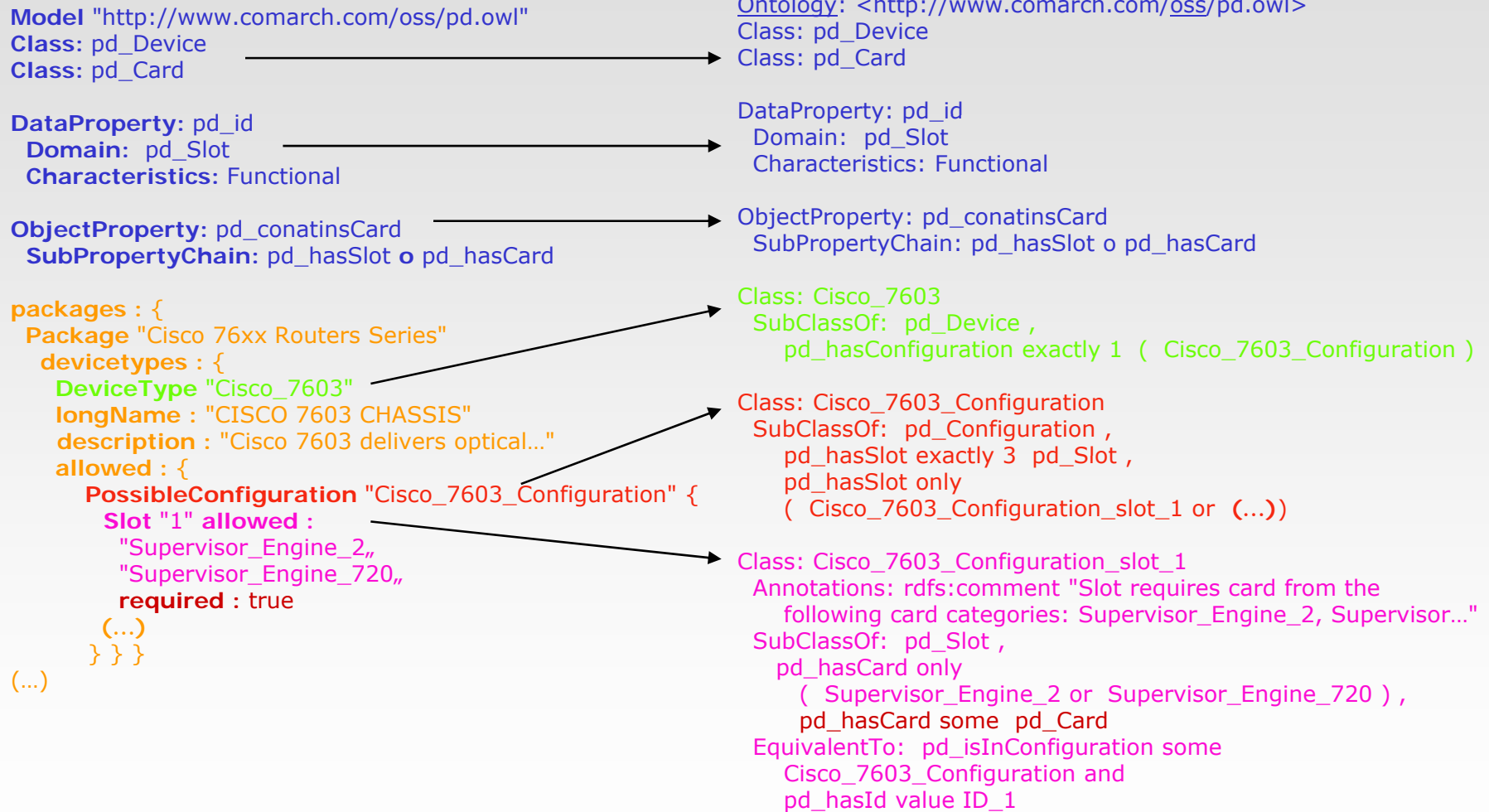
# Projection to OWL2



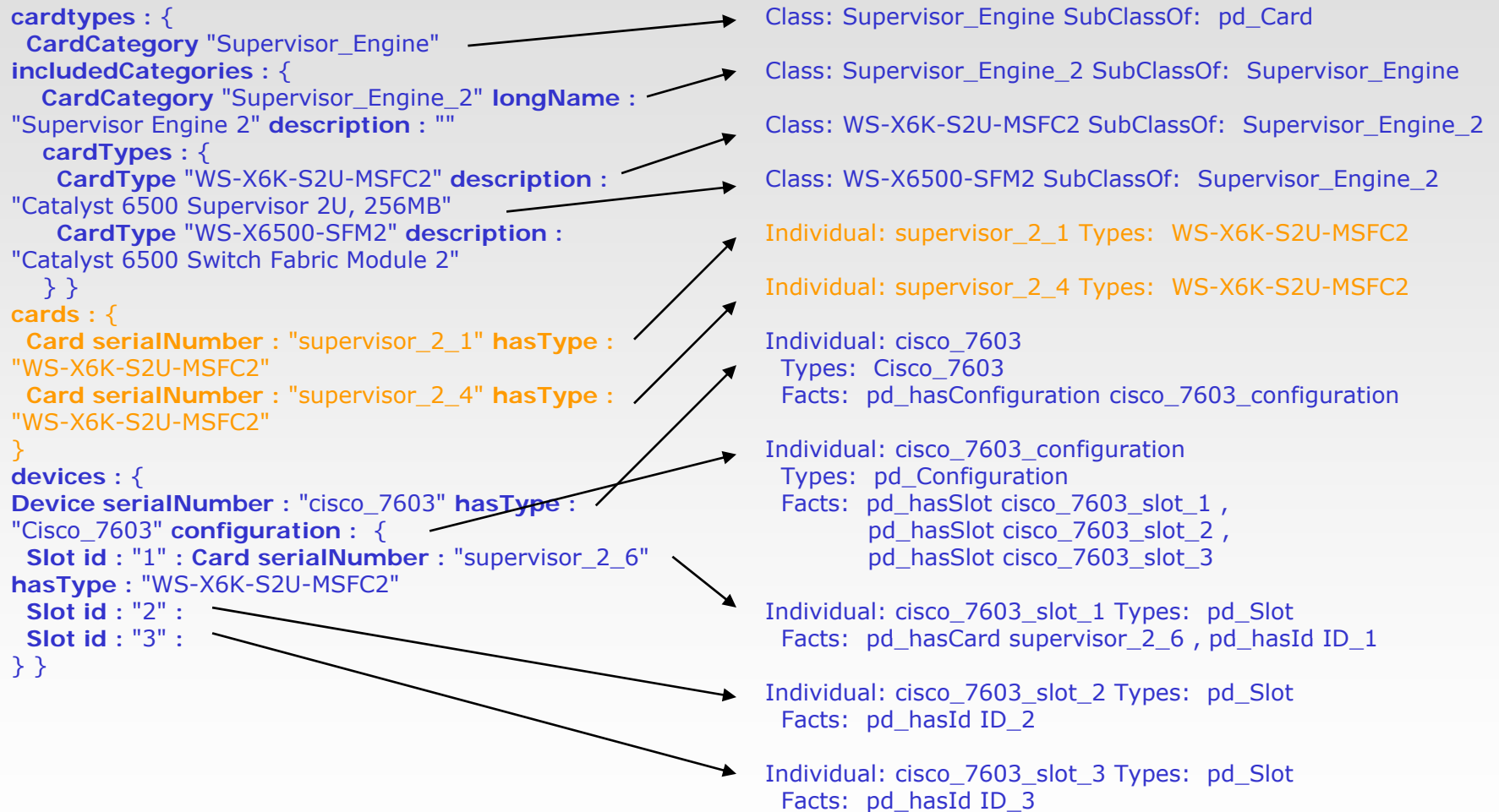
- Goal: define semantics of PDDSL constructs
- E.g. what is the meaning of *allowedCardTypes* association
- Informally: only the cards listed in *allowedCardTypes* can be connected into the slot
- Implemented as QVT Operational transformation
  - Model-to-model
- Another goal
  - Transform custom containment tree into flat frame representation
  - Map identifiers

```
mapping inout SlotType::updateSlotType() {  
  
    // Define a specific Slot subclass containing  
    // only allowed CardTypes, like in example:  
    //  
    // pd.hasCard only (Supervisor_Engine_2  
    //                   or Supervisor_Engine_720)  
    self.superClassesDescriptions += object ObjectPropertyOnly {  
        featureReference := object FeatureReference {  
            feature := hasCardProperty;  
        };  
        primary := object NestedDescription {  
            description := object Disjunction {  
                self.allowedCardTypes -> forEach (i) {  
                    conjunctions += object ClassAtomic {  
                        clazz := i;  
                    }  
                }  
            }  
        }  
    };  
};  
  
// Move all OWL::Class objects  
model.frames += model.allSubobjects() [OWL::Class] -> sortBy(iri);
```

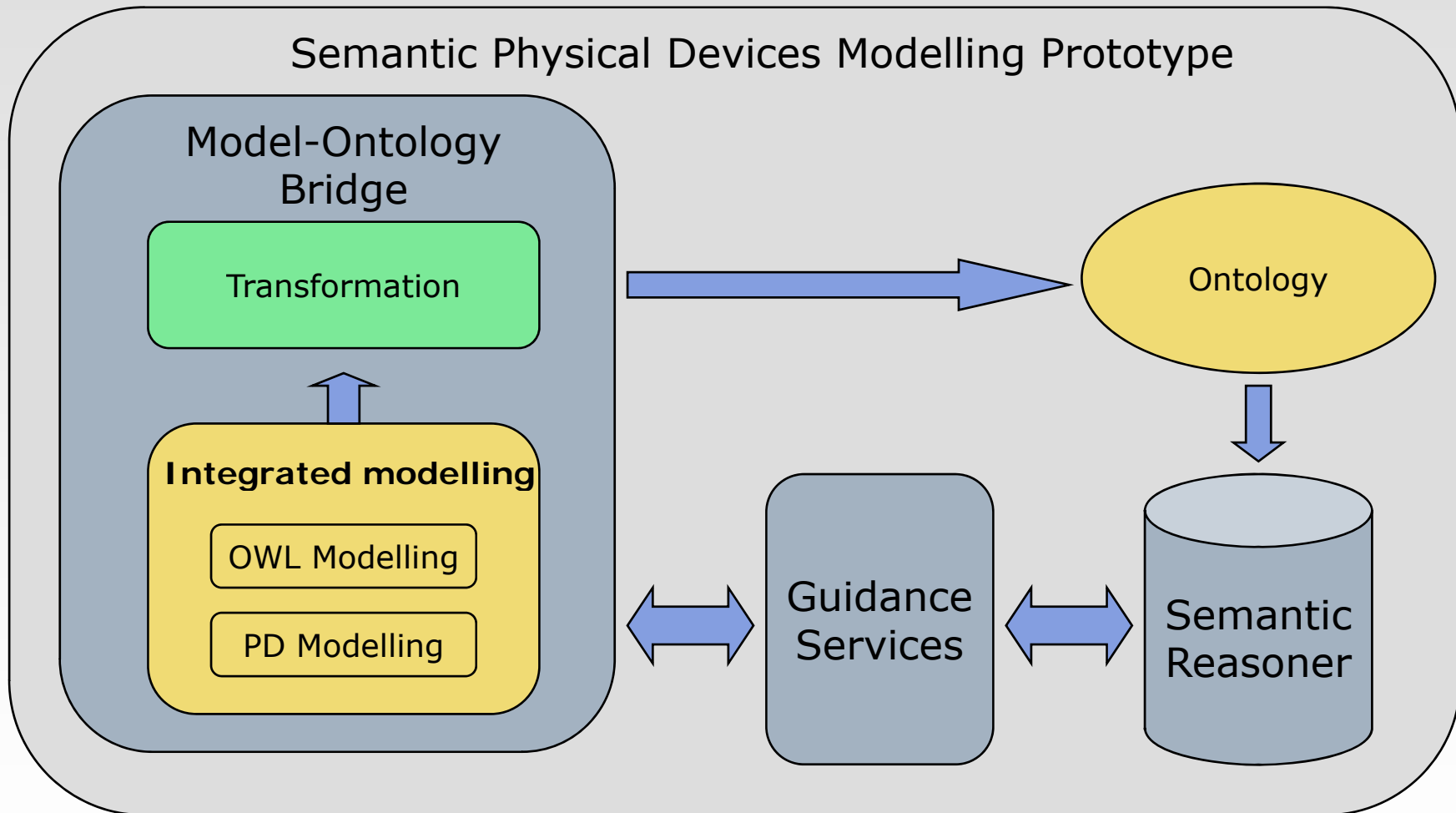
# Transformation example



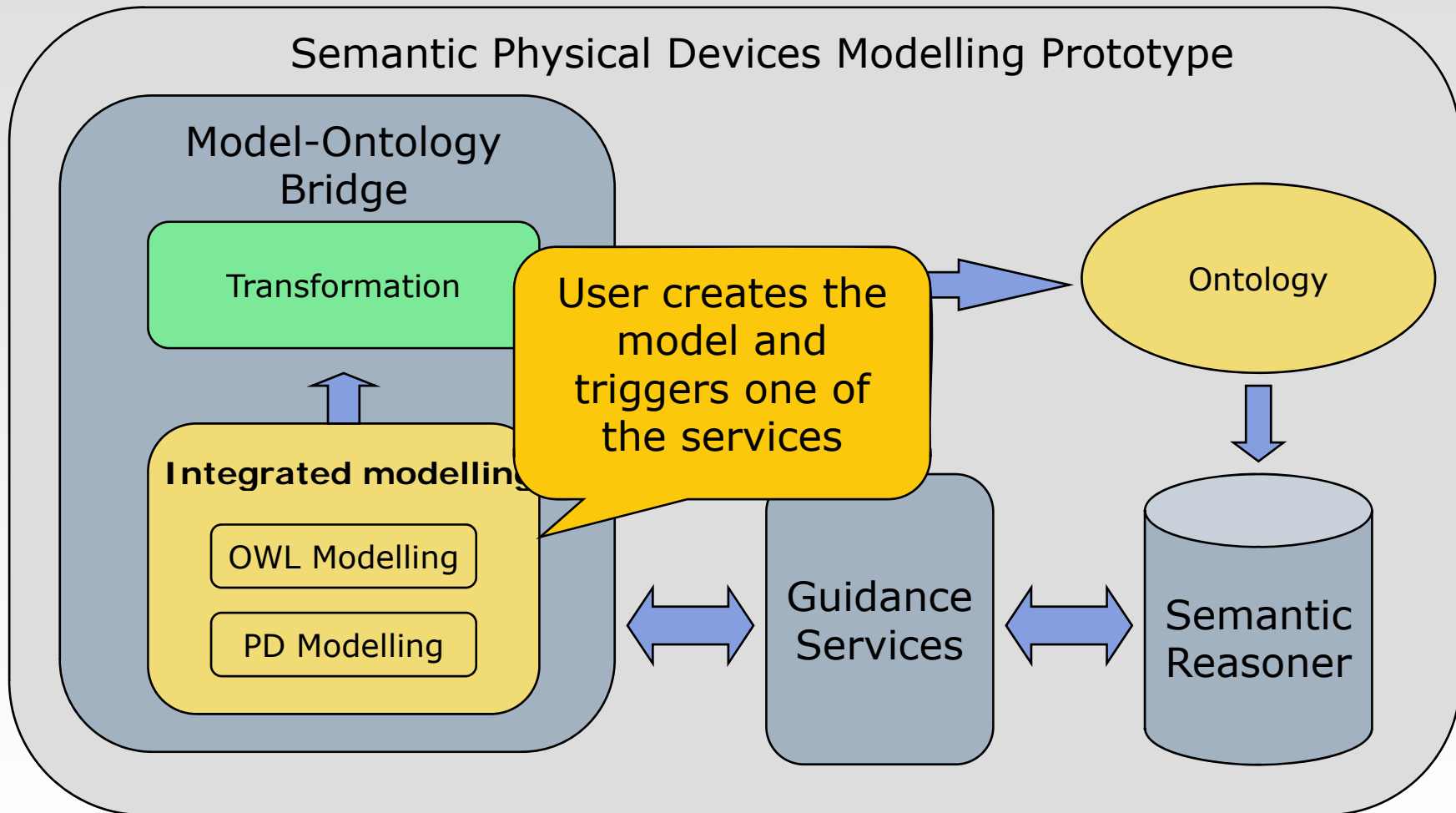
# Transformation example cont.



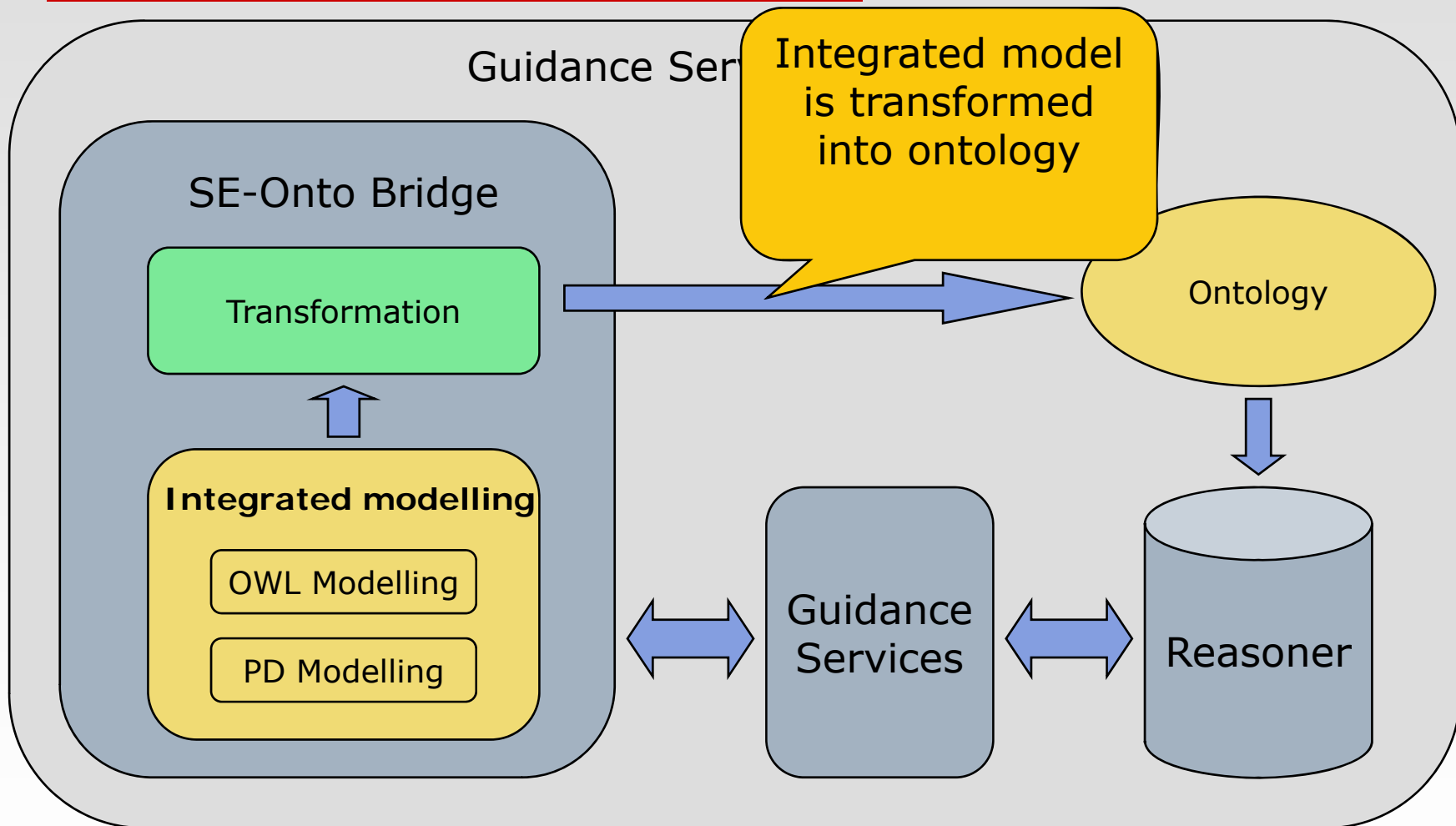
# Integrated modelling environment with consistency guidance



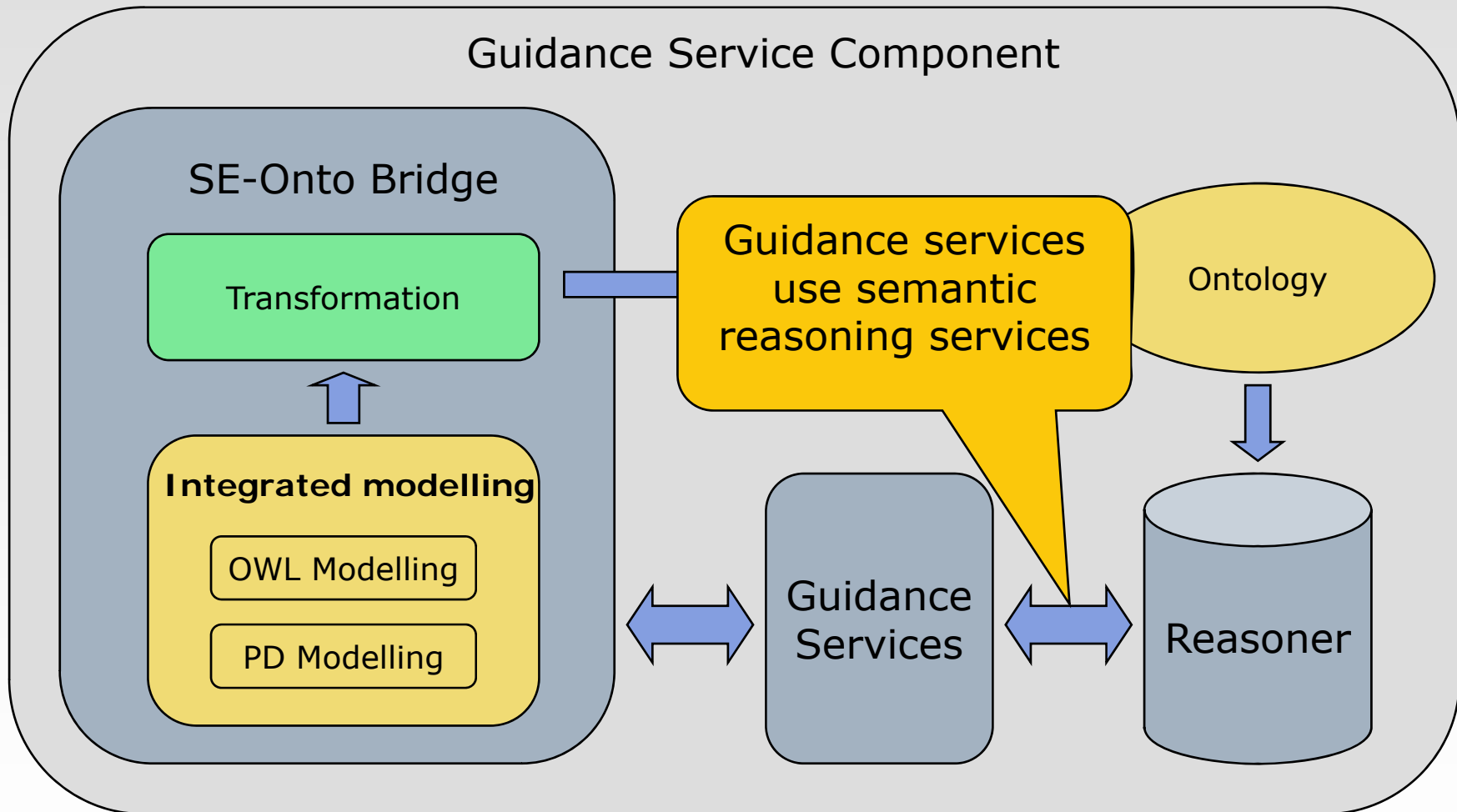
# Integrated modelling environment with consistency guidance



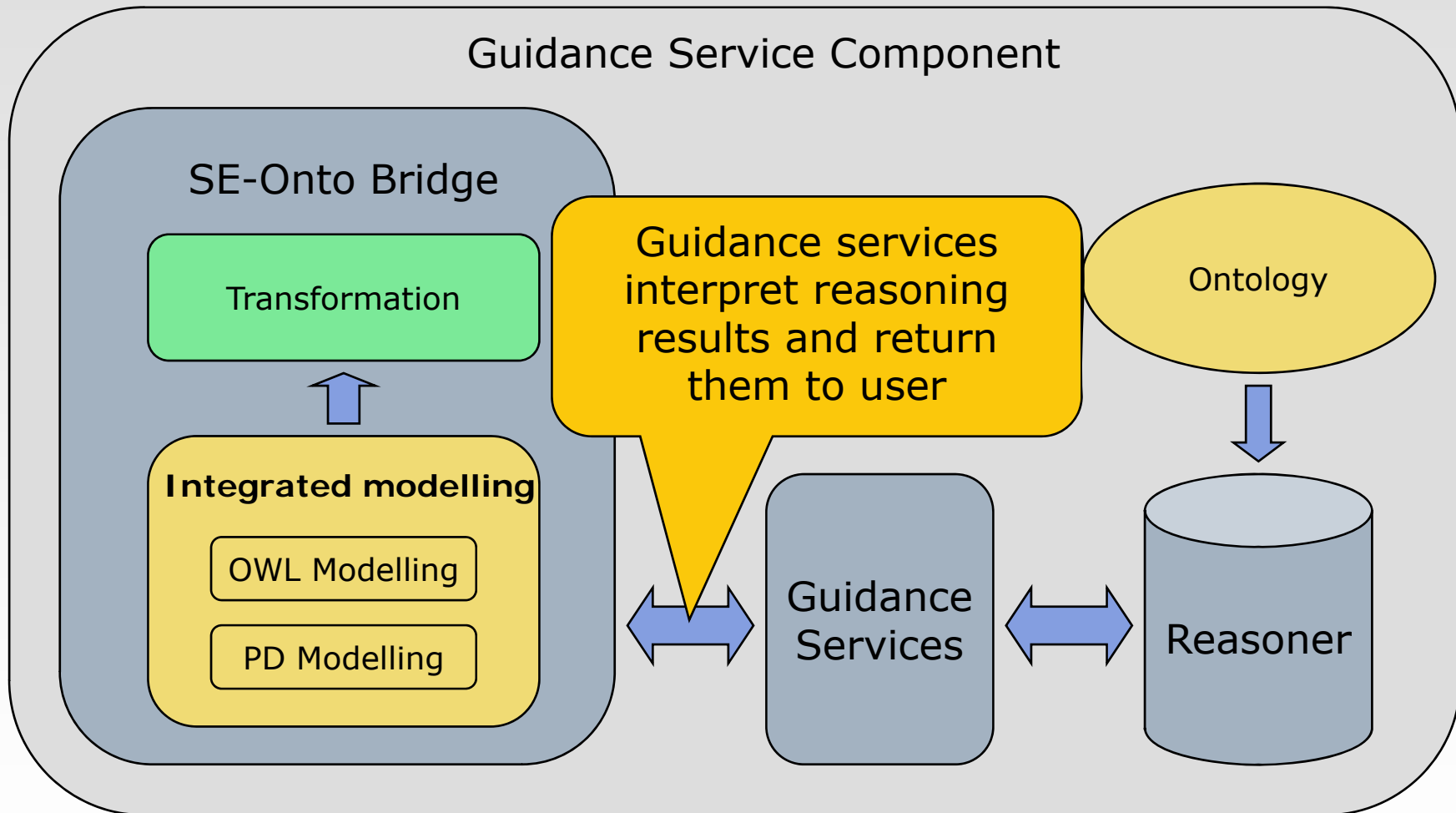
# Integrated modelling environment with consistency guidance



# Integrated modelling environment with consistency guidance



# Integrated modelling environment with consistency guidance



## Use cases

---



- ❑ **UC-1:** Detect errors in physical device type definition
- ❑ **UC-2:** Find wrongly configured instances of devices and explain errors
- ❑ **UC-3:** Suggest card categories which are allowed in a slot

# Implementation of the use cases

---



Use case	Description	Reasoning services
UC-1	Detect errors in physical device type definition	Satisfiability checking
UC-2	Find wrongly configured instances of devices and explain errors	Consistency checking, Explanations
UC-3	Suggest card categories which are allowed in a slot	Subsumption reasoning

# UC-1: user model

---



DeviceType "Cisco\_7603"

longName : "CISCO 7603 CHASSIS" description : "Cisco 7603 delivers optical LAN, WAN..."

allowed : {

    PossibleConfiguration "Cisco\_7603\_Configuration" {

        Slot "1" allowed : "Supervisor\_Engine\_2" "Supervisor\_Engine\_720" required : true

        Slot "2" allowed : "Supervisor\_Engine\_2" "Supervisor\_Engine\_720" "Catalyst\_6500\_Module"  
            required : false

        Slot "3" allowed : "Catalyst\_6500\_Module" required : false

    }

}

# UC-1: error

---



DeviceType "Cisco\_7603"

SubClassOf: pd:hasConfiguration some (pd:hasSlot some (pd:hasCard some Cisco\_7600\_SIP))

longName : "CISCO 7603 CHASSIS" description : "Cisco 7603 delivers optical LAN, WAN..."

allowed : {

    PossibleConfiguration "Cisco\_7603\_Configuration" {

        Slot "1" allowed : "Supervisor\_Engine\_2" "Supervisor\_Engine\_720" required : true

        Slot "2" allowed : "Supervisor\_Engine\_2" "Supervisor\_Engine\_720" "Catalyst\_6500\_Module"  
            required : false

        Slot "3" allowed : "Catalyst\_6500\_Module" required : false

    }

}

# UC-1: implementation



The screenshot displays two panels from a software application. The left panel, titled 'Inferred class hierarchy: Cisco\_7603', shows a tree structure starting with 'Thing' at the root. Under 'Thing', there are 'Nothing' and 'Cisco\_7603'. 'Cisco\_7603' is further divided into 'pd\_Card', 'pd\_Configuration', 'pd\_Device', and 'pd\_Slot'. The right panel, titled 'Description: Cisco\_7603', shows the details for this class. It lists 'Equivalent classes' as 'Nothing'. Under 'Superclasses', it lists 'pd\_Device', 'pd\_hasConfiguration some (pd\_hasSlot some (pd\_hasCard some Cisco\_7600\_SIP))', and 'pd\_hasConfiguration exactly 1 Cisco\_7603\_Configuration'. Each class entry in the right panel has a set of control icons (a circle with '@', a circle with 'x', and a circle with a dot).

1. Find unsatisfiable classes in PD Ontology
2. Find corresponding set of *Artefacts* in PDDSL model
3. Report the *Artefacts* to the user

# UC-2: user model



```
DeviceType "Cisco_7603"  
  longName : "CISCO 7603 CHASSIS" description : "Cisco 7603 delivers optical LAN, WAN..."  
  allowed : {  
    PossibleConfiguration "Cisco_7603_Configuration" {  
      Slot "1" allowed : "Supervisor_Engine_2" "Supervisor_Engine_720" required : true  
      Slot "2" allowed : "Supervisor_Engine_2" "Supervisor_Engine_720" "Catalyst_6500_Module",  
        required : false  
      Slot "3" allowed : "Catalyst_6500_Module" required : false  
    }  
  }  
  
Device serialNumber : "cisco_7603-INCORRECT" hasType : "Cisco_7603"  
configuration :  
{  
  Slot id : "1" :  
  Slot id : "2" :  
  Slot id : "3" :  
}  
  
CardCategory "Catalyst_6500_Module" includedCategories : {  
  CardCategory "Cisco_7600_Ethernet_Card_Line"  
  longName : "Cisco 7600 Series Ethernet Services Plus 20- and 40-Gbps Line Cards,"  
  cardTypes : {  
    CardType "7600-ES-2TG3C"  
    CardType "7600-ES-2TG3CXL,"  
  }  
}
```

# UC-2: error



```
DeviceType "Cisco_7603"  
  longName : "CISCO 7603 CHASSIS" description : "Cisco 7603 delivers optical LAN, WAN..."  
  allowed : {  
    PossibleConfiguration "Cisco_7603_Configuration" {  
      Slot "1" allowed : "Supervisor_Engine_2" "Supervisor_Engine_720" required : true  
      Slot "2" allowed : "Supervisor_Engine_2" "Supervisor_Engine_720" "Catalyst_6500_Module",  
        required : false  
      Slot "3" allowed : "Catalyst_6500_Module" required : false  
    }  
  }  
  
Device serialNumber : "cisco_7603-INCORRECT" hasType : "Cisco_7603"  
configuration :  
{  
  Slot id : "1" : Card serialNumber : "ethernet_card" hasType : "7600-ES-2TG3C"  
  Slot id : "2" :  
  Slot id : "3" :  
}  
  
CardCategory "Catalyst_6500_Module" includedCategories : {  
  CardCategory "Cisco_7600_Ethernet_Card_Line"  
  longName : "Cisco 7600 Series Ethernet Services Plus 20- and 40-Gbps Line Cards,,  
  cardTypes : {  
    CardType "7600-ES-2TG3C"  
    CardType "7600-ES-2TG3CXL"  
  }  
}
```

## UC-2: MIPS - Inconsistency explanation



---

```
7600-ES-2TG3C subClassOf Cisco_7600_Ethernet_Card_Line
Supervisor_Engine_2 subClassOf Supervisor_Engine
cisco_7603-INCORRECT type Cisco_7603
cisco_7603-INCORRECT_slot_1 pd_id 1
Cisco_7600_Ethernet_Card_Line subClassOf Catalyst_6500_Module
cisco_7603-INCORRECT pd_hasConfiguration cisco_7603-INCORRECT_configuration
Supervisor_Engine_720 subClassOf Supervisor_Engine
ethernet_card type 7600-ES-2TG3C
Cisco_7603_Configuration_slot_1 subClassOf pd_hasCard only
    Supervisor_Engine_2 or Supervisor_Engine_720
Functional pd_hasConfiguration
Cisco_7603 subClassOf pd_hasConfiguration exactly 1Cisco_7603_Configuration
cisco_7603-INCORRECT_slot_1 pd_hasCard ethernet_card
DisjointClasses(Catalyst_6500_Module
    Cisco_7600_SIP
    Supervisor_Engine)
pd_hasSlot inverseOf pd_isInConfiguration
Cisco_7603_Configuration_slot_1 equivalentTo pd_id value 1
    and pd_isInConfiguration some Cisco_7603_Configuration
cisco_7603-INCORRECT_configuration pd_hasSlot cisco_7603-INCORRECT_slot_1
```

## UC-2: User-friendly explanations

---



- Inconsistent individuals:
  - cisco\_7603-INCORRECT
  - cisco\_7603-INCORRECT\_configuration
  - cisco\_7603-INCORRECT\_slot\_1
- Textual explanation from axiom annotations:
  - "Slot requires card from the following card categories: Supervisor\_Engine\_2, Supervisor\_Engine\_720"
  - Annotation of Cisco\_7603\_slot\_1

Class: Cisco\_7603\_Configuration\_slot\_1

**Annotations: rdfs:comment "Slot requires card from the following card categories: Supervisor\_Engine\_2, Supervisor\_Engine\_720"**

SubClassOf: pd\_Slot ,

pd\_hasCard only

( Supervisor\_Engine\_2 or Supervisor\_Engine\_720 ) ,

pd\_hasCard some pd\_Card

# Device validation explanation interpretation



---

7600-ES-2TG3C subClassOf Cisco\_7600\_Ethernet\_Card\_Line  
Supervisor\_Engine\_2 subClassOf Supervisor\_Engine  
cisco\_7603-INCORRECT type Cisco\_7603  
cisco\_7603-INCORRECT\_slot\_1 pd\_id 1  
Cisco\_7600\_Ethernet\_Card\_Line subClassOf Catalyst\_6500\_Module  
cisco\_7603-INCORRECT pd\_hasConfiguration cisco\_7603-INCORRECT\_configuration  
Supervisor\_Engine\_720 subClassOf Supervisor\_Engine  
ethernet\_card type 7600-ES-2TG3C  
**Cisco\_7603\_Configuration\_slot\_1 subClassOf pd\_hasCard only  
Supervisor\_Engine\_2 or Supervisor\_Engine\_720**  
Functional pd\_hasConfiguration  
Cisco\_7603 subClassOf pd\_hasConfiguration exactly 1Cisco\_7603\_Configuration  
cisco\_7603-INCORRECT\_slot\_1 pd\_hasCard ethernet\_card  
DisjointClasses(Catalyst\_6500\_Module  
Cisco\_7600\_SIP  
Supervisor\_Engine)  
pd\_hasSlot inverseOf pd\_isInConfiguration  
Cisco\_7603\_Configuration\_slot\_1 equivalentTo pd\_id value 1  
and pd\_isInConfiguration some Cisco\_7603\_Configuration  
cisco\_7603-INCORRECT\_configuration pd\_hasSlot cisco\_7603-INCORRECT\_slot\_1

# UC-3: user model



```
DeviceType "Cisco_7603"  
  longName : "CISCO 7603 CHASSIS" description : "Cisco 7603 delivers optical LAN, WAN..."  
  allowed : {  
    PossibleConfiguration "Cisco_7603_Configuration" {  
      Slot "1" allowed : "Supervisor_Engine_2" "Supervisor_Engine_720" required : true  
      Slot "2" allowed : "Supervisor_Engine_2" "Supervisor_Engine_720" "Catalyst_6500_Module",  
        required : false  
      Slot "3" allowed : "Catalyst_6500_Module" required : false  
    }  
  }  
  
Device serialNumber : "cisco_7603-INCORRECT" hasType : "Cisco_7603"  
configuration :  
{  
  Slot id : "1" : Card serialNumber : "ethernet_card" hasType : "7600-ES-2TG3C"  
  Slot id : "2" :  
  Slot id : "3" :  
}  
  
CardCategory "Catalyst_6500_Module" includedCategories : {  
  CardCategory "Cisco_7600_Ethernet_Card_Line"  
  longName : "Cisco 7600 Series Ethernet Services Plus 20- and 40-Gbps Line Cards,"  
  cardTypes : {  
    CardType "7600-ES-2TG3C"  
    CardType "7600-ES-2TG3CXL"  
  }  
}
```

## UC-3: Computing suggestions

---



### □ Idea

1. What cannot be inserted to a slot given the current knowledge base?
2. Compute the complement
  - 1. is an OWA question
  - 2. requires local CWA
  - Answer both in terms of classes and individuals

# UC-3: Computing suggestions - example



- Subclasses
  - *CardCategories* which are not allowed in the slot
  
- Instances
  - *Cards* which are not allowed in the slot

Query (class expression)

```
pd_Card and not (inv(pd_hasCard) value cisco_7603_slot_1)
```

Execute    Add to ontology

---

Query results

Sub classes (4)		<input type="checkbox"/> Super classes
<input checked="" type="radio"/> Catalyst_6500_Module		<input type="checkbox"/> Ancestor classes
<input checked="" type="radio"/> Cisco_7600_SIP		<input type="checkbox"/> Equivalent classes
<input checked="" type="radio"/> Route_Switch_Processor_720		<input checked="" type="checkbox"/> Subclasses
<input checked="" type="radio"/> Supervisor_Engine_32		<input type="checkbox"/> Descendant classes
Instances (6)		<input checked="" type="checkbox"/> Individuals
<input checked="" type="checkbox"/> card_c2_1		
<input checked="" type="checkbox"/> card_c2_2		
<input checked="" type="checkbox"/> card_es20_2		
<input checked="" type="checkbox"/> card_es20_1		
<input checked="" type="checkbox"/> card_es20_3		
<input checked="" type="checkbox"/> supervisor_32_1		

# UC-3: Computing suggestions - complement



- Easier to perform in the modelling technical space
- OCL:  
*Model ->*  
*allCardCategories() ->*  
*excluding(disallowed)*
- In OWL, requires CWA

Query (class expression)

```
pd_Card and not {card_c2_1, card_c2_2, card_es20_1, card_es20_2, card_es20_3, supervisor_32_1}
```

Execute Add to ontology

Query results

Sub classes (3)

- ☰ Cisco\_7600\_SIP
- Supervisor\_Engine\_2
- Supervisor\_Engine\_720

Instances (7)

- ◆ supervisor\_2\_6
- ◆ supervisor\_720\_3
- ◆ supervisor\_2\_3
- ◆ supervisor\_2\_1
- ◆ supervisor\_720\_2
- ◆ supervisor\_720\_1
- ◆ supervisor\_2\_4

Super classes  
 Ancestor classes  
 Equivalent classes  
 Subclasses  
 Descendant classes  
 Individuals

## UC-3: other considerations



- Cards allowed in a slot, given the current configuration
  - `pd_Card and not (inv(pd_hasCard) value cisco_7603_slot_1)`
- Cards allowed in a slot, in general
  - `pd_Card and not (inv(pd_hasCard) some Cisco_7603_Configuration_slot_1 )`
- Cards allowed in a device, given the current configuration
  - `pd_Card and not (inv(pd_hasCard) some (inv(pd_hasSlot) some (inv(pd_hasConfiguration) cisco_7603)))`
- Cards allowed in a device, in general
  - `pd_Card and not (inv(pd_hasCard) some (inv(pd_hasSlot) some (inv(pd_hasConfiguration) Cisco7603)))`
- Which slots I can put a given car in
  - `pd_Slot and not (pd_hasCard value HS_OSM_1)`
- Which card types are compatible with a given card type
  - `pd_Card and not (inv(pd_hasCard) some (inv(pd_hasSlot) some (pd_hasSlot some (pd_hasCard some (Supervisor_32))))`
- What are the possible types of a device
  - `Device and not {cisco_11}`
- ...

# Agenda

---



- MOST project overview
- MDE & DSLs
- Case study
  - Motivation
  - 2D metamodelling
  - Physical Devices Ontology
  - DSL and OWL integration
  - Implementation of guidance services
- Evaluation**

# Initial findings: positives

---



- Creation of DSL tools is easier
  - Purely declarative approach
  - Various questions can be asked
  - No need to hard code semantics in tools
- Support for two-dimensional metamodelling
  - SE metamodelling contributes support for *linguistic instantiation*
  - Ontologies contribute support for *ontological (domain) instantiation*
  - Needed in other areas: service modelling, connectivity modelling
- Simple but expressive DSL
  - PDDSL part contributes simplicity, productivity
  - OWL part contributes rich expressiveness

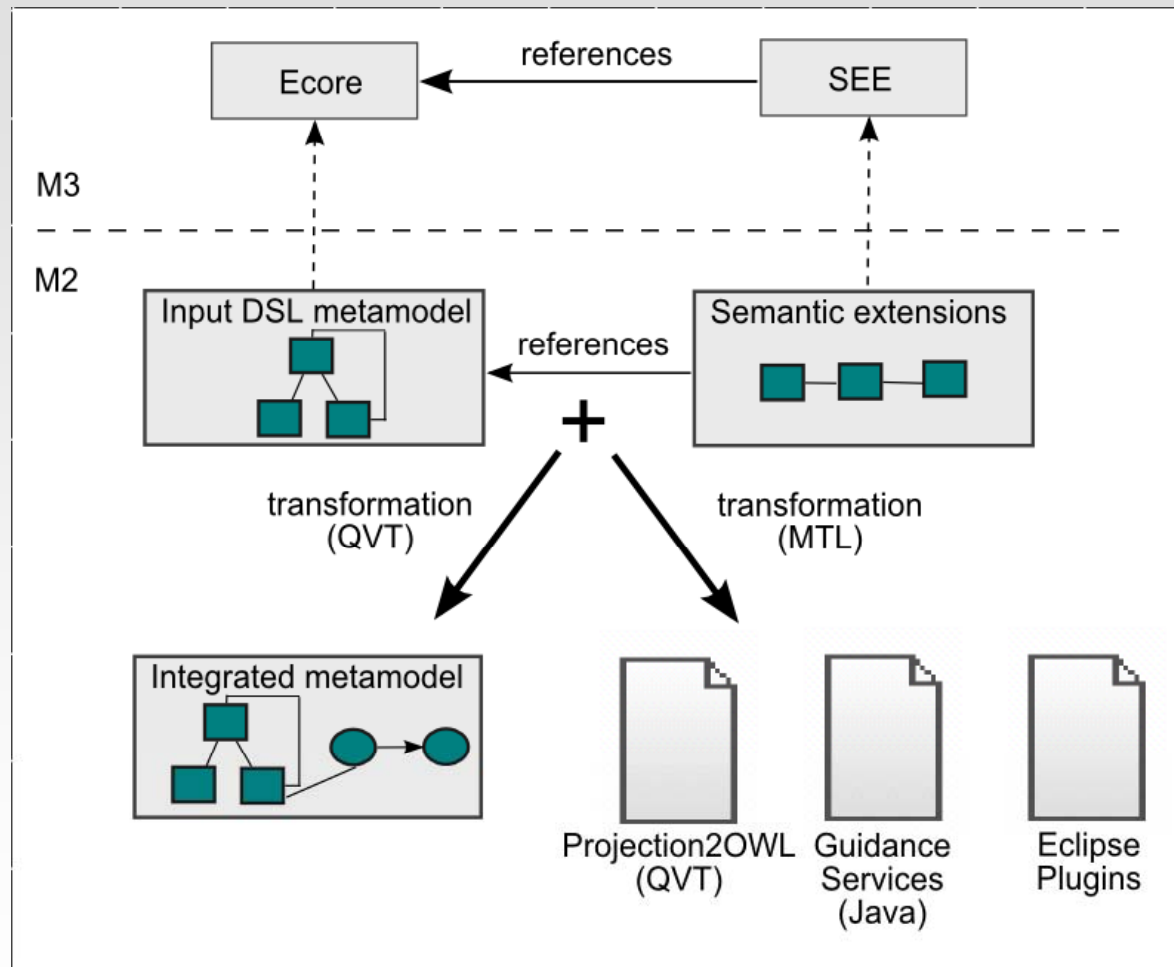
# Initial findings: challenges

---



- ❑ Multiple explanations and scalability
  - Single explanation result in early compiler style of work
  - Not possible with presently available tools
  - Expensive service: scalable implementation needed
  - Possible solution:
    - ❑ Approximate to OWL EL
  
- ❑ High cost of implementation
  - PDDSL is just one of many DSLs
  - Lots of manual work
  - Costly to reuse for other languages

# MOST extension: generalization of the case study





# Thank you

---

Further information:

[Krzysztof.Miksa@comarch.com](mailto:Krzysztof.Miksa@comarch.com)

[Pawel.Sabina@comarch.com](mailto:Pawel.Sabina@comarch.com)

[Marek.Kasztelnik@comarch.com](mailto:Marek.Kasztelnik@comarch.com)

**MOST - Marrying Ontology and Software Technology**