Model Federation using FML

Openflexo

Sylvain Guérin

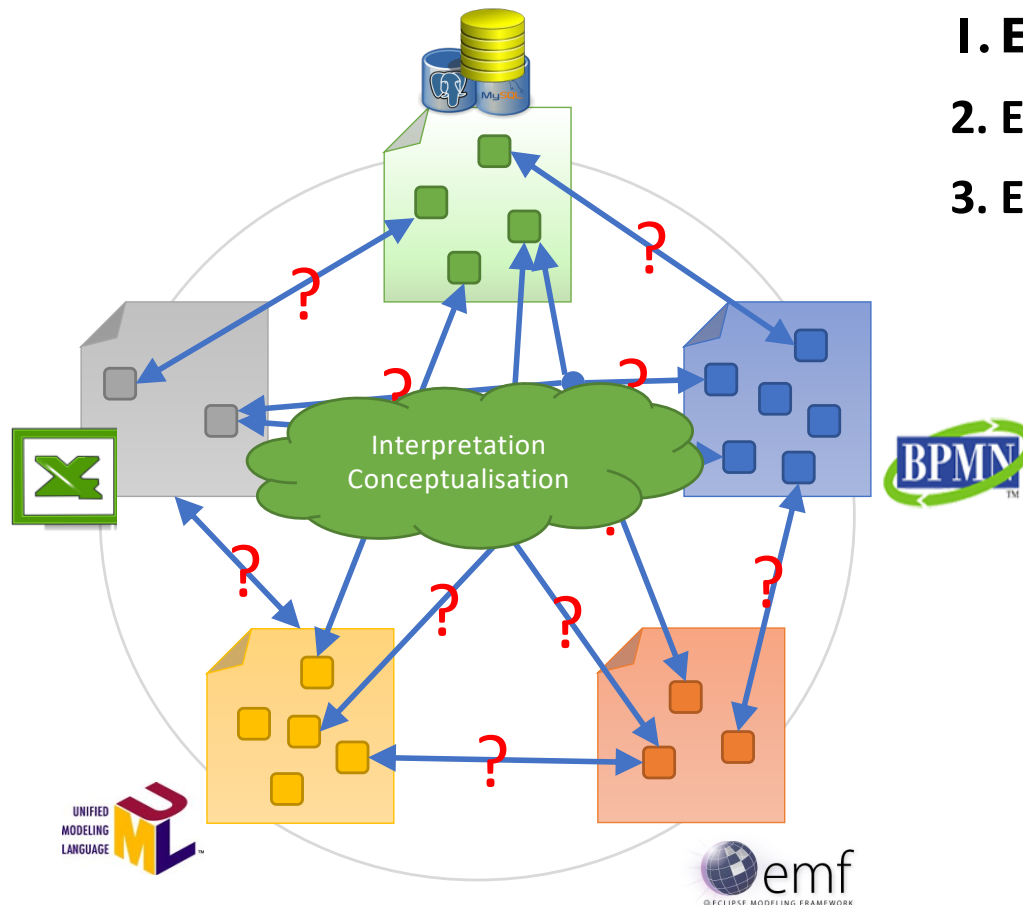15/12/2023

Multiple models ?

Semantic interoperability

Lots of experts, lots of concerns, lots of practices,
lots of formalisms, lots of models.

How can all these points of view be reconciled?
How do we connect all these models?

Model interoperability: different approaches

- **Integration**
  - Construction of a union, exhaustive and complete metamodel
  - All existing models 'conform' to this metamodel

- **Unification** (pivot + transformations)
  - Selection and/or construction of a pivot metamodel
  - All existing models are transformed to conform to this pivot metamodel

- **Model federation**
  - Definition of links/dependencies between concepts to be federated
  - Organisation of these links as models

ISO 14258 : Industrial automation systems - concepts and rules for enterprise models.
International Organization for Standardization, Geneva, Switzerland, 2000.

Problematic issues and research questions

1. **E1 : Heterogeneity**

2. **E2 : Data (re-) interpretation**

3. **E3 : Dynamicity**

Additional constraints and requirements

C1. Autonomy of the life cycle of federated information sources
C2. Intermittent connectivity
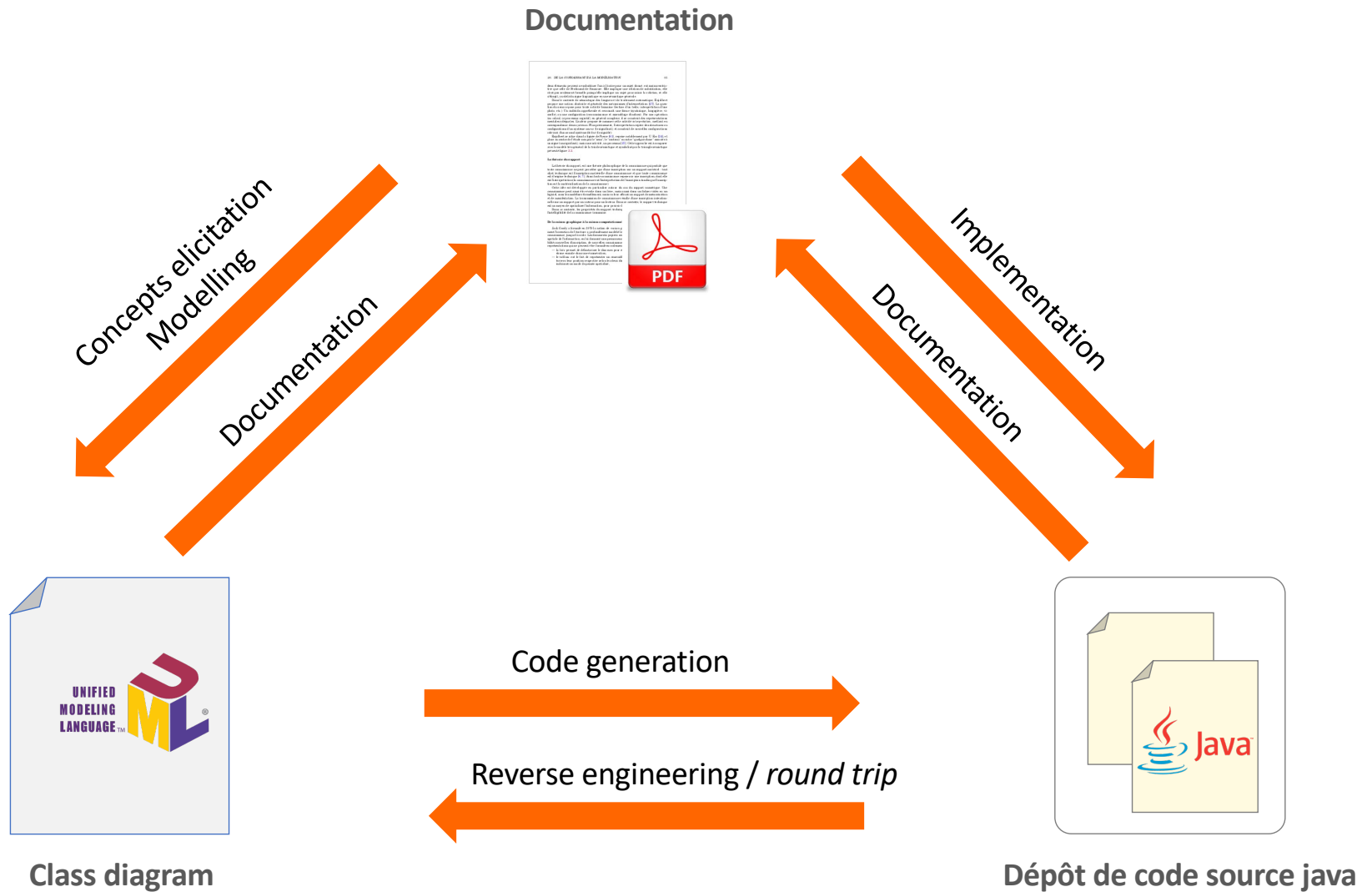**C3**. Non-intrusiveness
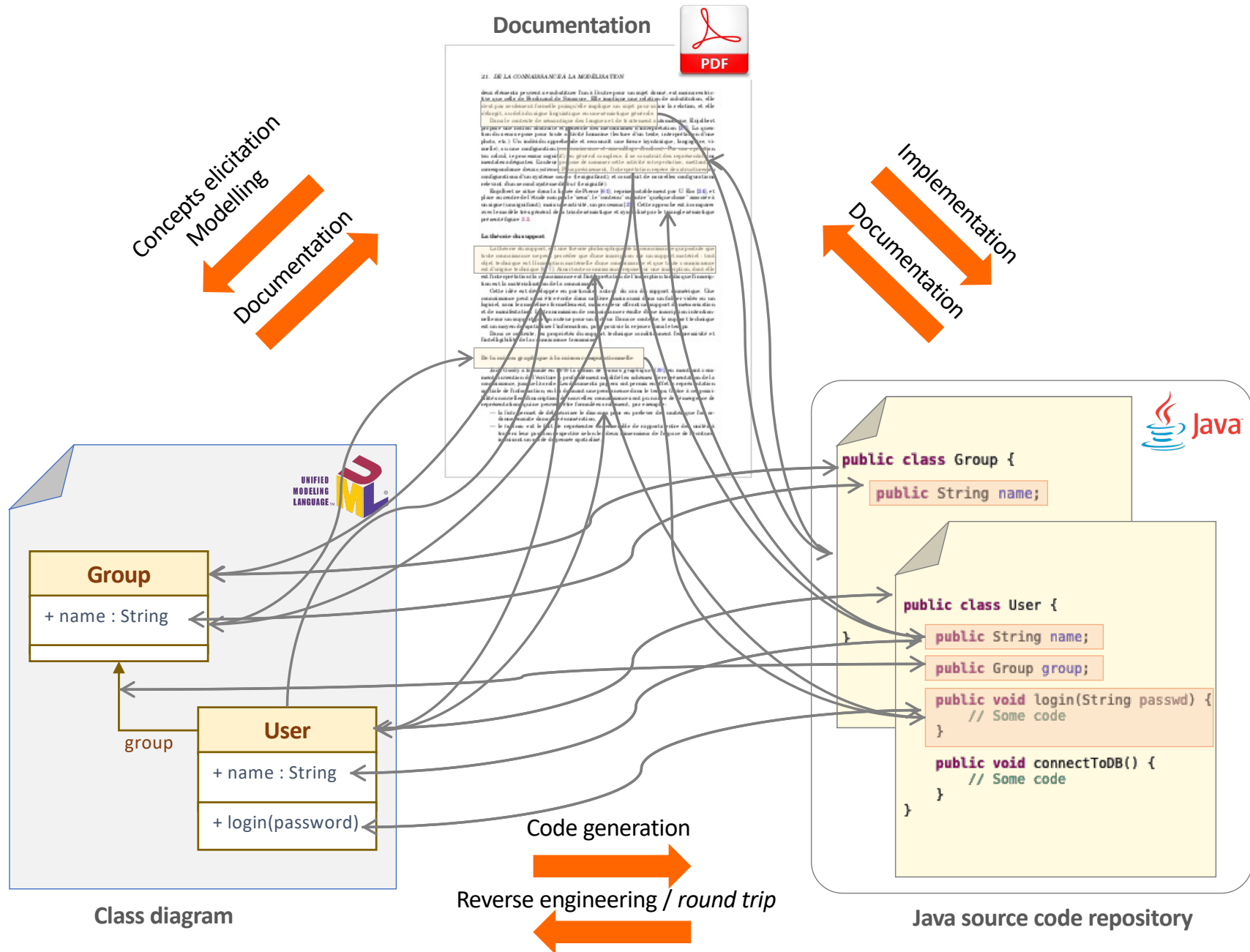**C4**. Source of Thruth management

Agenda

1. Introduction, context and research questions

2. Model federation

3. FML (Federation Modelling Language)

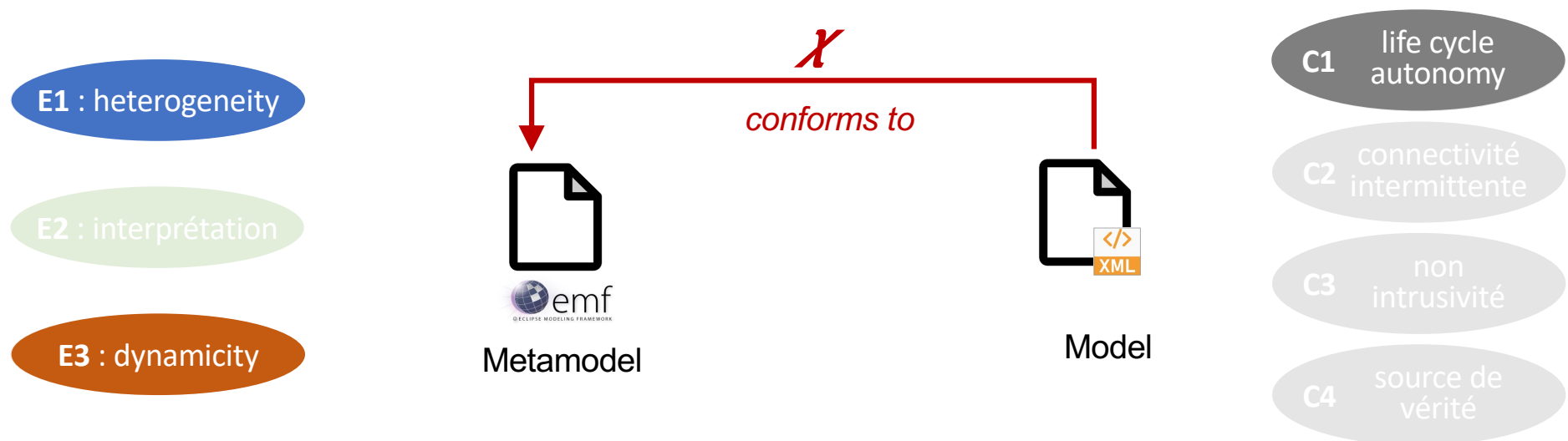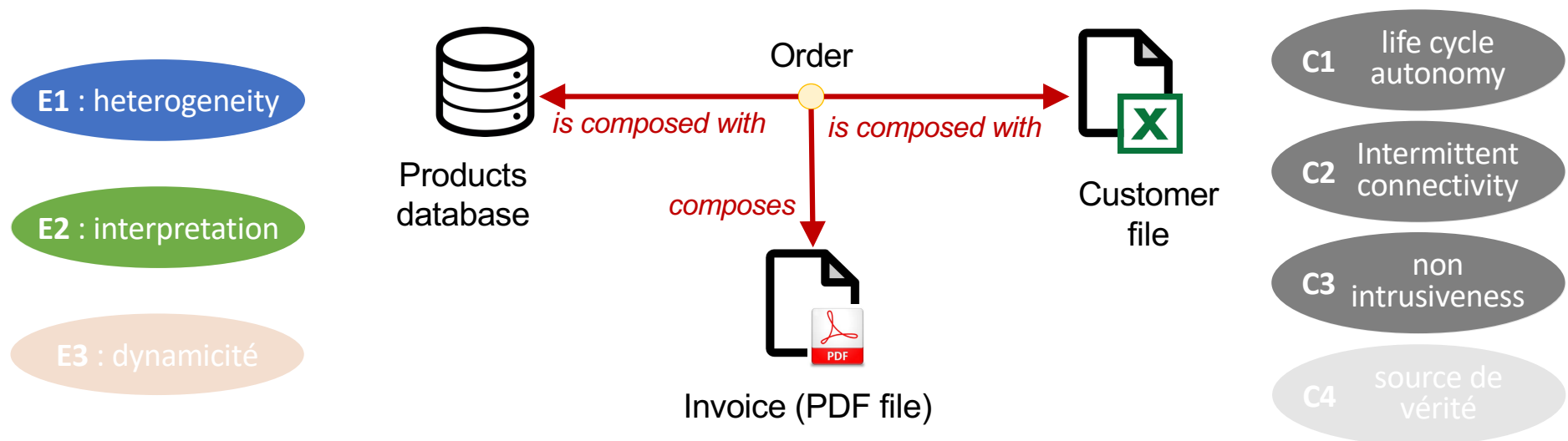4. Openflexo infrastructure

5. Conclusions

Agenda

Documentation

Concepts elicitation
Modelling

Documentation

Implementation

Documentation

Code generation

Reverse engineering / *round trip*

Class diagram

Dépôt de code source java

Documentation

Concepts elicitation
Modelling

Documentation

Implementation

Documentation

```java
public class Group {

    public String name;
```

```java
public class User {

    public String name;

    public Group group;

    public void login(String passwd) {
        // Some code
    }

    public void connectToDB() {
        // Some code
    }
}
```

Group

| + name : String |

group

User

| + name : String |
| + login(password) |

Code generation

Reverse engineering / *round trip*

Class diagram

Java source code repository

Some other modelling situations…

- Model construction

E1 : heterogeneity

E2 : interprétation

E3 : dynamicity

χ

conforms to

Metamodel

Model

C1 life cycle autonomy

C2 connectivité intermittente

C3 non intrusivité

C4 source de vérité

# Some other modelling situations…

- Model construction
- Model composition

E1 : heterogeneity

E2 : interpretation

E3 : dynamicité

Order

is composed with   is composed with

Products database

composes

Customer file

Invoice (PDF file)

C1 life cycle autonomy

C2 Intermittent connectivity

C3 non intrusiveness

C4 source de vérité

Some other modelling situations…

- Model construction
- Model composition
- Model mapping

alignment management

E1 : heterogeneity

E2 : interpretation

E3 : dynamicity

EMF model

Mapping model

OWL ontology

C1 life cycle autonomy

C2 Intermittent connectivity

C3 non intrusiveness

C4 source of thruth

Some other modelling situations...

- Model construction
- Model composition
- Model mapping
- Model edition
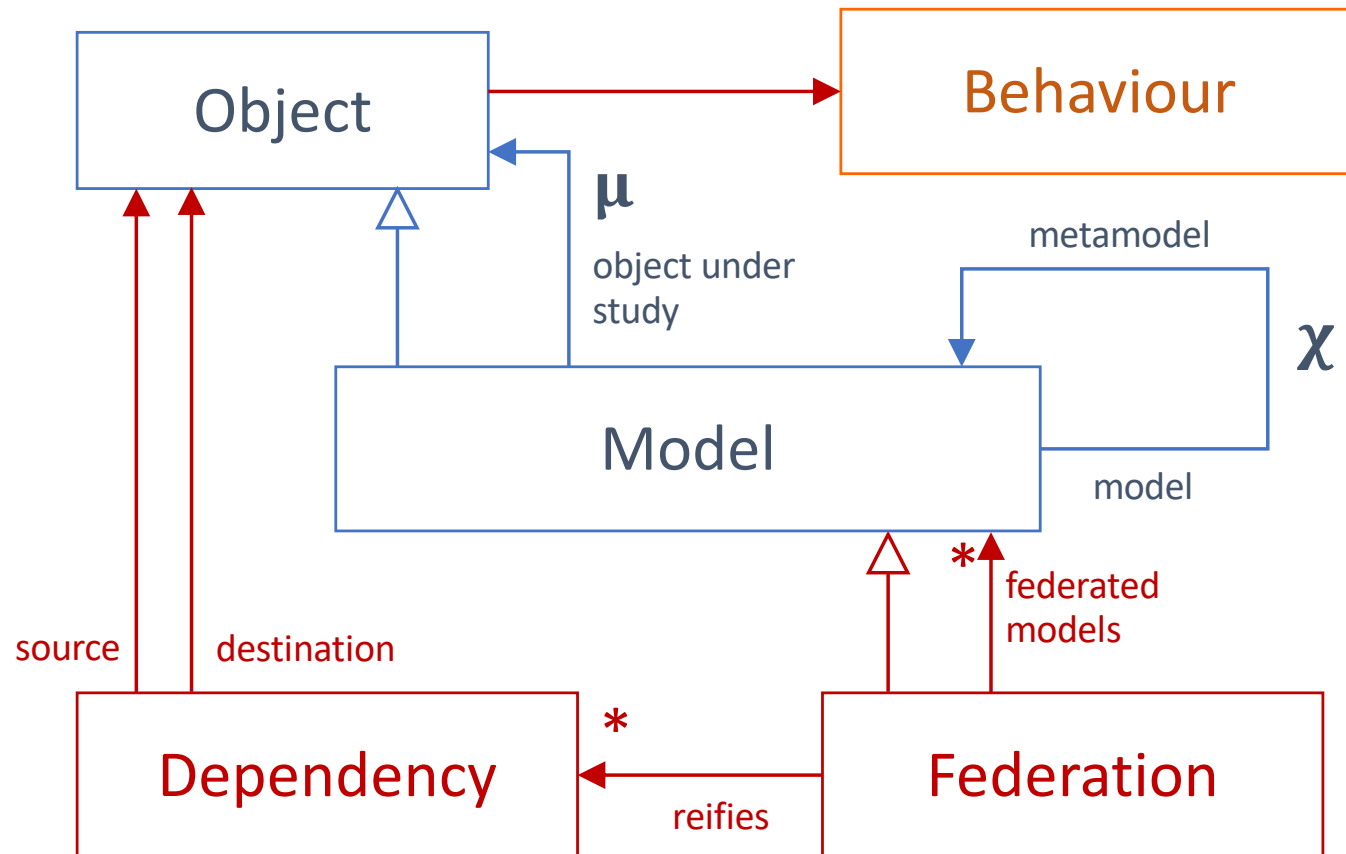
E1 : heterogeneity

E2 : interpretation

E3 : dynamicity

represents

is edited with

UML model

Diagram
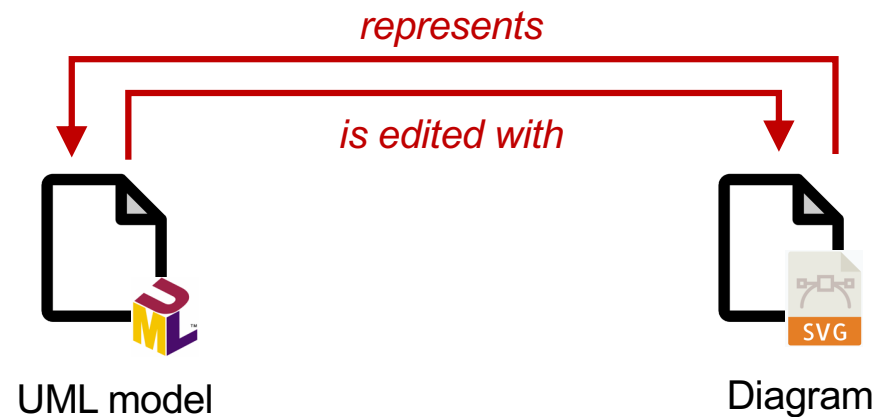
C1 life cycle autonomy

C2 connectivité intermittente

C3 non intrusiveness

C4 source de vérité

Model federation : a conceptual approach

F. Dagnat, S. Guérin, A. Beugnard, S Martínez, J Champeau, J.C. Bach *Working with Multiple Models through Model Federation*
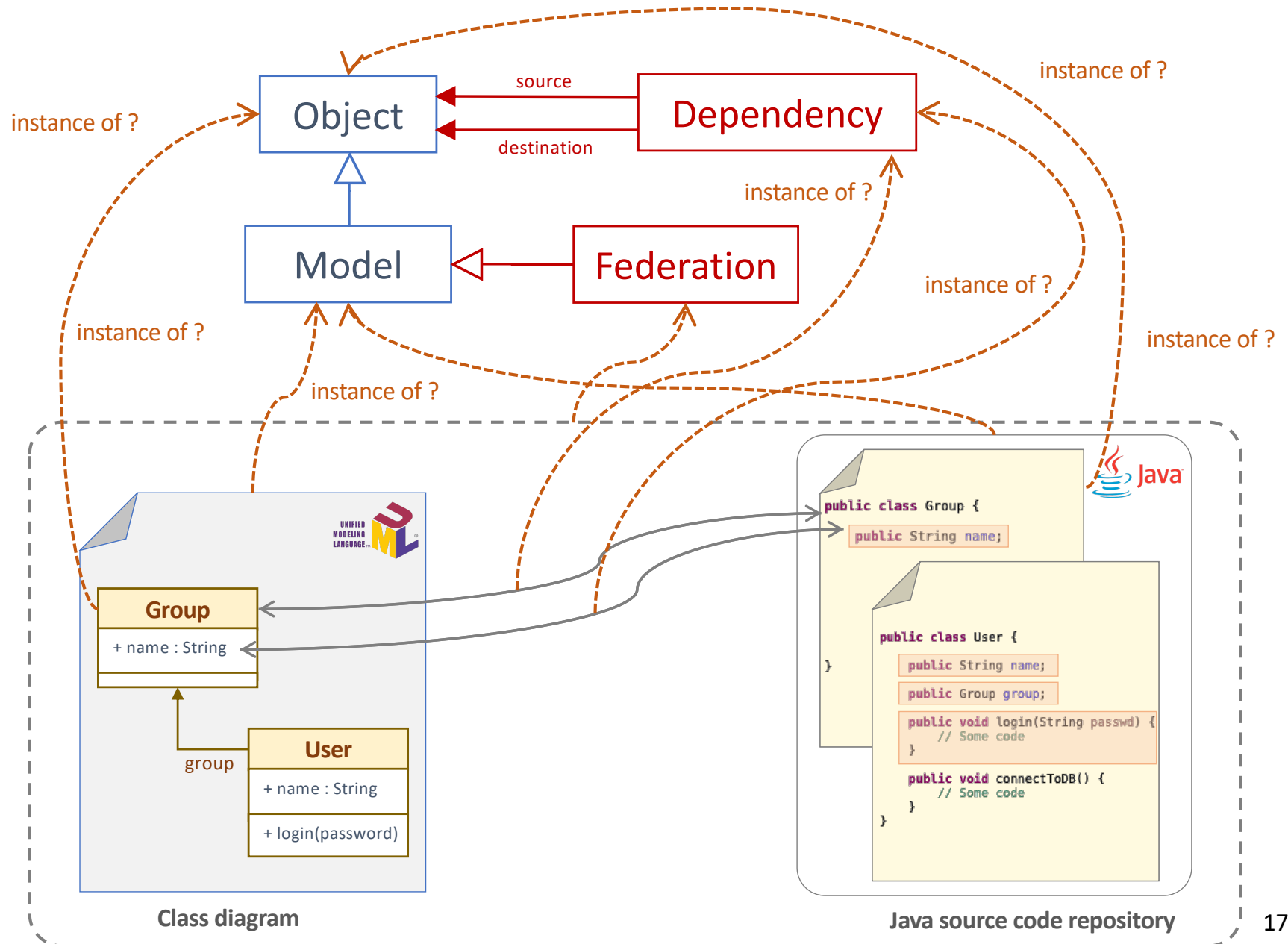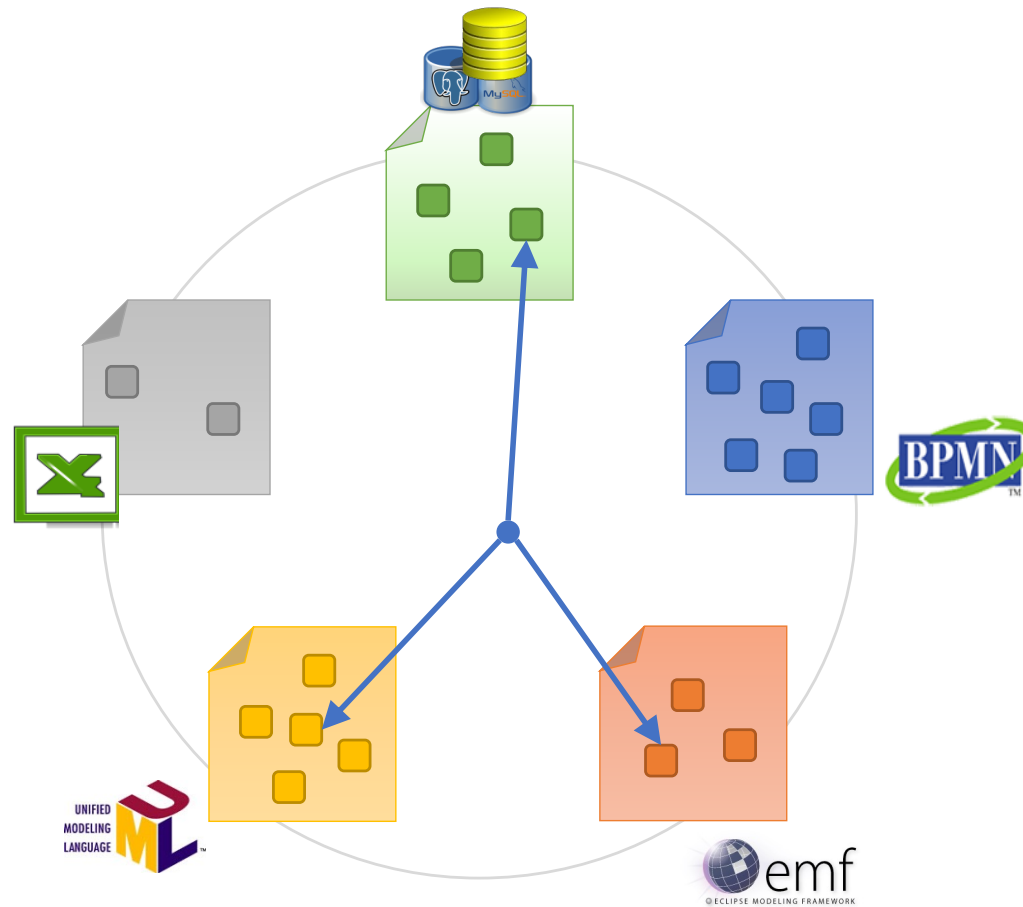
14 /36

Agenda

1. Introduction, context and research questions

2. Model federation

3. FML (Federation Modelling Language)
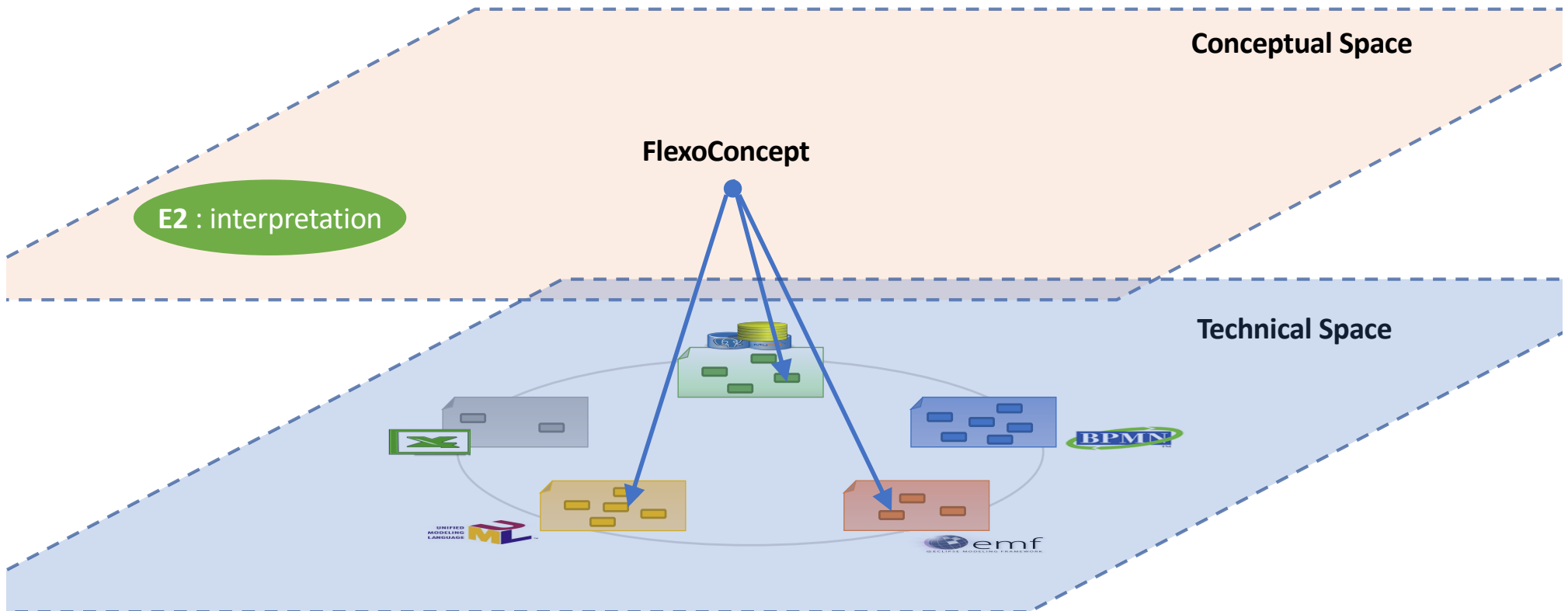
4. Openflexo infrastructure

5. Conclusions

Documentation

PDF

Concepts elicitation
Modelling

Documentation

Implementation

Documentation

UNIFIED
MODELING
LANGUAGE™

Java

```java
public class Group {

    public String name;
```

Group

+ name : String

```java
public class User {

    public String name;

    public Group group;

    public void login(String passwd) {
        // Some code
    }

    public void connectToDB() {
        // Some code
    }
}
```

group

User

+ name : String

+ login(password)

Code generation

Reverse engineering / *round trip*

Class diagram

Java source code repository
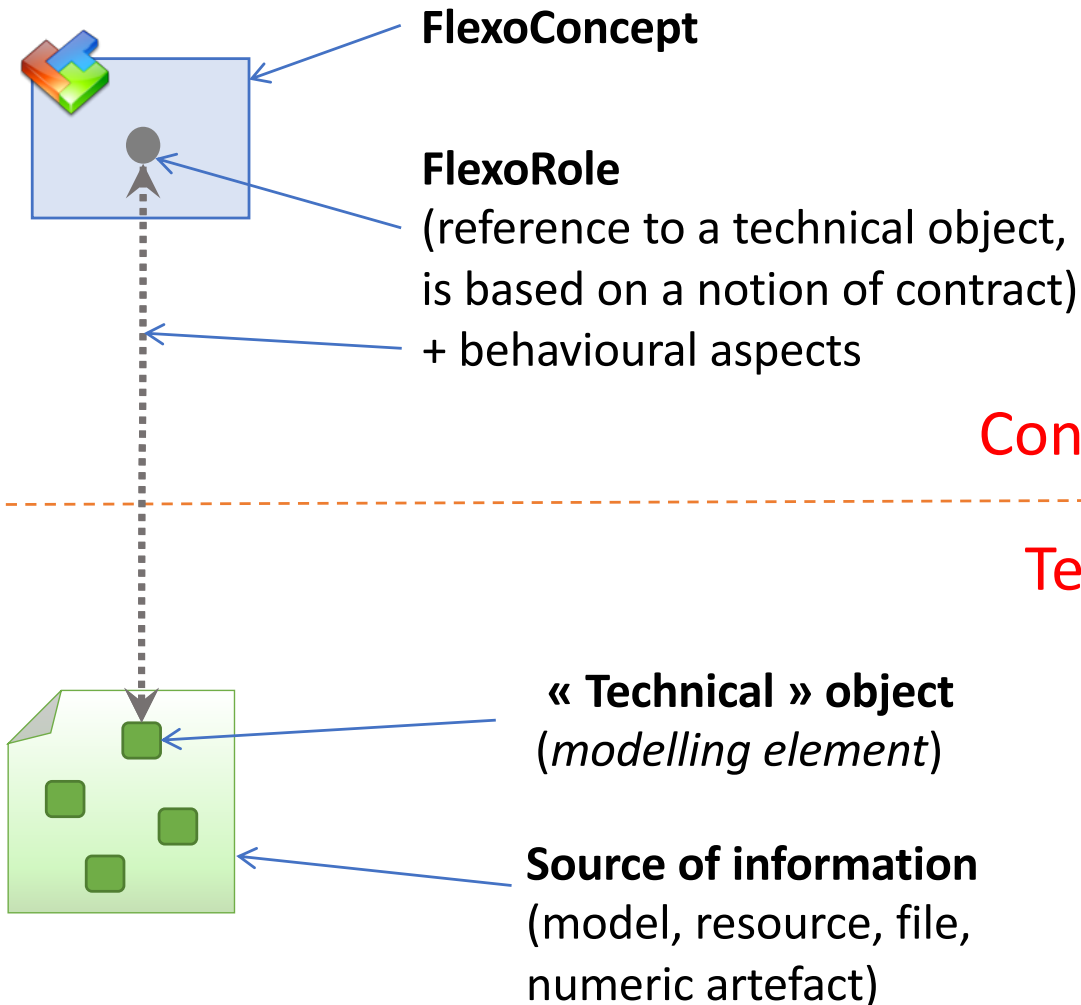
Principles and design choices for FML

1. Modelling spaces partition

- **Conceptual Space**: carries conceptual semantics
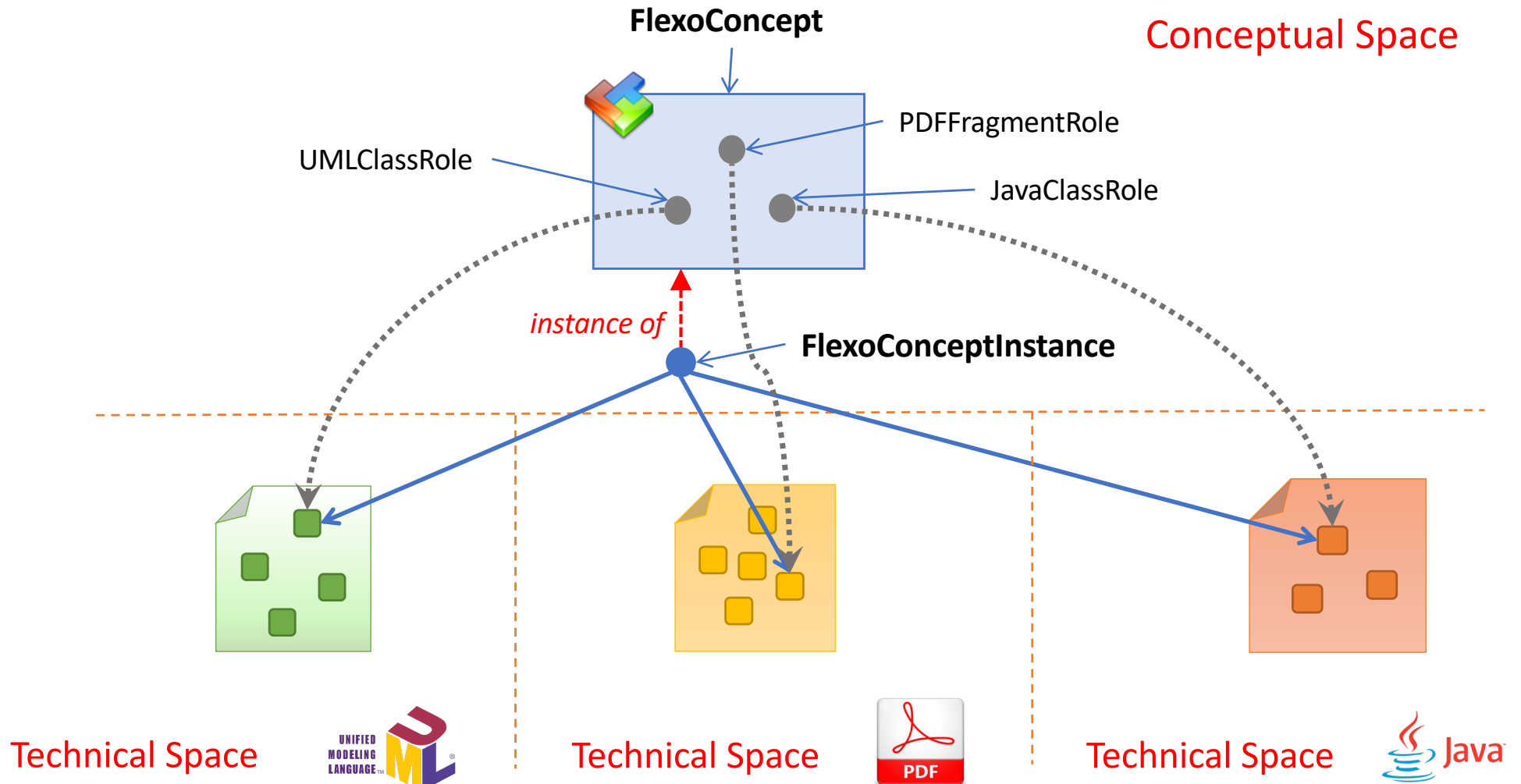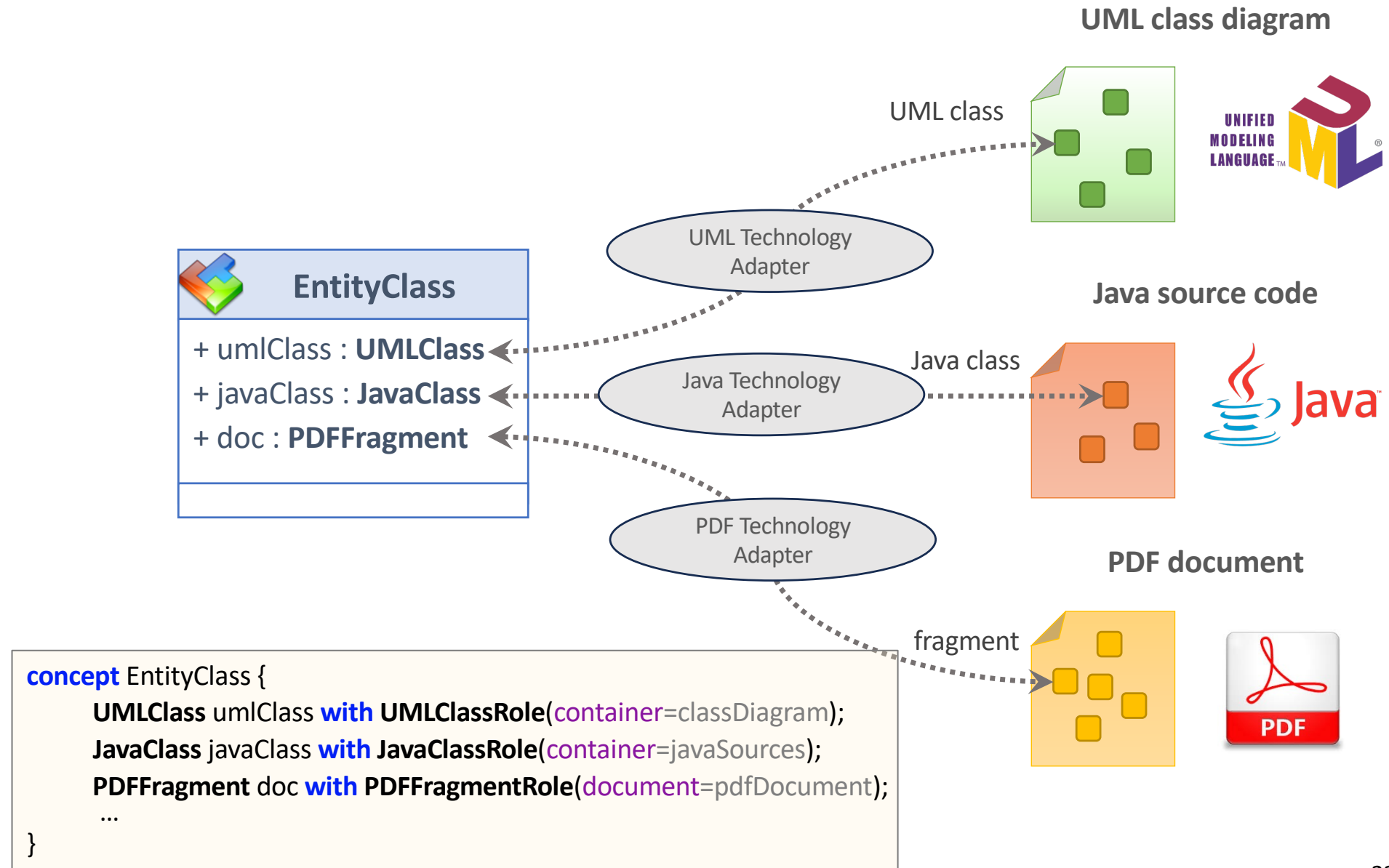- **Technical Space**: hosts federated models
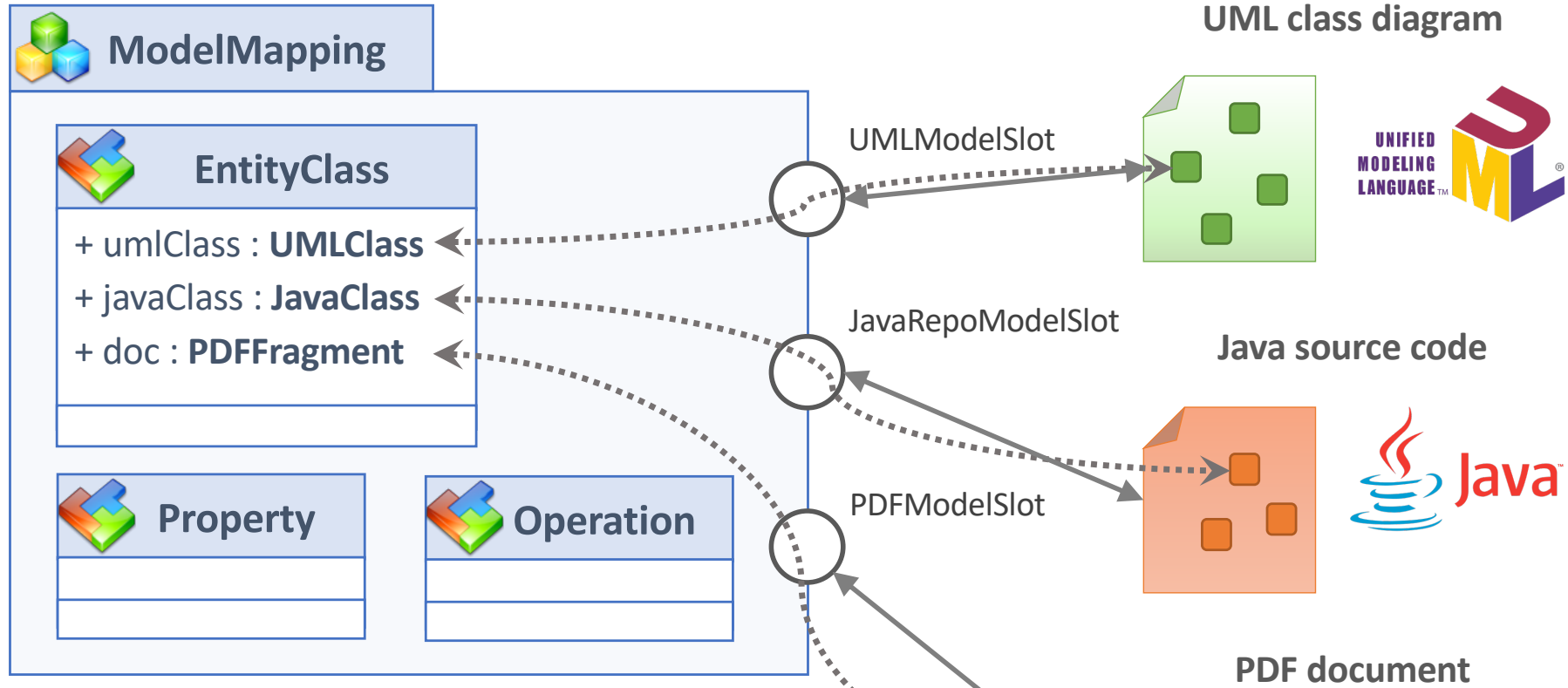
2. Designation mechanism

FlexoConcept

E1 : heterogeneity

FlexoRole
(reference to a technical object,
is based on a notion of contract)
+ behavioural aspects

Conceptual Space

Technical Space

« Technical » object
(*modelling element*)

Source of information
(model, resource, file,
numeric artefact)

UNIFIED MODELING LANGUAGE™

3. Conceptual Space structuration

FlexoConcept

Conceptual Space

UMLClassRole

PDFFragmentRole

JavaClassRole

instance of

FlexoConceptInstance

Technical Space

Technical Space

Technical Space

UML class diagram

UML class

Java source code

EntityClass

+ umlClass : **UMLClass**
+ javaClass : **JavaClass**
+ doc : **PDFFragment**

UML Technology Adapter

Java Technology Adapter

Java class

PDF Technology Adapter

PDF document

fragment

```
concept EntityClass {
    UMLClass umlClass with UMLClassRole(container=classDiagram);
    JavaClass javaClass with JavaClassRole(container=javaSources);
    PDFFragment doc with PDFFragmentRole(document=pdfDocument);
     ...
}
```

UML class diagram

ModelMapping

EntityClass

+ umlClass : **UMLClass**

+ javaClass : **JavaClass**

+ doc : **PDFFragment**

Property   **Operation**

UMLModelSlot

JavaRepoModelSlot

Java source code

PDFModelSlot
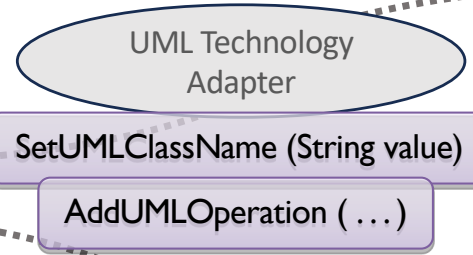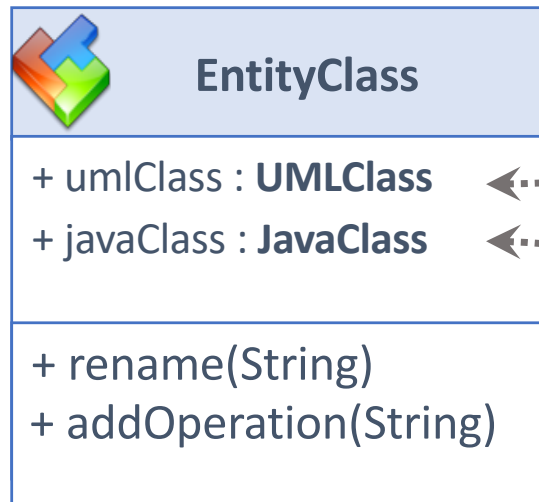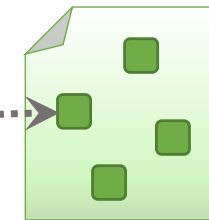
PDF document

```
model ModelMapping {
    UMLClassDiagram classDiagram with UMLModelSlot( . . . );
    SourceRepository javaSources with JavaRepoModelSlot(jdk="1.7+");
    PDFDocument pdfDocument with PDFModelSlot ( . . . );
    concept EntityClass { . . . }
    concept Property { . . . }
    concept Operation { . . . }
}
```
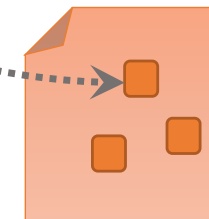
Behavioural aspects

UML class diagram

EntityClass

+ umlClass : **UMLClass**
+ javaClass : **JavaClass**

+ rename(String)
+ addOperation(String)

UML Technology Adapter

SetUMLClassName (String value)

AddUMLOperation (…)

Java Technology Adapter

RenameFileName (String fileName)

CreateMethod (…)
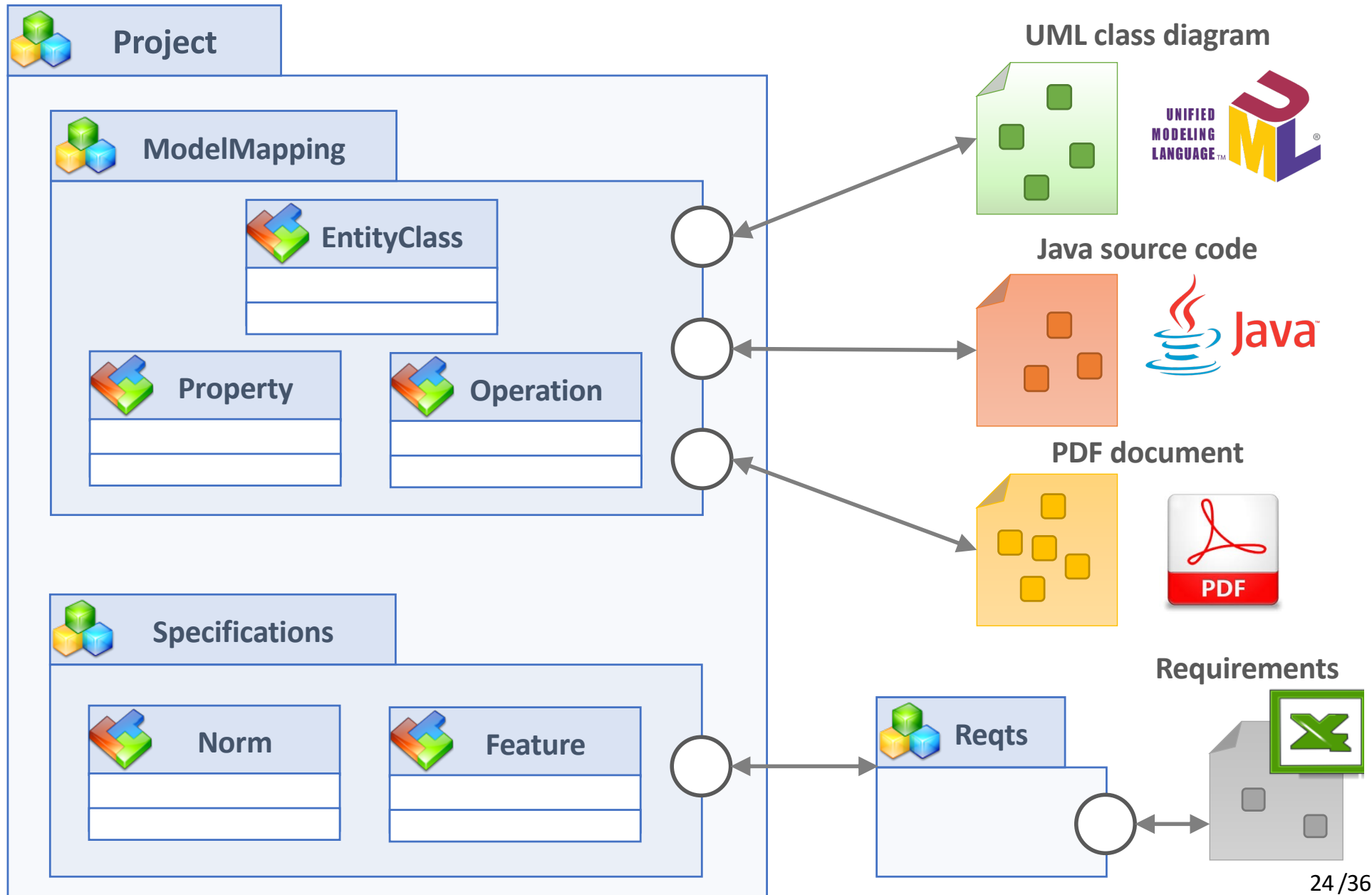
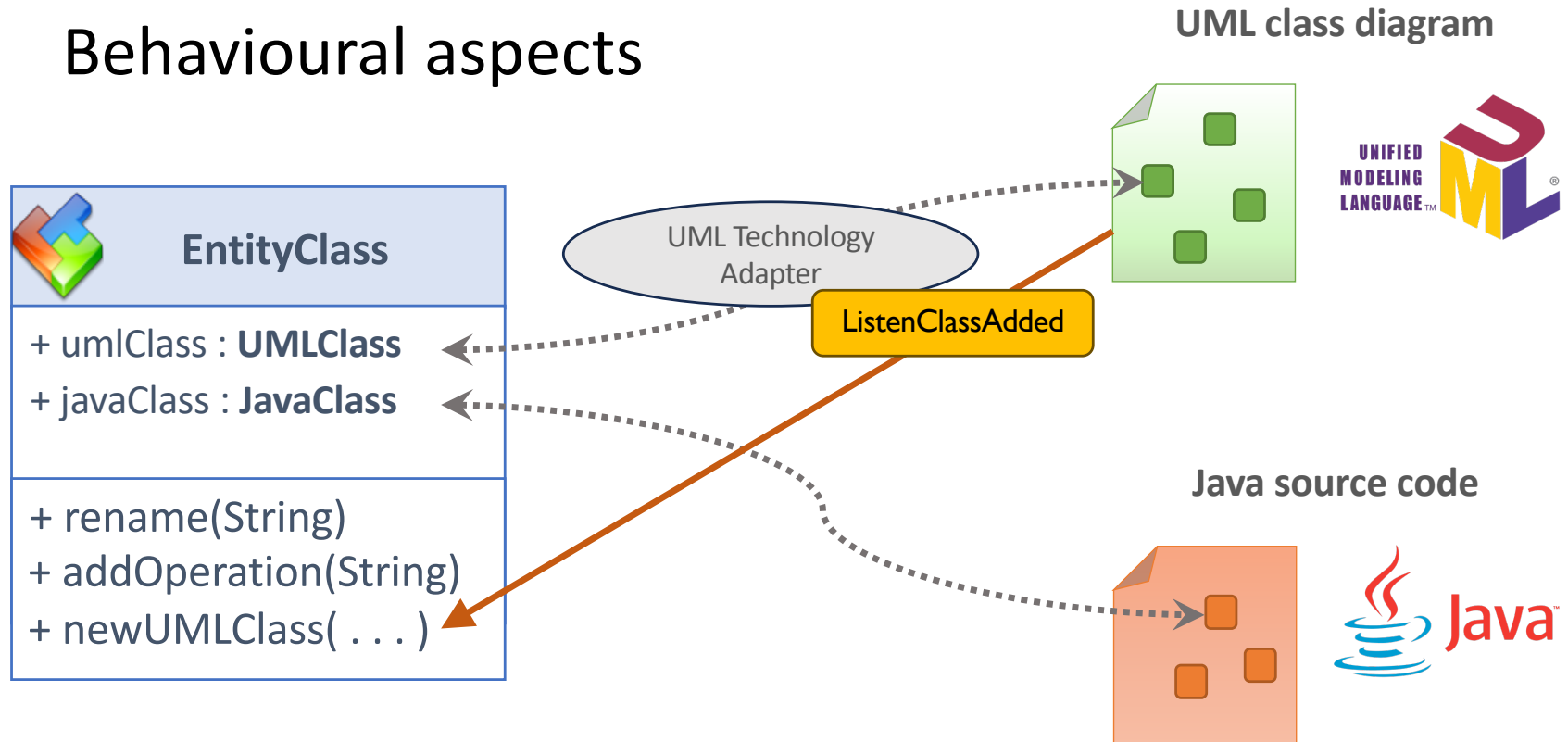**Java source code**

```
concept EntityClass {
    . . .
    rename (String newName) {
        UML::SetUMLClassName(newName) in umlClass;
        JAVA::RenameFileName(newName) in javaClass;
    }
}
```

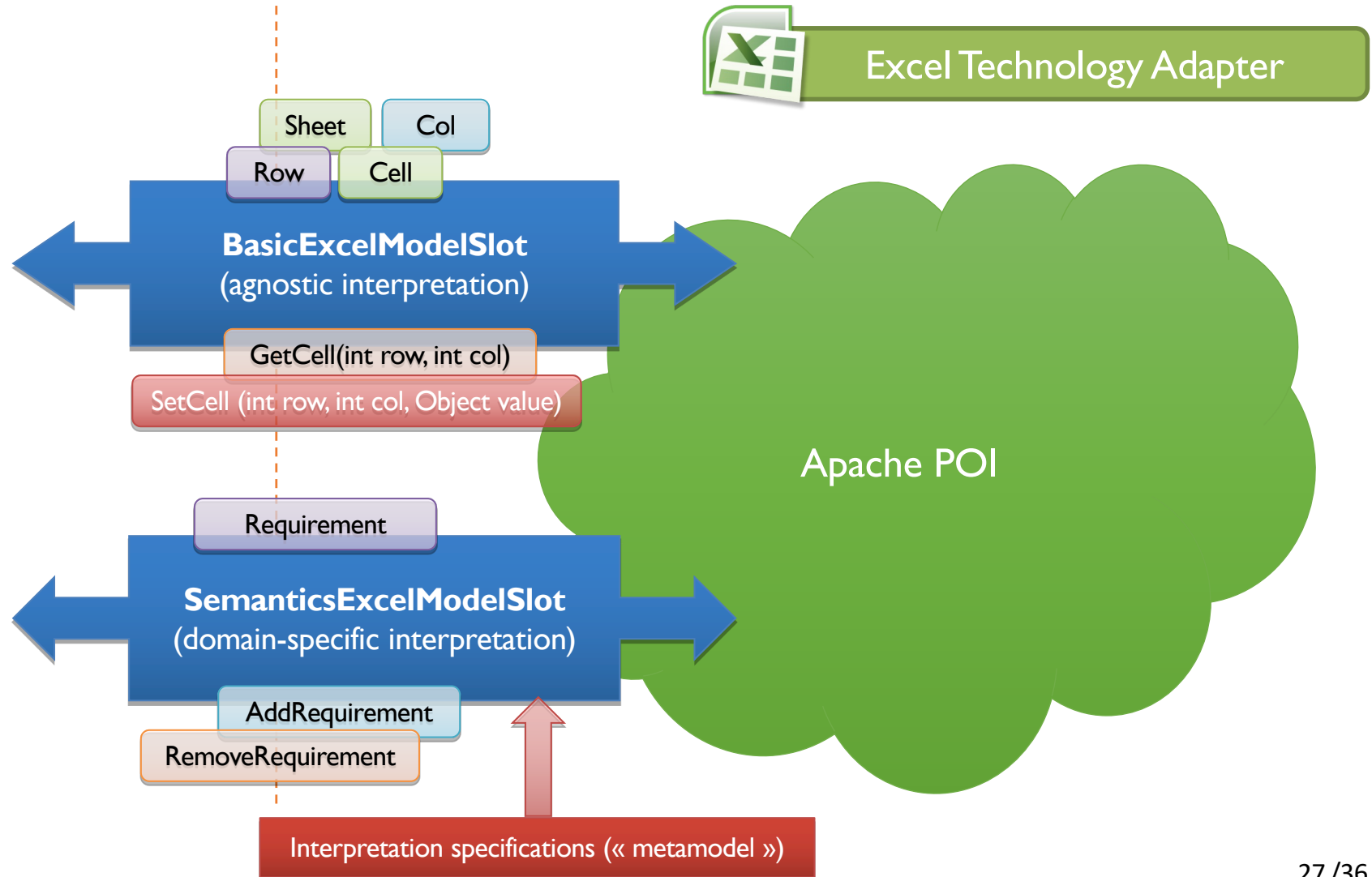**E3** : dynamicity

Behavioural aspects

UML class diagram

EntityClass

+ umlClass : **UMLClass**
+ javaClass : **JavaClass**

+ rename(String)
+ addOperation(String)
+ newUMLClass(. . .)

UML Technology Adapter

ListenClassAdded

Java source code

```
concept EntityClass {
    . . .
    newUMLClass (UMLClass newClass)
        with ListenClassAdded(observed=classDiagram) {
        . . .
    }
}
```
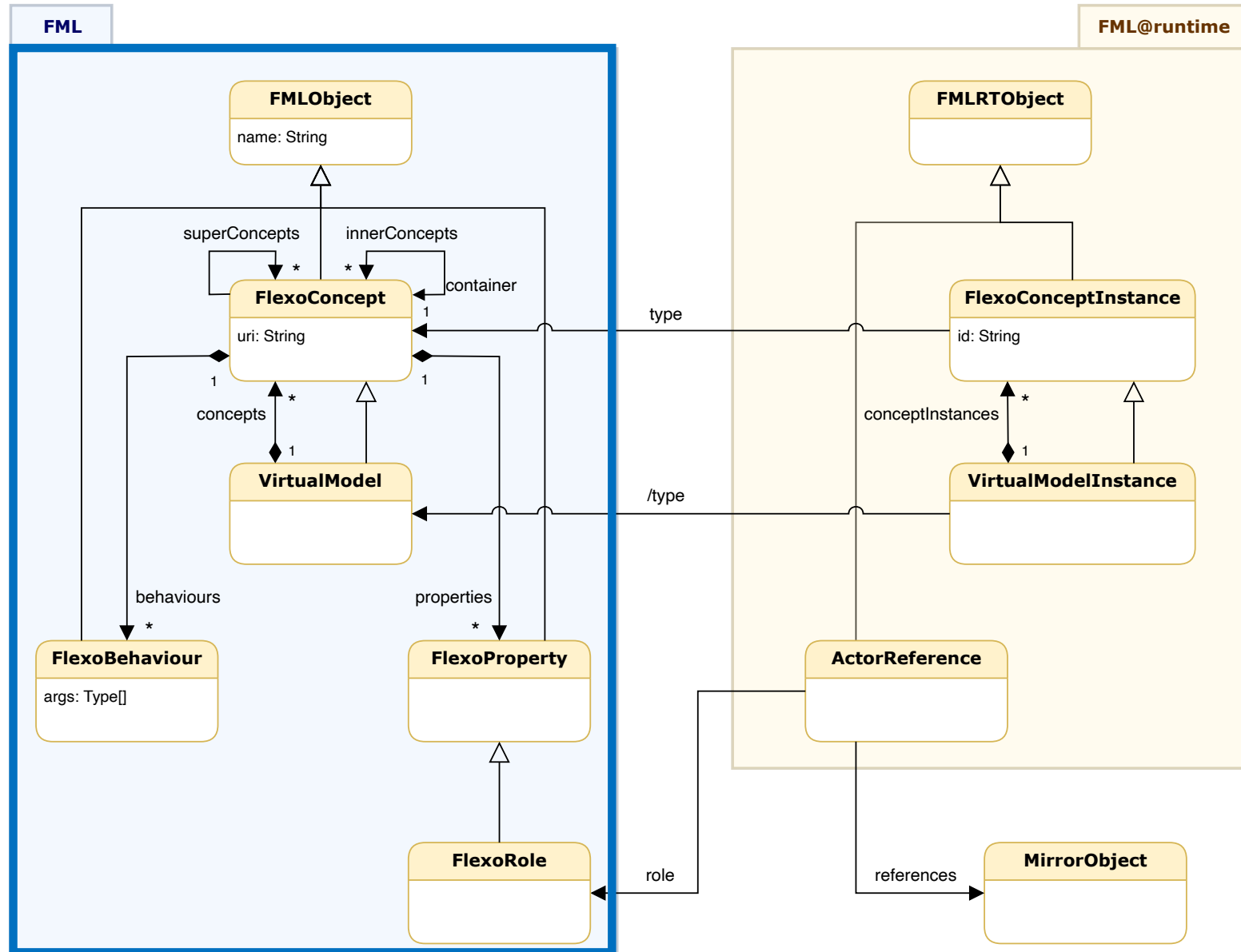
E3 : dynamicity

C1 life cycle autonomy
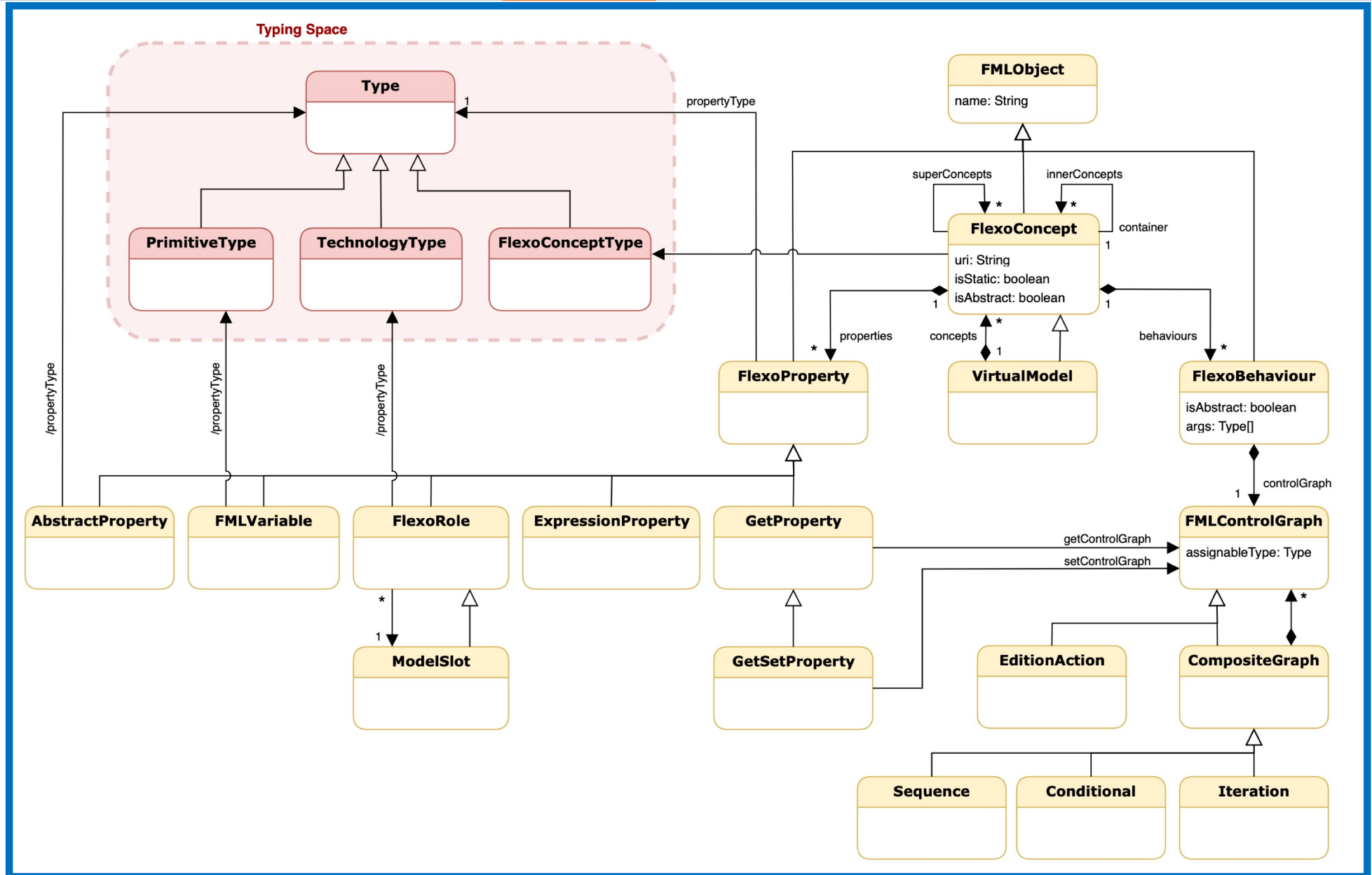
C2 intermittent connectivity

Conceptual Space   Technical Space

E1 : heterogeneity

Excel Technology Adapter

Sheet   Col

Row   Cell

BasicExcelModelSlot
(agnostic interpretation)

GetCell(int row, int col)

SetCell (int row, int col, Object value)

Apache POI

Requirement

SemanticsExcelModelSlot
(domain-specific interpretation)

AddRequirement

RemoveRequirement

Interpretation specifications (« metamodel »)

FML language

- Modelling spaces partition

- Designation mechanism, technology adapter

- Conceptual space structuration

  - FlexoConcept / VirtualModel : structure + behaviour

  - Object-oriented approach

  - Types/instances partition (ontological instantiation)

  - Abstraction, genericity, modularity, composability

- Interpreted langage

- Imperative langage (control structures)

- Static and strong typing, composed type system

- FML/Connie expression language

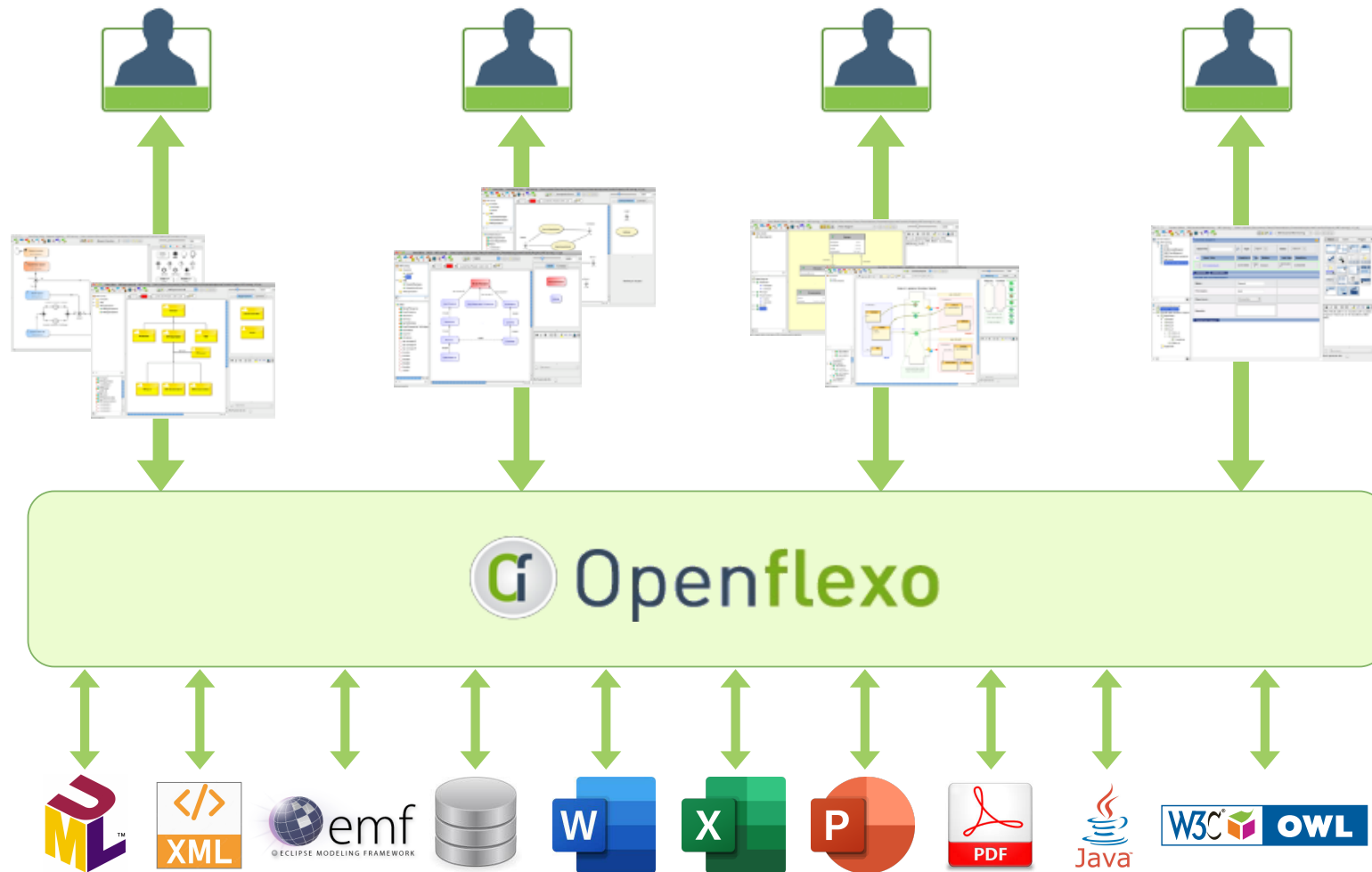- Multiples concrete syntaxes (textual and graphical)

E1 : heterogeneity
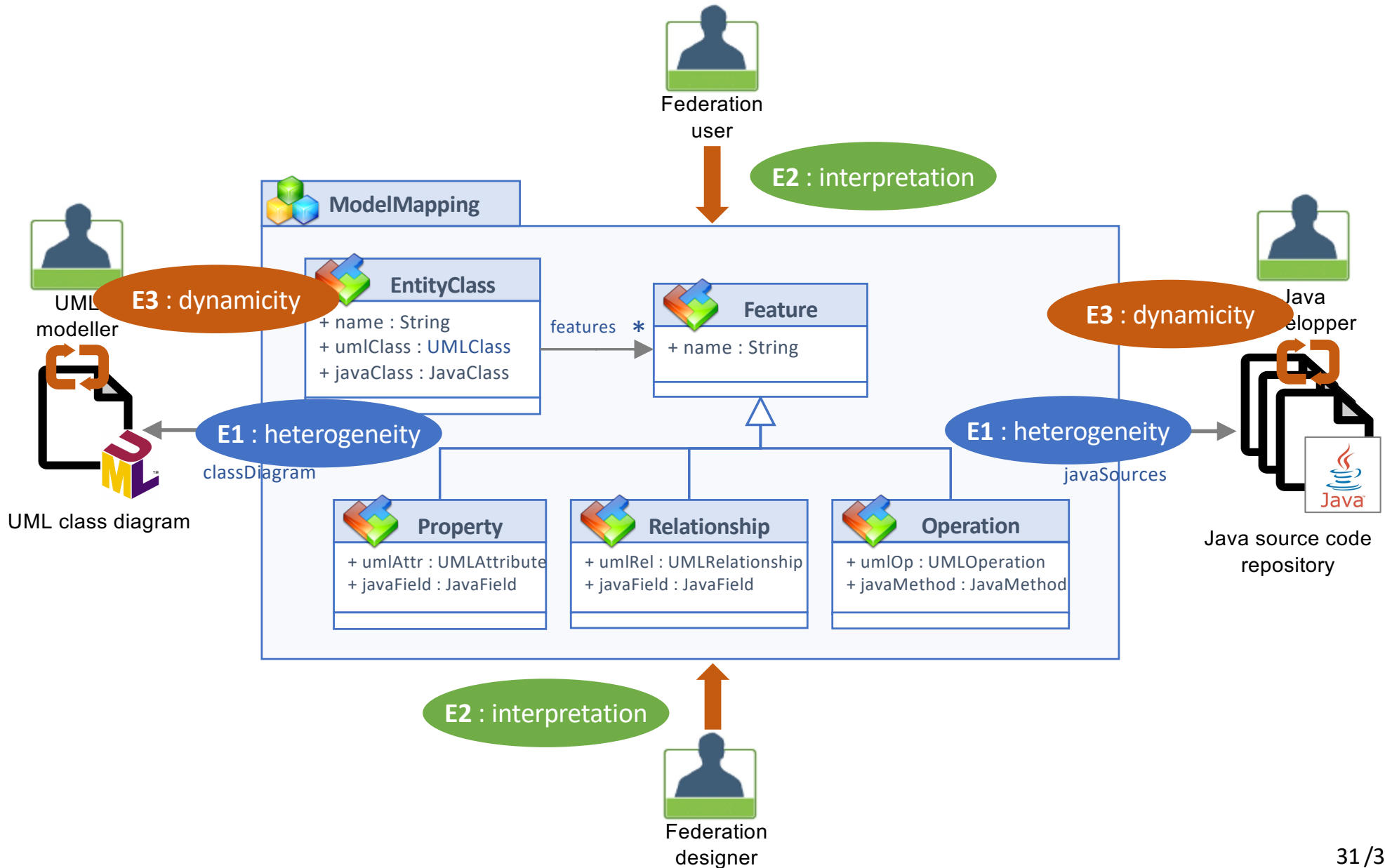
E2 : interpretation

E3 : dynamicity

Agenda

1. Introduction, context and research questions

2. Model federation

3. FML (Federation Modelling Language)

4. Openflexo infrastructure

5. Conclusions

Openflexo software infrastructure

- FML language implementation
  - FML integrated development environment (IDE)
  - FML execution environment
  - Dedicated tools (scripting language, terminal, console, debugger, etc.)
- Technology Adapters
  - (EMF, UML, BPMN, OWL, Word, Excel, PowerPoint, XML, JDBC, PDF, Rest, etc.)
- Infrastructure
  - Reusable software components (ex: PAMELA, *Model Oriented Programming*)
  - Graphical tools (Java/Swing technology)
  - Packaged applications (Java/Swing)
  - Openflexo server (web)
  - Reusable technical and business model libraries
- Free software with double licencing EUPL/GPLv3
- https://github.com/openflexo-team

Agenda

1. Introduction, context and research questions

2. Model federation

3. FML (Federation Modelling Language)

4. Openflexo infrastructure

5. Conclusions

Conclusions and perspectives

- Explanation and conceptualization of the model federation approach

- The FML language

- Openflexo software infrastructure

- Validation on many use cases (research projects and industrial experiments)

- Model federation as enabler of new usages

  - Industry 4.0 and digital twins

  - Reverse engineering of software and systems

  - Co-evolution

  - Model execution

  - Free modelling

  - …

http://www.openflexo.org