Maturing to Metamodels

Richard Paige and Dimitris Kolovos

Department of Computer Science

University of York

@richpaige, @kolovos, #mde2014

Motivation

- How do we help organisations adopt MDE, or MDE approaches to modelling?
  - But who don't extensively use MDE platforms.
- How do we help students learn MDE, coming from a programming background?
- That is, how do we help organisations/individuals *mature towards full use of MDE?*

Context

- Aims to improve the quality and efficiency of the software development process
  - Promotes **models** to first-class citizens
  - Reduces the need for human compilers
- Not restricted to a particular modelling technology / model representation format

Models?

Spreadsheets

Visio

RDBMS

EMF

XML

XSD

The MDE Community

Spreadsheets

Visio

EMF

RDBMS

XML

XSD

Everyone else

Spreadsheets

Visio EMF RDBMS

XML XSD

Why Adopt MDE?

- Benefit from modern modelling languages, DSLs, language workbenches, tools.
- Benefit from use of generic tools for model management (e.g., for transformation, validation, migration).
- Consolidate technology platforms.
  - Converge on EMF/Ecore, enable interoperability.

What Hinders Adoption?

- EMF and GMF learning curve.
- Bootstrapping
  - particularly working with several abstraction levels at once (model, metamodel).
- Investment in legacy models.
- Up-front metamodelling.
- Eclipse (and UI issues).
- ...

Solutions?

- Eliminate metamodelling?
  - Unrealistic: metamodelling is like typing, it can catch or prevent errors.
- Delay metamodelling?
  - Is it necessary in early stages of adoption?
- Infer metamodels?
  - Support legacy models and modelling languages.
- Some combination of the above may be needed!

‘Solutions’!

- XML for modelling.
- Spreadsheets for modelling.
- Drawing tools for modelling.
  - Useful for some modelling problems (and organisational contexts) but not all!

XML for modelling

- Pros:
  - Simple syntax, plenty of examples/tutorials
  - No need for specialist tools
  - Most developers familiar with it
  - Much data already stored in this format.
- Cons:
  - Verbose
  - No support for types, good for tree-based metadata (references via XML Schema).

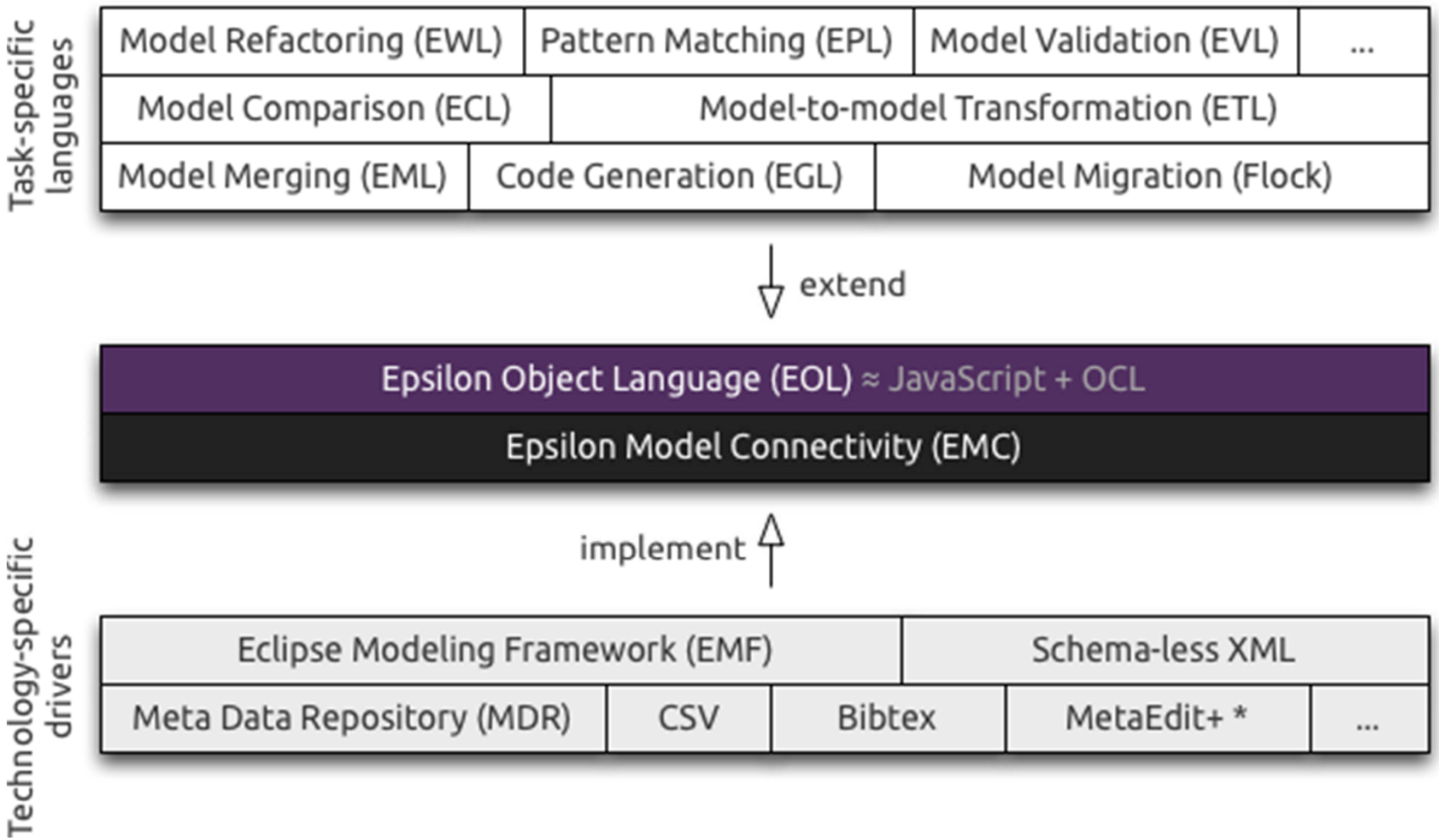
XML

- Bring MDE to XML-literate developers without forcing them to engage with EMF.
- Mature to full MDE by introducing one concept at a time, i.e.,
  - Models; then
  - Model management (e.g., M2M); *then*
  - Metamodelling.
- If there's payoff, transition from XML to EMF should be less of a challenge.

How?

- Enable model management languages to manipulate schema-less XML documents.
  - XML documents play the role of models.
  - Infer a simple schema from a document.
- Access the key benefits of MDE (automation) without paying the initial price (metamodelling).
  - Sometimes you will never need to pay the price!

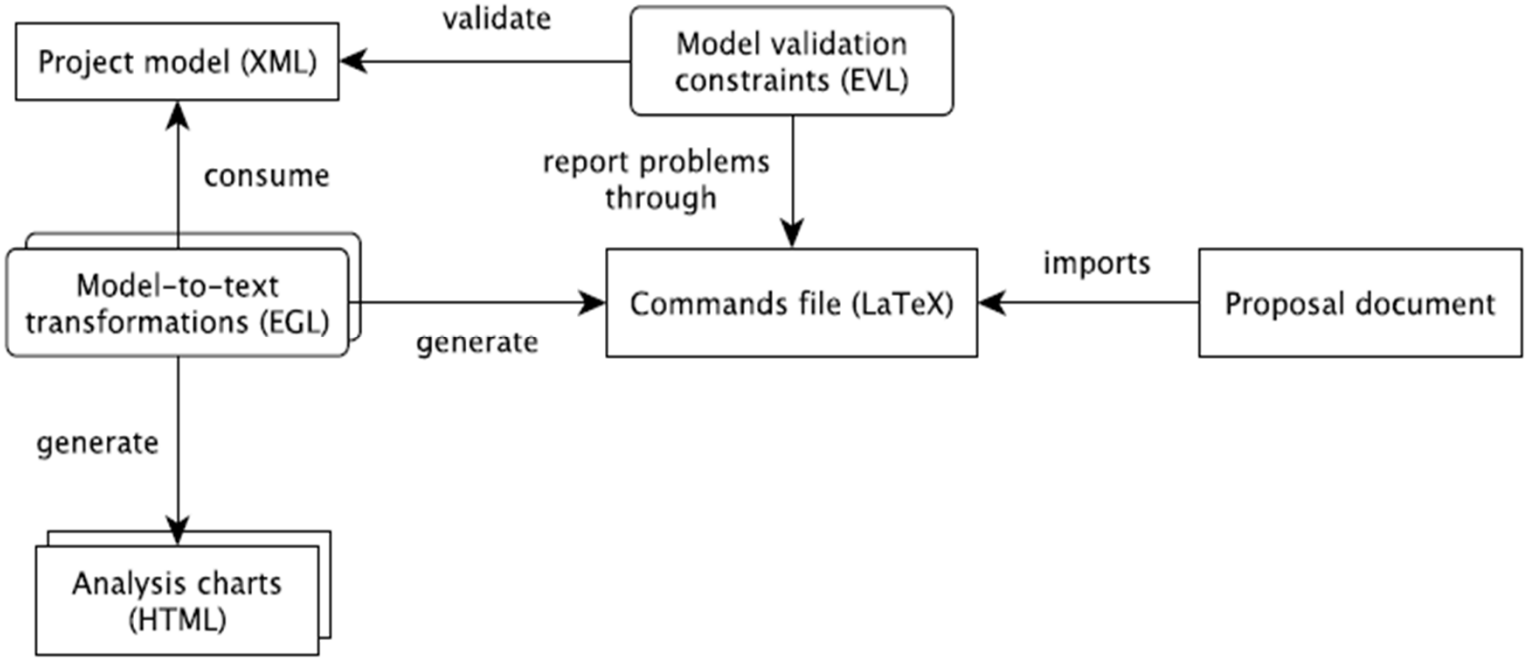
eclipse.org/Epsilon



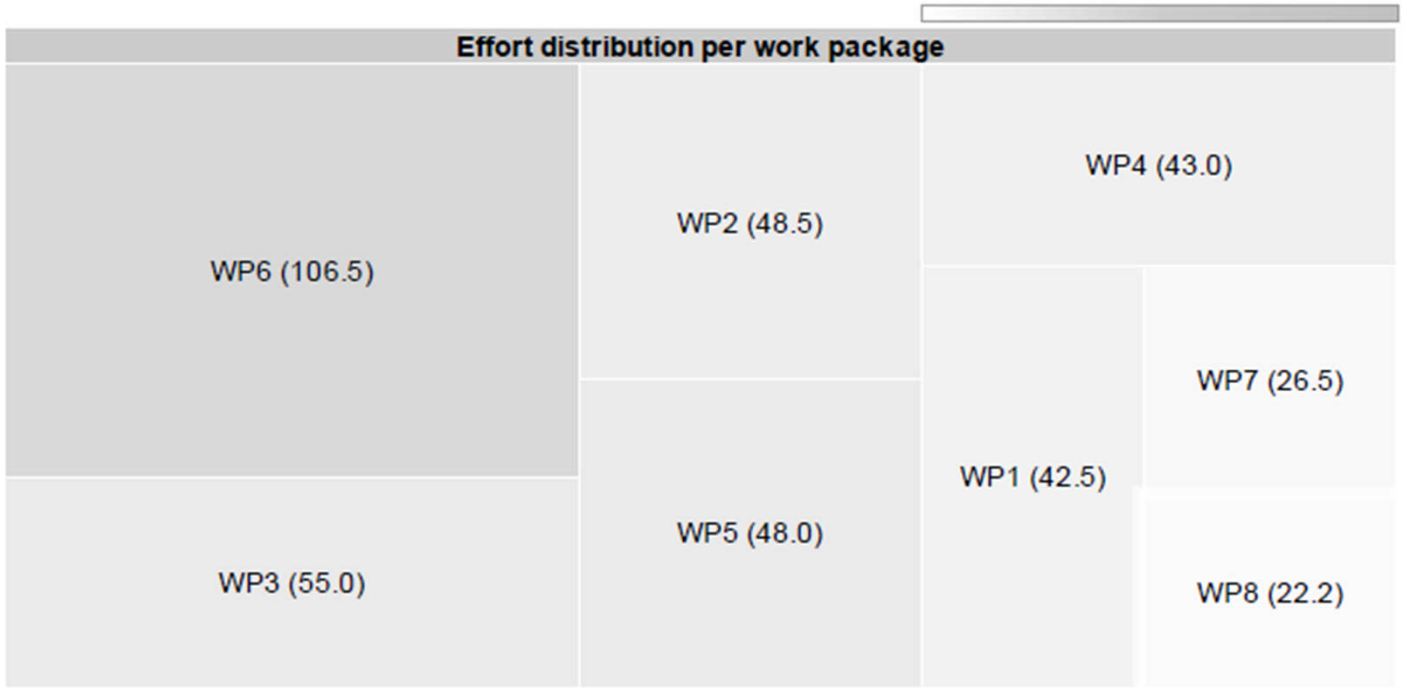
Example


```
1  <?xml version="1.0"?>
2  <project name="OSSMETER" duration="30"
3   title="Automated Measurement and Analysis of Open-Source
        Software" >
4
5   <wp title="Requirements & Use Cases" leader="TOG" type="RTD">
6    <effort partner="TOG" months="6"/>
7    <effort partner="York" months="6"/>
8
9    <task title="Use Case Analysis"
10    start="1" end="6" partners="TOG, York, ..."/>
11   <deliverable title="Project Requirements"
12    due="6" nature="R" dissemination="PU" partner="TOG"/>
13  </wp>
14
15  <milestone title="Requirements and case studies completion"
        month="6"/>
16  <milestone title="Project completion" month="30"/>
17
18  <partner id="YORK" name="University of York"
19   country="United Kingdom"/>
20 </project>
```

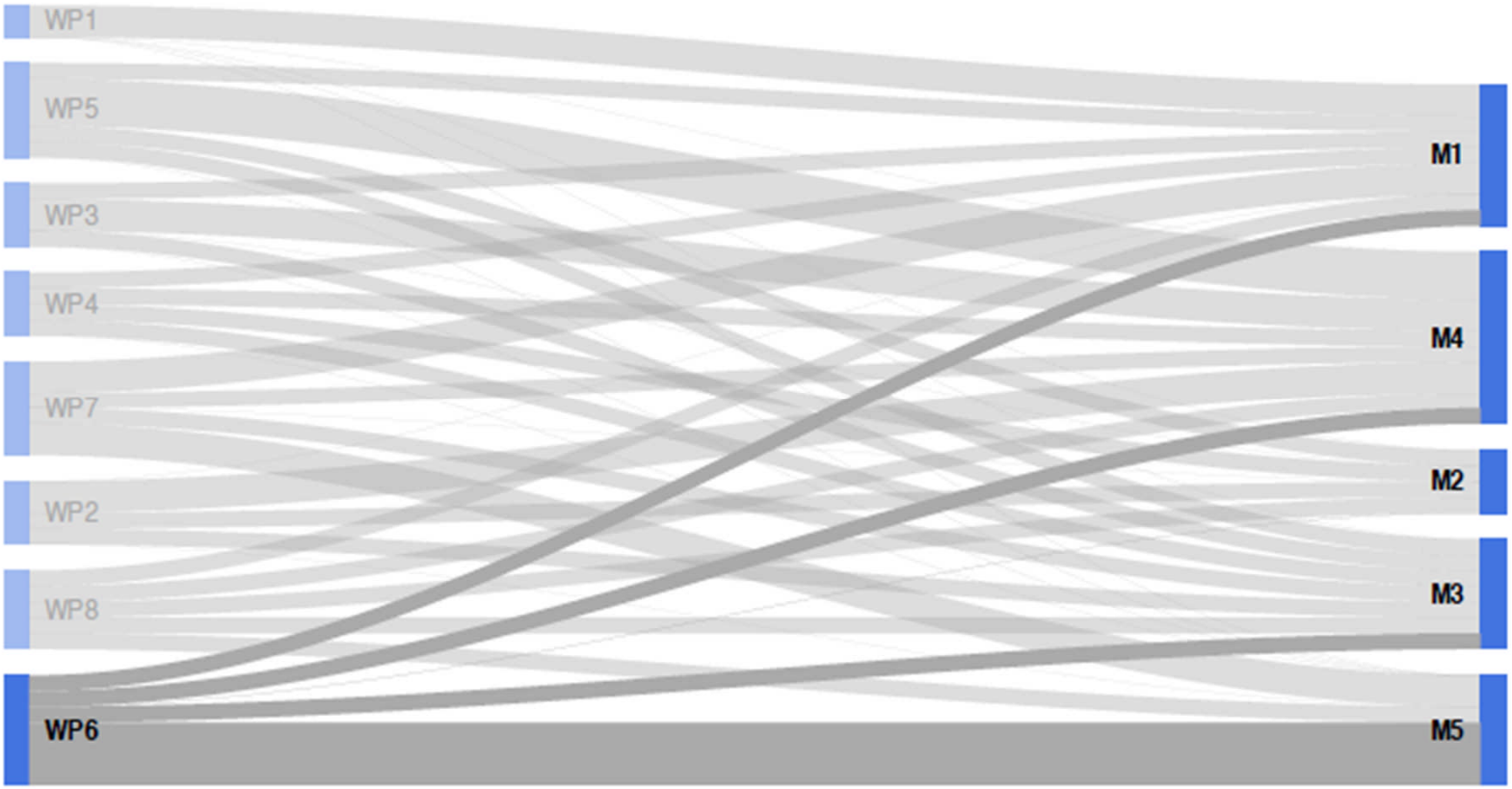
Example


Project model (XML)
validate
Model validation constraints (EVL)
consume
report problems through
Model-to-text transformations (EGL)
generate
Commands file (LaTeX)
imports
Proposal document
generate
Analysis charts (HTML)

Generation


Effort distribution per work package
WP6 (106.5)
WP2 (48.5)
WP4 (43.0)
WP7 (26.5)
WP1 (42.5)
WP3 (55.0)
WP5 (48.0)
WP8 (22.2)

Generation



Spreadsheets for modelling

- Versatile and intuitive
- Fill gaps in software development process
  - When no specialised tools exist for the job
  - When specialised tools are too expensive/complicated
  - When information needs to be collected from non-programmers
- Immense amount of data already in this form – can we support it as a modelling formalism?

How?

- Add support for spreadsheets to languages for
  - Model querying
  - Model-to-model transformation
  - Model validation
  - Model-to-text transformation
  - Model comparison
  - Model merging
- ... so that spreadsheets can be used in any step of an MDE process

Concept Mapping

- Spreadsheet -> Model
- Worksheet -> Type
- Column -> Property
- Row -> Model element

	A	B	C	D	E	F
1	id	firstname	lastname	age	supervisor	modules
2	jd501	Joe	Thompson	23	mt506	MSD,RQE
3	jd502	Jane	Smith	22	mt506	MSD,HCI
4						

Student	Staff	Module	Mark

Example (Query)

	A	B	C	D	E	F
1	id	firstname	lastname	age	supervisor	modules
2	jd501	Joe	Thompson	23	mt506	MSD,RQE
3	jd502	Jane	Smith	22	mt506	MSD,HCI
4						

Student | Staff | Module | Mark

```
Student.allInstances->
  select(s | s.age >= 18).println();
```

Example (M2T)

	A	B	C	D	E	F
1	id	firstname	lastname	age	supervisor	modules
2	jd501	Joe	Thompson	23	mt506	MSD,RQE
3	jd502	Jane	Smith	22	mt506	MSD,HCI
4						

Student | Staff | Module | Mark

```
<ul>
[%for (s in Student.allInstances){%]
  <li> [%=s.firstname] [%=s.lastname]
[%}%]
</ul>
```

Example (Validation)

	A	B	C	D	E
1	student	module	mark		
2	jd501	TPOP	62		
3	jd502	ICAR	74		

Student	Staff	Module	**Mark**

```
context Mark {
  constraint WithinRange {
    check: self.mark <= 100 and
           self.mark >= 0
    message: “Mark ” + self.mark +
      “ must be between 0-100”
  }
}
```

How?

- Develop a native spreadsheets driver for Epsilon’s pluggable type system (EMC)
- Pros
  - No intermediate artefacts
  - No stale data
  - Direct updates to the spreadsheet
  - Can leverage native querying capabilities

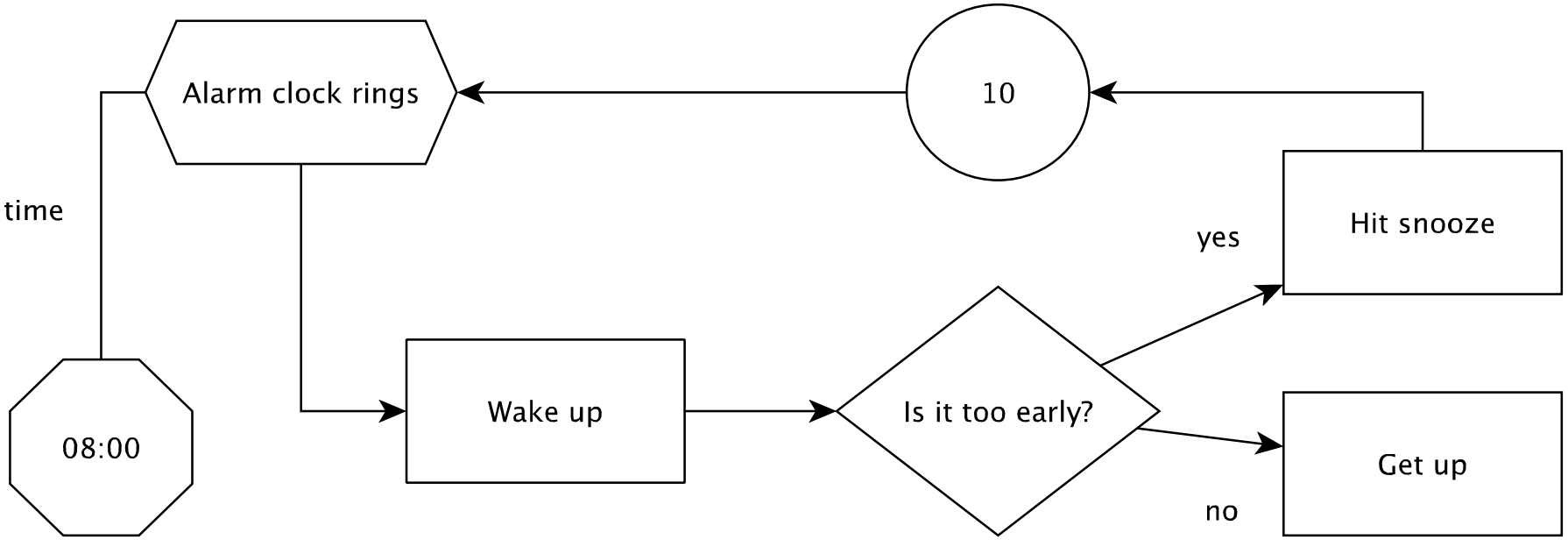
Query Translation

- The implementation of primitive query operations can be overloaded.
- The Google Spreadsheets driver rewrites (EOL) queries at runtime, e.g., to:
  - https://spreadsheets.google.com/feeds/list/tb-<mark-worksheet-guid>/od6/private/full?sq=mark>70
- Search is performed on the server and a **`Set(Mark)`** is returned
  - Composite queries are supported

Drawing for Modelling

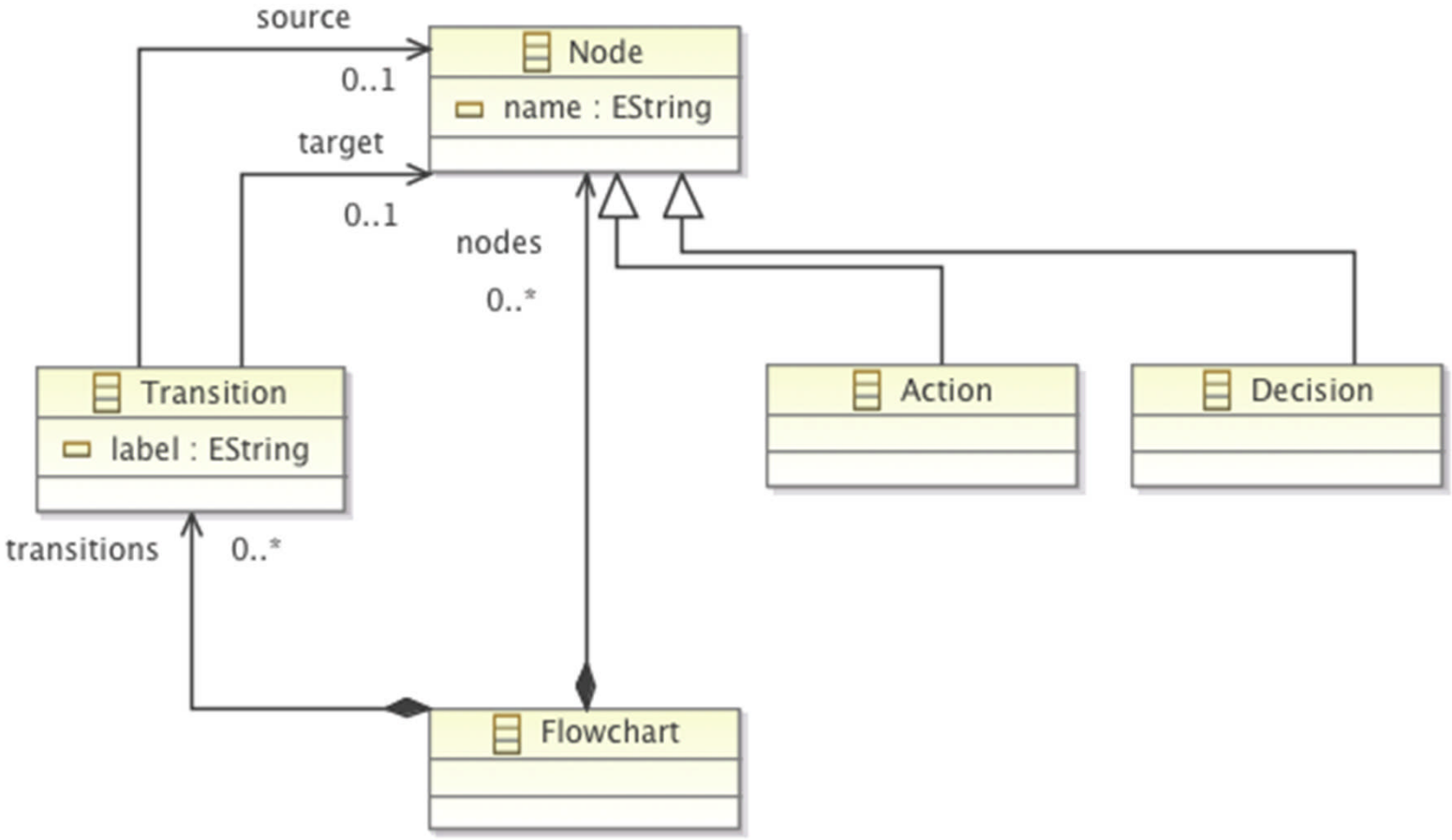
- What if we want to support the development of new DSLs?
  - Not just legacy models (spreadsheets etc) or non-MDE technologies (XML etc).
- EMF rage?

Novice vs. EMF

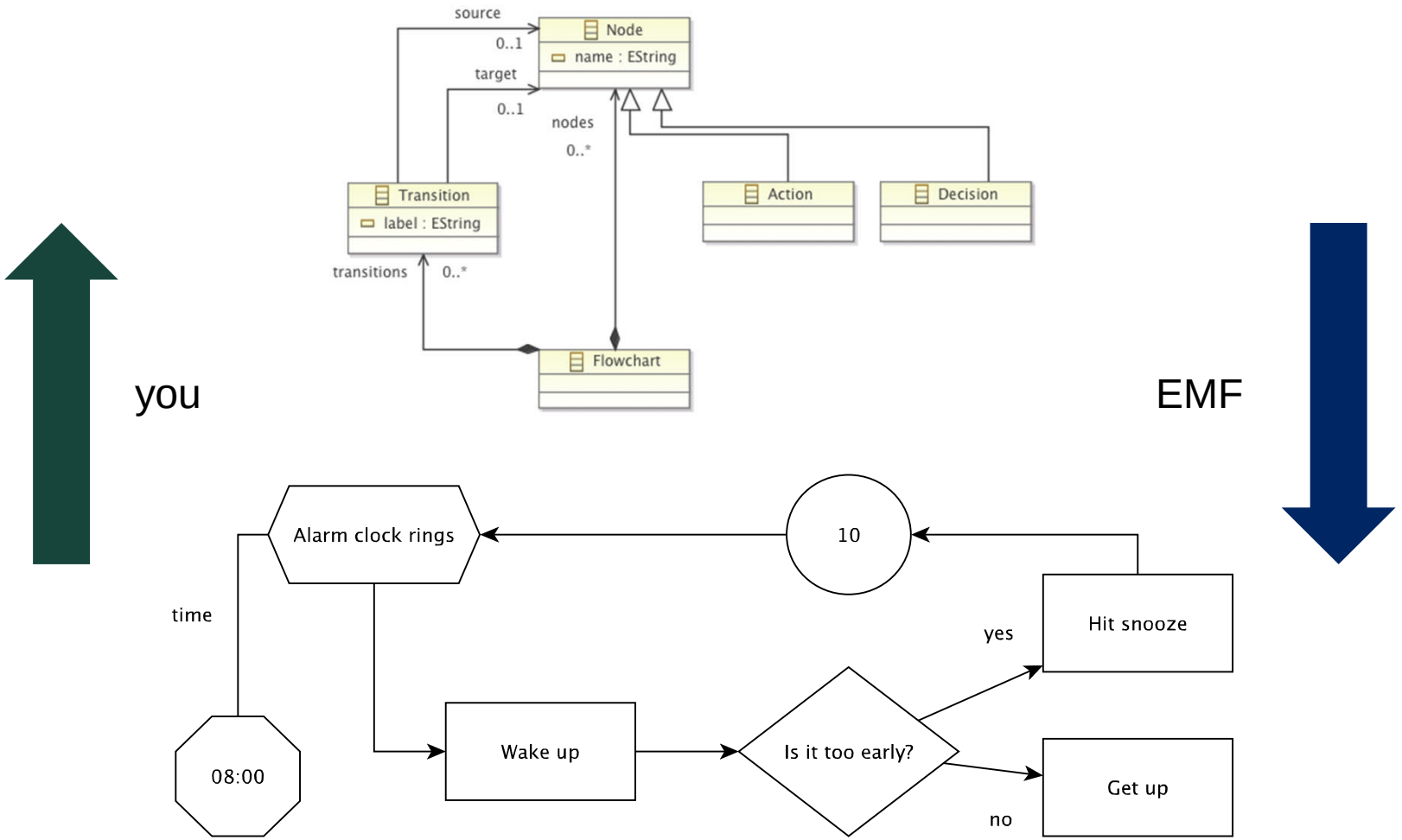


Novice vs. EMF*

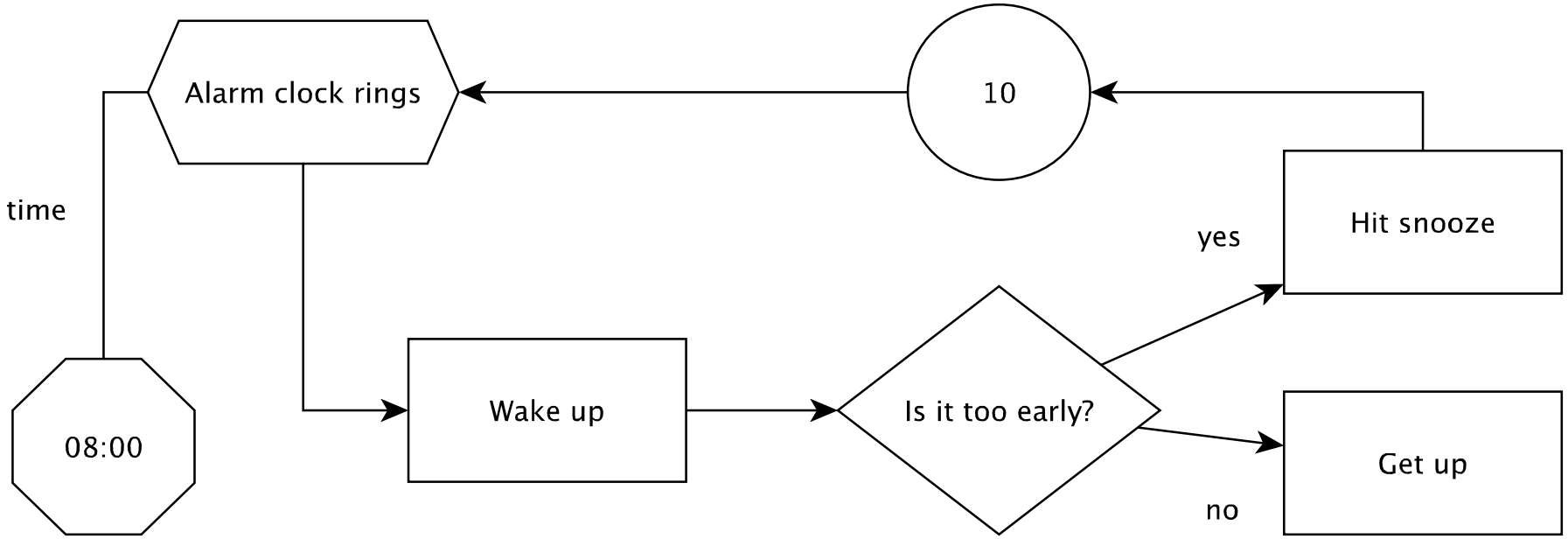
* or any similar 3-level metamodelling architecture



Novice vs. EMF



Draw, don’t model



Validation

- The diagram looks acceptable – but is it?
- Write a program against it!
  - e.g. a transformation, simulator
- Useful for evaluating the completeness of the underlying DSL
  - Concepts, relationships, captured information
- ... but all you have is a drawing ☹

Solution #1

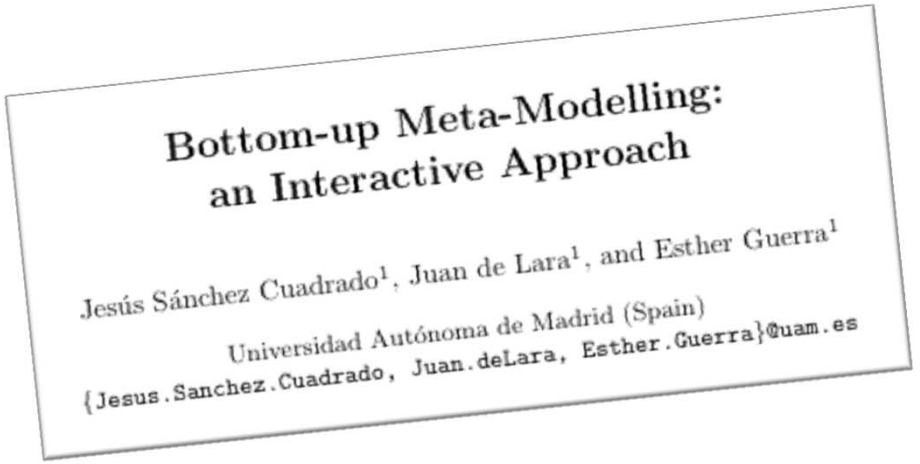
1. Draw the diagram using a GraphML-compliant tool (e.g. yEd)
2. Script the XML representation of the diagram

```
<graph edgedefault="directed" id="G">
  <data key="d10"/>
  <node id="n0">
    <data key="d4"><![CDATA[Action]]></data>
    <data key="d5"><![CDATA[boolean start = true]]></data>
    <data key="d6"><![CDATA[name]]></data>
    <data key="d9">
      <y:ShapeNode>
        <y:Geometry height="61.0" width="119.0" x="-594.0" y="-263.0"/>
        <y:Fill color="#FFFFFF" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content" fontFamily="Dialog" fontSize="12"
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0" nodeRatioX="0.0" no
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="rectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="n1">
    <data key="d4"><![CDATA[Decision]]></data>
    <data key="d6"><![CDATA[name]]></data>
    <data key="d9">
      <y:ShapeNode>
        <y:Geometry height="106.0" width="138.0" x="-603.5" y="-120.0"/>
        <y:Fill color="#FFFFFF" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content" fontFamily="Dialog" fontSize="12"
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0" nodeRatioX="0.0" no
          </y:ModelParameter>
```

Too low level

Solution #2

1. Draw the diagram using a GraphML-compliant tool (e.g. Dia/yEd)
2. Draw a separate *legend* diagram
3. Export the diagram as a textual *fragment*
4. Annotate the fragment
5. Derive an Ecore metamodel
6. Transform the diagram to a conforming EMF model
7. Use your favourite EMF-compliant model management languages



Bottom-up Meta-Modelling:
an Interactive Approach

Jesús Sánchez Cuadrado[1], Juan de Lara[1], and Esther Guerra[1]

Universidad Autónoma de Madrid (Spain)
{Jesus.Sanchez.Cuadrado, Juan.deLara, Esther.Guerra}@uam.es

Or?

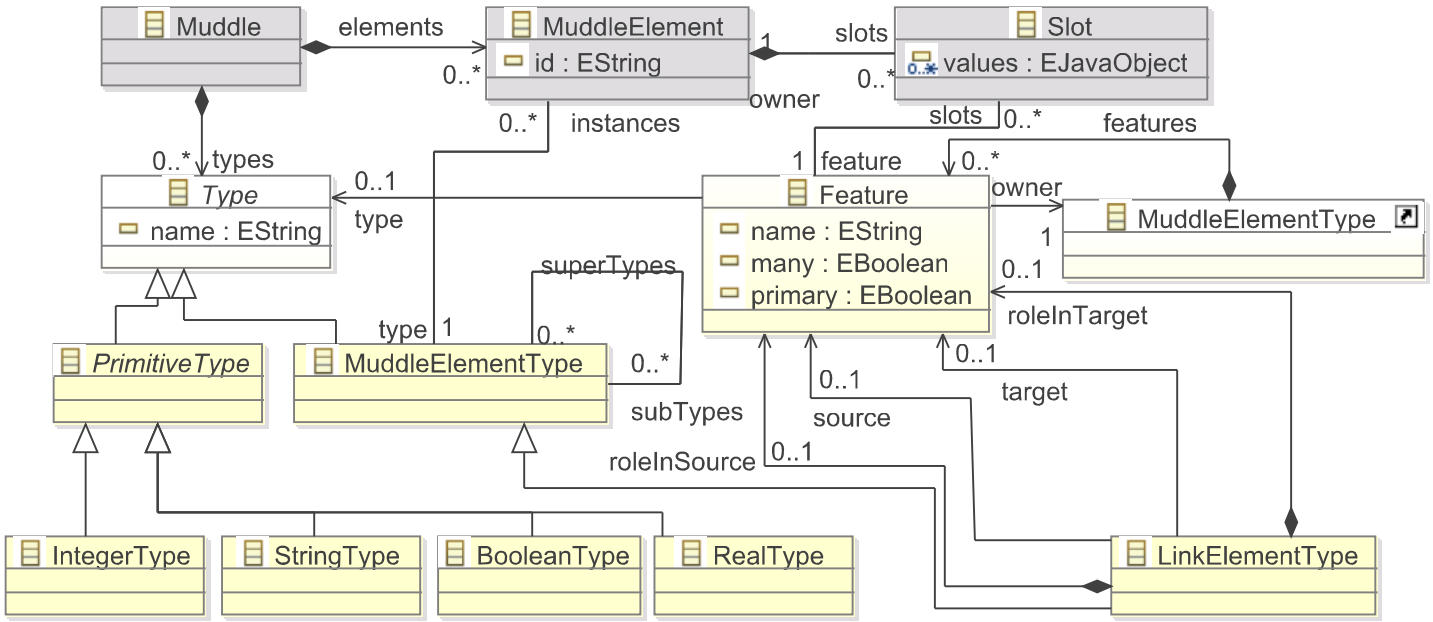
1. Draw a diagram using GraphML
2. Annotate the *diagram*
3. Write model management programs against your diagram as if it was a real model using

Annotations

- Use extension mechanisms of GraphML.
- For example, declare that the “type” of directed edges is “Transition”.
  - GraphML doesn’t support types.
  - But it does support extension fields on nodes, and these can encode simple patterns.
  - The patterns are sufficient to encode basic role ends, multiplicities, types, … aka simple meta-constructs.

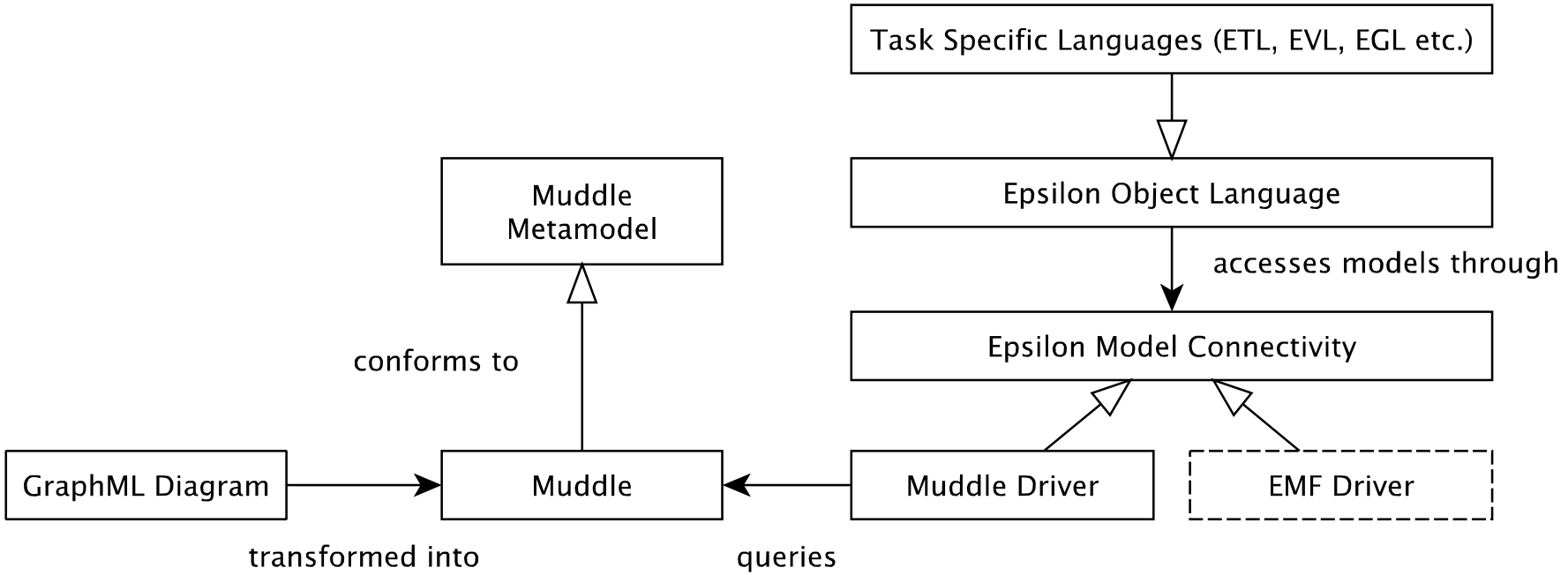
How it works (1/2)

- The annotated diagram is translated to an intermediate EMF *muddle* behind the scenes

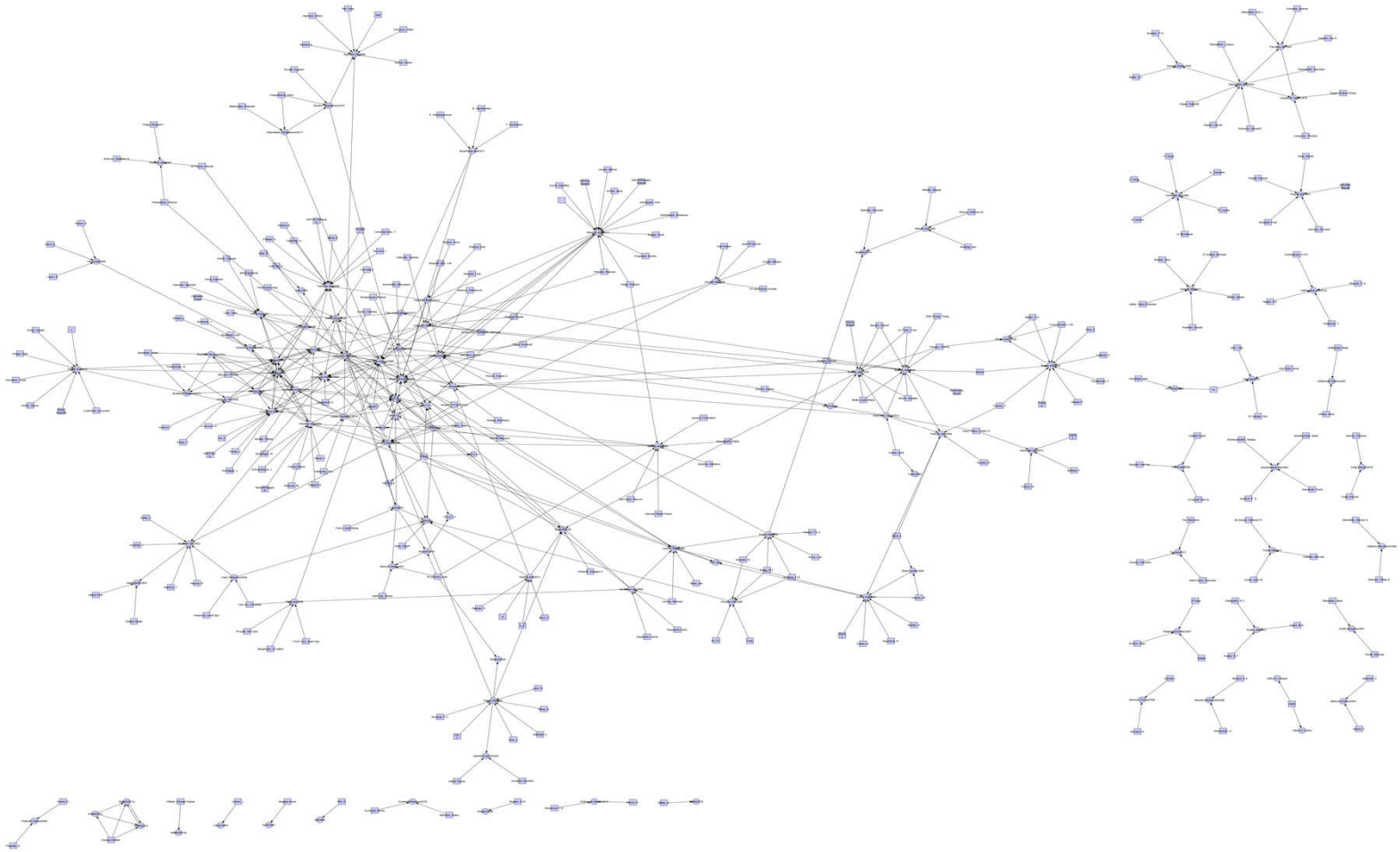


How it works (2/2)

- The muddle is accessed by a specialised Epsilon Model Connectivity driver



Citation Map



Lessons Learned

- EMF and jumping into metamodelling turns off some companies/students.
  - Especially those not already comfortable with Eclipse.
- Starting with yEd (anecdotally) helps in letting them exploit MDE.
  - Some organisations keep using it, preferring not to fully migrate to EMF!
  - Risks of this.

Lessons Learned

- Annotation-based approaches (like the GraphML approach) are a pragmatic stepping stone to metamodelling.
  - It’s a recurring pattern for language design!
  - As soon as you use metamodels, you will encounter migration issues.

Lessons Learned

- Working with pure XML is ideal for small tasks that benefit from automation, but benefit minimally from metamodelling.
  - E.g., where typing rules are straightforward, models are relatively small.
- Also for tasks where insufficient MDE expertise exists.
- But requires ability to package up MDE programs in digestible bundles.

Lessons Learned

- Spreadsheets processed as models are extremely useful, but normally are used in concert with other artefacts.
  - E.g., defining trace-links between spreadsheets and SysML models.
- The spreadsheets are generally incomplete and inconsistent, and very large.
  - More efficient querying, bad smell detection, and automated repair needed.
  - “Software Engineering for Spreadsheets” Workshop in Delft, 2 July 2014.
  - http://spreadsheetlab.org/sems-14/

Take-home Message

- We need to **reach out** to mainstream developers and **embrace** the types of *models* they are using
  - and the way in which they are using models.
- Or risk becoming irrelevant
- We can’t ignore the adoption difficulties.
- MDE is something you grow into.

Code

- All Epsilon EMF-based code available at
  - http://www.eclipse.org/epsilon
- Non-EPL licensed code at Epsilon Labs:
  - http://code.google.com/p/epsilonlabs/