3rd Int'l Workshop on Multi-Level Modelling
October 4, 2016

FUJITSU
shaping tomorrow with you

An Evaluation of Multi-Level Modeling Frameworks for Extensible Graphical Editing Tools

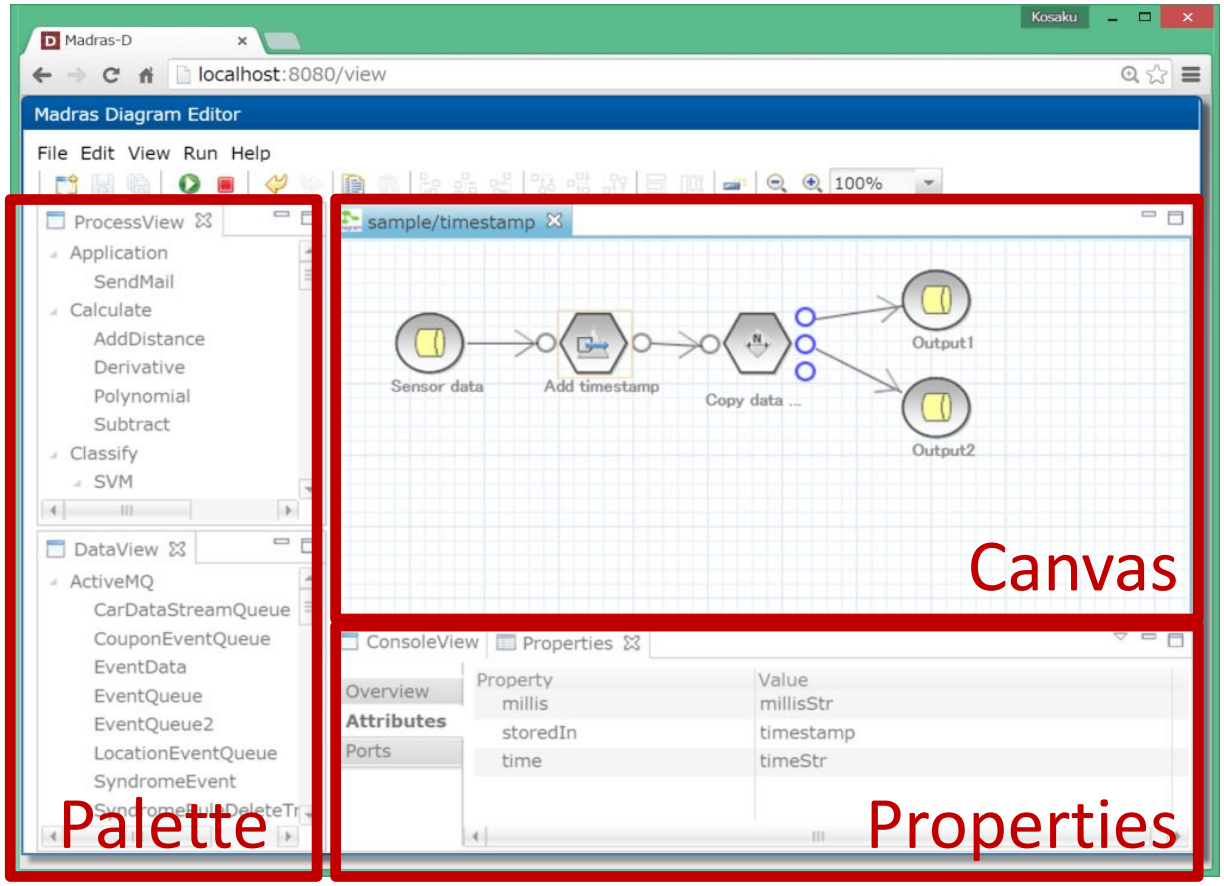
Kosaku Kimura, Fujitsu Labs
Kazunori Sakamoto, National Institute of Informatics

Background - Graphical Editing Tool (GET)

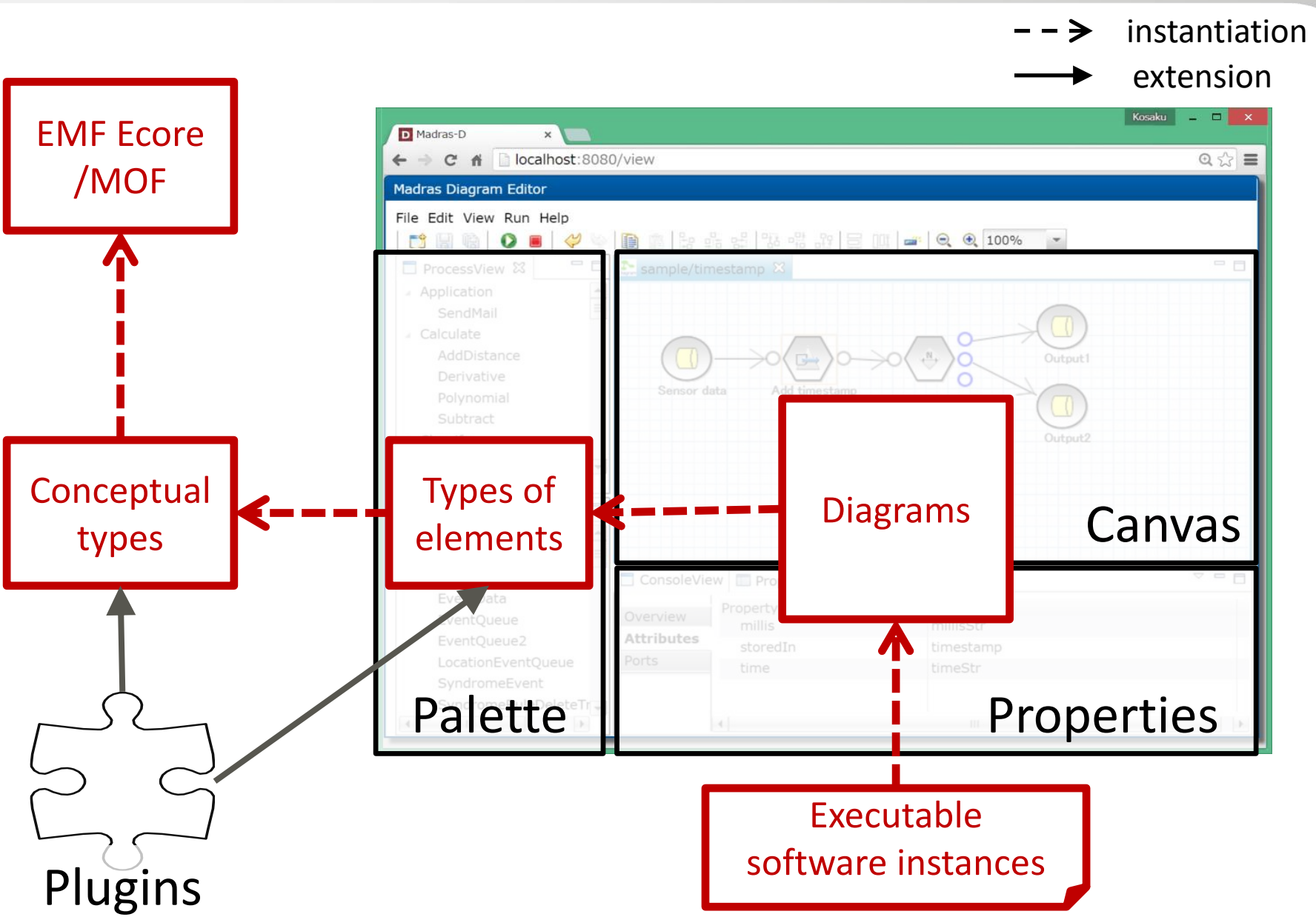
FUJITSU

- Used as an approach of abstraction and automation for developing complex software with less effort
  - Define software as a diagram editing on the canvas
  - Generate and execute a software instance from a diagram model
- Encapsulates information to help software developers to concentrate what really matters for their work
- In order to facilitate to develop GET, various mature MDE tools (e.g., model transformation, verification, repository, visualization) should be utilized

Background - Graphical Editing Tool (GET)



Background - Graphical Editing Tool (GET)



Background - Graphical Editing Tool (GET)

FUJITSU

- Should be easily extensible by software developers
  - Adding a type of building blocks for a new function
  - Adding a conceptual type in order to describe information about a new concern in the diagrams

- Needs to access (meta-) metamodel for the extensions, but it is difficult for standardized specifications and frameworks
  - e.g., MetaObject Facility (MOF), Eclipse Modeling Framework (EMF)

Multi-Level Modeling (MLM)

FUJITSU

- Can define and manipulate arbitrary number of levels of metamodels
- Various methodologies and frameworks exist
  - Orthogonal Classification Architecture (OCA)
  - Potency-based
  - Dual deep instantiation
  - Powertype-based, etc.
- We need to have knowledge about when, where and how to use them
- Question: which MLM framework is the most appropriate for developing extensible GET?
  - Evaluate them by using a dataflow model as an example of GET models

Dataflow Model (DFM)

https://github.com/dataflow-mlm/dataflow-mlm

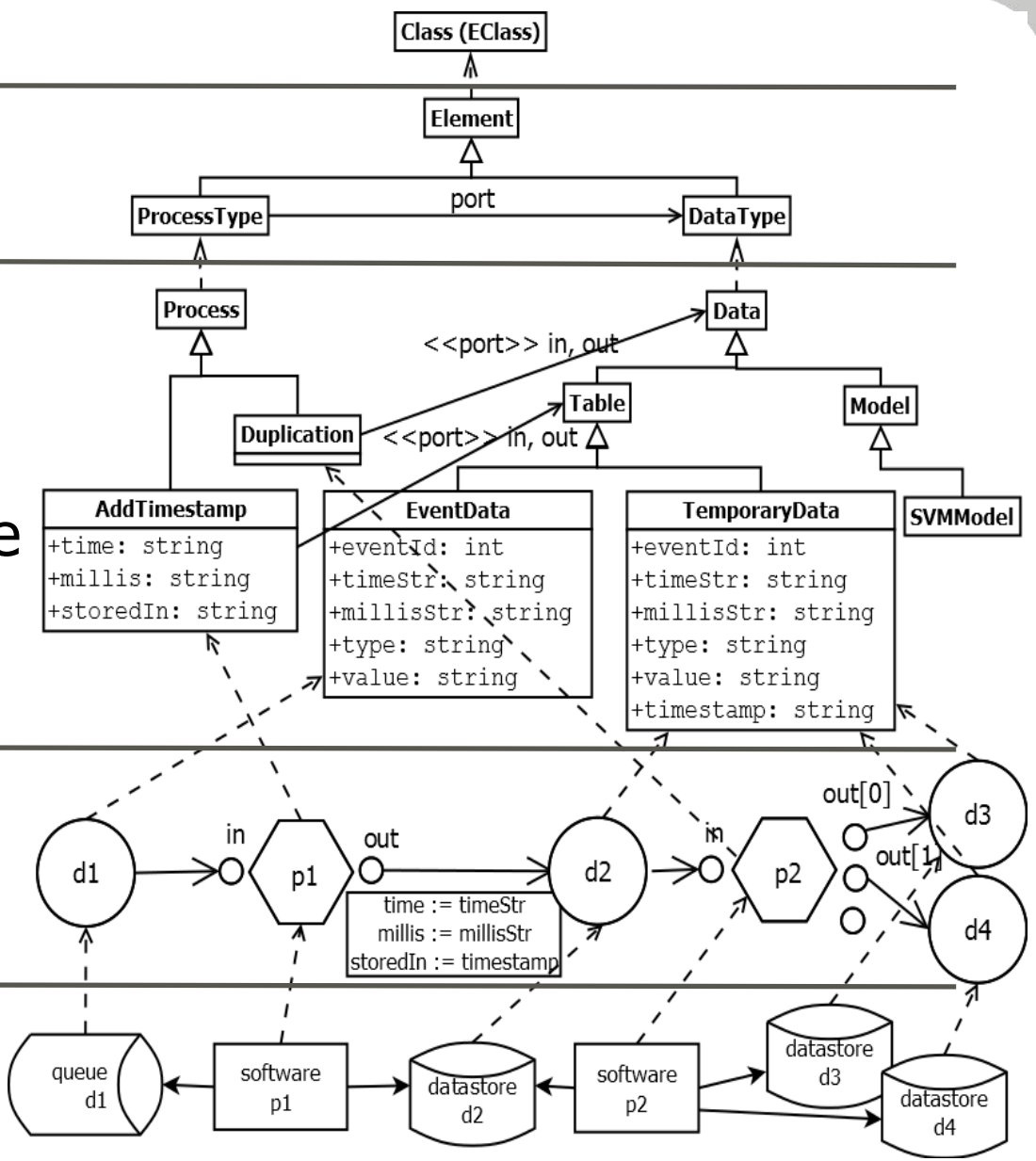
M4 MOF/EMF Ecore

M3 Conceptual types

M2 Types of elements displayed in palette

M1 Diagrams edited on canvas

M0 Executable software instances



Dataflow Model (DFM)

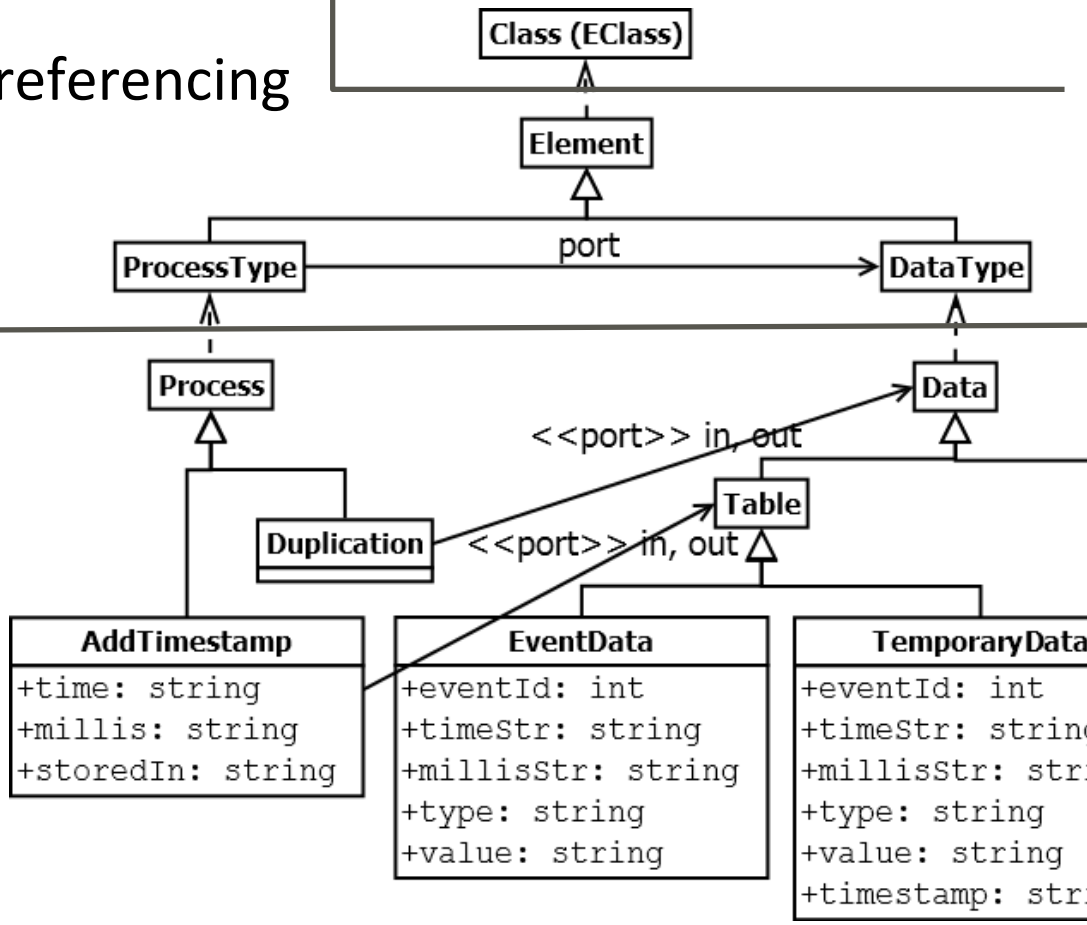
- - → instantiation
→ reference

FUJITSU

M3: Conceptual types

- DFM consists of process nodes and data nodes
- Process nodes has ports for referencing specific kind of data nodes
  - Ports are unidirectional

M4: MOF/EMF Ecore



M2: Types of elements

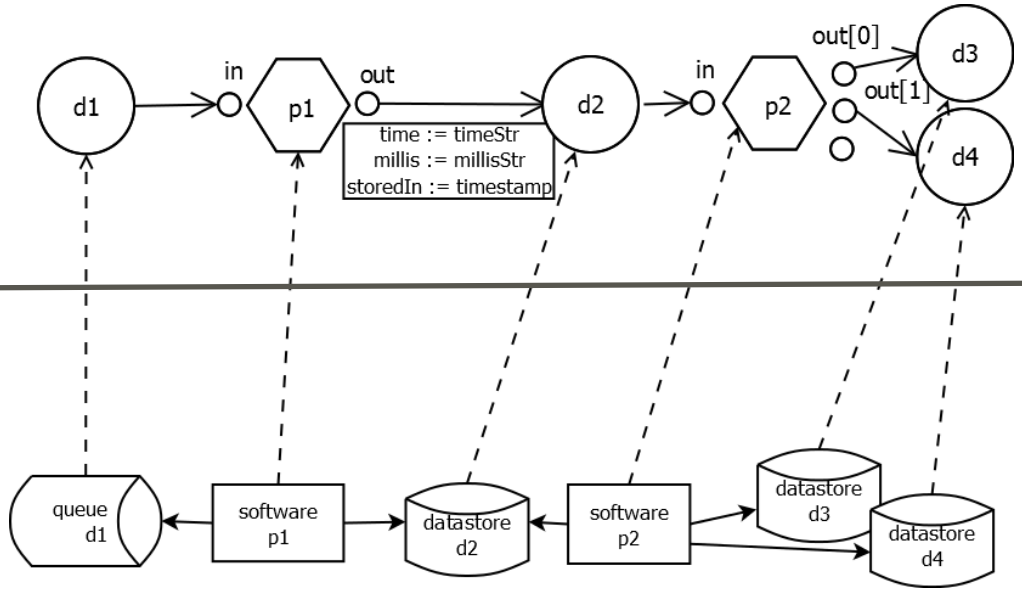
- Two classifications: Process and Data
- “in” and “out” in M2 are instances of port

Dataflow Model (DFM)

- - → instantiation
→ flow direction
→ dependency

M1: Diagrams

- Nodes and connections are edited on canvas
- Attributes are edited on properties editor



M0: Software instances

- Achieved by model transformation from diagrams in level M1
- Run on different execution environments (e.g., containers, VMs)

MLM Frameworks

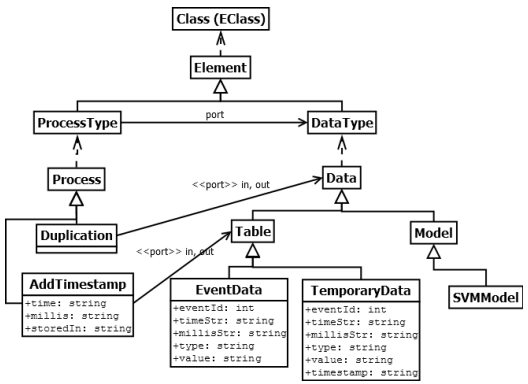
We selected frameworks that are actively maintained and support MDE tools (e.g., model transformation, constraints) from MLM Wiki: http://homepages.ecs.vuw.ac.nz/Groups/MultiLevelModeling/

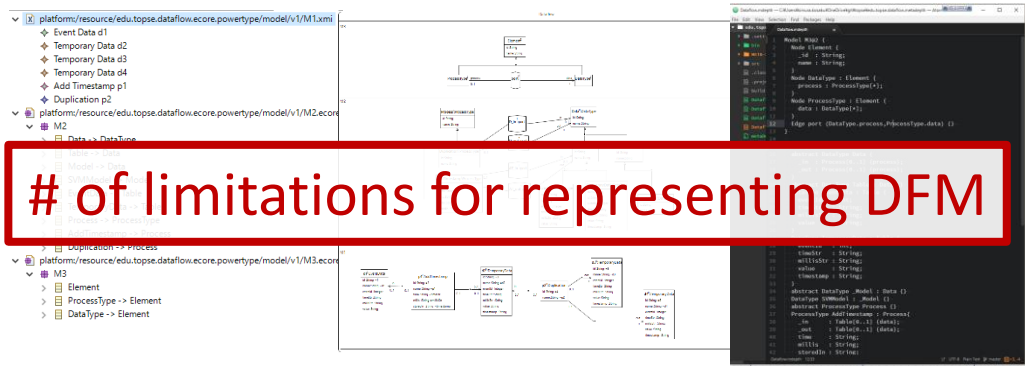
	Methodology
EMF	Two-level
Melanee	OCA
MetaDepth	Potency-based

Evaluation of the MLM frameworks

FUJITSU

How well can we represent DFM as it is?



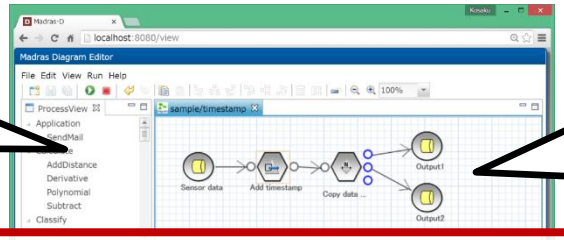


# of limitations for representing DFM

EMF Melanee MetaDepth

How easy can we extend (meta-) metamodel?

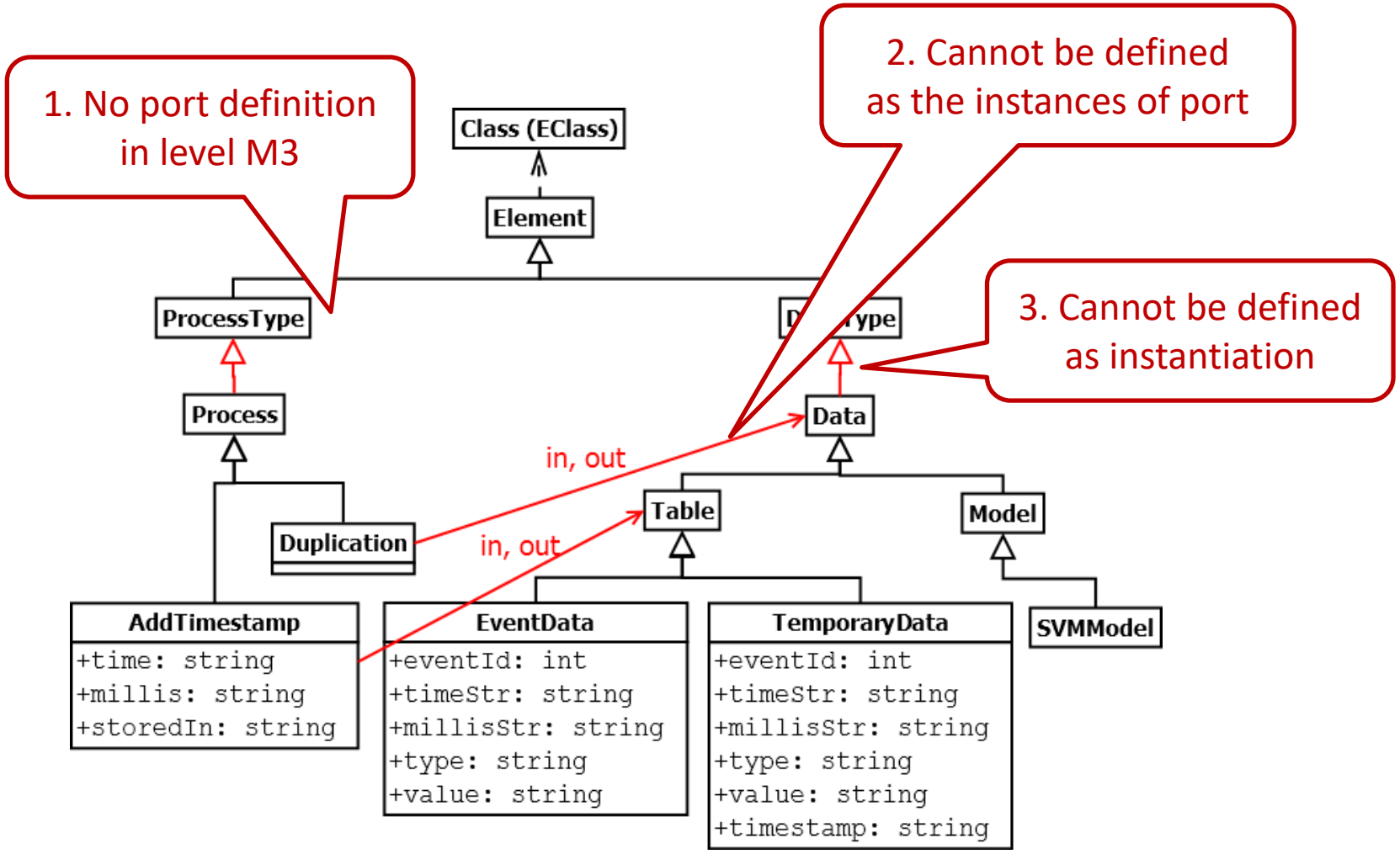
Adding a new type in level M2



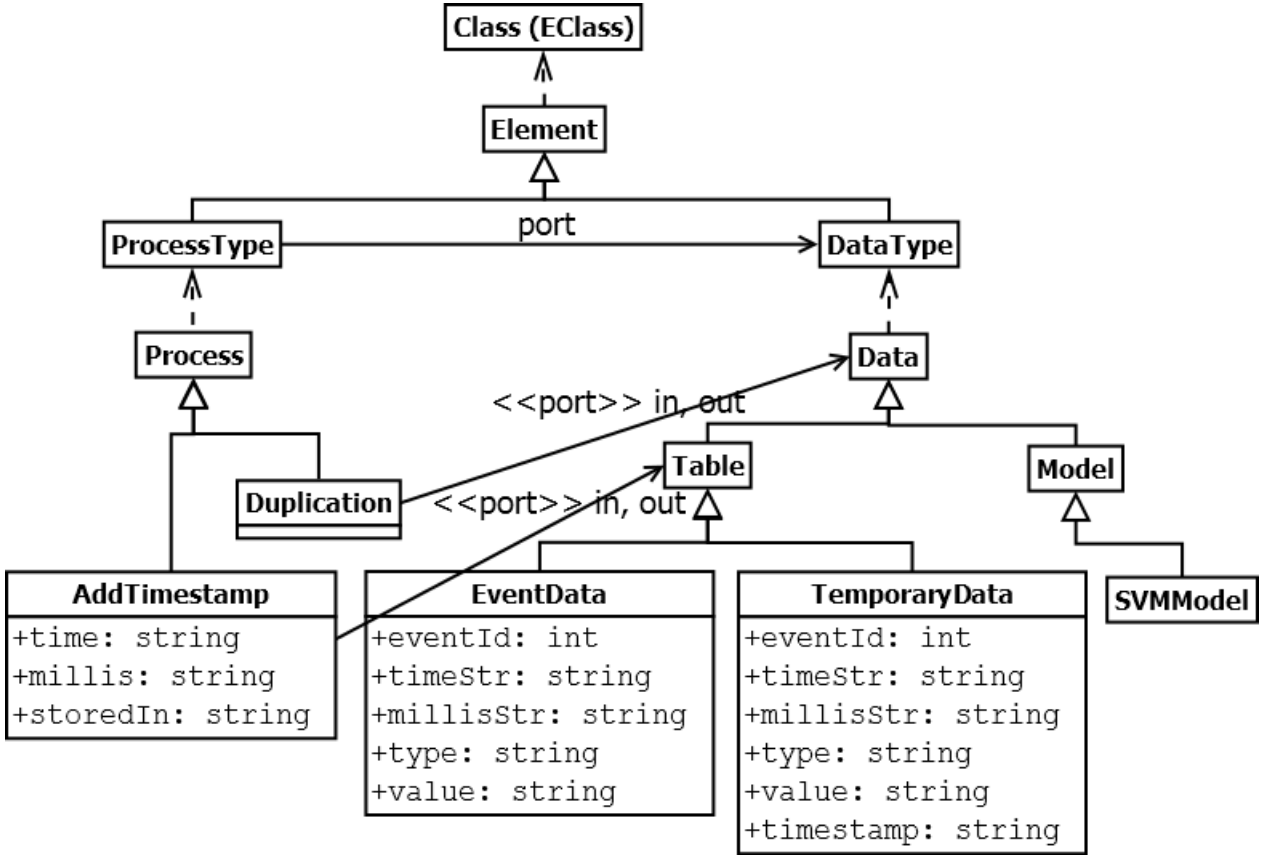
Adding a new conceptual type in level M3

# of existing elements to be modified in the two scenarios.
(The more the # increases, the more likely conflictions due to extensions can be caused)

Limitations of EMF

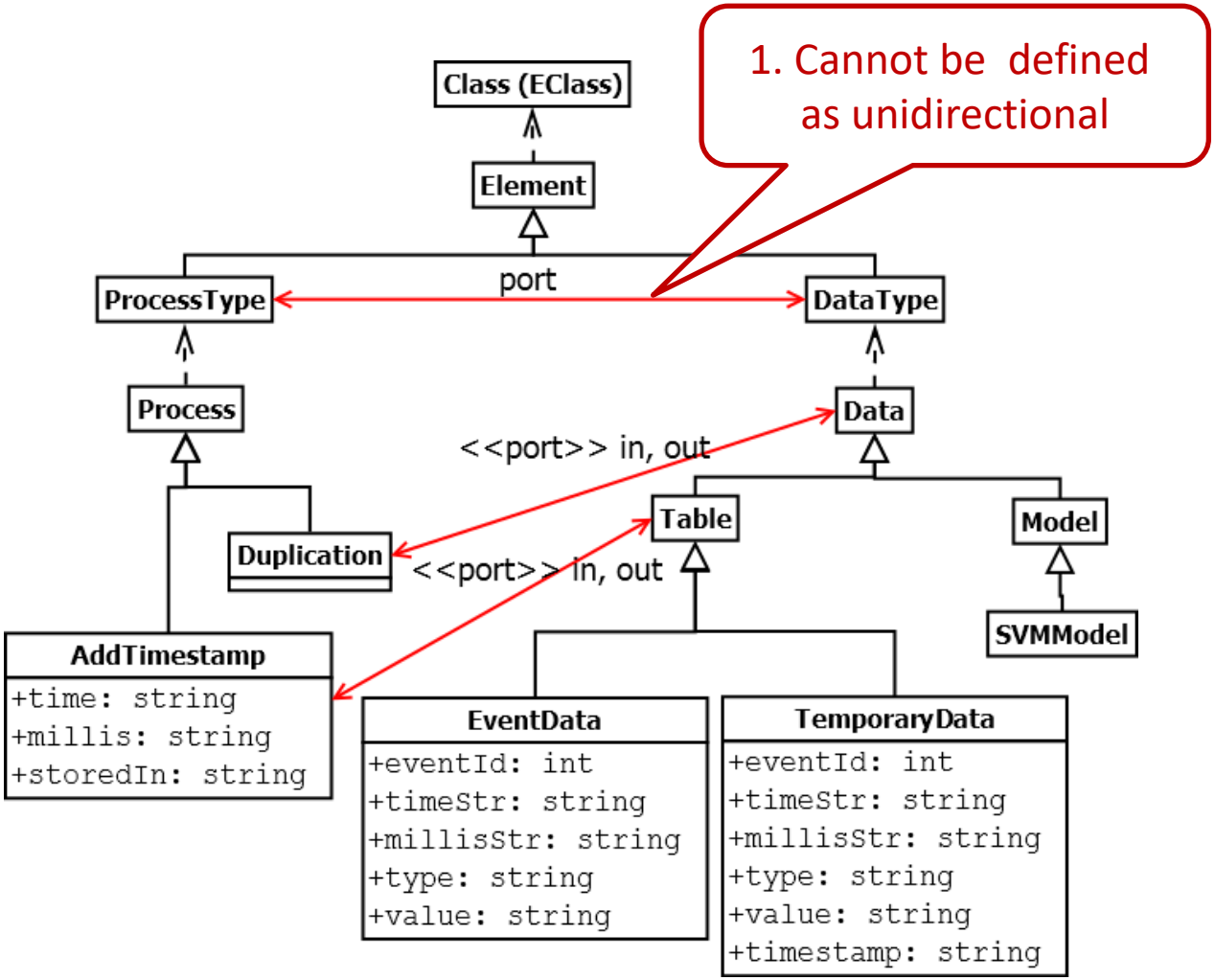


Limitations of Melanee



No limitation

Limitations of MetaDepth



1. How well we can represent DFM as it is

FUJITSU

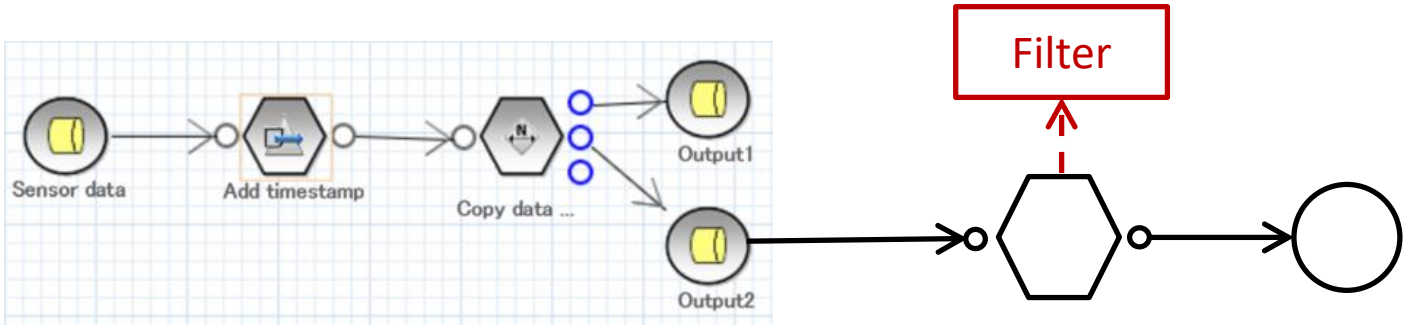
- Melanee has no limitation for representing DFM

# of limitations for defining DFM

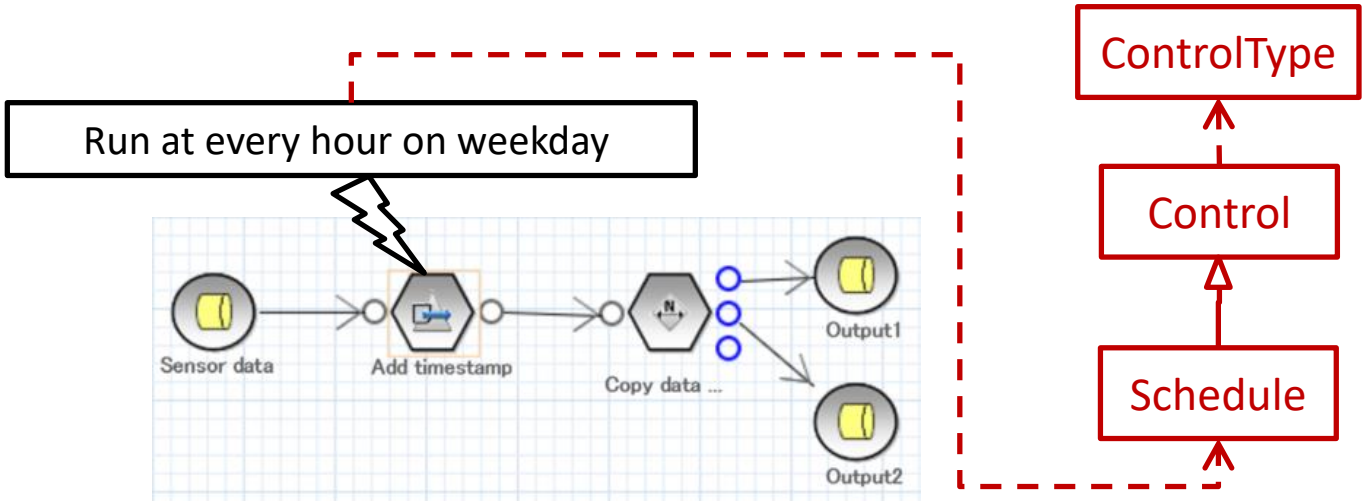
	# limitations
EMF	3
Melanee	0
MetaDepth	1

Two scenarios for extending DFM

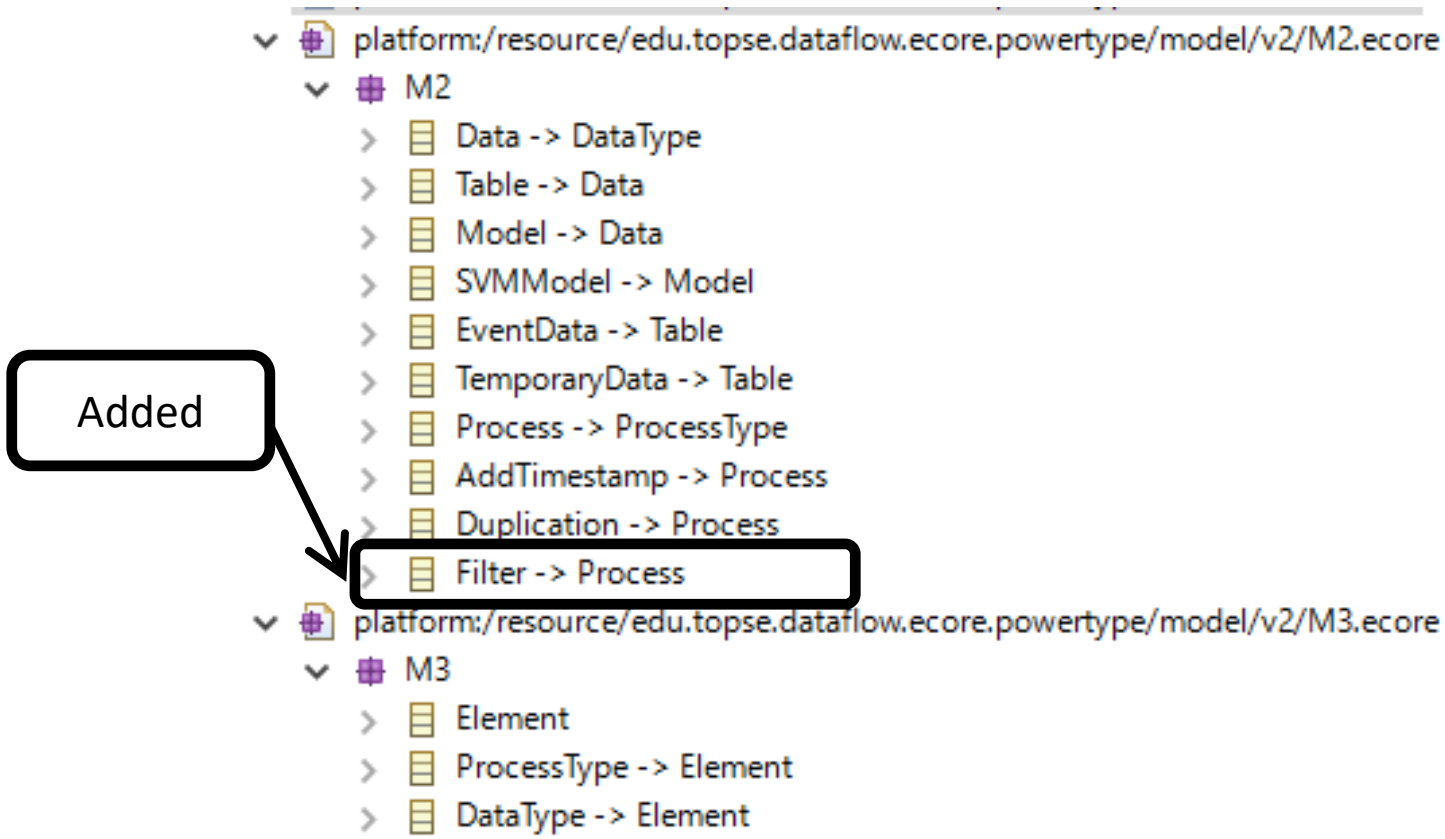
A: Adding a new type in level M2



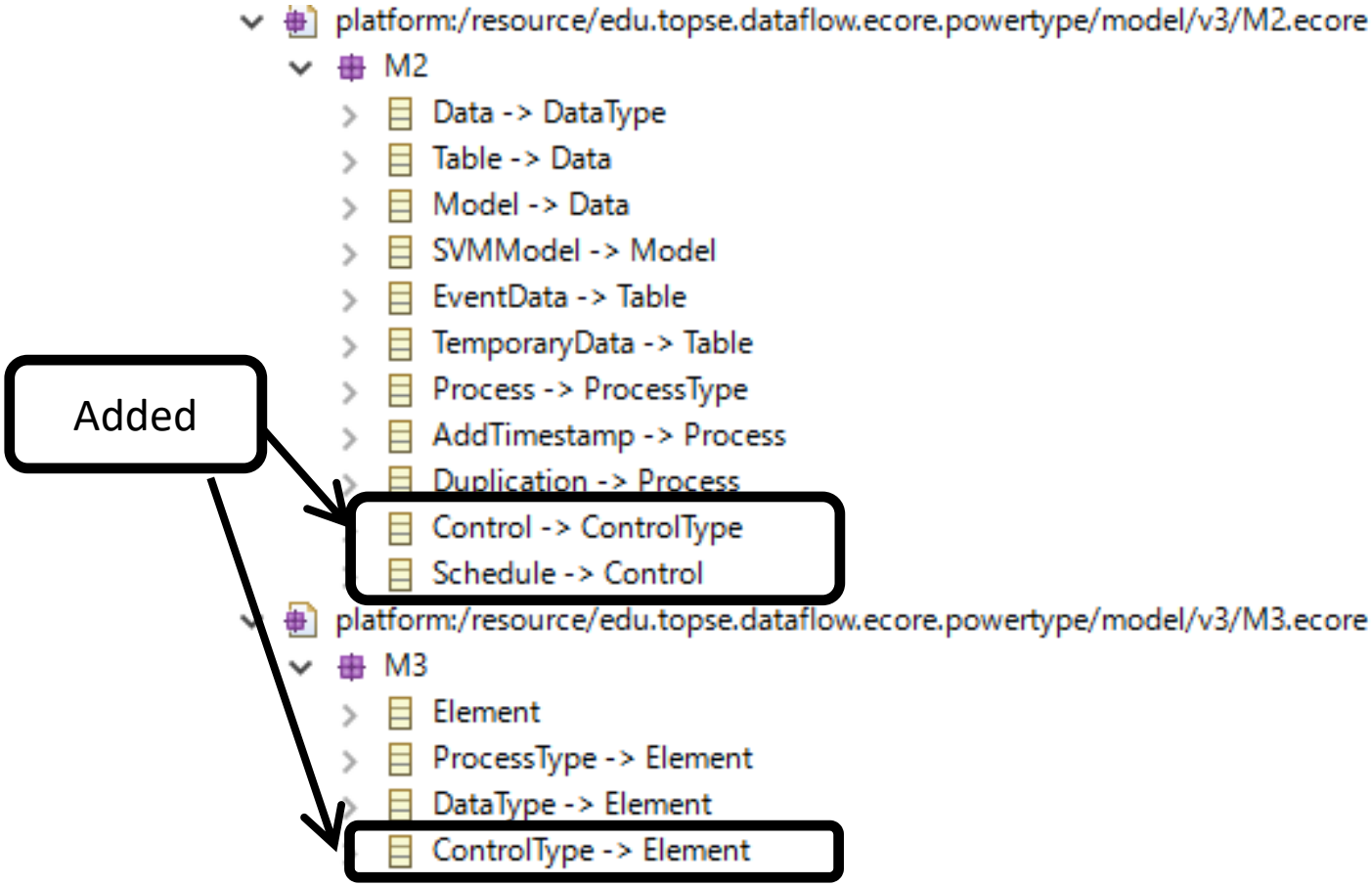
B: Adding a new conceptual type in level M3



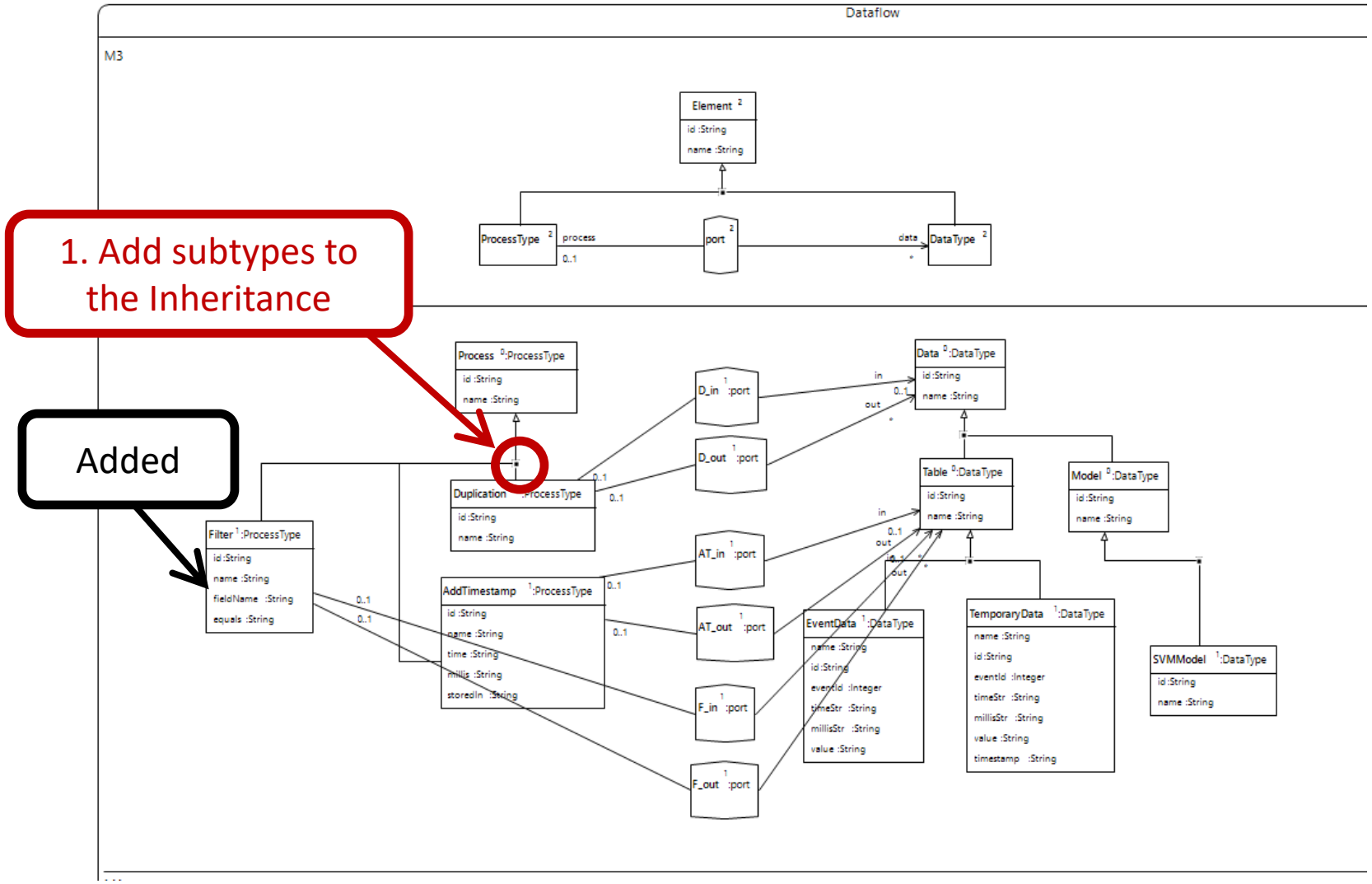
Scenario A EMF



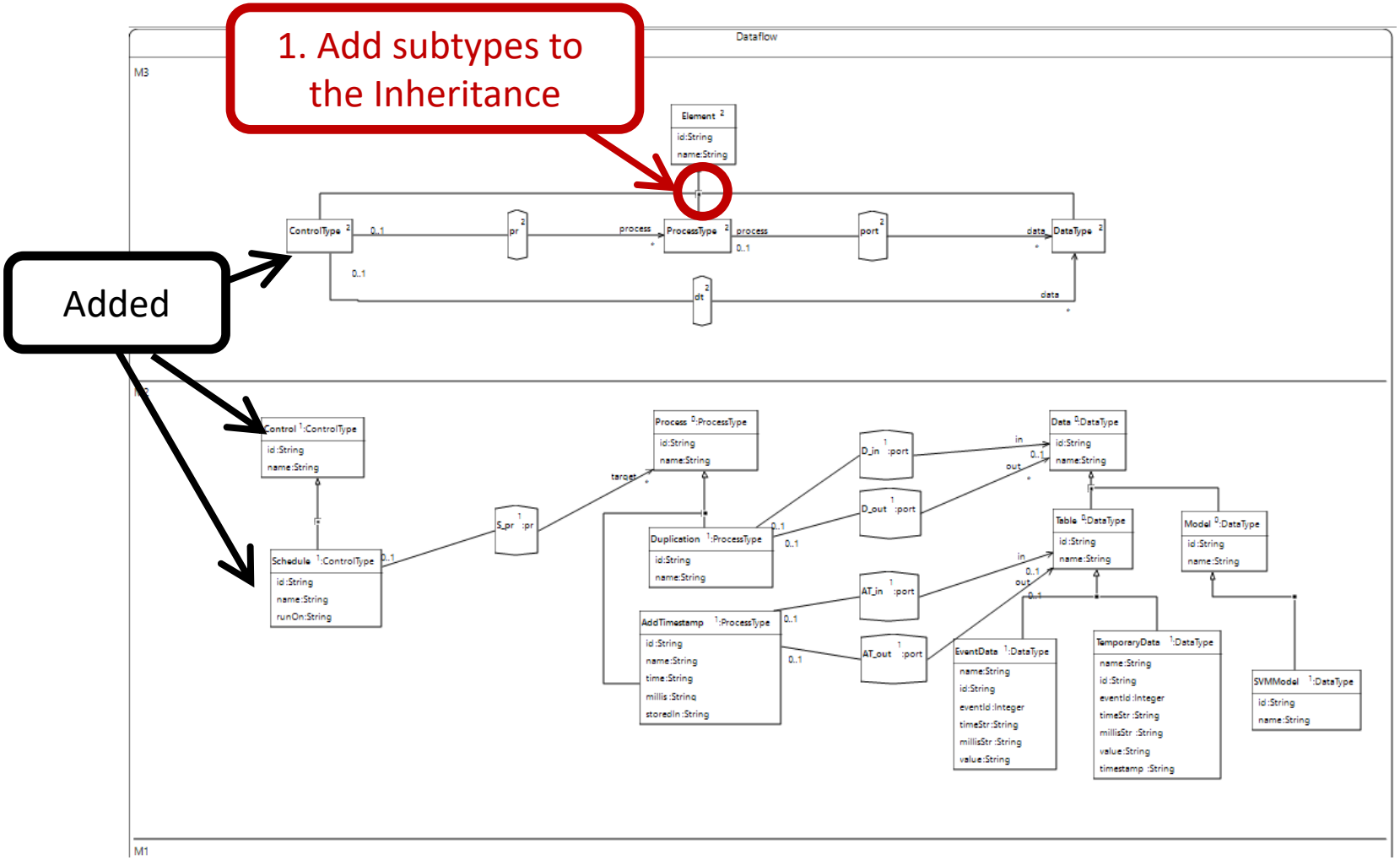
Scenario B EMF



Scenario A Melanee



Scenario B Melanee



Scenario A MetaDepth

M2

```
M3 M2 {
…
  abstract ProcessType Process {}
  ProcessType AddTimestamp : Process{
    _in      : Table[0..1] {data};
    _out     : Table[0..1] {data};
    time     : String;
    millis   : String;
    storedIn : String;
  }
  ProcessType Duplication : Process {
    _in  : Data[0..1] {data};
    _out : Data[*] {data};
  }
  ProcessType Filter : Process {
    _in  : Data[0..1] {data};
    _out : Data[0..1] {data};
    fieldName : String;
    equals    : String;
  }
  port AT_in  (Data._out, AddTimestamp._in) {}
  port AT_out (Data._in, AddTimestamp._out) {}
  port D_in   (Data._out, Duplication._in) {}
  port D_out  (Data._in, Duplication._out) {}
  port F_in   (Data._out, Filter._in) {}
  port F_out  (Data._in, Filter._out) {}}
```

Added

Scenario B MetaDepth

FUJITSU

M3

```
Model M3@2 {
  Node Element {
    _id  : String;
    name : String;
  }
  Node DataType : Element {
    process : ProcessType[*];
    control : ControlType[*];
  }
  Node ProcessType : Element {
    data : DataType[*];
    control : ControlType[*];
  }
  Node ControlType : Element {
    process : ProcessType[*];
    data : DataType[*];
  }
  Edge port (DataType.process, ProcessType.data) {}
  Edge pr (ControlType.process, ProcessType.control) {}
  Edge dt (ControlType.data,DataType.control) {}
}
```

1~4. Modifications for bidirectional references

Added

M2

```
M2 {
abstract DataType Data {
  _in  : Process[0..1] {process};
  _out : Process[0..1] {process};
  ctrl : Control[*] {control};
}
…
abstract ProcessType Process {
  ctrl : Control[*] {control};
}
…
abstract ControlType Control {}
ControlType Schedule : Control {
  target : Process[*] {process};
  runOn : String;
}
port AT_in  (Data._out, AddTimestamp._in) {}
port AT_out (Data._in, AddTimestamp._out) {}
port D_in   (Data._out, Duplication._in) {}
port D_out  (Data._in, Duplication._out) {}
pr S_pr   (Schedule.target, Process.ctrl) {}
}
```

2. How easy we can extend (meta-) metamodel of DFM

FUJITSU

- EMF has no modification in both scenarios
- Melanee has fewer modifications next to EMF

# of existing elements to be modified

	Scenario A	Scenario B
EMF	0	0
Melanee	1	1
MetaDepth	0	4

Conclusion and Future work

FUJITSU

Which MLM framework is the most appropriate for developing extensible GET?

How well can we represent DFM as it is? ⇨ Melanee

How easy can we extend (meta-) metamodel? ⇨ EMF, Melanee

- ✓ Melanee is better than the other two frameworks for developing extensible GET
  - Taking into account unidirectional associations is important
- Limitation
  - May not be valid for models having similar structure of DFM
- Furure work
  - Further evaluation with other models
  - Evaluation regarding compatibility to MDE tools and simplicity of model transformation rules and constraints

https://github.com/dataflow-mlm/dataflow-mlm

FUJITSU

shaping tomorrow with you