How to succeed with your Modelling Approach using Enterprise Architect

About Meta-Models and Metamodeling



Part 1.5

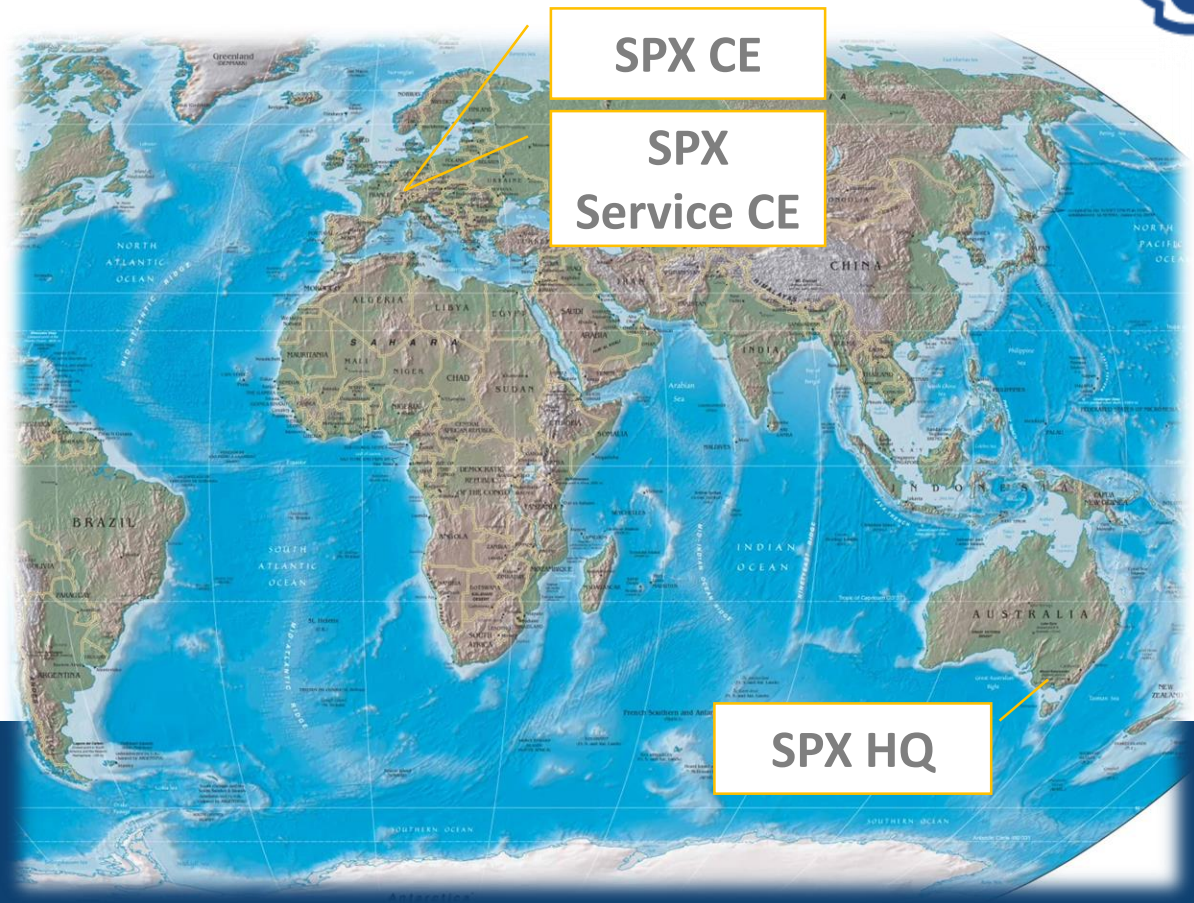
Horst Kargl

SPARX SYSTEMS CENTRAL EUROPE

© Sparx Systems Software GmbH

Who is **Sparx Systems** - Who is **Sparx Services**

- **Sparx Systems Central Europe** is a sister Company of Sparx Systems HQ.
  - Licenses, Training
- **Sparx Services Central Europe**
  - Consulting
  - Customizations



ENTERPRISE ARCHITECT

SPARX SYSTEMS CENTRAL EUROPE

Agenda

- Theory
  - Meta-Model
  - Model
  - Meta Meta-Model, etc.
- Praxis in EA
  - UML Profile
  - Meta Constraints
  - Additional Validation Possibilities
  - MDG Technology

ENTERPRISE ARCHITECT
SPARX SYSTEMS CENTRAL EUROPE

What is a Meta-Model?

- **Meta-Model**: is a Model about a Model.

- **Metamodeling**: is the process of creating such meta-models.



ENTERPRISE ARCHITECT

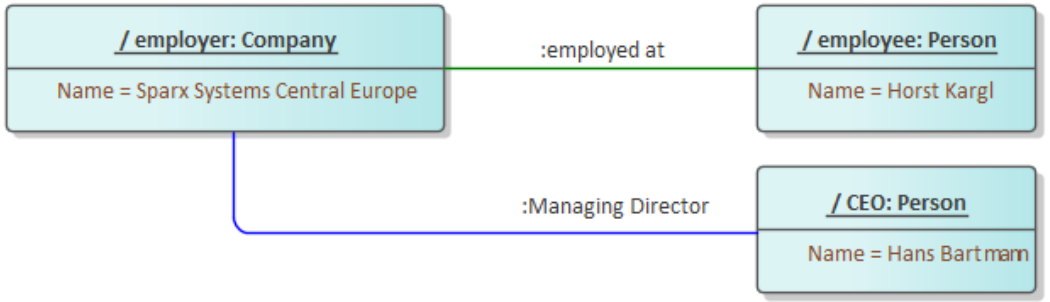
SPARX SYSTEMS CENTRAL EUROPE

Meta-Model Example



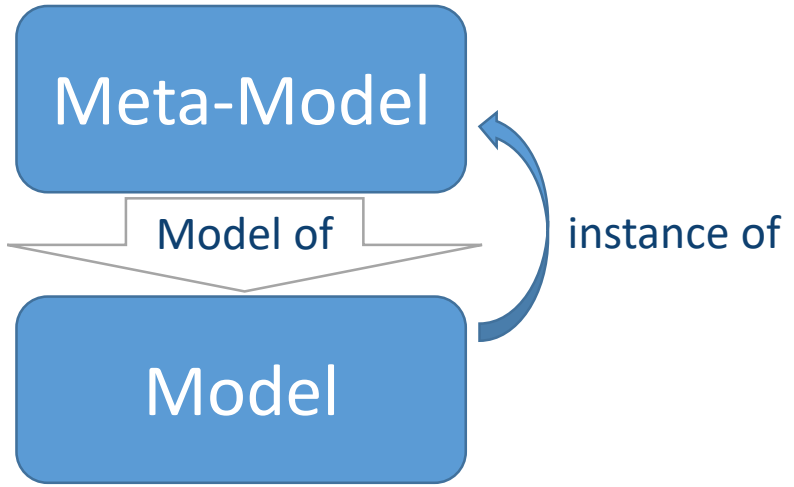
Metamodel

Model



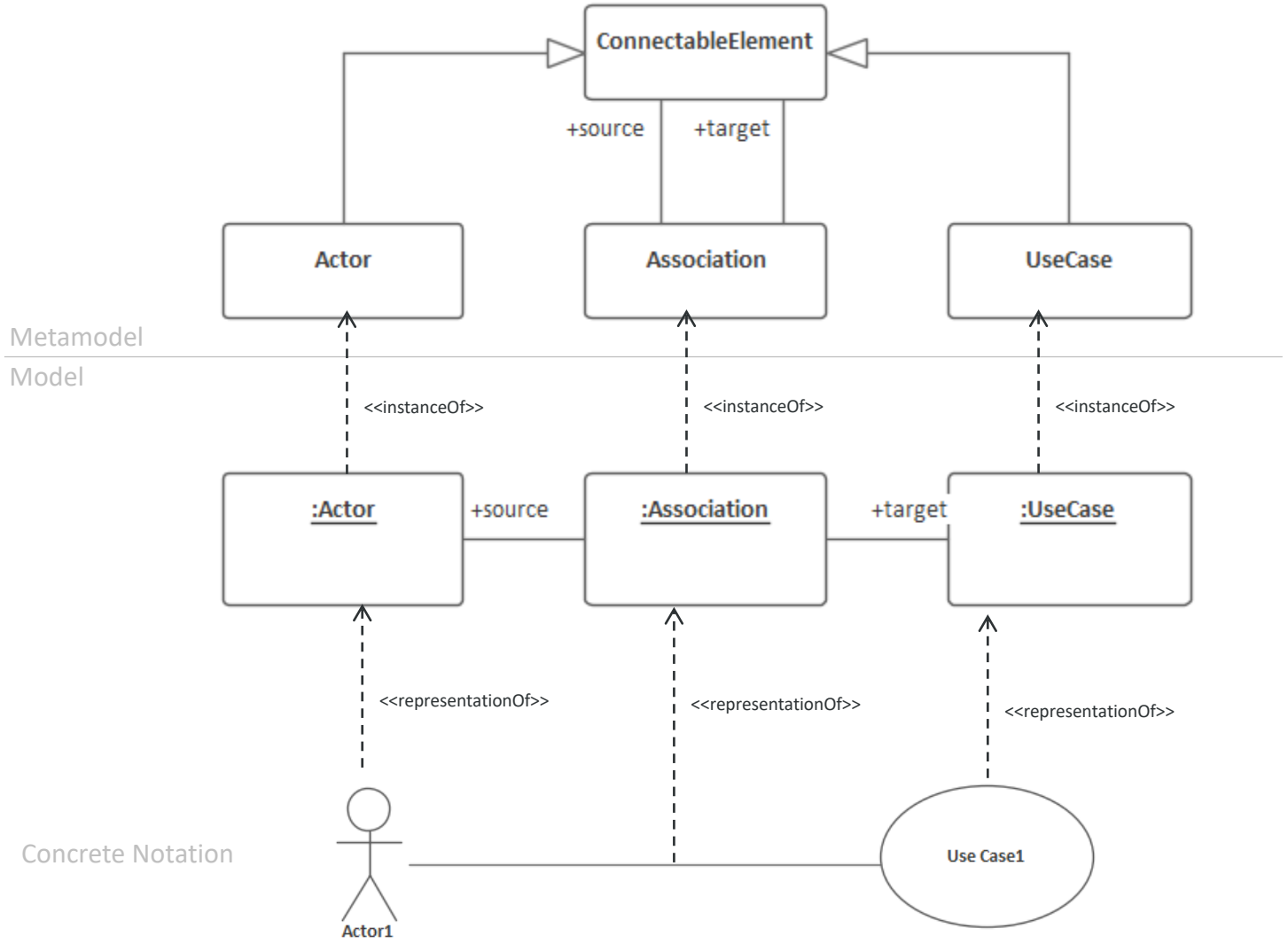


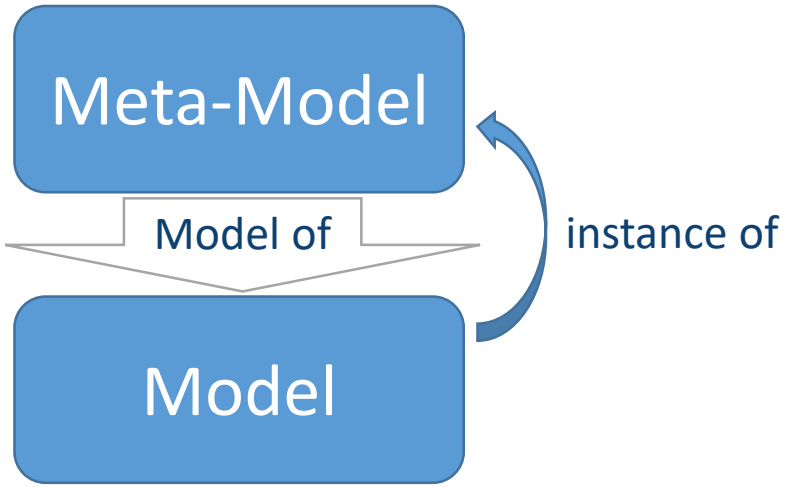
Possible
Concrete Notation



ENTERPRISE ARCHITECT

Meta-Model Example

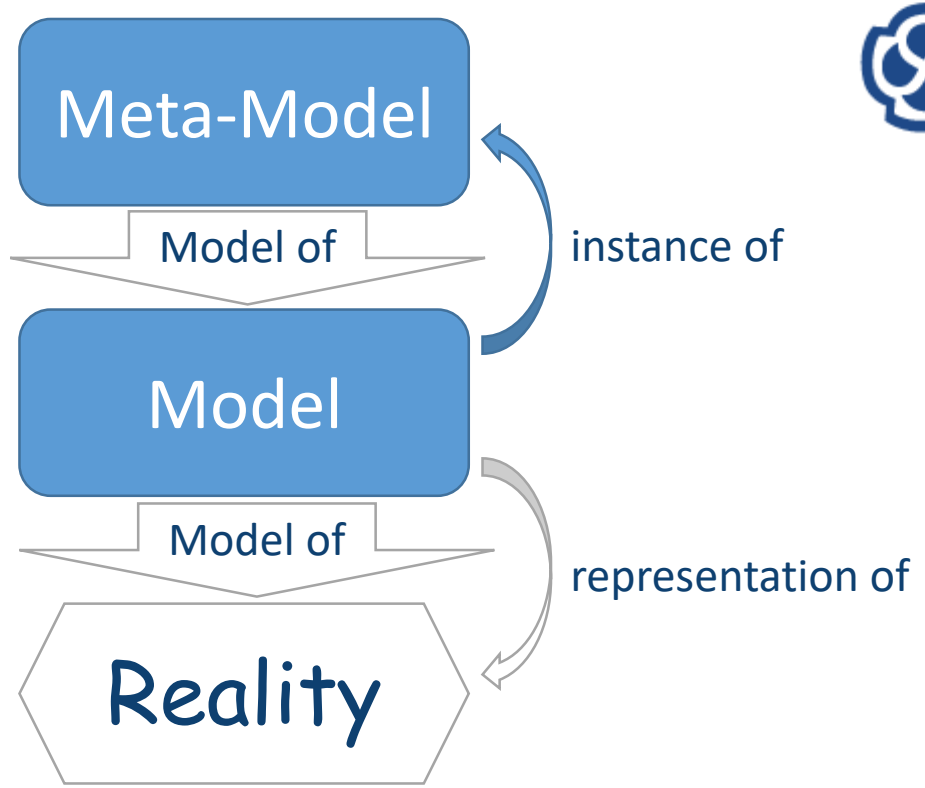




ENTERPRISE ARCHITECT

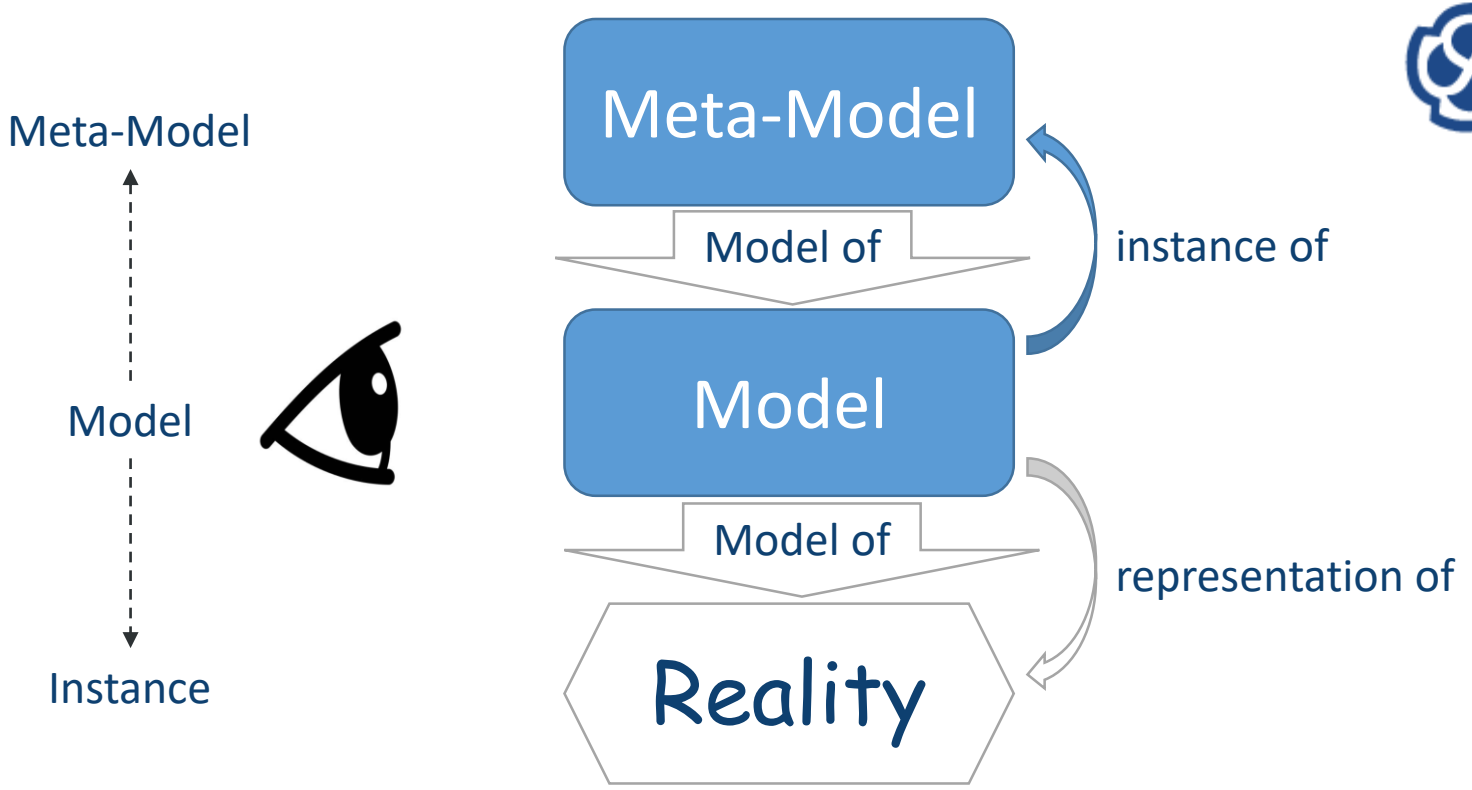
What is a Model?

- **Model**: A model is a simplified representation of a certain reality.
- **Modelling**: is the process and method of building models for a specific purpose.

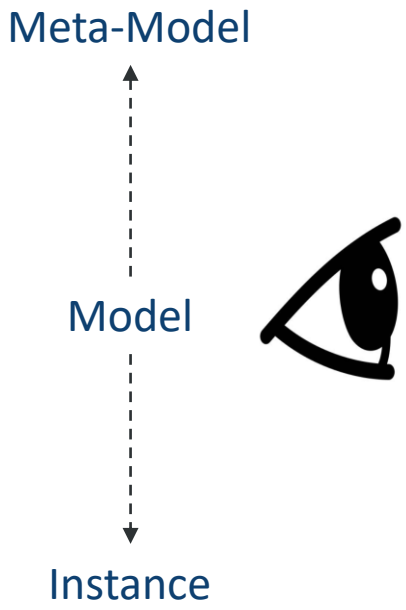


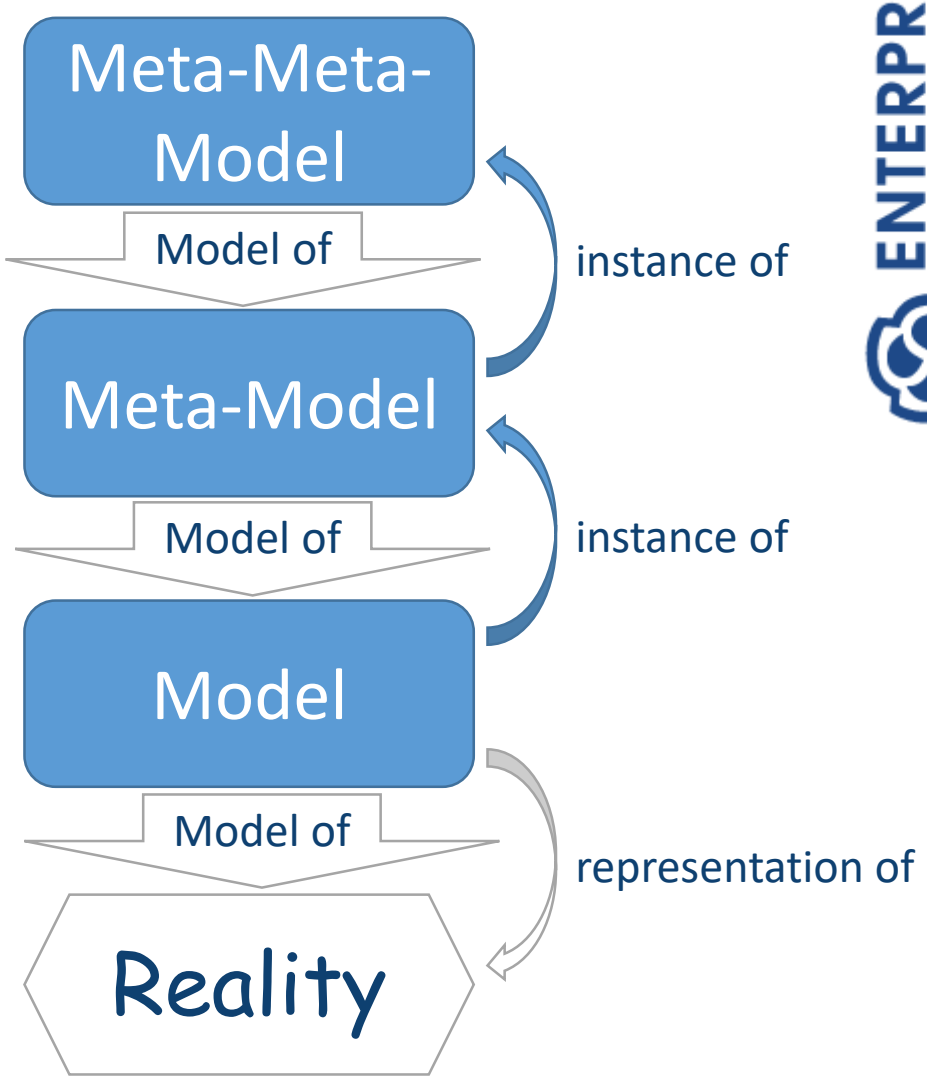
ENTERPRISE ARCHITECT
SPARX SYSTEMS CENTRAL EUROPE

What is a Meta Meta-Model?



What is a Meta Meta-Model?

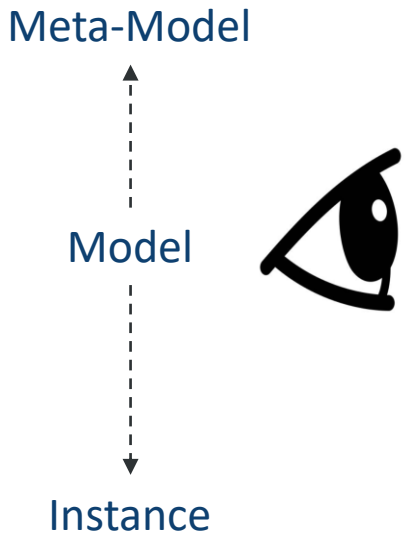


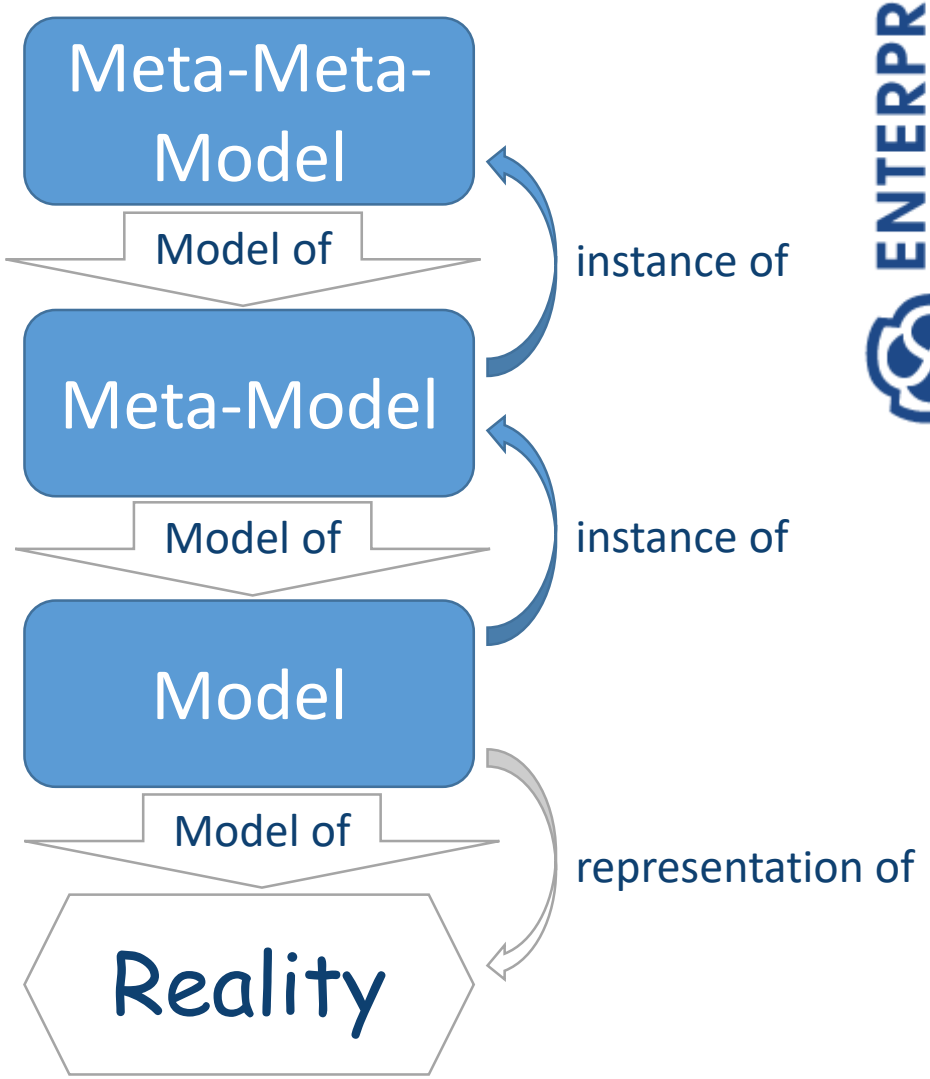


ENTERPRISE ARCHITECT

SPARX SYSTEMS CENTRAL EUROPE

What is a Meta Meta-Model?

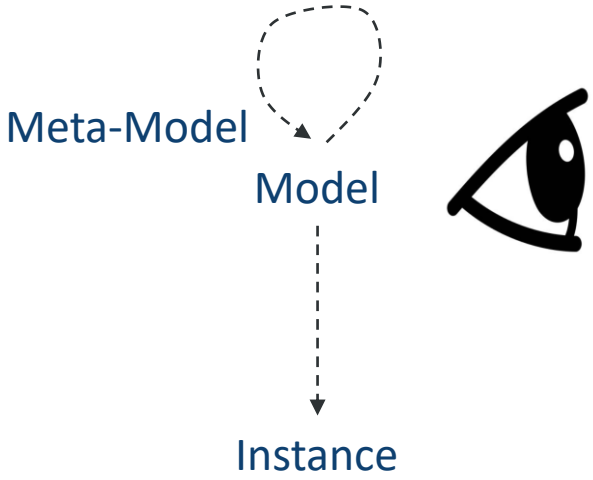


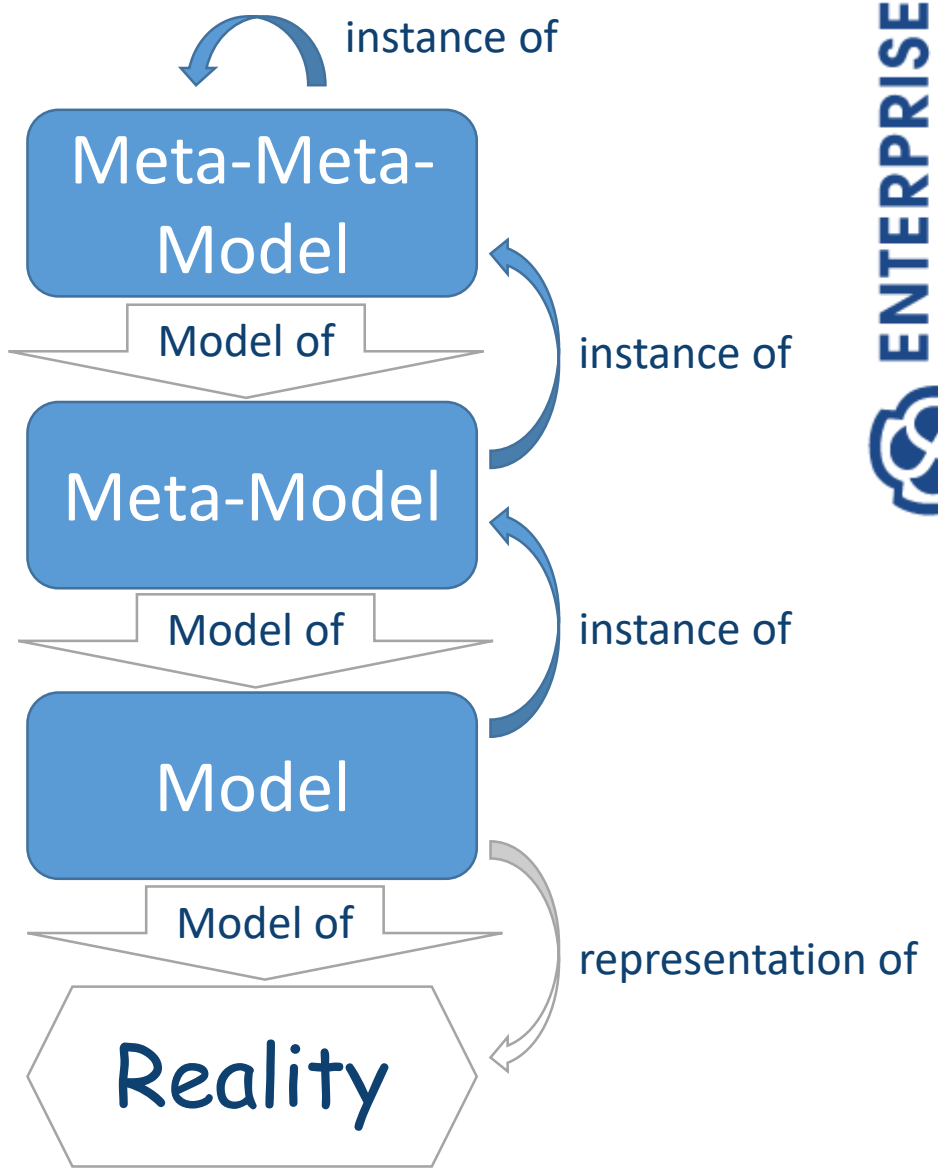


ENTERPRISE ARCHITECT

SPARX SYSTEMS CENTRAL EUROPE

What is a Meta Meta-Model?

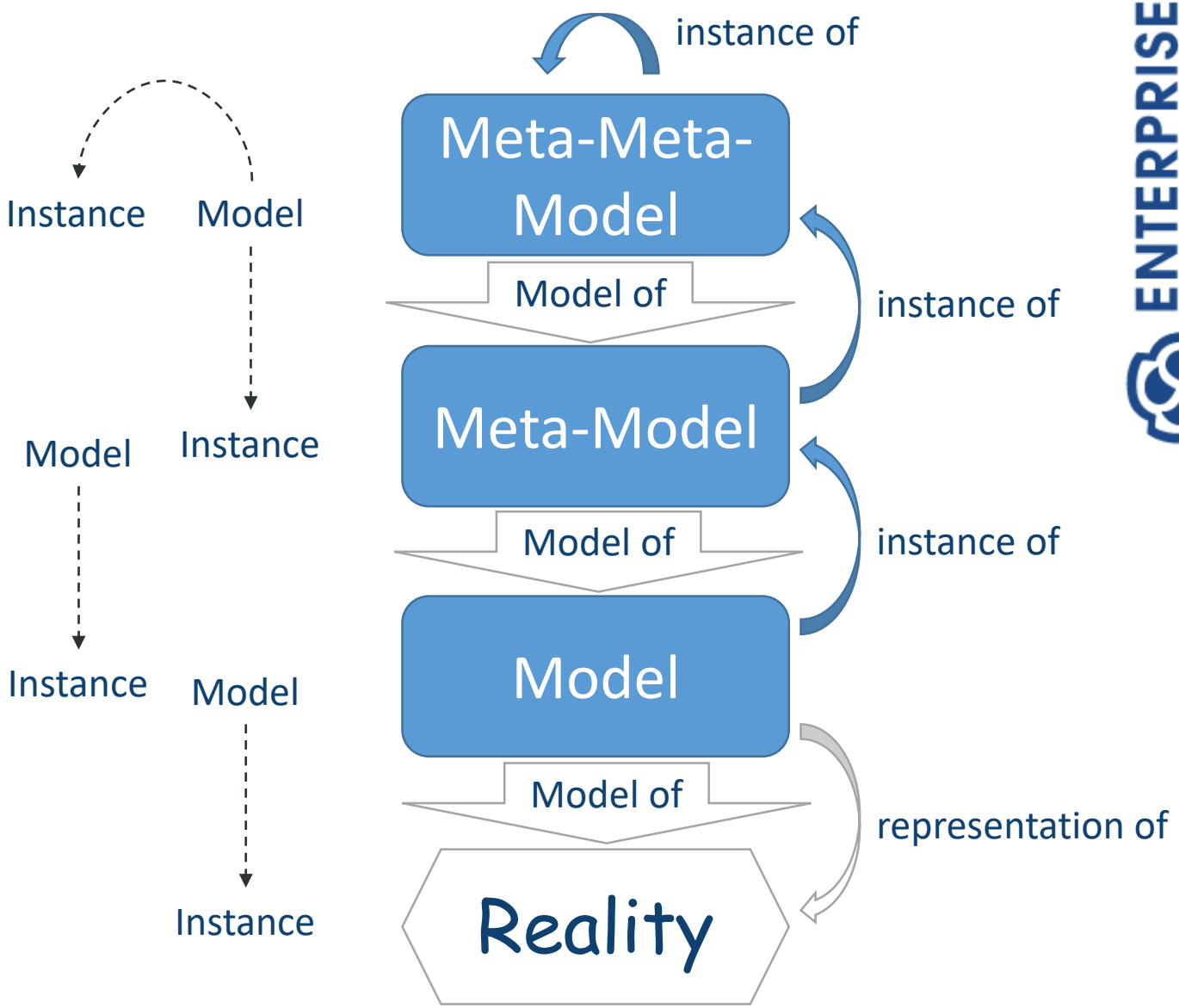




ENTERPRISE ARCHITECT

SPARX SYSTEMS CENTRAL EUROPE

What is a Meta Meta-Model?

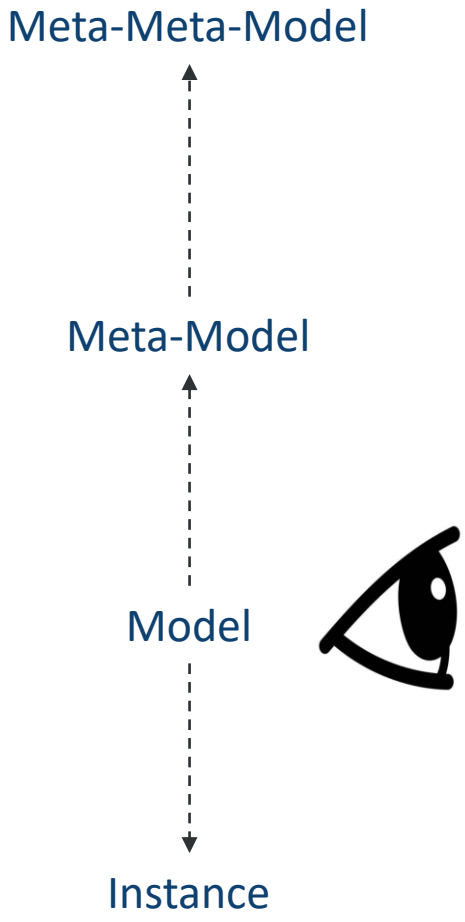


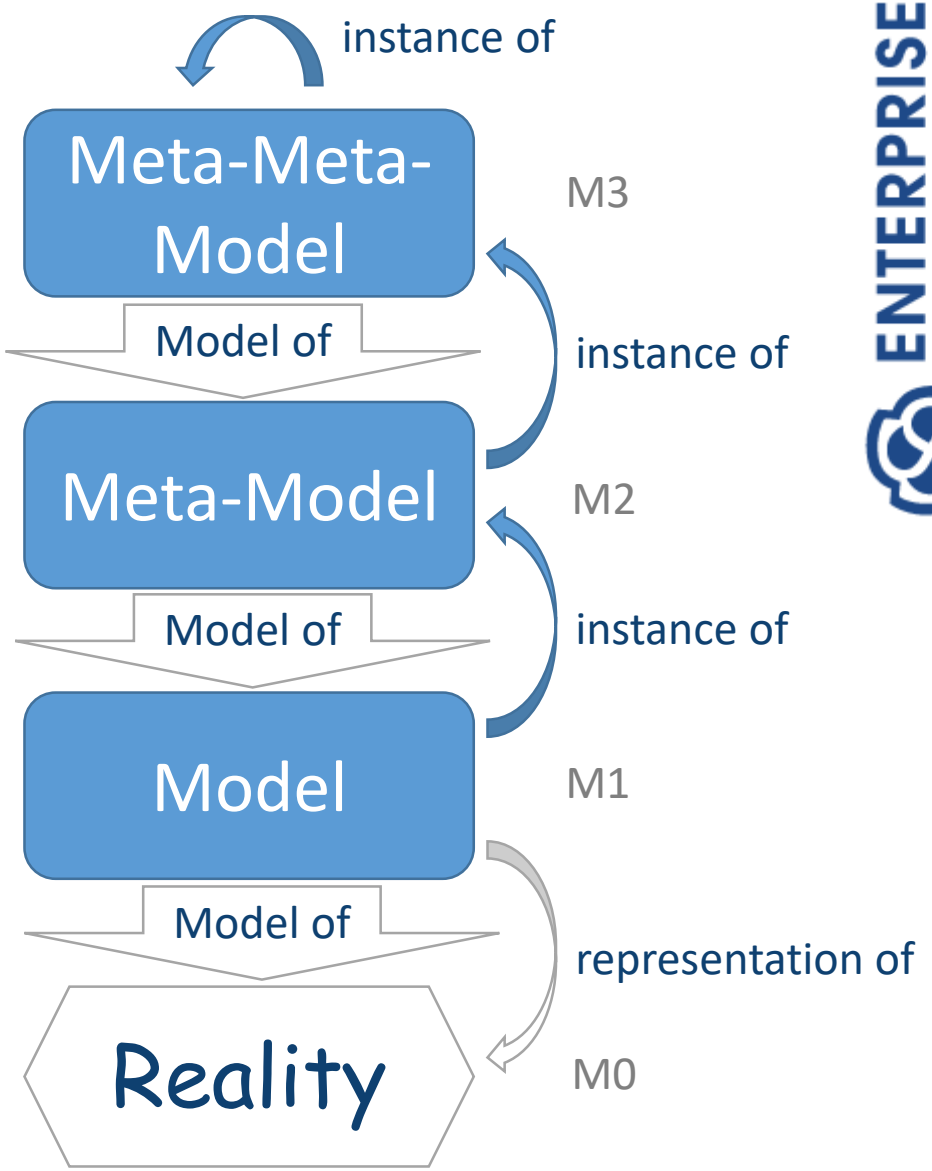
ENTERPRISE ARCHITECT

SPARX SYSTEMS CENTRAL EUROPE

What is a Meta Meta-Model?

The Language Stack



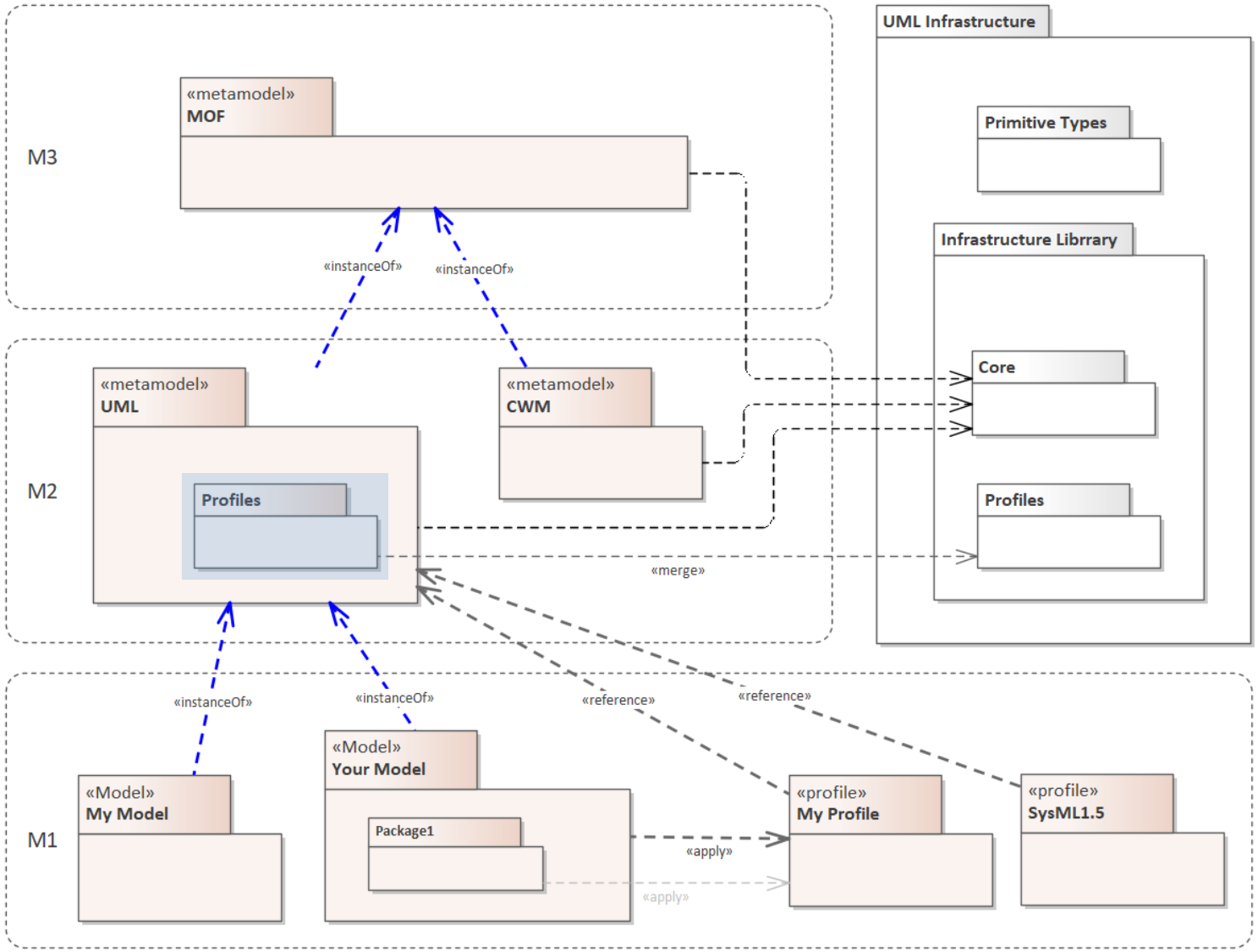


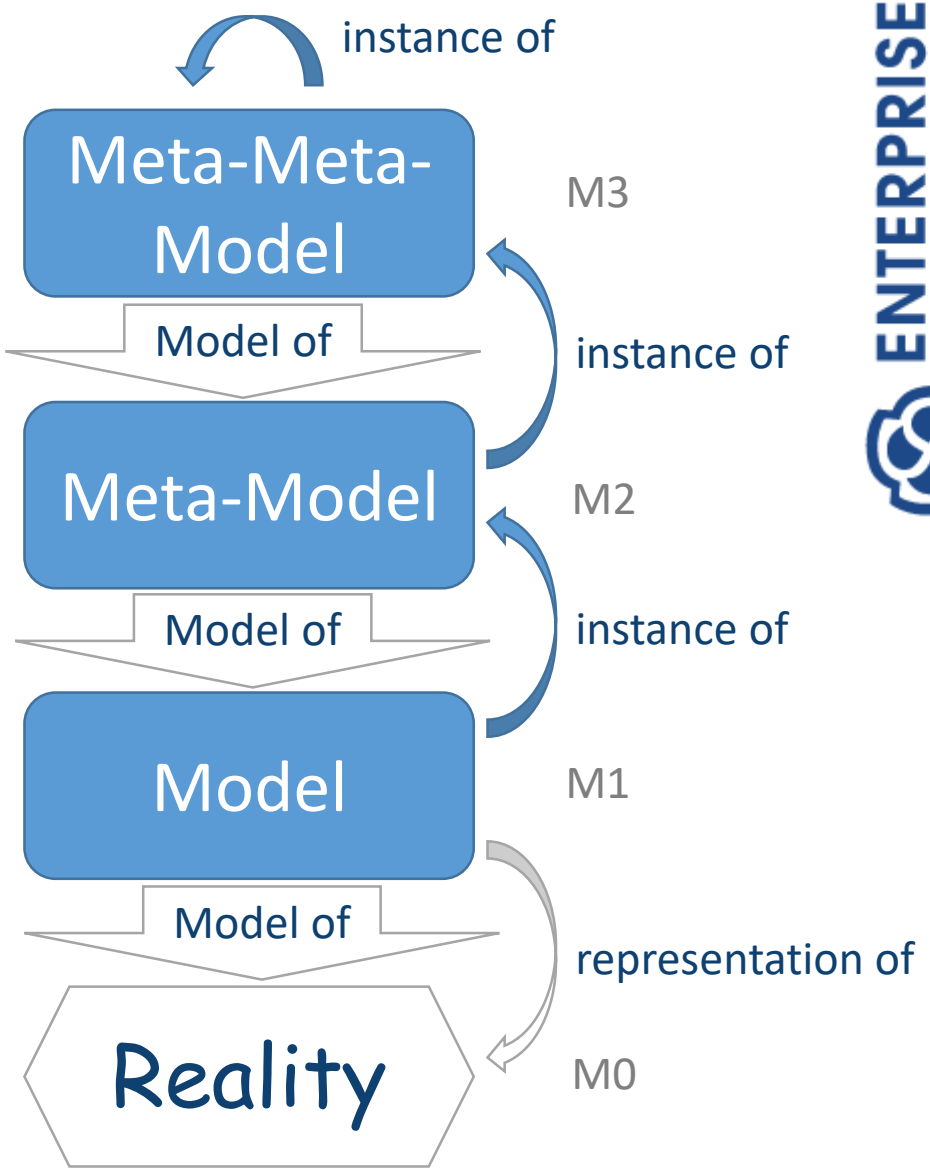
ENTERPRISE ARCHITECT

SPARX SYSTEMS CENTRAL EUROPE

Language Stack

based on the Meta Object Facility

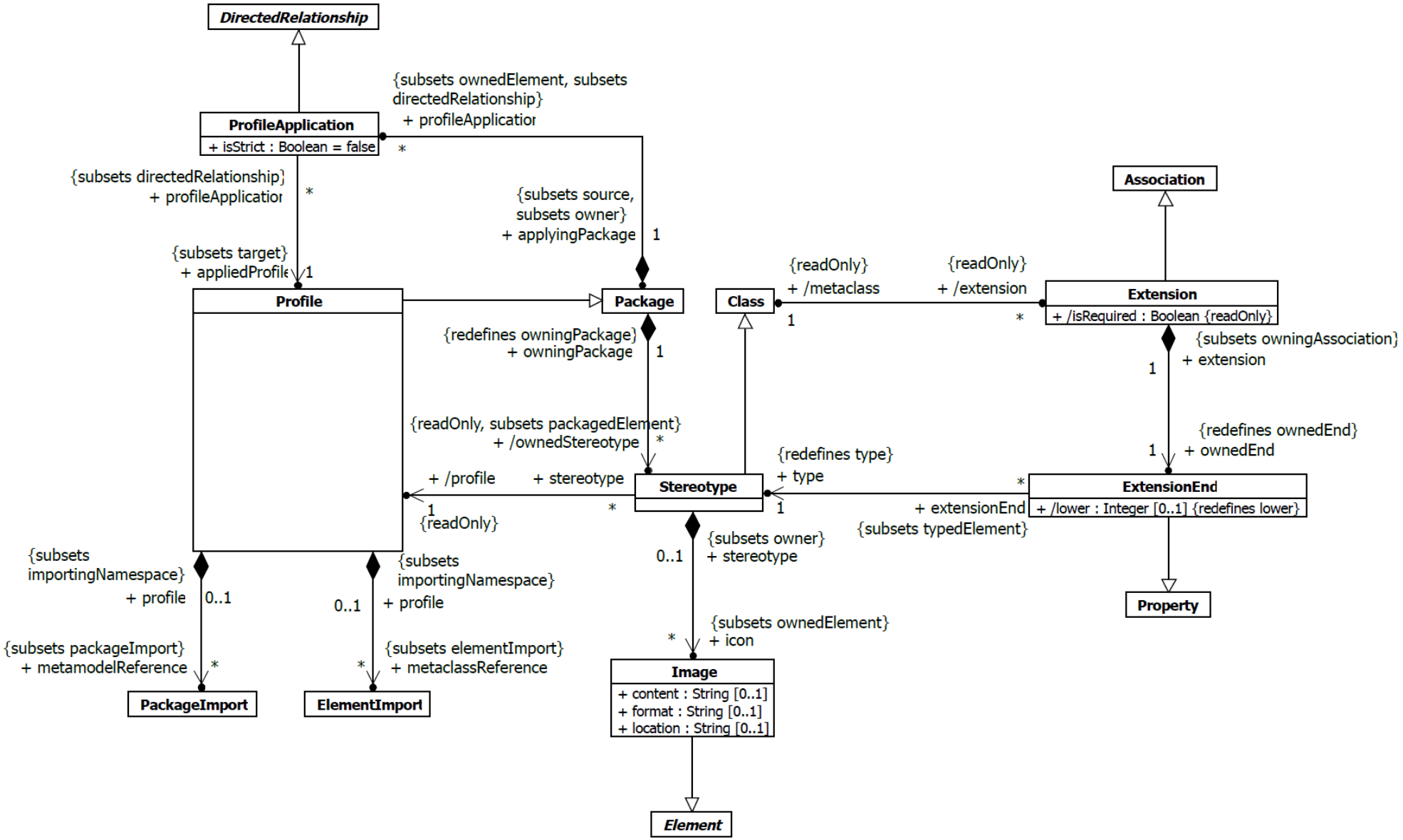




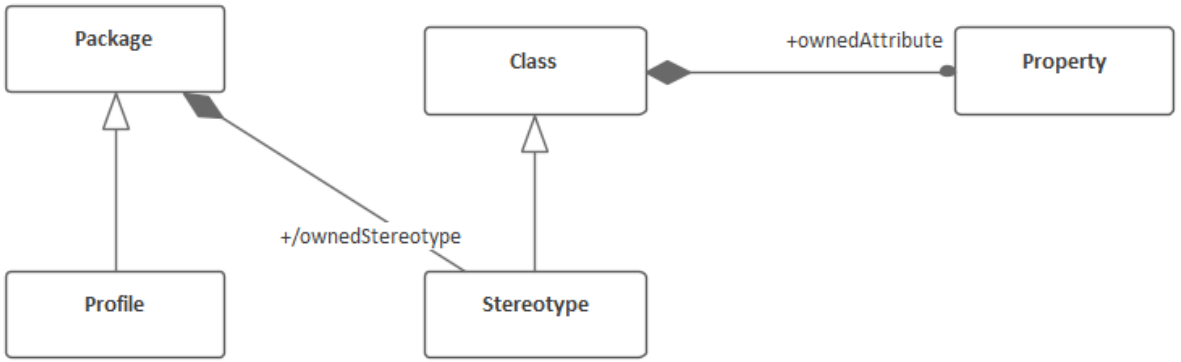
ENTERPRISE ARCHITECT

Profile

M2

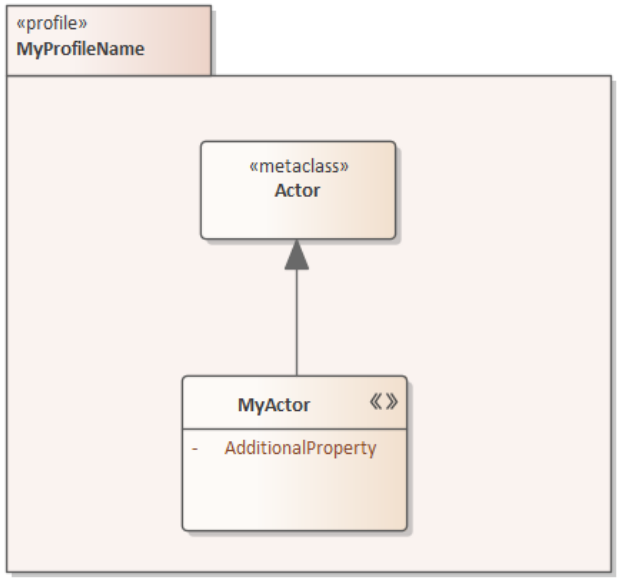

DirectedRelationship
ProfileApplication
+ isStrict : Boolean = false
{subsets ownedElement, subsets directedRelationship}
+ profileApplicatior
{subsets directedRelationship}
+ profileApplicatior
{subsets target}
+ appliedProfile
{subsets source, subsets owner}
+ applyingPackage
Profile
Package
{redefines owningPackage}
+ owningPackage
{readOnly, subsets packagedElement}
+ /ownedStereotype
+ /profile
+ stereotype
{readOnly}
Stereotype
Class
{readOnly}
+ /metaclass
{readOnly}
+ /extension
Association
Extension
+ /isRequired : Boolean {readOnly}
{subsets owningAssociation}
+ extension
{redefines ownedEnd}
+ ownedEnd
{redefines type}
+ type
+ extensionEnd
{subsets typedElement}
ExtensionEnd
+ /lower : Integer [0..1] {redefines lower}
Property
{subsets importingNamespace}
+ profile
{subsets packageImport}
+ metamodelReference
PackageImport
{subsets importingNamespace}
+ profile
{subsets elementImport}
+ metaclassReference
ElementImport
{subsets owner}
+ stereotype
{subsets ownedElement}
+ icon
Image
+ content : String [0..1]
+ format : String [0..1]
+ location : String [0..1]
Element

Profile – Simple Representation



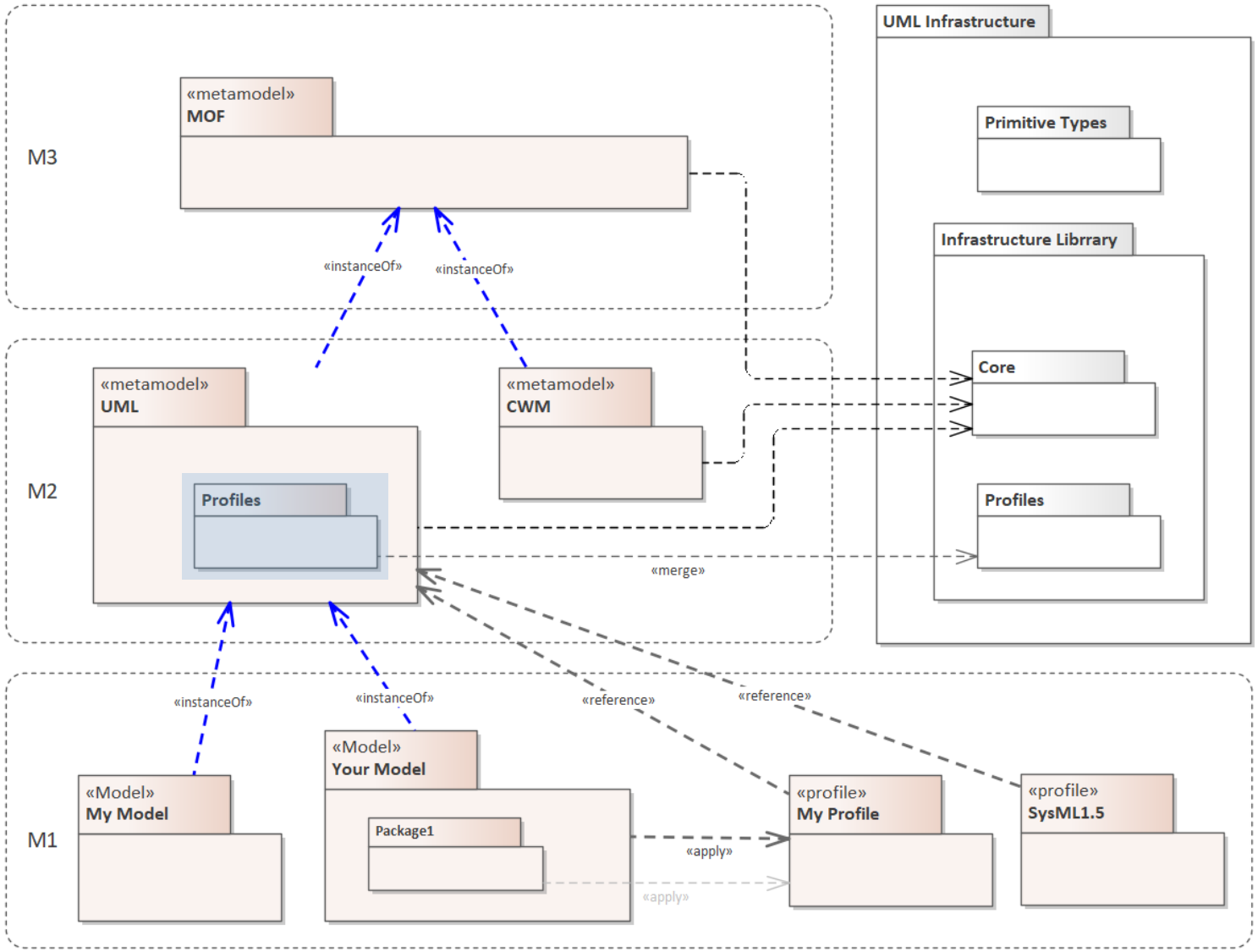
M2

M1

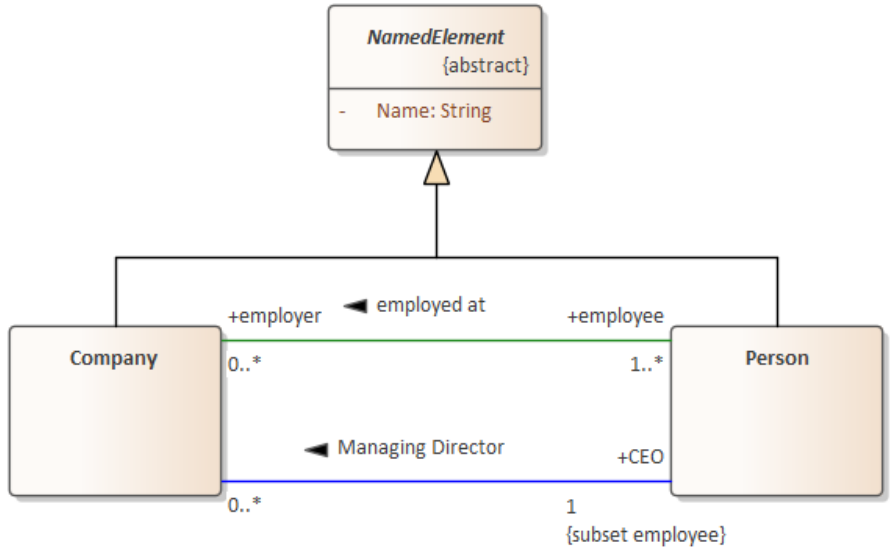


Meta Object Facility

The Language Stack

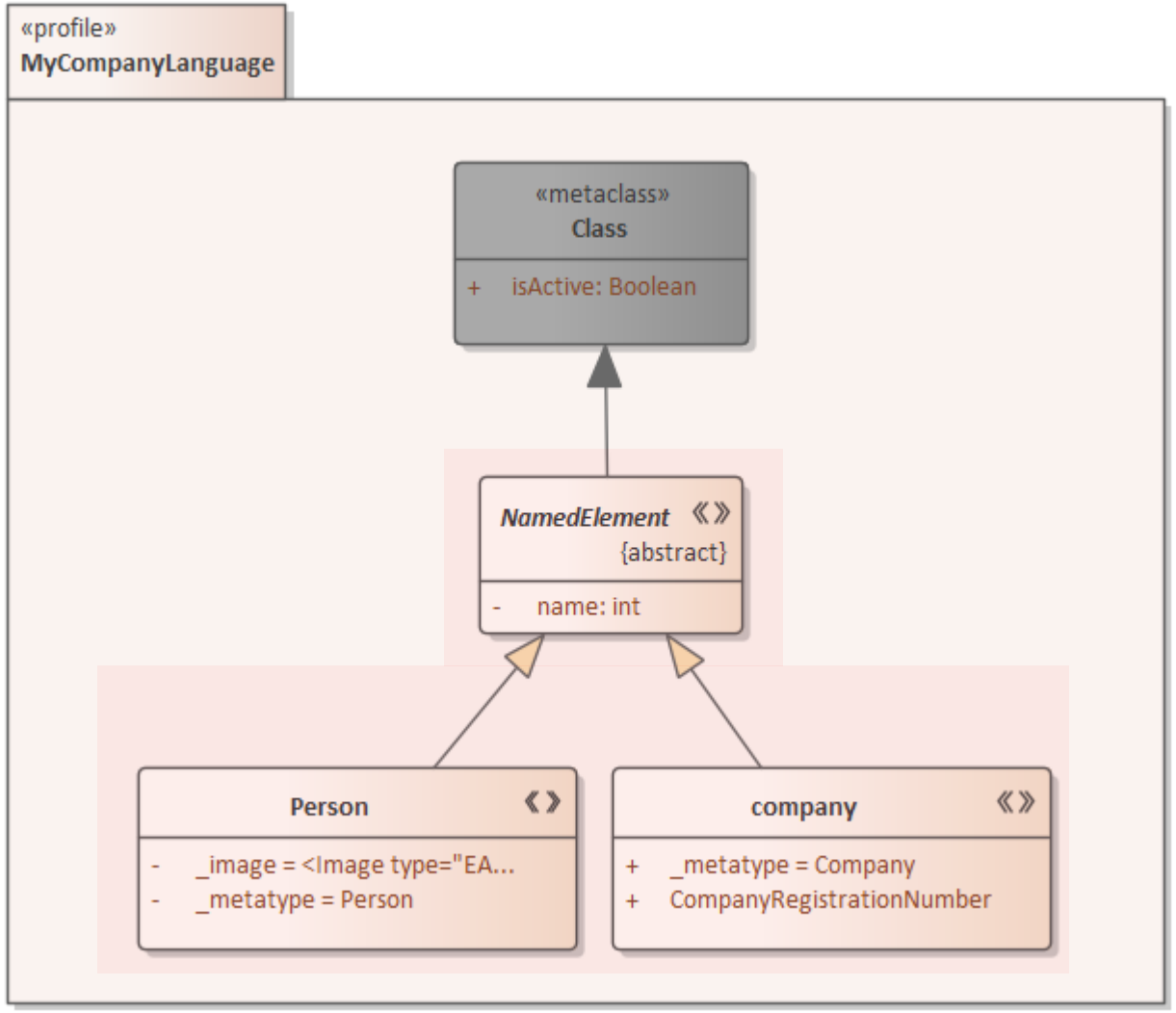


A Simple MetaModel



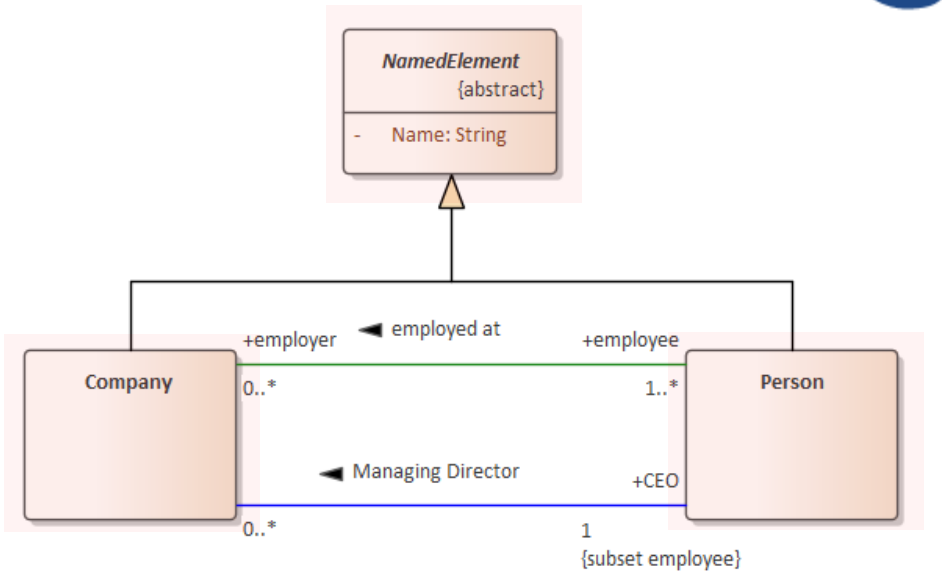
We translate the meta-model now into a UML Profile.

Translate a Meta-Model into a UML-Profile



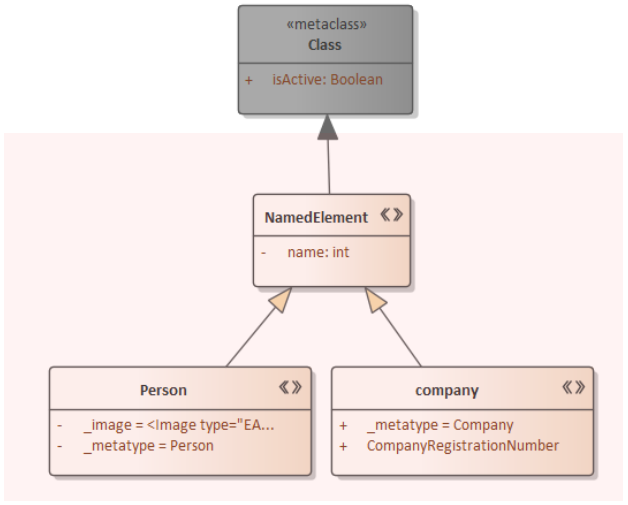
Transform

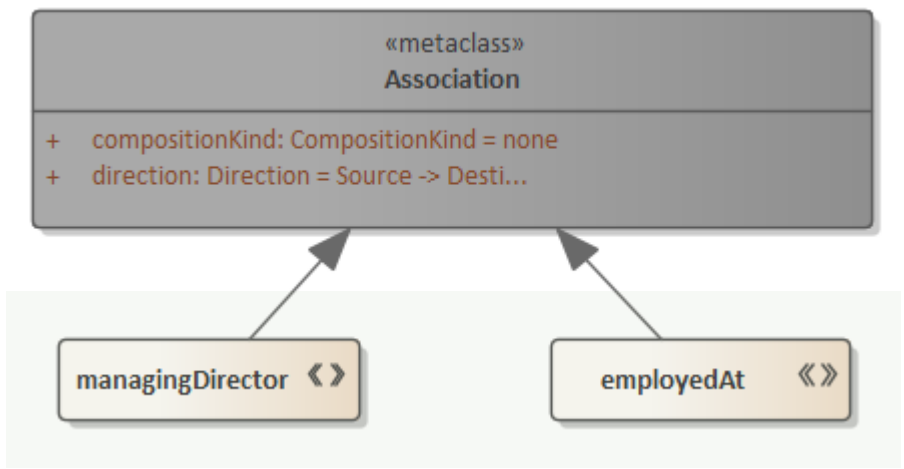
A Simple MetaModel



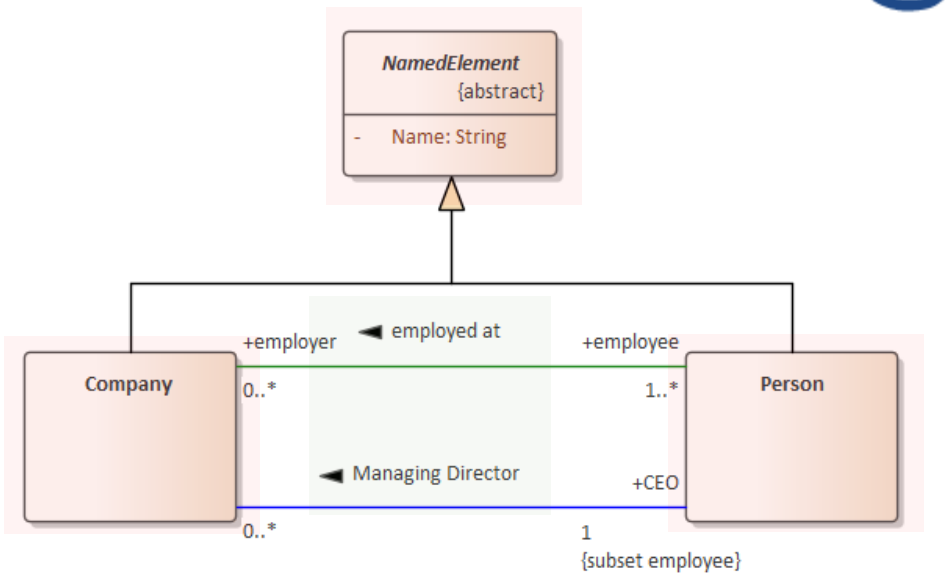
We translate the meta-model now into a UML Profile.

Translate a Meta-Model into a UML-Profile



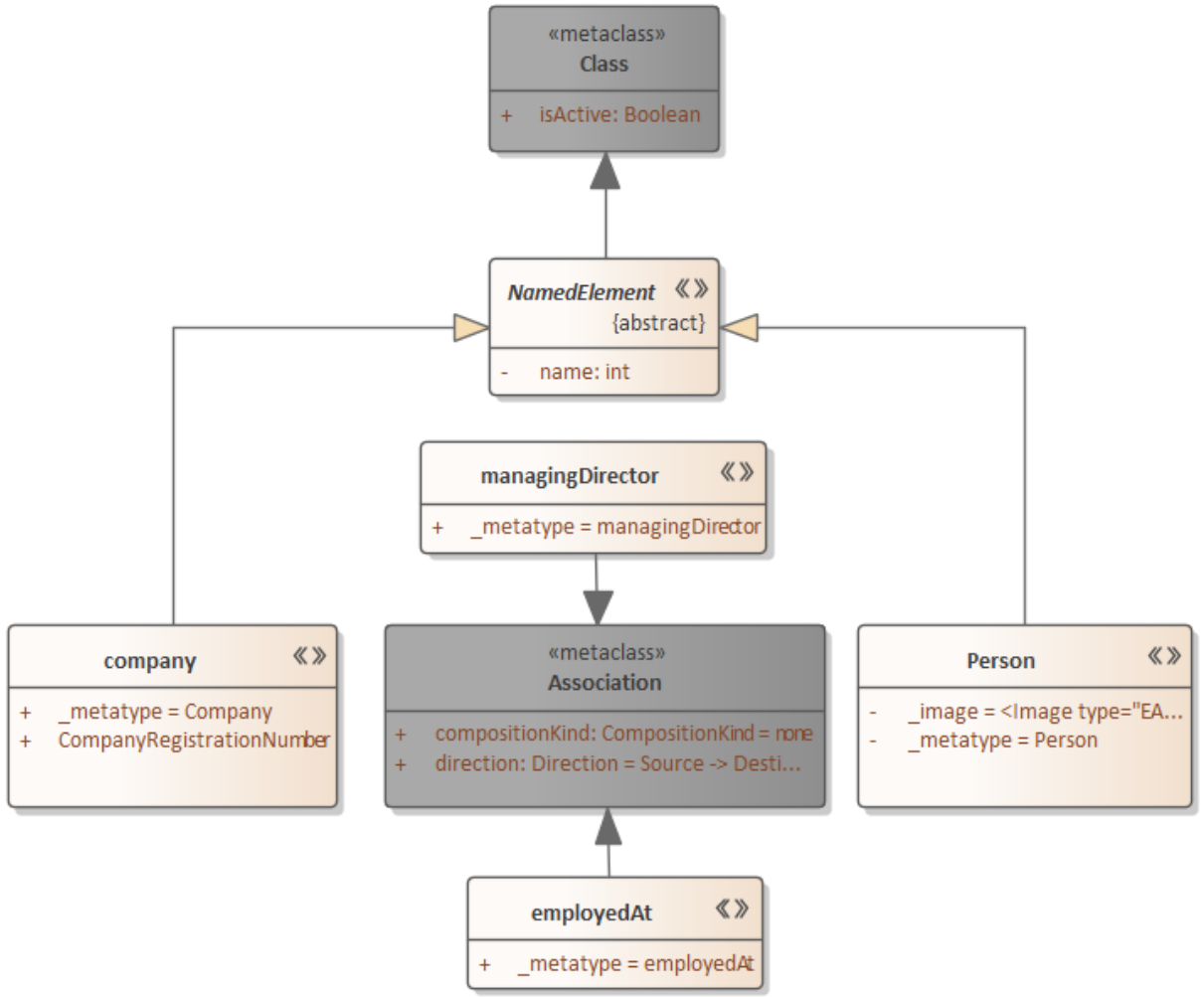


Transform

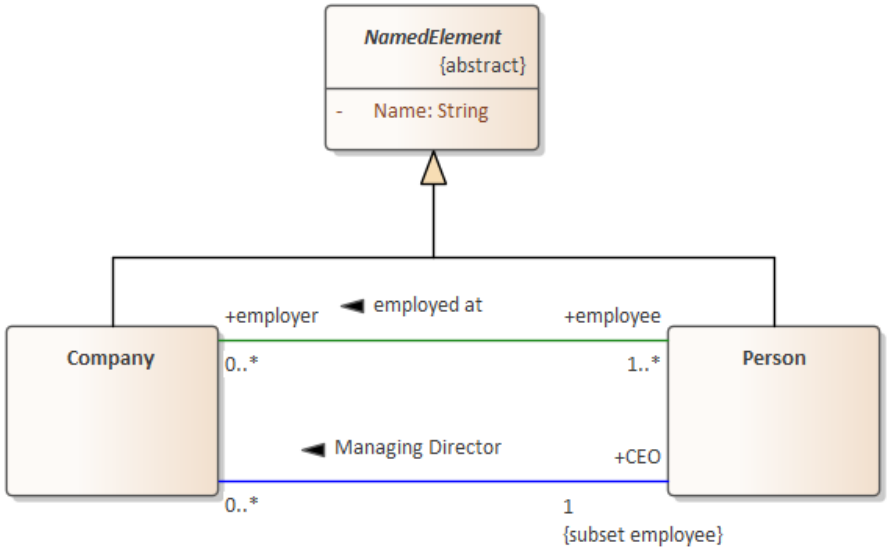


We translate the meta-model now into a UML Profile.

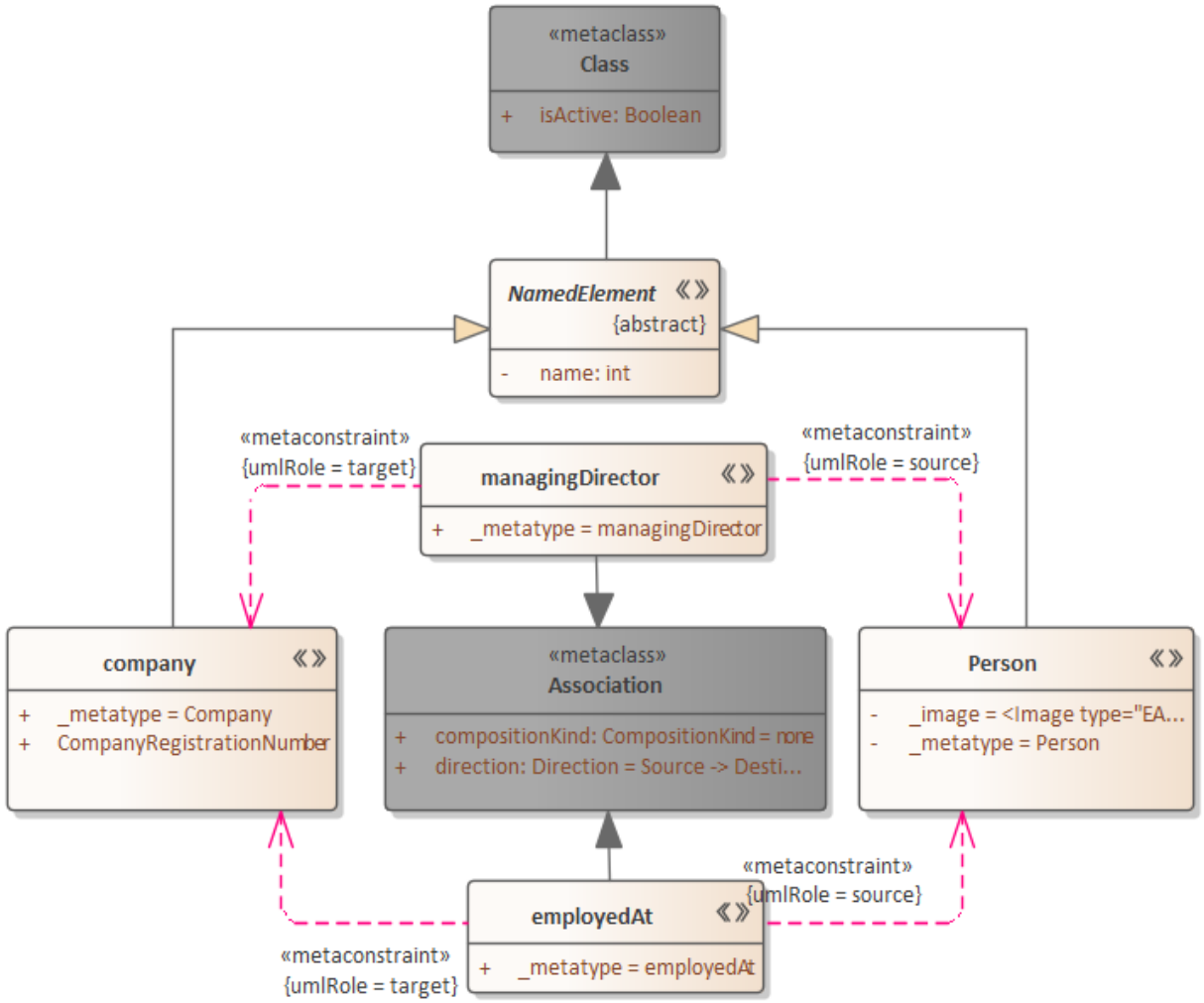
Translate a Meta-Model into a UML-Profile



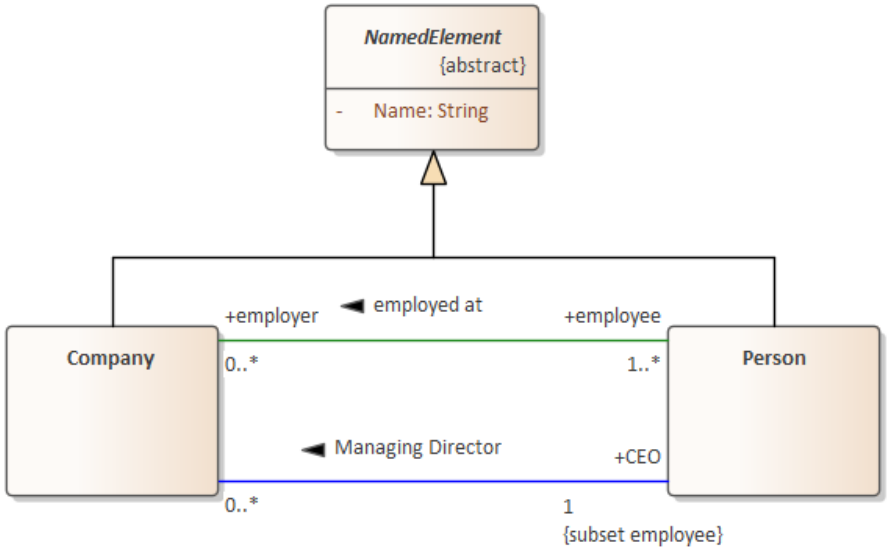
Transform



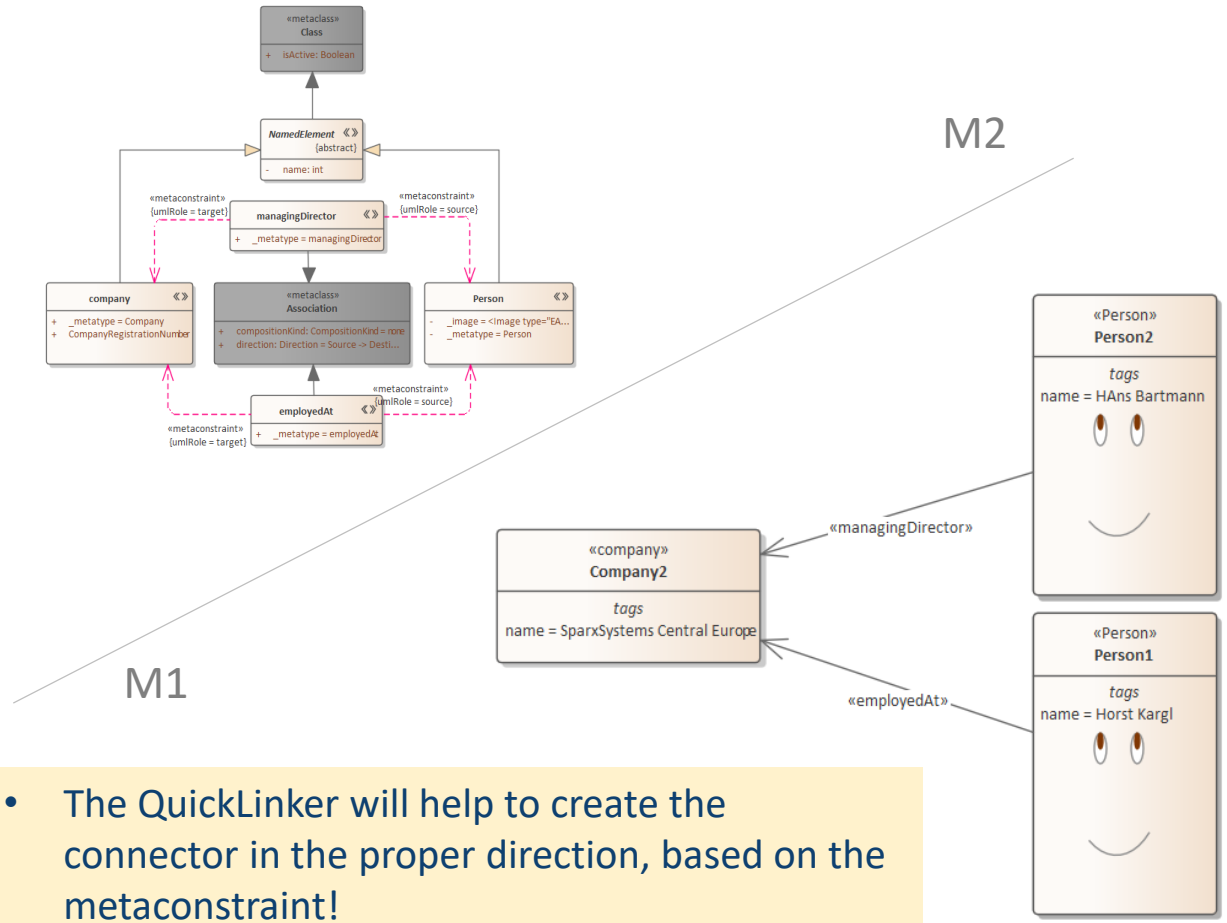
Add MetaModel Constraints



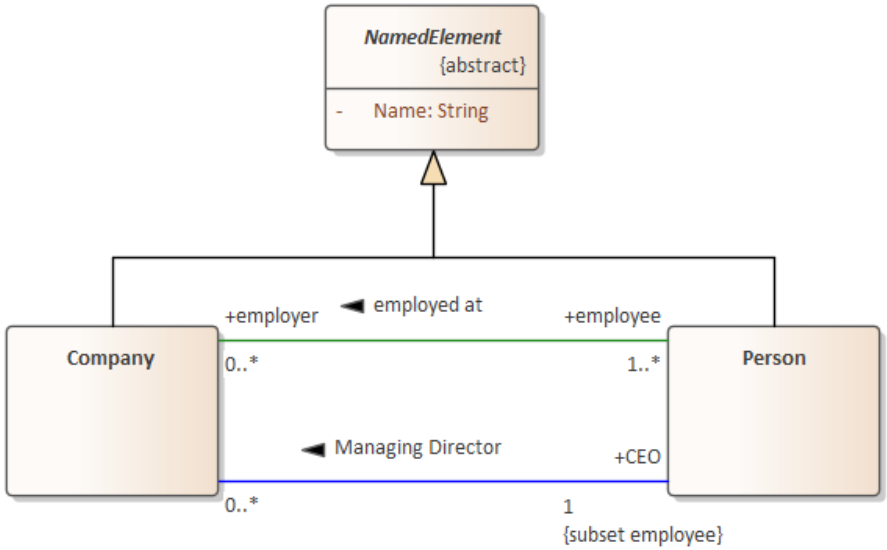
Transform



How the Model could Look Like



Transform

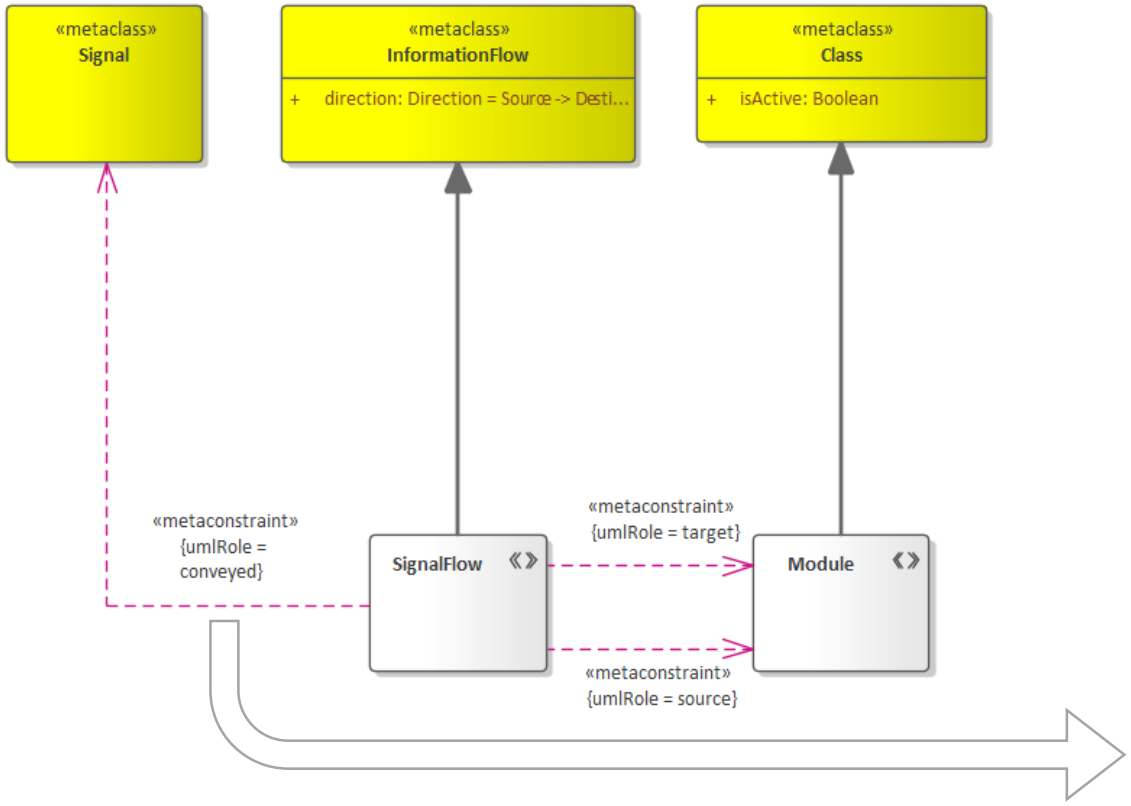


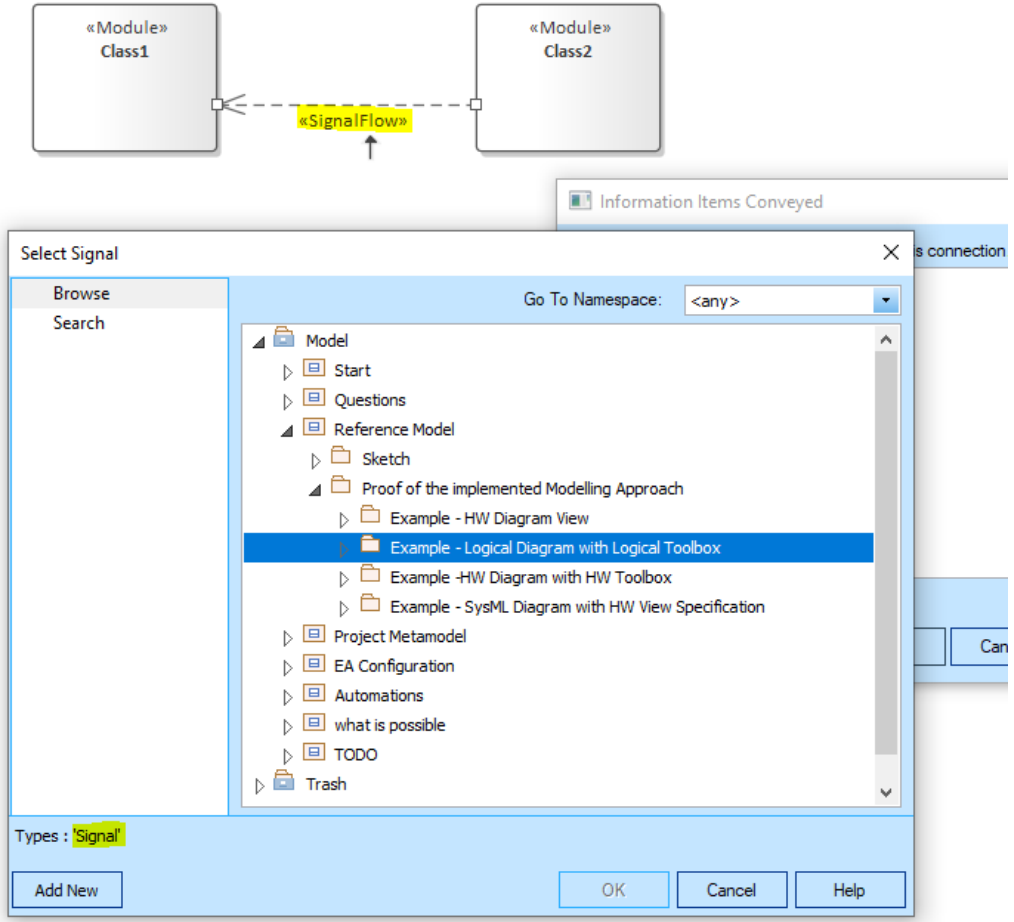
- The QuickLinker will help to create the connector in the proper direction, based on the metaconstraint!
- Model Validation will also find connectors with wrong direction based on the metaconstraint!

Other MetaModel Constraints

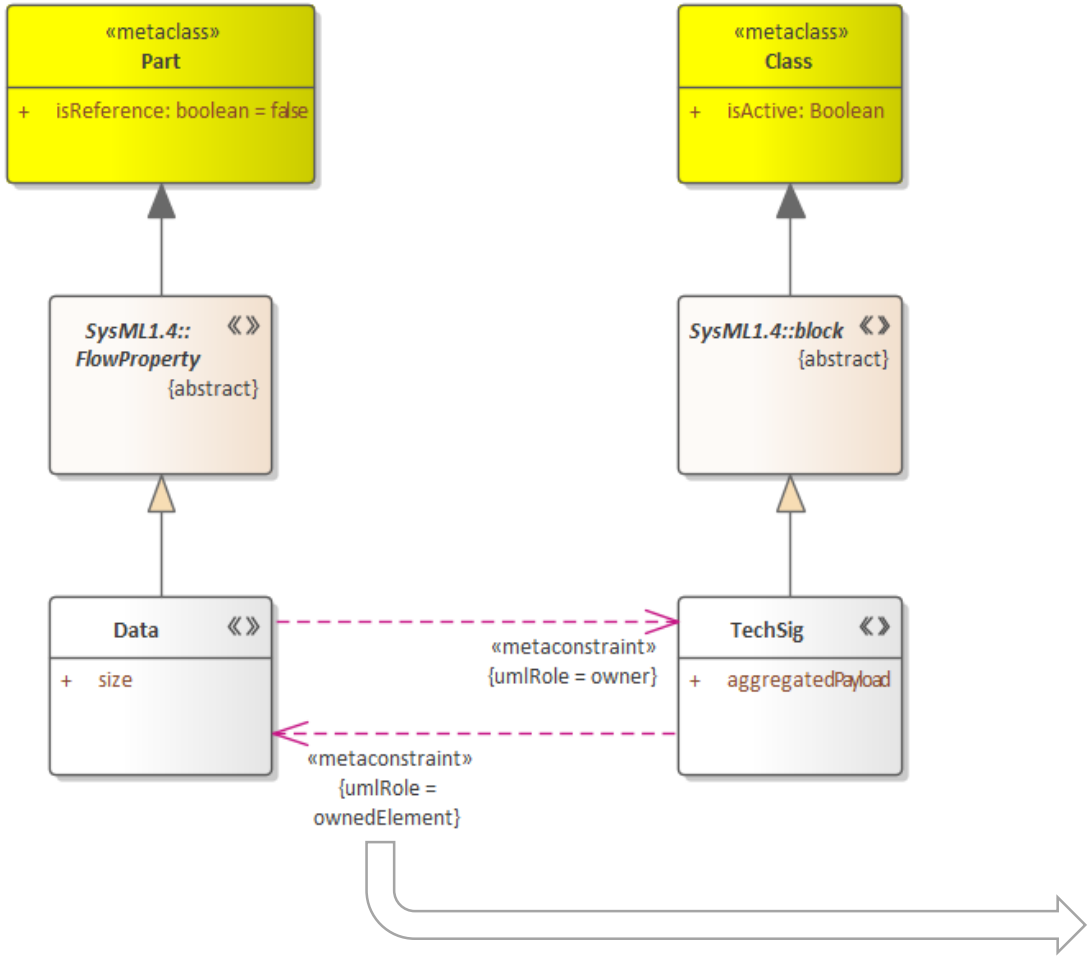
- …
- **Classifier/Type** – restricts the classifier/Type for the source Stereotype element to the target Stereotype element
- **behavior** - restricts the behavior for the source Stereotype () element to the target Stereotype element
- **conveyed** - restricts the conveyed element for the source Stereotype element to the target Stereotype element
- **slot** - restricts the slot for the source Stereotype element to the target Stereotype element
- **client/source/end[0].role/informationSource** – restricts the source of a connector to the target Stereotype element
- **supplier/target/end[1].role/informationTarget** - restricts the target of a connector to the target Stereotype element
- **realizingConnector/realizingActivityEdge/realizingMessage** - restricts the relationship that can realize an information flow
- **typedElement/instanceSpecification** – when dropping as classifier from the **Browser window**, this constraint restricts the type to the target Stereotype element
- **owner/class/activity/owningInstance** – restricts the container of this element to the target Stereotype element; this constraint is used to create embedded element rules for the Quick Linker and validate nesting during Model Validation
- **ownedElement/ownedAttribute/ownedOperation/ownedParameter/ownedPort** – restricts the element/attribute/operation/parameter/port that can be owned by the source Stereotype element; this constraint is typically used to validate nesting during Model Validation
- **annotatedElement/constrainedElement** – restricts the target of a Note Link connector to the target Stereotype element
- …

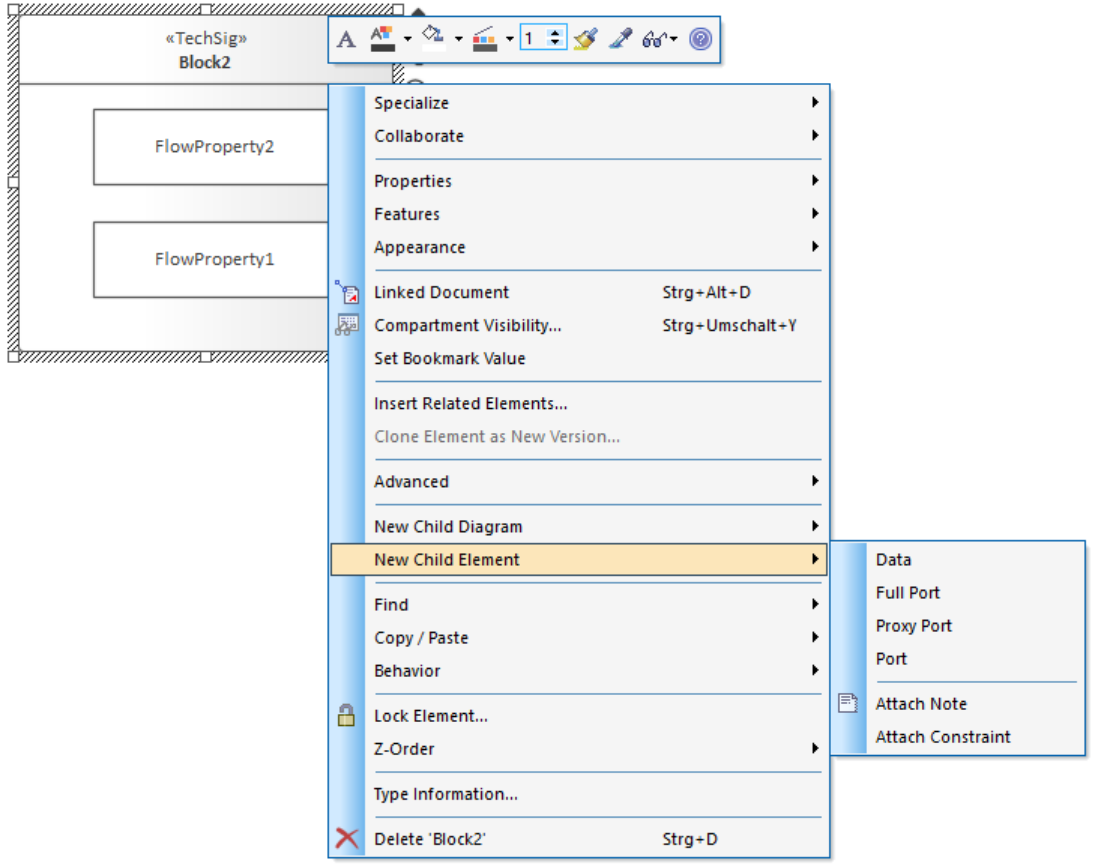
Other MetaModel Constraints



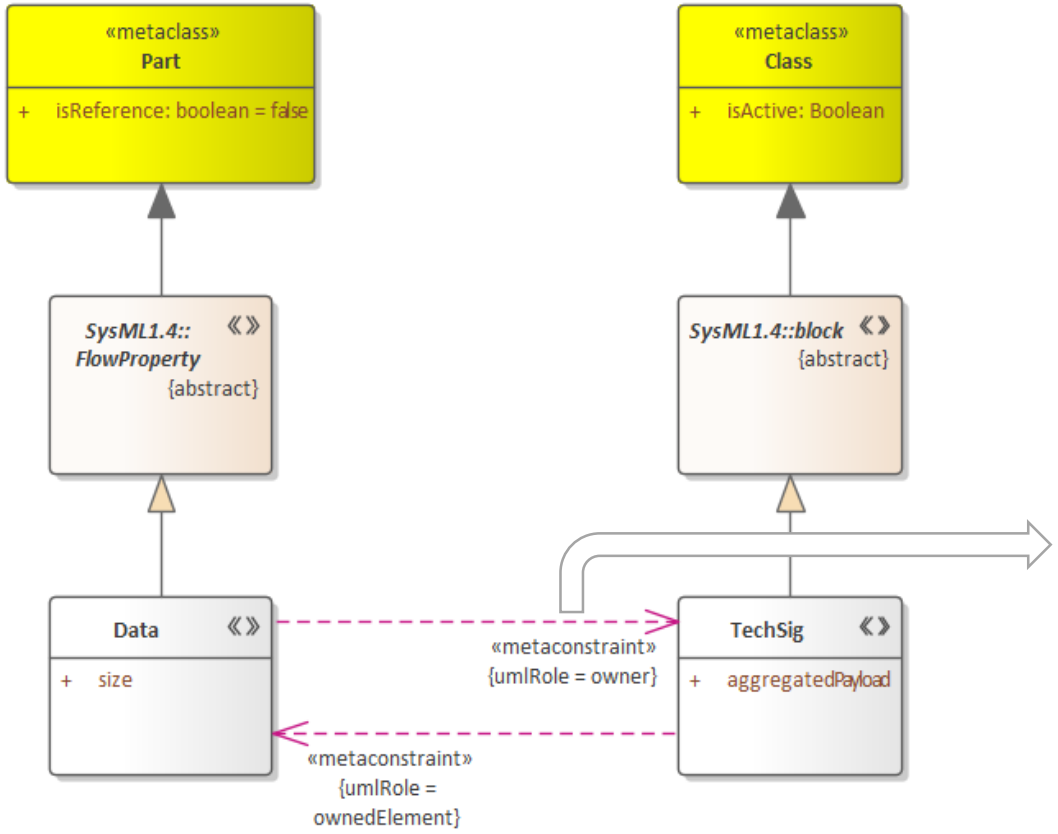


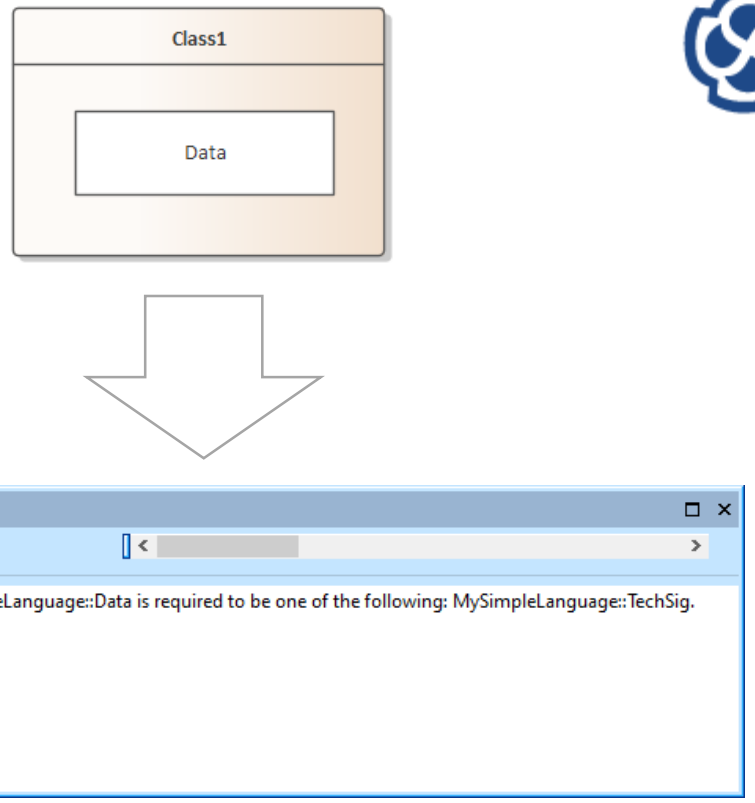
Other MetaModel Constraints



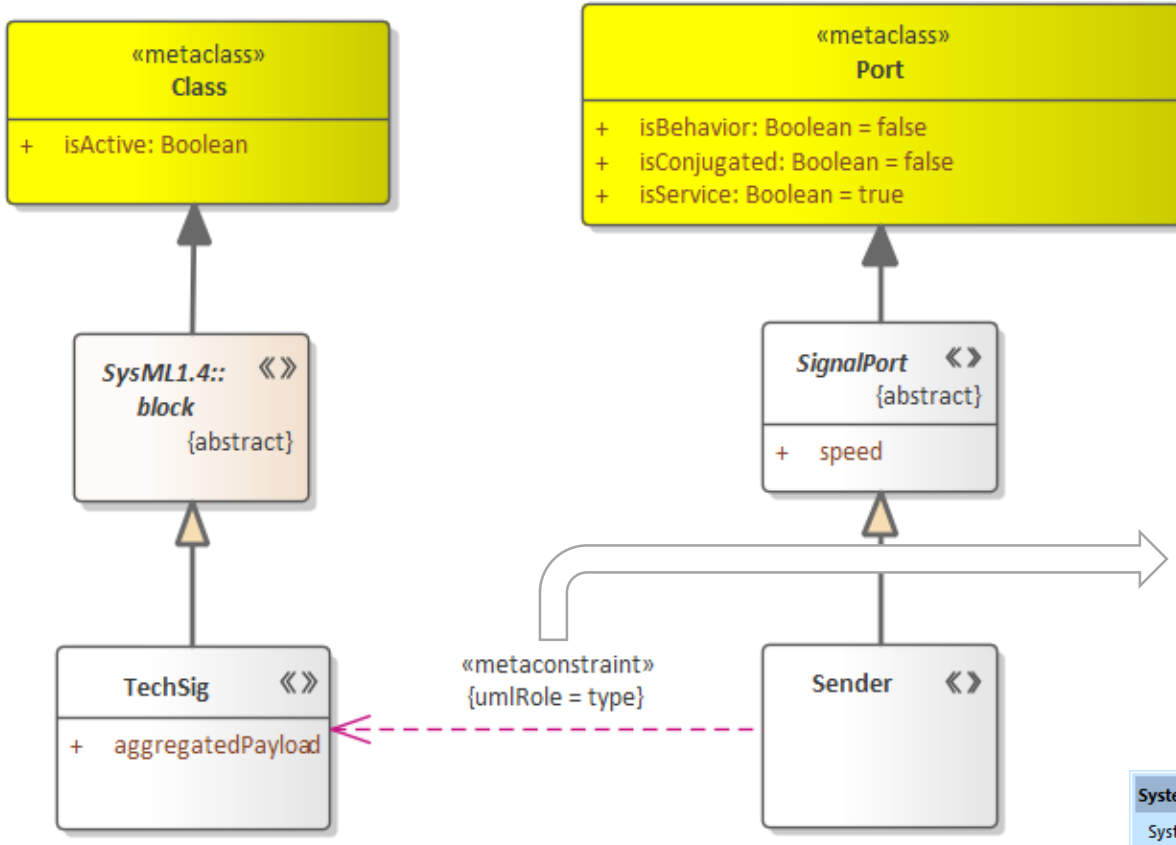


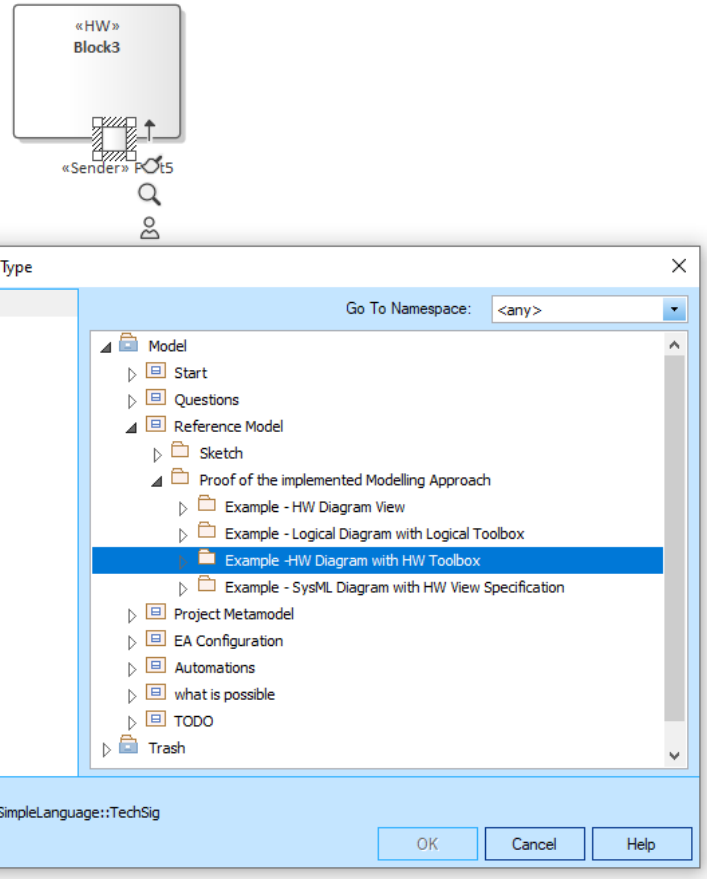
Other MetaModel Constraints

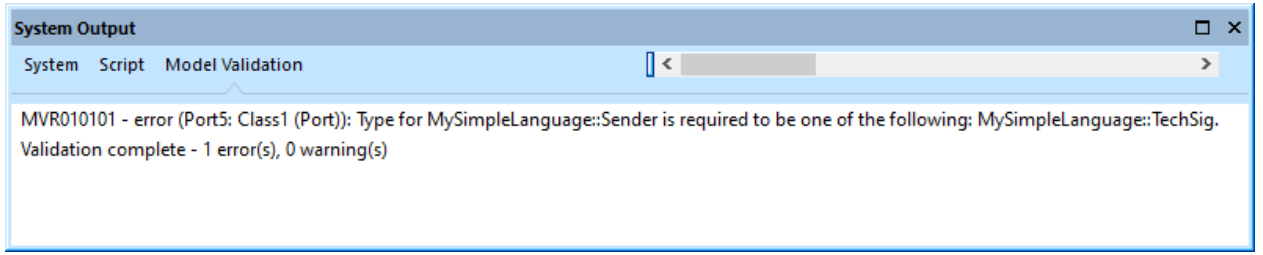




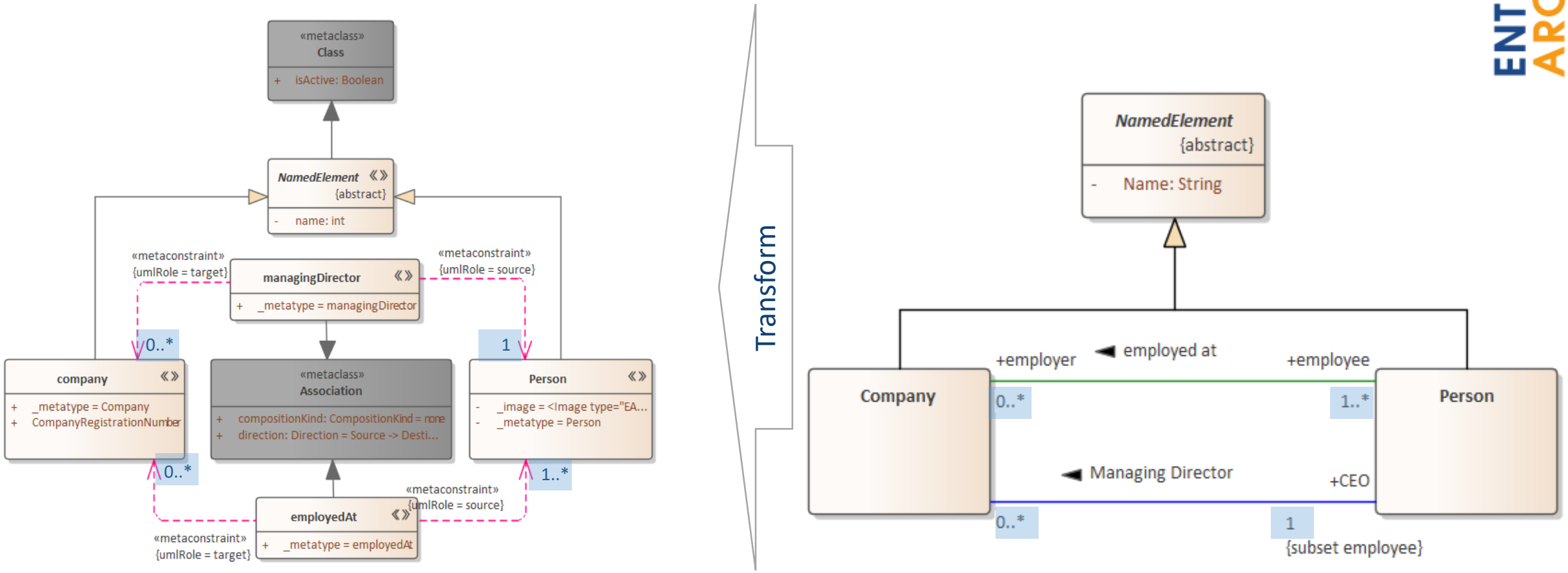
Other MetaModel Constraints





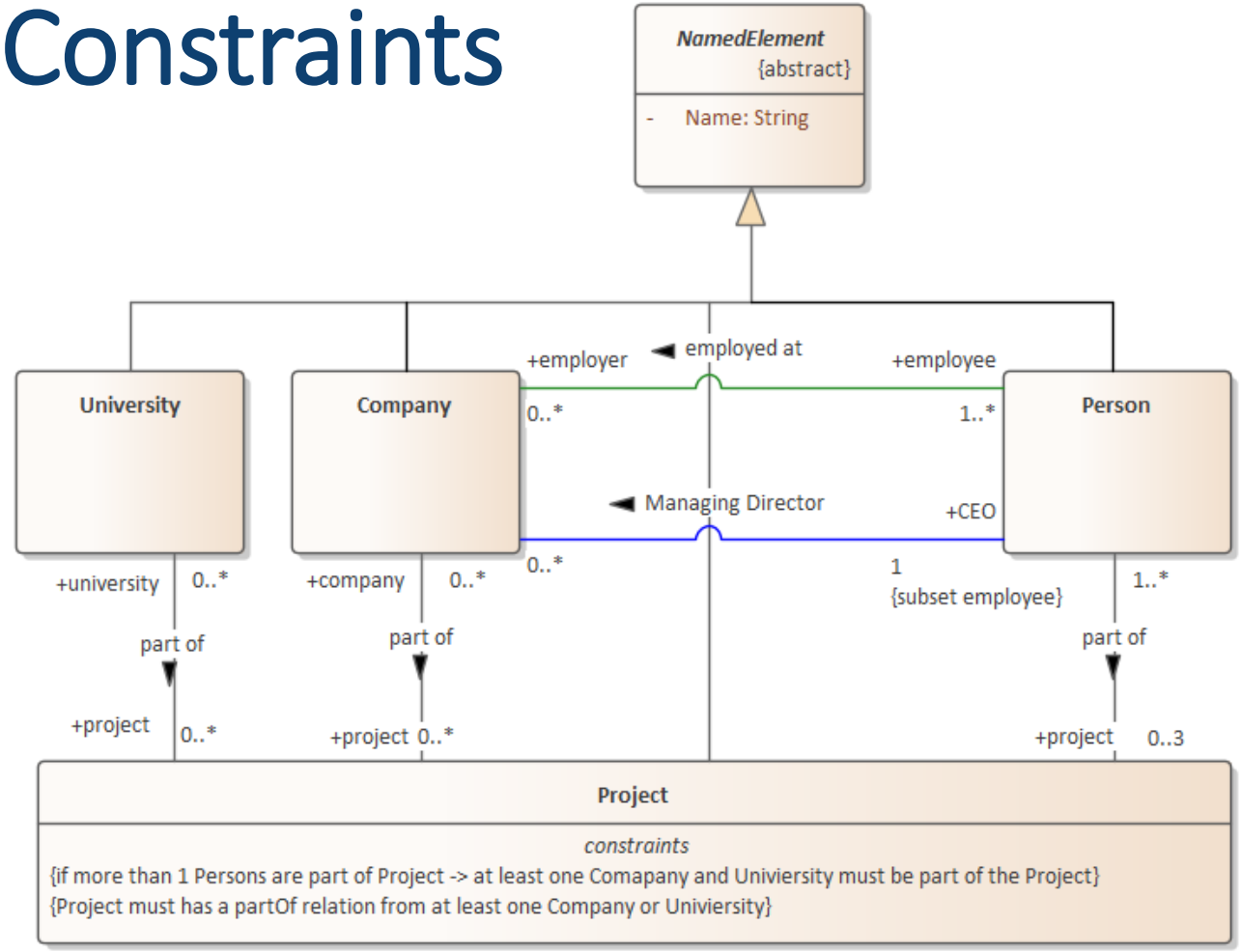


Additional Constraints



But what about constraints which cannot be expressed in a MetaModel?

Additional Constraints



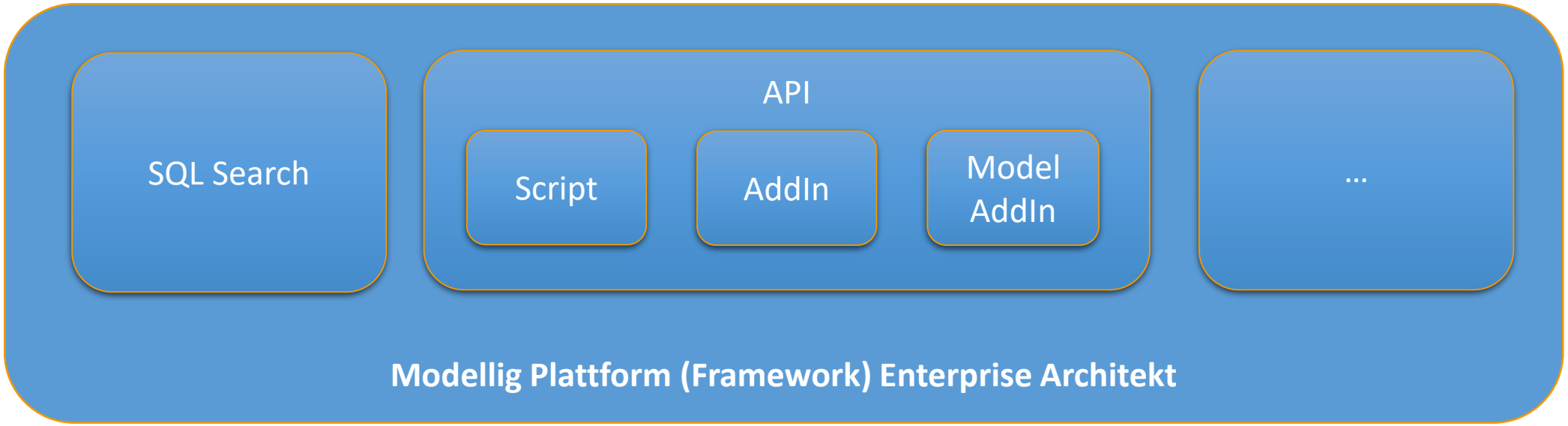
But what about constraints which cannot be expressed in a MetaModel?

ENTERPRISE ARCHITECT

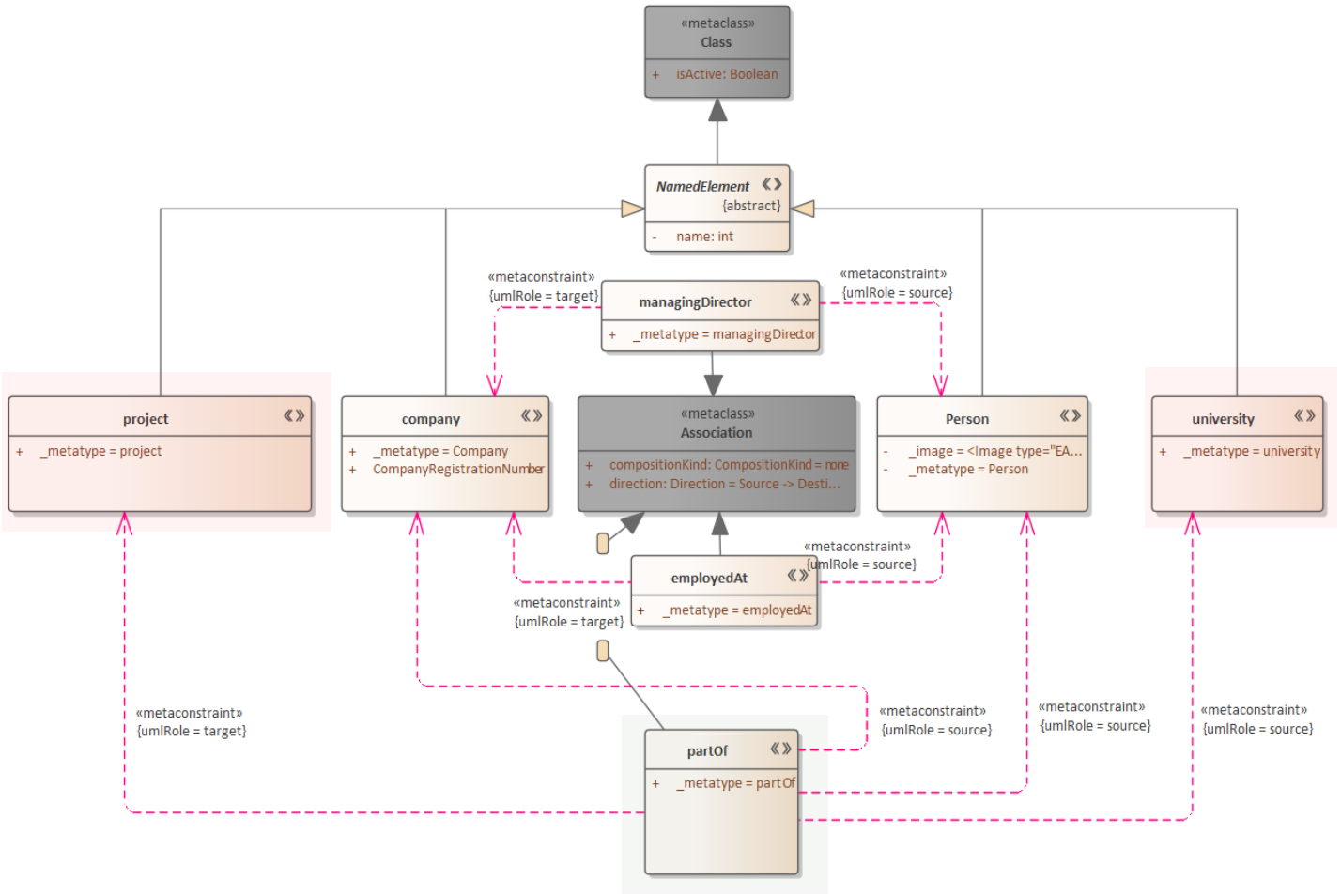
Additional Constraints

But what about constraints which cannot be expressed in a MetaModel?

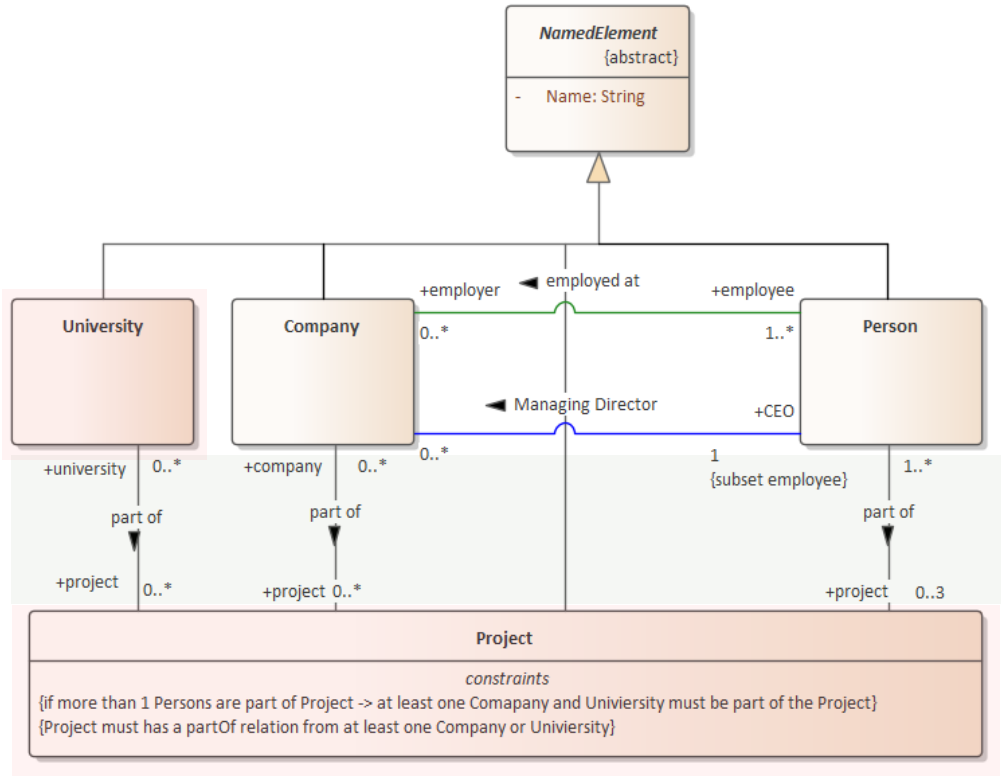
We can use the Platform (Framework) Features of EA in order to customize EA to our needs.



Translate the Extended Metamodel into a Profile



Transform

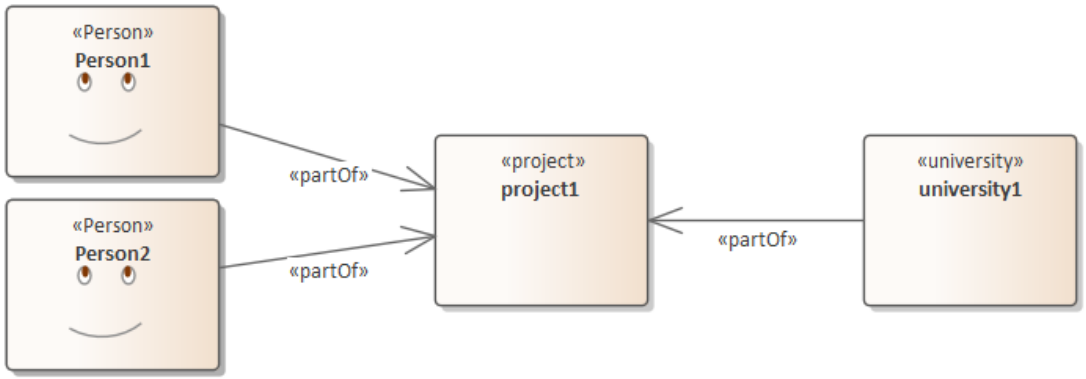


Additional Constraints in the MetaModel

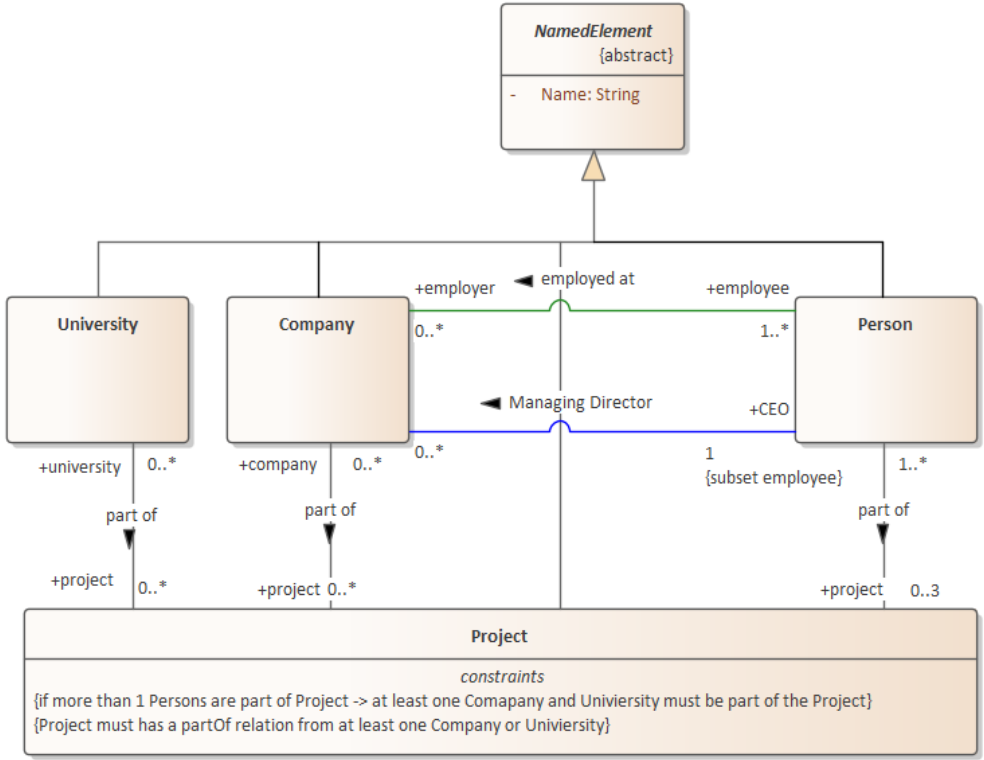
But what about constraints which cannot be expressed in a MetaModel?

Additional Constraint:

If more than one ***Person*** is ***part of*** a ***Project***, at least one ***University*** and one ***Company*** must be also ***part of*** the ***Project***.



… at lease one ***Company*** must be also ***part of*** the ***Project***



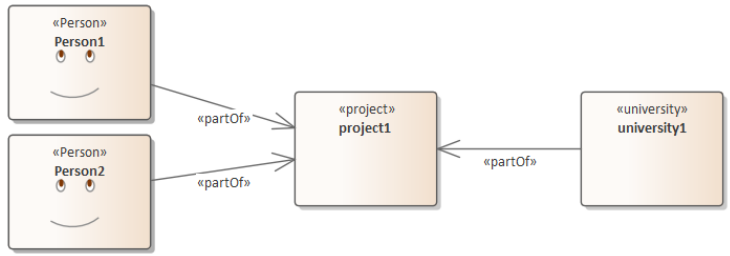
Using the EA API to define Validation Rules

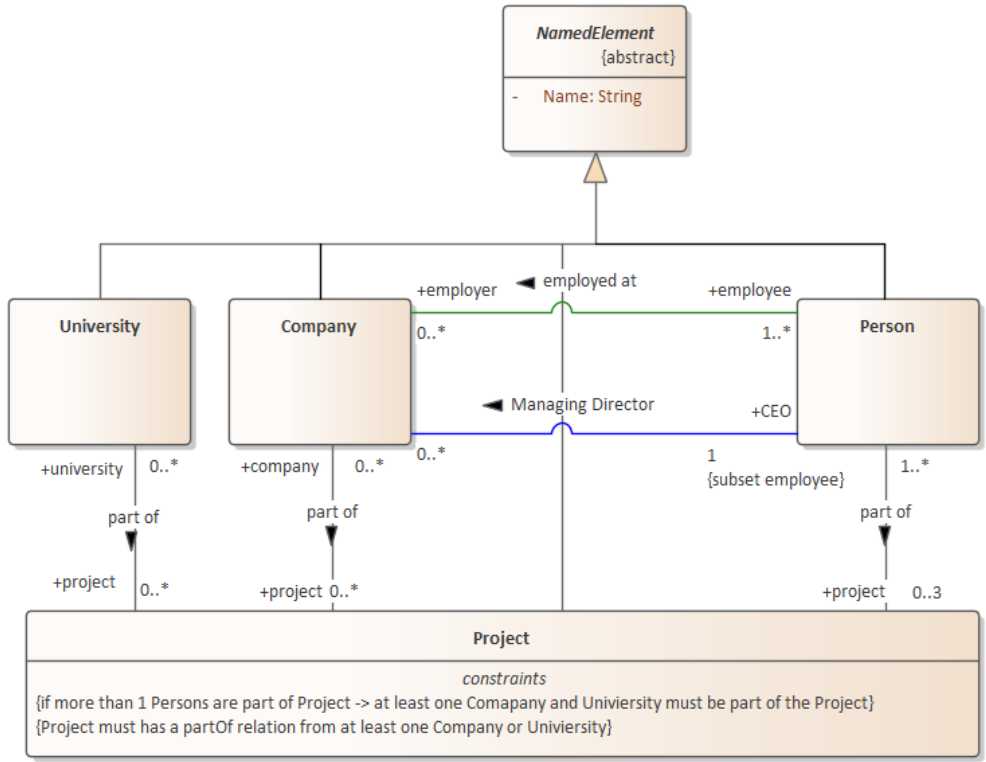
But what about constraints which cannot be expressed in a MetaModel?

```
public bool ProjectConstraintEaApi(Element self)
{
    if (self.MetaType != "project")
        return true;

    int personCount = 0;
    int universityCount = 0;
    int companyCount = 0;

    foreach(Connector connector in self.Connectors)
    {
        if (connector.MetaType != "partOf")
            continue;

        int oppositeEnd = self.ElementID == connector.ClientID ? connector.SupplierID : connector.ClientID;
        Element oppositeElement = _repo.GetElementByID(oppositeEnd); //may rais an exception -> should be in try Catch

        if(oppositeElement.MetaType == "university")
        {
            universityCount++;
        }
        else if (oppositeElement.MetaType == "company")
        {
            companyCount++;
        }
        else if (oppositeElement.MetaType == "Person")
        {
            personCount++;
        }
    }

    return personCount > 1 ? universityCount >= 1 && companyCount >= 1 : true;
}
```

Additional Constraint:

If more than one ***Person*** is ***part of*** a ***Project***, at least one ***University*** and one ***Company*** must be also ***part of*** the ***Project***.





Search for ***EA API*** at: https://blog.sparxsystems.de/

Search for ***Model Validation Example*** at: https://www.sparxsystems.com

Using Extensions to the EA API

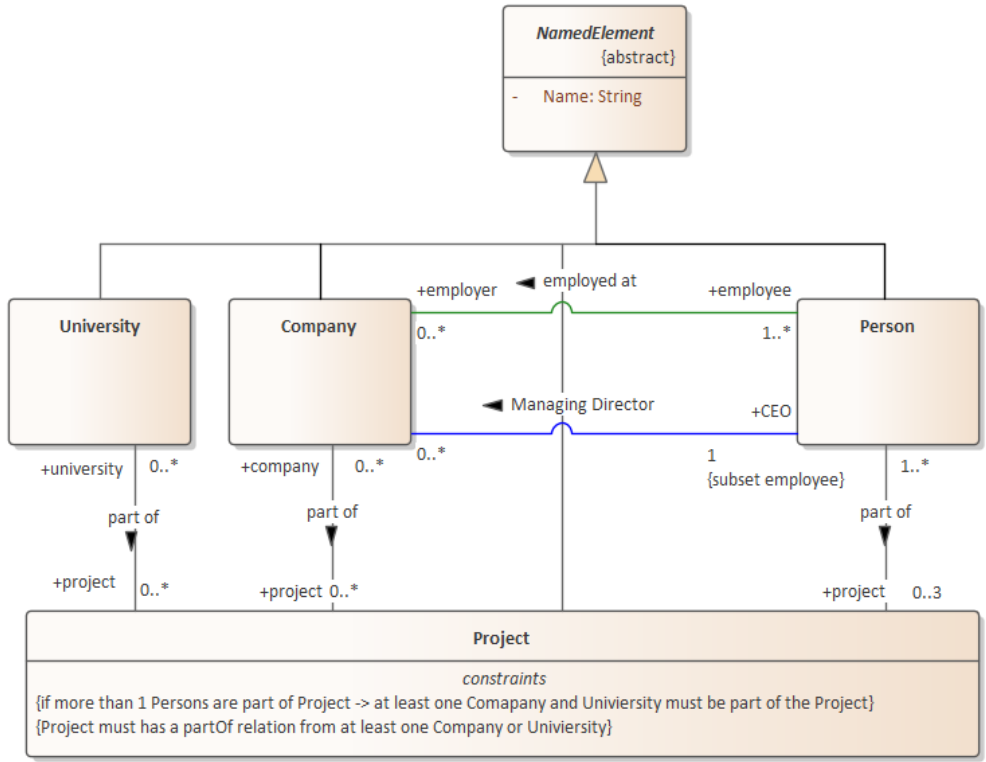
But what about constraints which cannot be expressed in a MetaModel?

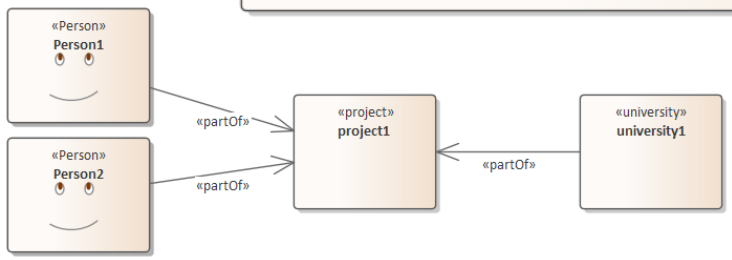
Additional Constraint:

If more than one ***Person*** is ***part of*** a ***Project***, at least one ***University*** and one ***Company*** must be also ***part of*** the ***Project***.

```
public bool ProjectConstraintExtendedApi(Element self)
{
    if (self.MetaType != "project")
        return true;

    return (self.ConnectorWithType("partOf").ToType("Person").Count() >= 1).Implies(
        (self.ConnectorWithType("partOf").ToType("university").Count() >= 1)
        && (self.ConnectorWithType("partOf").ToType("company").Count() >= 1));
}
```

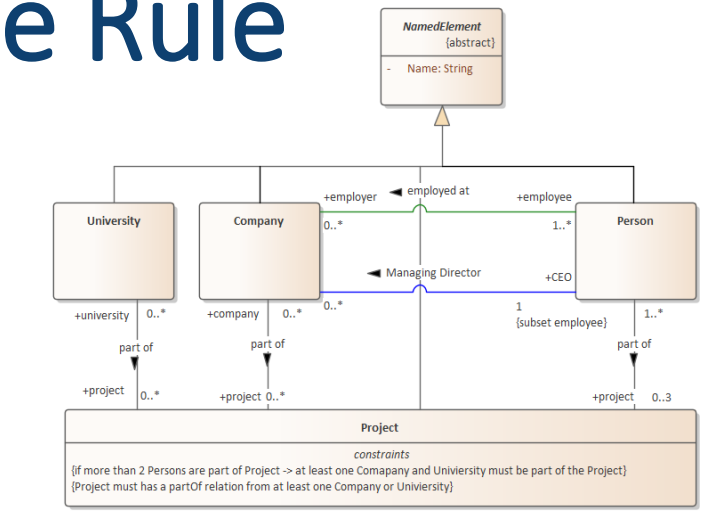
Fluent API with additional OCL like operations





A more Integrated way to Define the Rule

But what about constraints which cannot be expressed in a MetaModel?



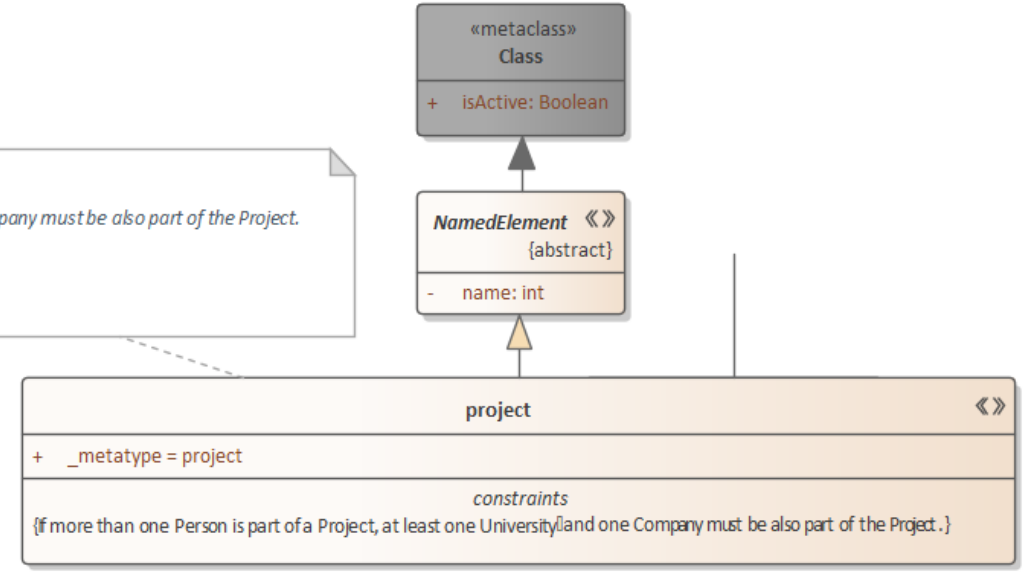
Additional Constraint:

If more than one ***Person*** is ***part of*** a ***Project***, at least one ***University*** and one ***Company*** must be also ***part of*** the ***Project***.

Part of derived UML Profile

```
public bool ProjectConstraintExtendedApi(Element self)
{
    if (self.MetaType != "project")
        return true;

    return (self.ConnectorWithType("partOf").ToType("Person").Count() >= 1).Implies(
        (self.ConnectorWithType("partOf").ToType("university").Count() >= 1)
        && (self.ConnectorWithType("partOf").ToType("company").Count() >= 1));
}
```

«Invariant»
{If more than one Person is part of a Project, at least one Universityand one Company must be also part of the Project.
: (self.ConnectorWithType("partOf").ToType("Person").Count() >= 1).Implies(
(self.ConnectorWithType("partOf").ToType("university").Count() >= 1)
&& (self.ConnectorWithType("partOf").ToType("company").Count() >= 1))}



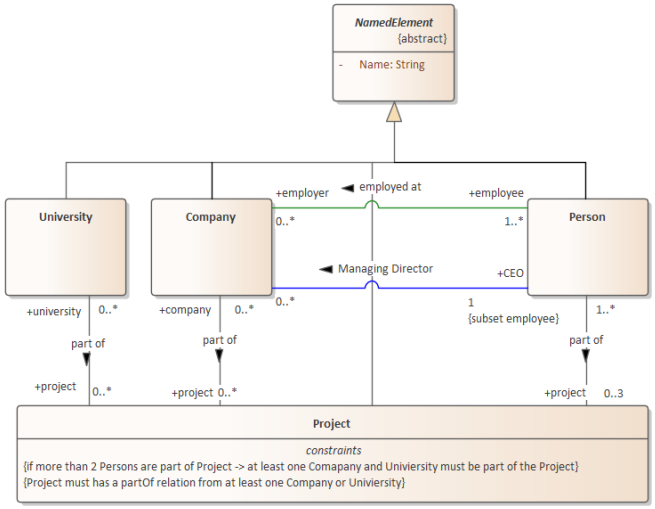
Generated, compiled and executed during EA's Model Validation

modeling assistants
by LieberLieber Software GmbH

Other Examples of Fluent Rules

Plain EA API

```
public bool DuplicatedPartOfTest(Element self)
{
    bool result = false;
    var connectorList = new Dictionary<string, int>();

    foreach (Connector connector in self.Connectors)
    {
        if (connector.MetaType != "partOf")
            continue;

        string key = connector.ClientID + connector.MetaType + connector.SupplierID;
        if (connectorList.ContainsKey(key))
        {
            connectorList[key]++;
            result = true;
        }
        else
        {
            connectorList.Add(key, 1);
        }
    }

    return result;
}
```



Plain EA API + C# LINQ + Custom Extensions

```
public bool DuplicatedPartOfTest2(Element self)
{
    //group by ElementId //wehre group contains more than one, skip the group with 1
    //re-expand the group and return the duplicate
    return ! self.ConnectorWithType("partOf").ToType("project").GroupBy(project => project.ElementID).
        Where(group => group.Skip(1).Any()).SelectMany(entry => entry).Any();
}
```

Only one ***partOf*** relation between ***Person*** and ***Project*** is allowed.



Plain EA API + C# LINQ + Custom Extensions for Duplicated

```
public bool DuplicatedPartOfTest3(Element self)
{
    //more easy solution with a custom Duplicated operation
    return ! self.ConnectorWithType("partOf").Duplicated().ToType("project").Any();
}
```

Possibilities of the EA API

		Performance	UI	Deployment	Event
COM, C#, Delphi, VB	Add In	++	+++	Installer	+++
JavaScript	Script	- -	-	MDG (-)	-
	Model Add In	- -	0	Modell/Template	+++
	Scriptlet	- -	0	Model/Template	+

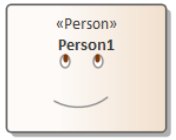
ENTERPRISE ARCHITECT
SPARX SYSTEMS CENTRAL EUROPE

What we can do with UML Profiles in EA

- Extend the Language UML with additional Type
  - Elements and Connectors
  - Additional Properties as Tagged Values



- Define alternate graphical representations.



- Validate the model based on constraint Metamodel Rules.

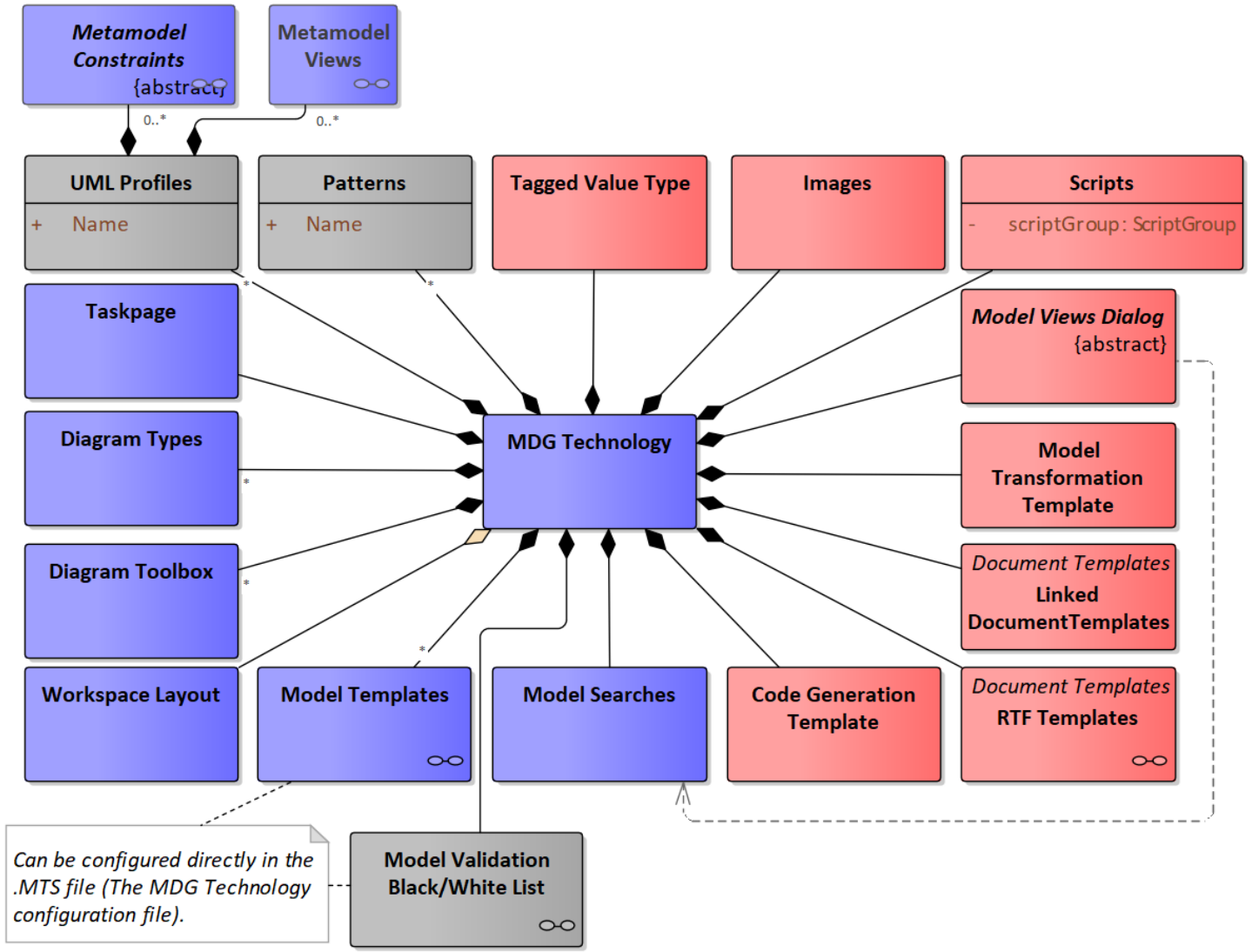
ENTERPRISE ARCHITECT

MDG Technologies in EA

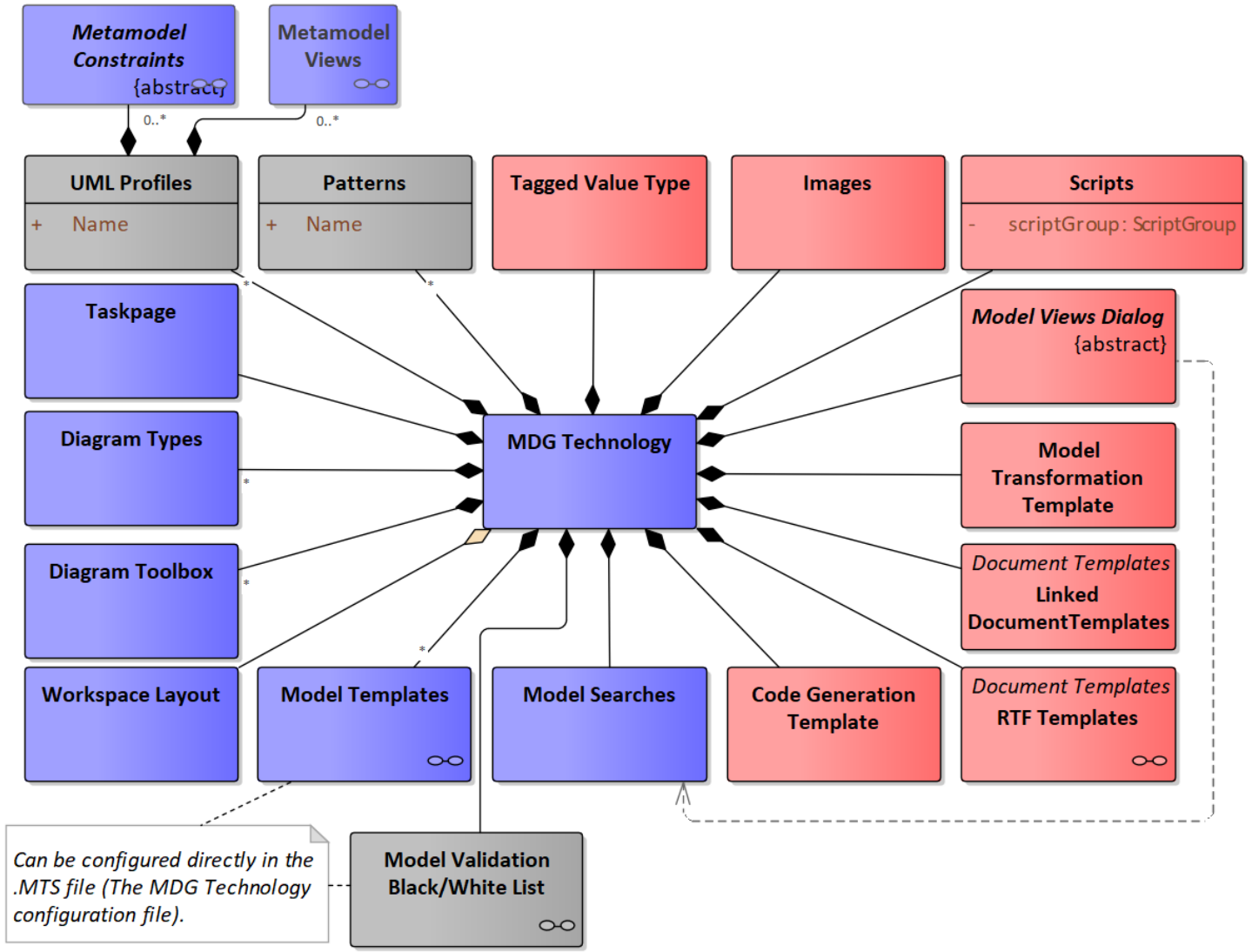


ENTERPRISE ARCHITECT

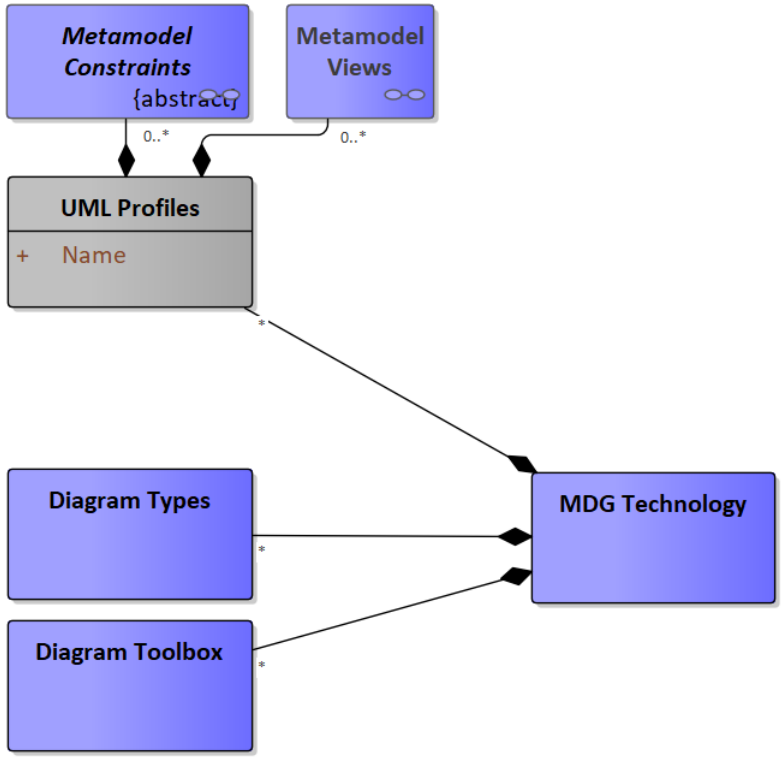
MDG Technologies in EA



MDG Technologies in EA



MDG Technologies in EA



ENTERPRISE ARCHITECT

What we have learned

- ✓MetaModels are defined using UML Profiles in Enterprise Architect.
- ✓Constraint Metamodel provide the possibility to:
  - ✓Automatically get adopted UI support based on the constraints.
  - ✓Automatic model validation based on constraints in the UML Profile.
- ✓Possibility to add custom validation rules with the help of the Framework API.
  - ✓Additional extensions of the API will make it easier for you.

Look for the next Edition of the



modeling assistants

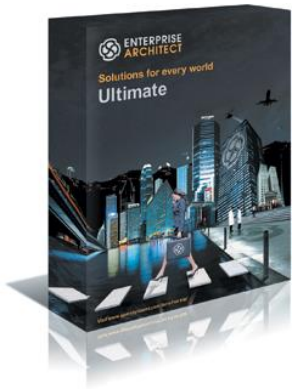
by LieberLieber Software GmbH



ENTERPRISE ARCHITECT

We are happy to support you to bring
your modelling approach alive in EA.

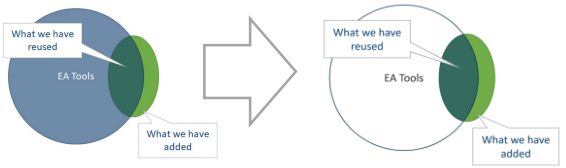
Recommendation: ***How to Succeed with your Modelling Approach in EA***



Part 1: *About the benefit of a model based specification and how you find a suitable Modelling Approach.*

Part 2: Support the Modelling Approach with MetaModels, Constraints and Configurations.

Part 3: Support the Modelling Approach with Wizards and Automation.



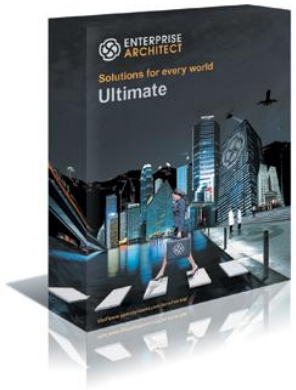
Dr. Horst Kargl
Horst.Kargl@sparxsystems.eu
vendor@sparxsystems.eu



ENTERPRISE ARCHITECT

We are happy to support you to bring
your modelling approach alive in EA.

Recommendation: ***How to Succeed with your Modelling Approach in EA***

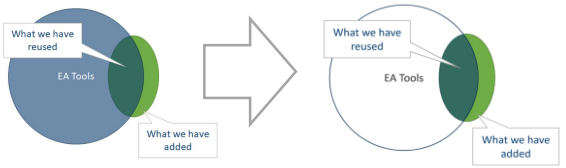


Part 1: *About the benefit of a model based specification and how you find a suitable Modelling Approach.*

Part 1.5*: About Meta-Models and MetaModelling.*

Part 2: Support the Modelling Approach with MetaModels, Constraints and Configurations.

Part 3: Support the Modelling Approach with Wizards and Automation.



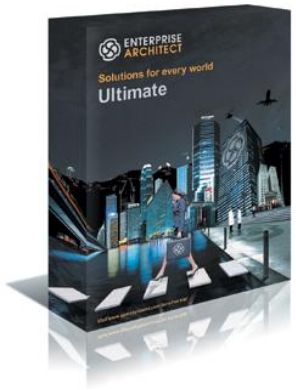
Dr. Horst Kargl
Horst.Kargl@sparxsystems.eu
vendor@sparxsystems.eu



ENTERPRISE ARCHITECT

We are happy to support you to bring
your modelling approach alive in EA.

Recommendation: ***How to Succeed with your Modelling Approach in EA***



Part 1: *About the benefit of a model based specification and how you find a suitable Modelling Approach.*
Part 1.5*: About Meta-Models and MetaModelling.*
Part 2: Support the Modelling Approach with MetaModels, Constraints and Configurations.
Part 3: Support the Modelling Approach with Wizards and Automation.



Dr. Horst Kargl
Horst.Kargl@sparxsystems.eu
vendor@sparxsystems.eu