2nd Int'l Workshop on Multi-Level Modelling
September 27, 2015

FUJITSU
shaping tomorrow with you

# Practical Multi-level Modeling on MOF-compliant Modeling Frameworks

Kosaku Kimura <kimura.kosaku@jp.fujitsu.com>,
Yoshihide Nomura, Yuka Tanaka, Hidetoshi Kurihara, and Rieko Yamamoto

Fujitsu Laboratories, Kawasaki, Japan

# Outline

1. Background
   - Graphical editing tools and their plugin development
2. Problems
   - Limitation of EMF
3. Multi-level modeling on EMF
   - Using modeling patterns
4. Preliminary evaluation
5. Conclusions

# Background

FUJITSU

- Model-driven engineering (MDE) facilitates to develop various graphical editing tools
  - Extract-Transform-Load, Business Analytics, Workflow Management, …
  - Utilize code generation feature based on model transformation
- How about plugin development of the tools?
  - For third-party developers
  - Simplifying plugin development method is quite important for popularizing the tools

Need a methodology to facilitate both tools and plugins

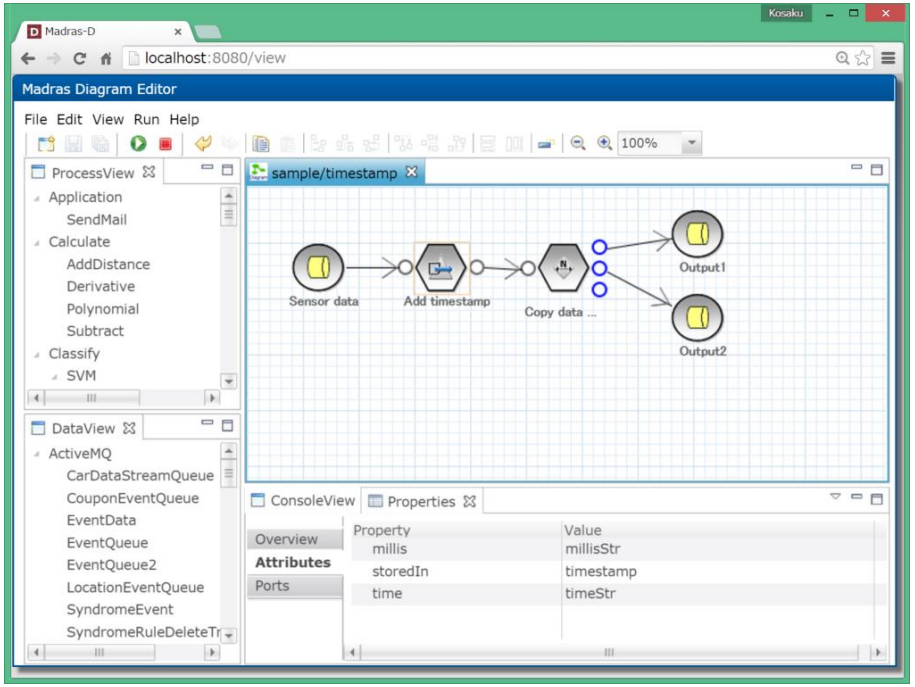
# (A part of) current state on MDE

FUJITSU

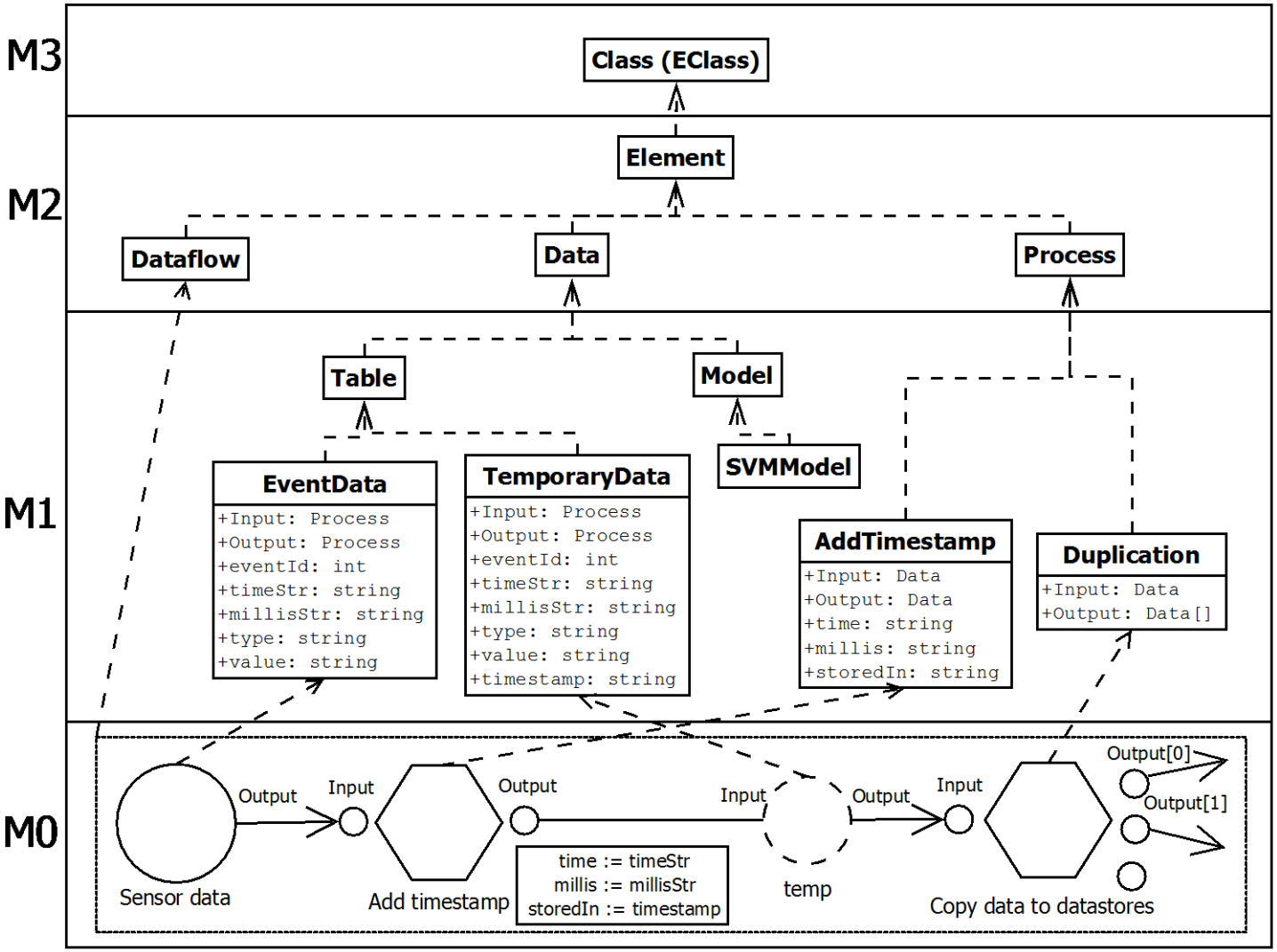
- Meta-Object Facility (MOF)
  - Object Management Group (OMG)'s standard
  - Four-layer model architecture
    - Object, Model, Metamodel, and Metametamodel
  - Various specifications for MOF are available
    - Object Constraint Language (OCL), MOFM2T, etc.
- Eclipse Modeling Framework (EMF)
  - One of mature MOF-compliant framework
  - Provides the Ecore metamodel
    - Compatible with Essential MOF
  - Various toolkits are available in the community
    - Acceleo, QVTo, ATL, etc.

# Motivating example

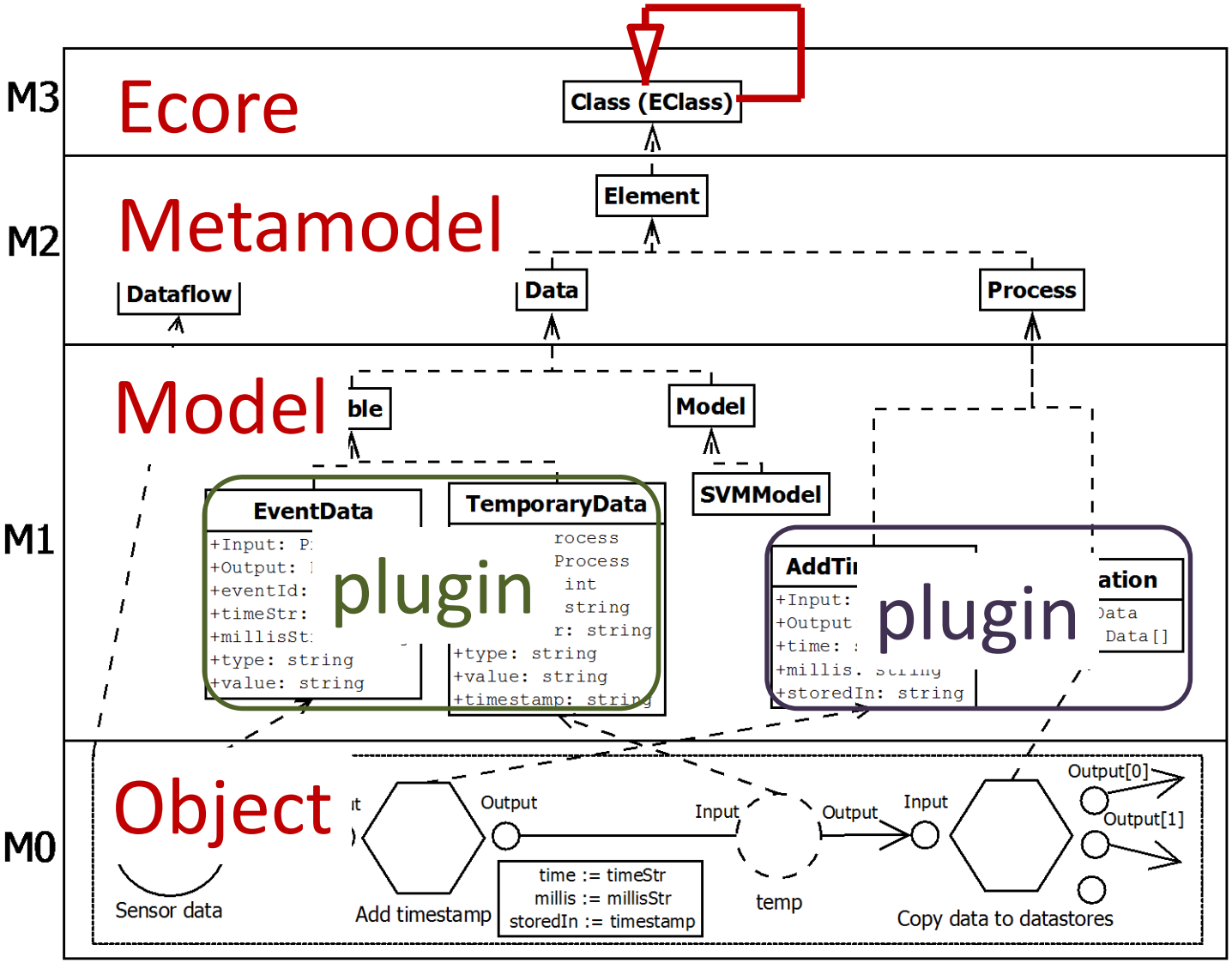
- Madras[18]: A cloud-based graphical editing tool for big data processing apps based on a dataflow model
- Developing apps on a dataflow diagram editor
- All building blocks (types of data and process nodes) are given by plugins



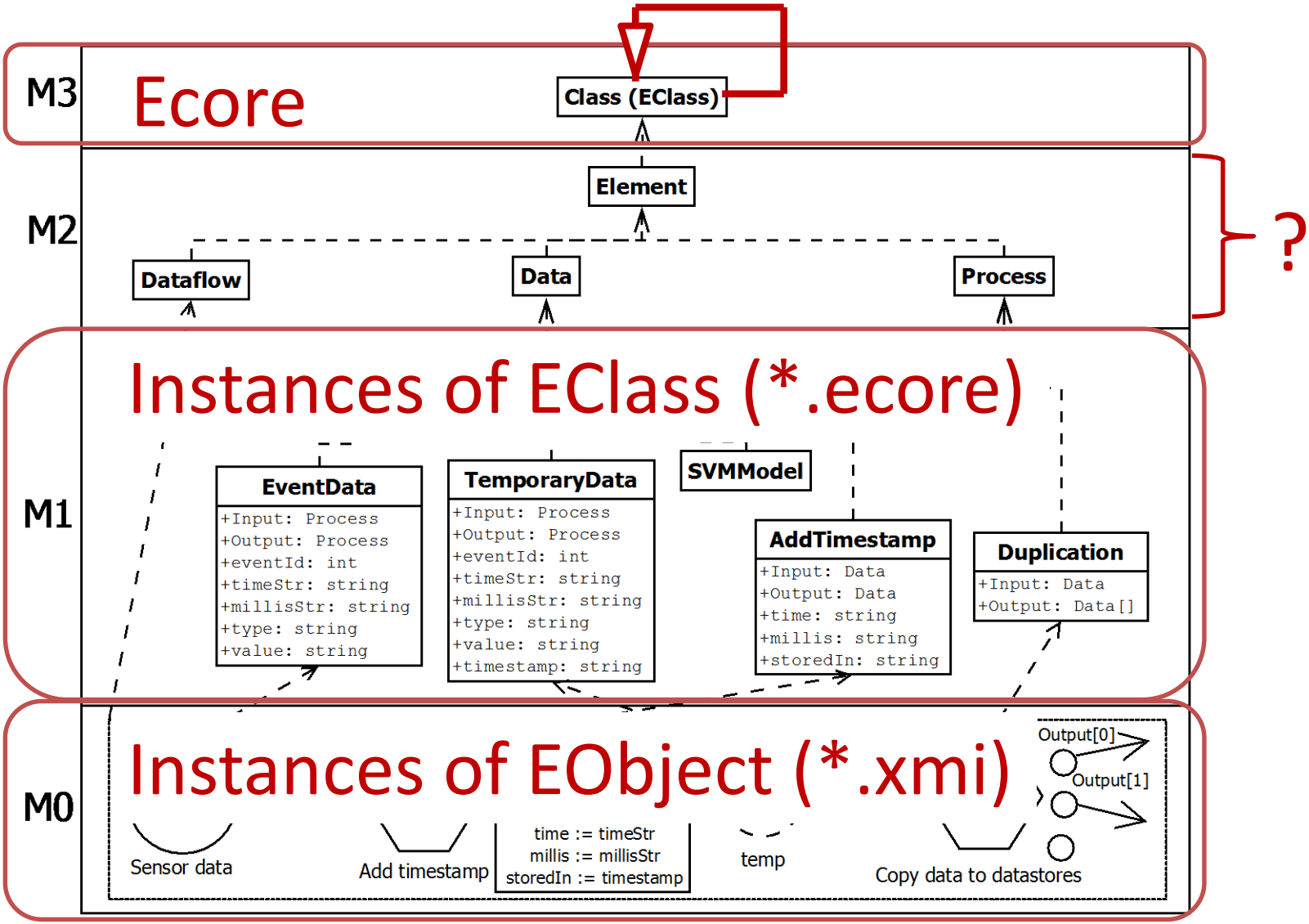
# Dataflow model on EMF



# Dataflow model on EMF



# Dataflow model on EMF



# Problem: how can we define layer M2?

FUJITSU

- Fit the four layers in the three layers of EMF
  - By a simple solution for developing tools and plugins
- Several multi-level modeling methodologies and toolkits may provide the simple solution
  - Orthogonal Classification Architecture (OCA)[6, 7, 9, 11]
  - Powertype-based metamodeling[13, 14]
  - Deep instantiation[12, 17], etc.
- However, it is premature to be locked in one of them
  - Need more discussion for achieving consensus

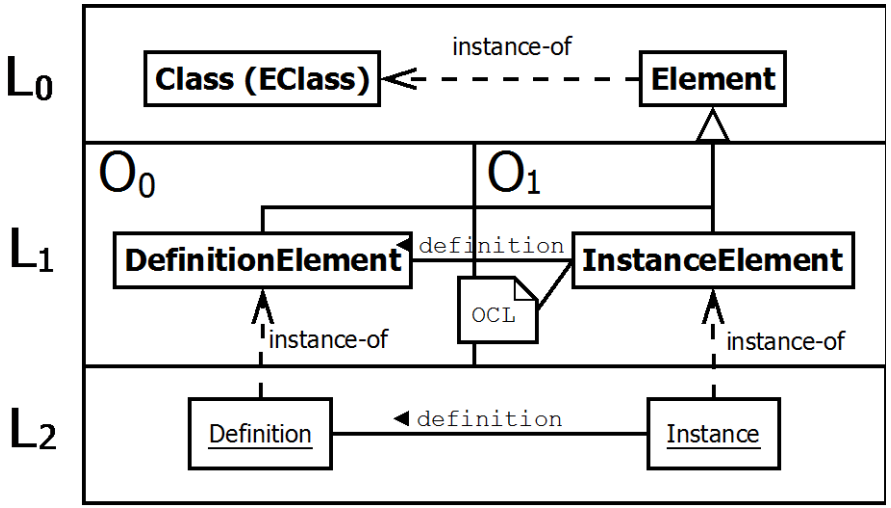
Achieve multi-level models on EMF without using any special toolkit

at the moment…

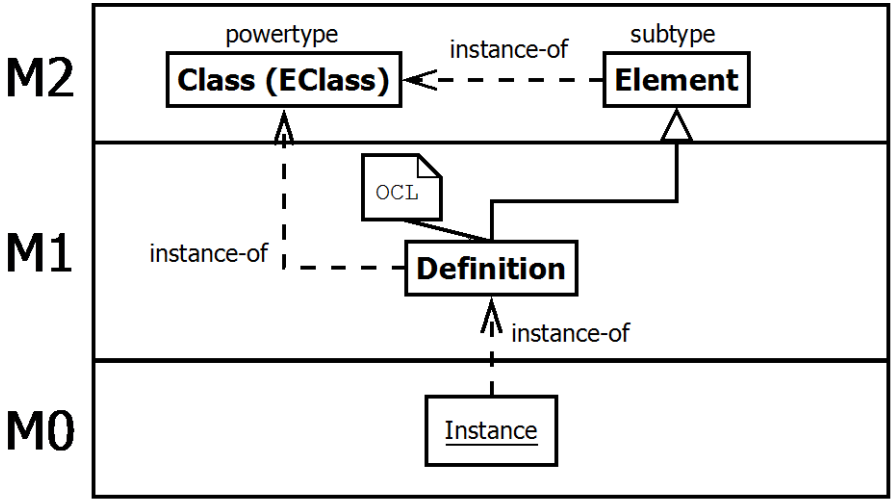
# Multi-level modeling on EMF

- **OCA** and **powertypes** can be “encoded” just on EMF
  - **Deep instantiation** introducing **clabjects** with **potency** is difficult
- Some objects have two different kinds of relations that cannot be defined simultaneously on EMF
- Basic idea: **use OCL verification** for either two relations
  - Verify their conformance passively

## OCA

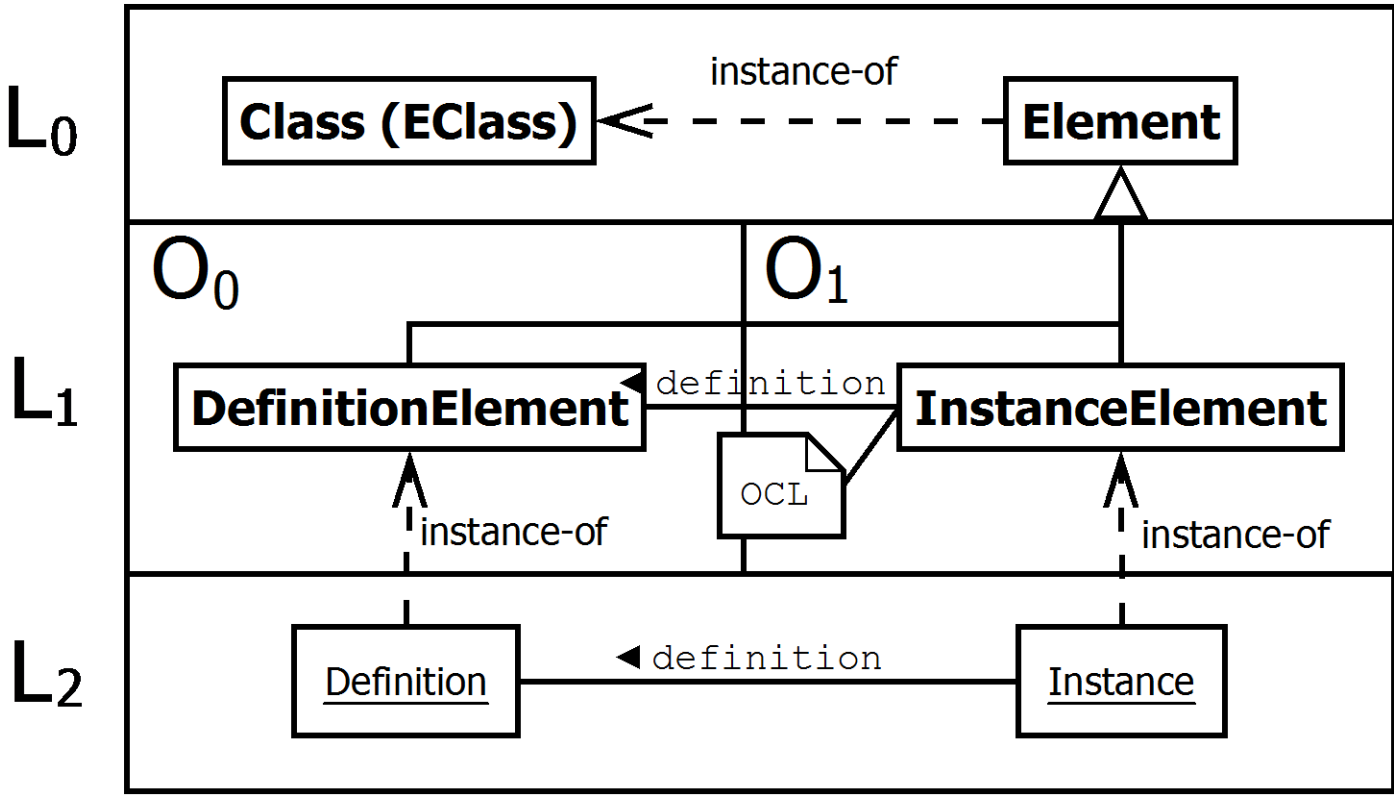



## Powertypes

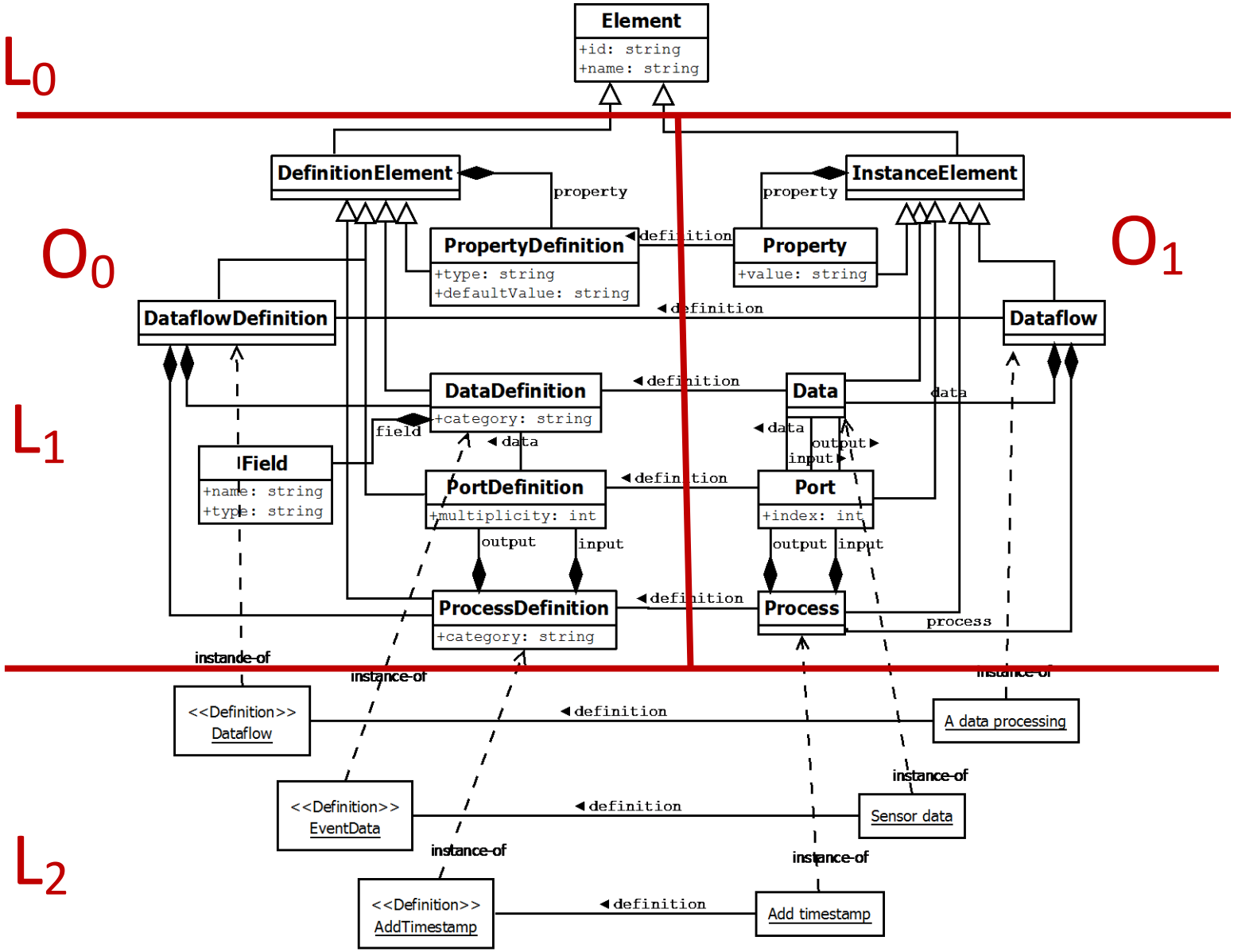


# OCA Pattern

- Two kinds of instantiation
  - Ontological and linguistic
- OCL verification for ontological instantiation relations



# Dataflow model by OCA pattern



# Dataflow model by OCA pattern

FUJITSU

$L_2$

L2_Instance_oca.xmi

- platform:/resource/com.thirdparty.app/model/L2_Instance_oca.xmi
  - Dataflow
    - Data Sensor data
    - Data temp
    - Data Output1
    - Data Output2
    - Process Add timestamp
      - Property time
      - Property millis
      - Property storedIn
      - Port in
      - Port out
    - Process Copy data to datastore
- platform:/resource/com.fujitsu.paas.model.oca/model/dataflow_oca.ecore
- platform:/resource/com.thirdparty.app/model/L2_Definition_oca.xmi
  - Dataflow Definition
    - Data Definition EventData
      - Field eventId
      - Field timeStr
      - Field millisStr
      - Field type
      - Field value
    - Data Definition TemporaryData
    - Process Definition AddTimestamp
      - Property Definition time
      - Property Definition millis
      - Property Definition storedIn
      - Port Definition in
      - Port Definition out
    - Process Definition Duplication
- platform:/resource/com.fujitsu.paas.model.oca/model/dataflow_oca.ocl

Instance

Definition

dataflow_oca.ecore

- platform:/resource/com.fujitsu.paas.model.oca/model/dataflow_oca.ecore
  - dataflow_oca
    - Element
    - DefinitionElement -> Element
    - PropertyDefinition -> Element
    - DataflowDefinition -> DefinitionElement
    - ProcessDefinition -> DefinitionElement
    - PortDefinition -> DefinitionElement
    - DataDefinition -> DefinitionElement
    - Field
    - InstanceElement -> Element
    - Property -> Element
    - Dataflow -> InstanceElement
    - Process -> InstanceElement
    - Port -> InstanceElement
    - Data -> InstanceElement

$L_0$

$O_0$

$O_1$

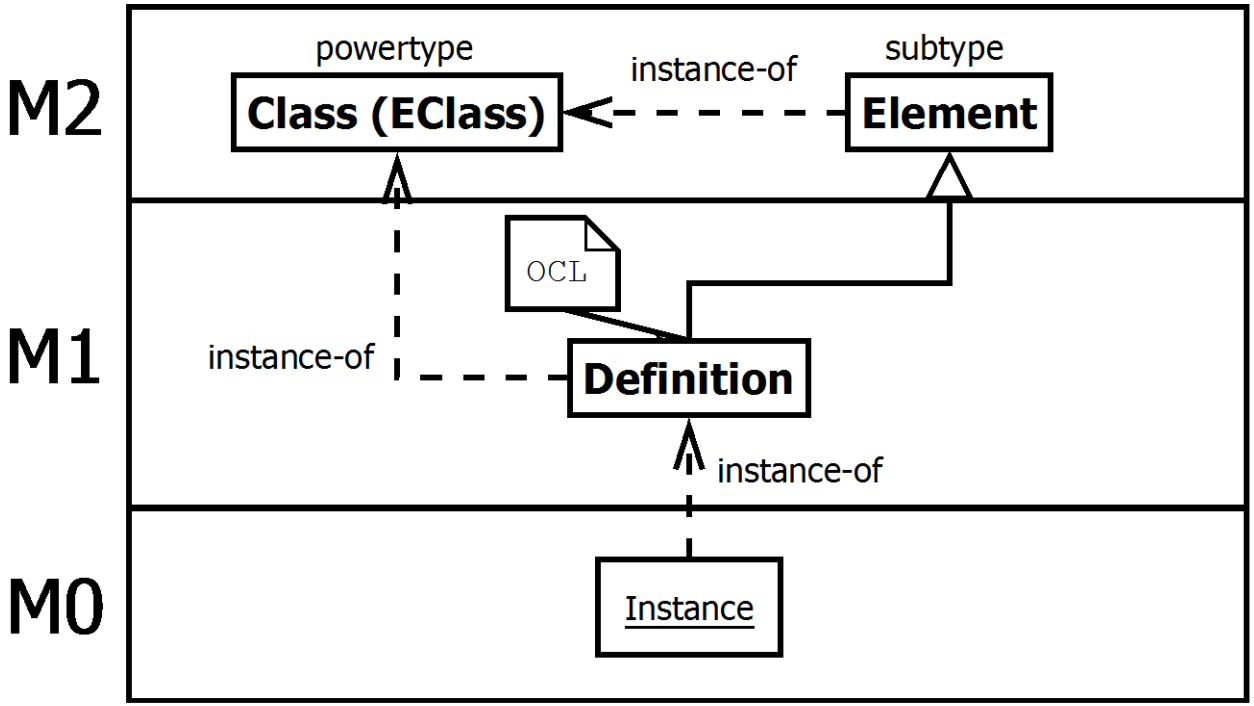
dataflow_oca.ocl

```
import 'platform:/resource/com.fujitsu.paas.model.oca/model/dataflow_oca.ecore'
package dataflow_oca

--- for instance objects of Data
context Data
    inv DataHasDefinition: definition <> null
    inv DataHasValidProperties:
        definition.property->forAll(i | property->exists(definition = i))
    inv DataHasValidFields:
        definition.field->forAll(name <> null and type <> null)

--- for instance objects of Process
context Process
    inv ProcessHasDefinition: definition <> null
    inv ProcessHasValidProperties:
        definition.property->forAll(i | property->exists(definition = i))
    inv ProcessHasValidInputPorts:
        definition.input->forAll(i | input->exists(definition = i))
    inv ProcessHasValidOutputPorts:
        definition.output->forAll(i | output->exists(definition = i))

endpackage
```

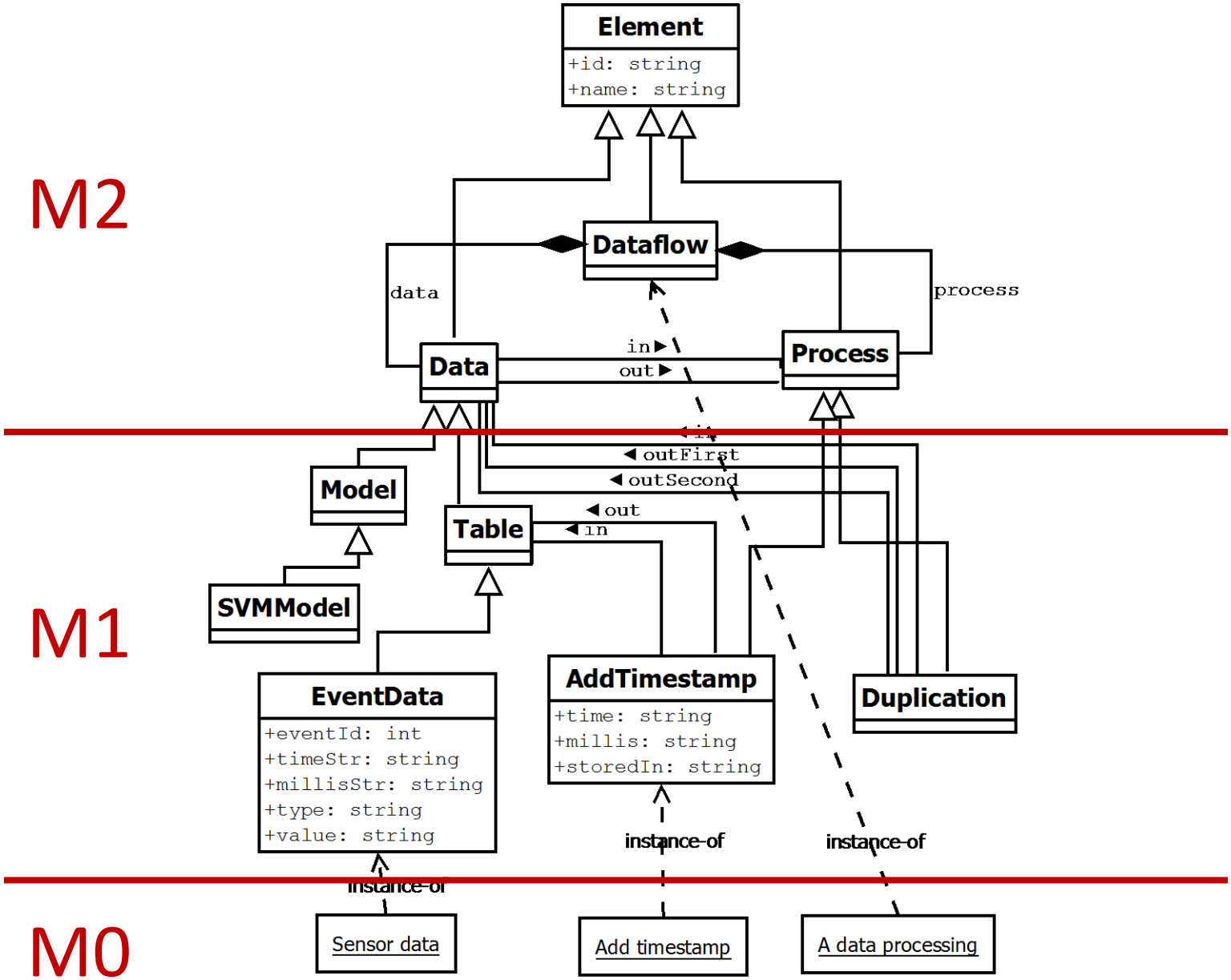
OCL for Instance

# Powertypes Pattern

- Object in layer M1 must be
  - Instance of the powertype (EClass)
  - Subclass of the subtype (Element)
- OCL verification for inheritance relations from subtype



# Dataflow model by Powertypes pattern



# Dataflow model by Powertypes pattern

FUJITSU

M0_Instance_powertype.xmi

- platform:/resource/com.thirdparty.app/moc
  - Dataflow
    - Event Data Sensor data
    - Temporary Data temp
    - Add Timestamp Add timestamp
    - Duplication Copy data to datastore
- platform:/resource/com.fujitsu.paas.model.
- platform:/resource/com.thirdparty.app/moc

M0

M1_Definition_powertype.ecore

- platform:/resource/com.thirdparty.app/model/M1_D
  - M1_Definition_powertype
    - Table -> Data
    - Model -> Data
    - SVMModel -> Model
    - EventData -> Table
      - eventId : EString
      - timeStr : EString
      - millisStr : EString
      - type : EString
      - value : EString
    - TemporaryData -> null
    - AddTimestamp -> Process
      - in : null
      - out : null
      - time : EString
      - millis : EString
      - storedIn : EString
    - Duplication -> Process
- platform:/resource/com.fujitsu.paas.model.powerty
- platform:/resource/com.fujitsu.paas.model.powerty

M1

dataflow_powertype.ecore

- platform:/resource/com.fujitsu.paas.model.powertype
  - dataflow_powertype
    - Element
    - Dataflow -> Element
    - Process -> Element
    - Data -> Element

M2

dataflow_powertype.ocl

```
import 'http://www.eclipse.org/emf/2002/Ecore#/'
package ecore
context EClass
def : isA(typeName : String) : Boolean =
    name = typeName or oclIsKindOf(EClass)
    and oclAsType(EClass).eAllSuperTypes->exists(name = typeName)

-- for subclasses of Data
inv DataHasNoExtraProcessRefs:
    isA('Data'
    ) implies eReferences->forAll(
        eReferenceType.isA('Process') implies name.matches('in|out')
    )

-- for subclasses of Process
inv ProcessHasValidInputPorts:
    isA('Process') implies eReferences->forAll(
        name.matches('^in.*') implies eReferenceType.isA('Data')
    )
inv ProcessHasValidOutputPorts:
    isA('Process') implies eReferences->forAll(
        name.matches('^out.*') implies eReferenceType.isA('Data')
    )

endpackage
```

OCL for layer M1

# Preliminary evaluation

FUJITSU

- How easy can we develop our tool and plugins by applying the modeling patterns on EMF?

A) Model manipulation for our tool

B) Template description for plugins

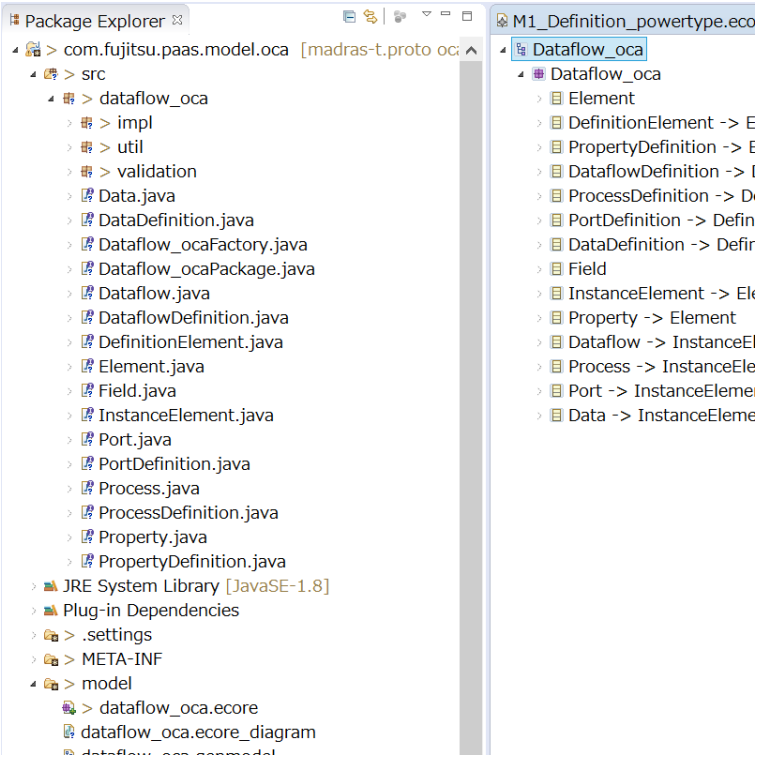
# A) Model manipulation for our tool

FUJITSU

## OCA pattern

☺ Can utilize code generation feature of EMF

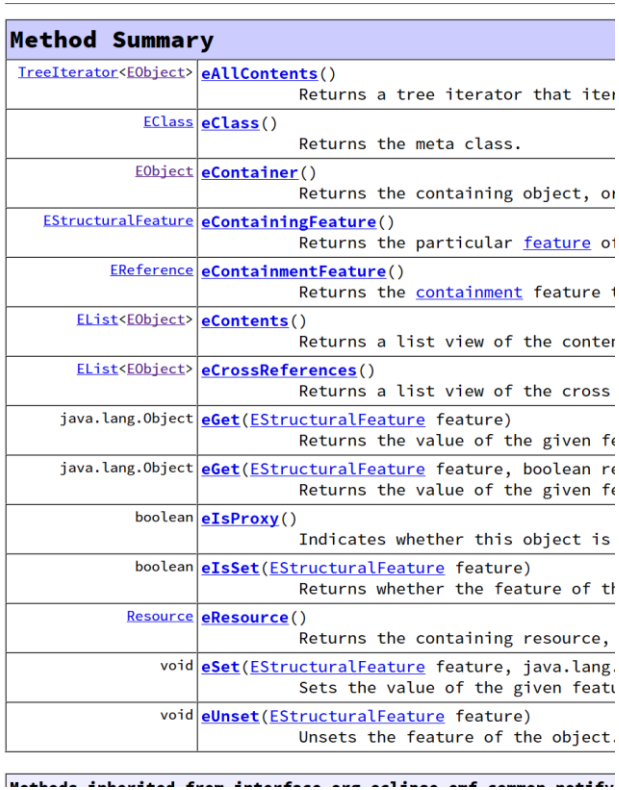
- Not only model code but also editor and test code




## Powertypes pattern

☹ Need to manipulate models by using Ecore APIs

- Require high skills for using them



**Method Summary**

| | |
|---|---|
| TreeIterator<EObject> | eAllContents() Returns a tree iterator that ite |
| EClass | eClass() Returns the meta class. |
| EObject | eContainer() Returns the containing object, o |
| EStructuralFeature | eContainingFeature() Returns the particular feature o |
| EReference | eContainmentFeature() Returns the containment feature |
| EList<EObject> | eContents() Returns a list view of the conte |
| EList<EObject> | eCrossReferences() Returns a list view of the cross |
| java.lang.Object | eGet(EStructuralFeature feature) Returns the value of the given f |
| java.lang.Object | eGet(EStructuralFeature feature, boolean r Returns the value of the given f |
| boolean | eIsProxy() Indicates whether this object is |
| boolean | eIsSet(EStructuralFeature feature) Returns whether the feature of t |
| Resource | eResource() Returns the containing resource, |
| void | eSet(EStructuralFeature feature, java.lang. Sets the value of the given feat |
| void | eUnset(EStructuralFeature feature) Unsets the feature of the object. |

# B) Template description for plugins

FUJITSU

## OCA pattern

☹ Too many indirect expressions

generate_oca.mtl

```
[comment encoding = UTF-8 /]
[module generate_oca('platform:/resource/com.fujitsu.paas.model.
[template public generate(aProcess : Process)
? (definition.name='AddTimestamp')]
[comment @main/]
insert into [output->any(definition.name='out').data.name/]
select [for (input->any(definition.name='in')
.data.definition.field) separator(',')][name/][/for]
, UDF.timestamp(
[property->any(definition.name='time').value/],
[property->any(definition.name='millis').value/]
) as
[property->any(definition.name='storedIn').value/]
from [input->any(definition.name='in').data.name/]
[/template]
```

## Powertypes pattern

☺ Simple access to attributes

generate_powertype.mtl

```
[comment encoding = UTF-8 /]
[module generate_powertype('platform:/resource/com.thirdparty.app/model
[template public generate(aProcess : AddTimestamp)]
[comment @main/]
insert into [out.name/]
select [for (_in.eClass().eAttributes) separator(',')][name/][/for]
, UDF.timestamp([time/],[millis/]) as [storedIn/] from [_in.name/];
[/template]
```

```
insert into <Output>
select <Field of Input>[,<Field of Input> ...],
UDF.timestamp(<time>, <millis>) as <storedIn>
from <Input>
```

# Quick summary of the evaluation

FUJITSU

- How easy can we develop our tool and plugins by applying the modeling patterns on EMF?

A) Model manipulation for **our tool**

B) Template description for **plugins**

| | OCA | Powertypes |
|---|---|---|
| A) | ☺ | ☹ |
| B) | ☹ | ☺ |

# Conclusions

- Multi-level modeling on EMF
  - OCA pattern
  - Powertypes pattern
- Preliminary evaluation using the dataflow model
  - Found trade-off between the patterns
- We recommend to prioritize ease of plugin development over ease of developing our tool

# Future work

- Further evaluation of the patterns
- Migrate a multi-level modeling toolkit that will be well standardized

FUJITSU

shaping tomorrow with you