Project Presentation #2

Synchronization between display objects and representation templates in graphical language construction

François Helg
Fabien Rohrer

Assistant :
Frédéric Fondement

Overview

- François
  - Reminder
    - Goal of the project
    - Theoretical issues
    - Problematic
    - Key technologies
- Fabien
  - What we have already done
  - What we're working on
  - Problems
  - To do

Plan

- Reminder
- What we have already done
- What we're working on
- Problems
- To do

Reminder
General issues

- This project concentrates on defining a graphical concrete syntax for a language, if the abstract syntax is given.
- Tool a generic language editor
  - Part of TopModL project

Reminder
Define a language

- Definition :
  - Abstract synthax : MetaModel
  - Concrete synthax : Graphical templates
- Instanciation :
  - Abstract : Model
  - Concrete : Graphical representation

Reminder
ProBXS

- Project of Fabrice Hong (Semester Project in Winter 2004)
- Concrete syntax graphical edition tool
- « SVG image becomes an editing tool »
  - Display representation based on SVG templates
  - DopiDom extension (dynamical behavior)

Reminder
DoPIDom

- Tool to build interactive environnements, masking the concept of application from user point of view
- Based on : DOM (documents)
  SVG (their representation)
- DoPIDom defines the concept of component & defines the behaviour that each component can have (Consumable actions or queries)
- DoPIDom also defines the concept of instrument & actions they can produce

Reminder
DoPIDom

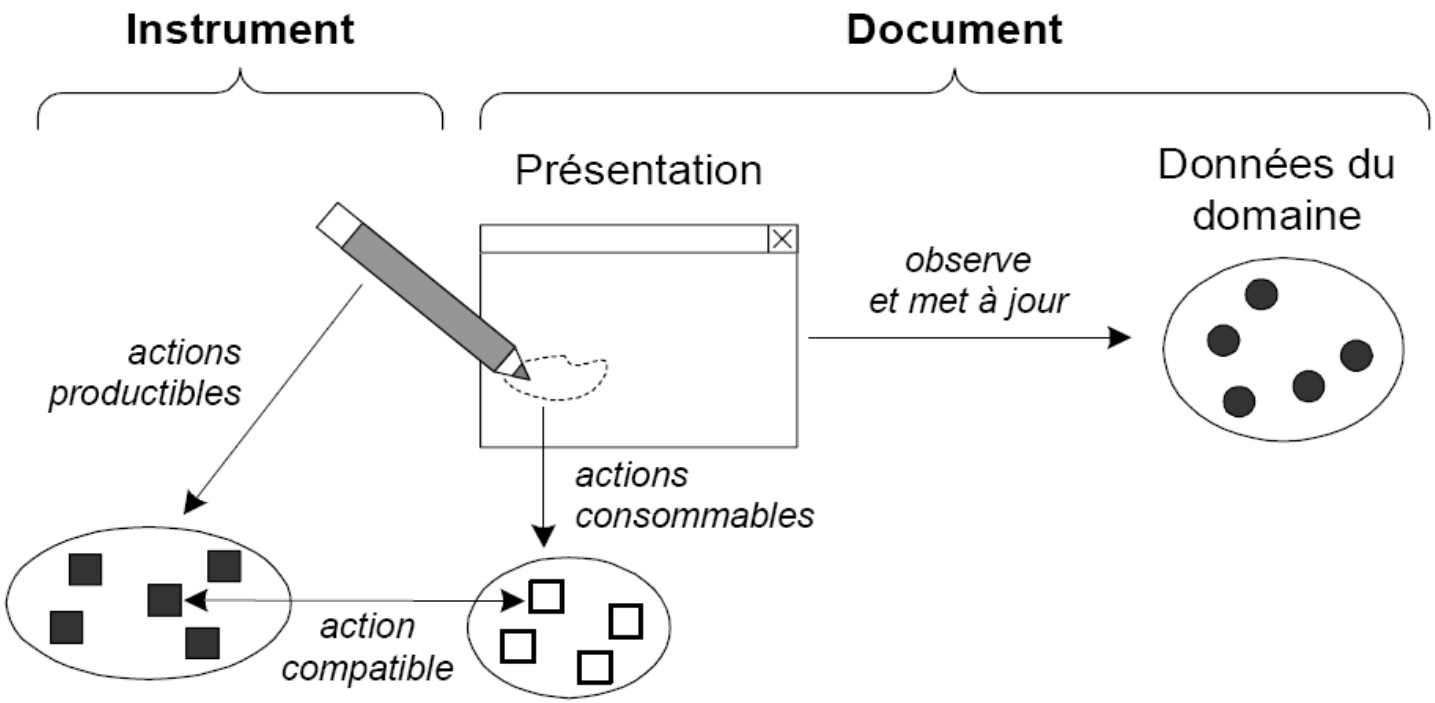


Figure 1: Principe du modèle DPI

Reminder
DoPIDom

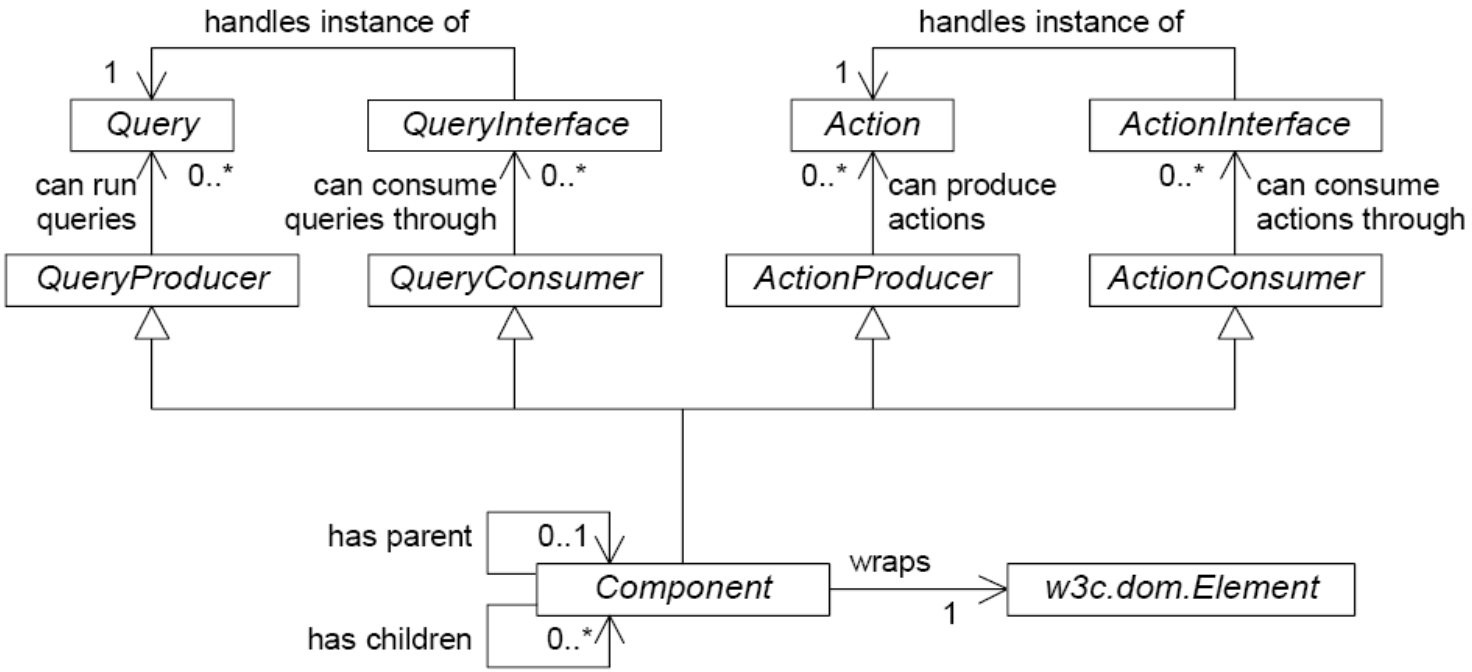
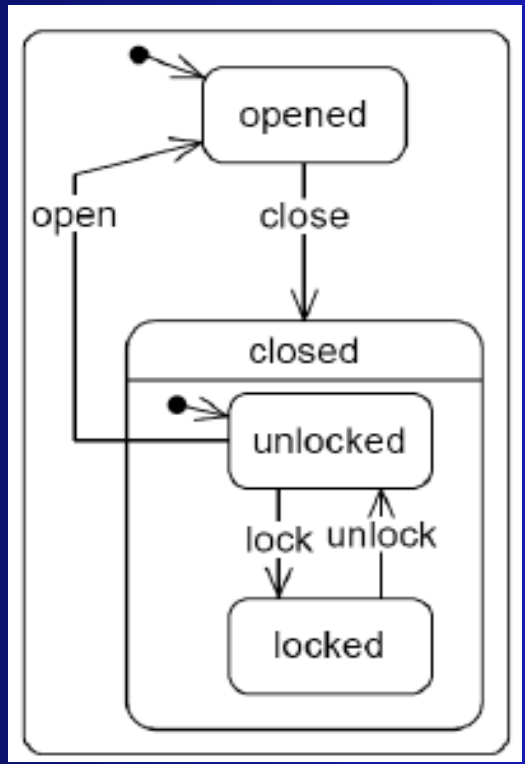


Figure 2: Modèle UML des composants interactifs

Reminder
Exemple of interaction

- Change the name of a state in statechart language



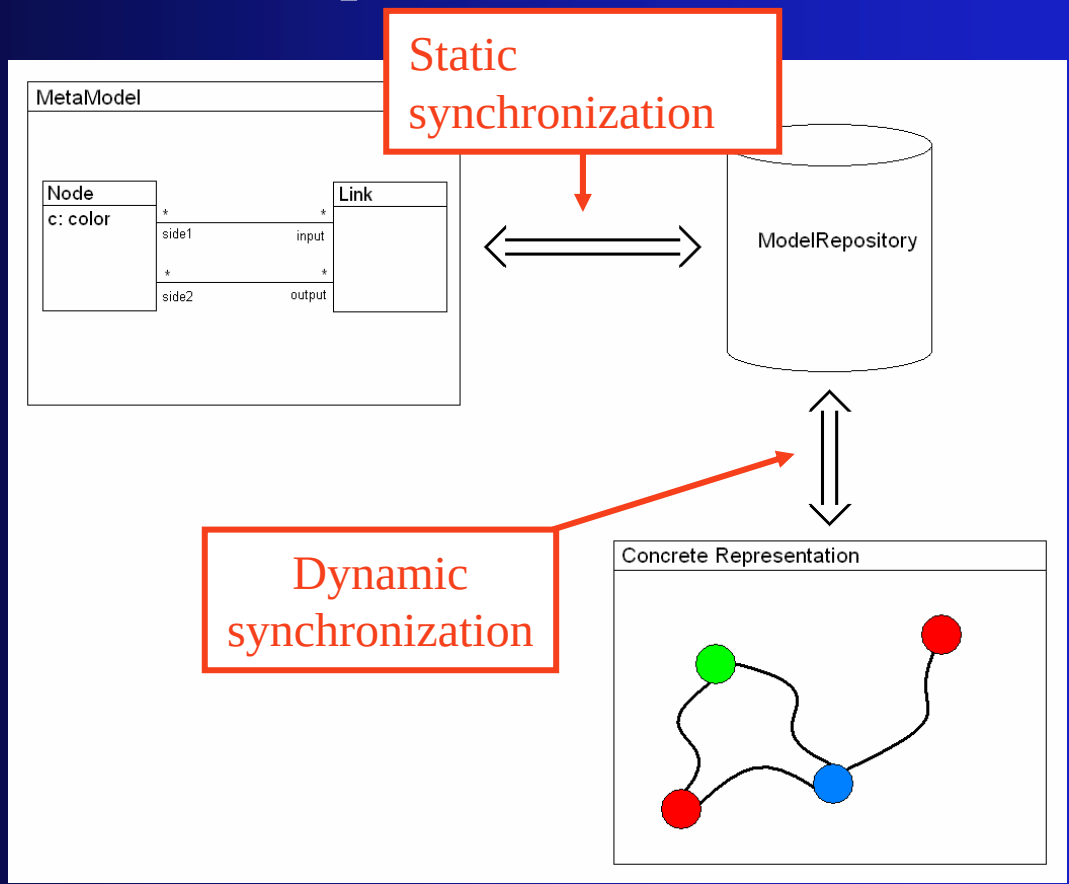
Reminder
Exemple of interaction

- Component « State » can consume the action « changeName » because it implements the interface « Namechangable »
- The interactor produces the action by clicking on the component and changing the name.

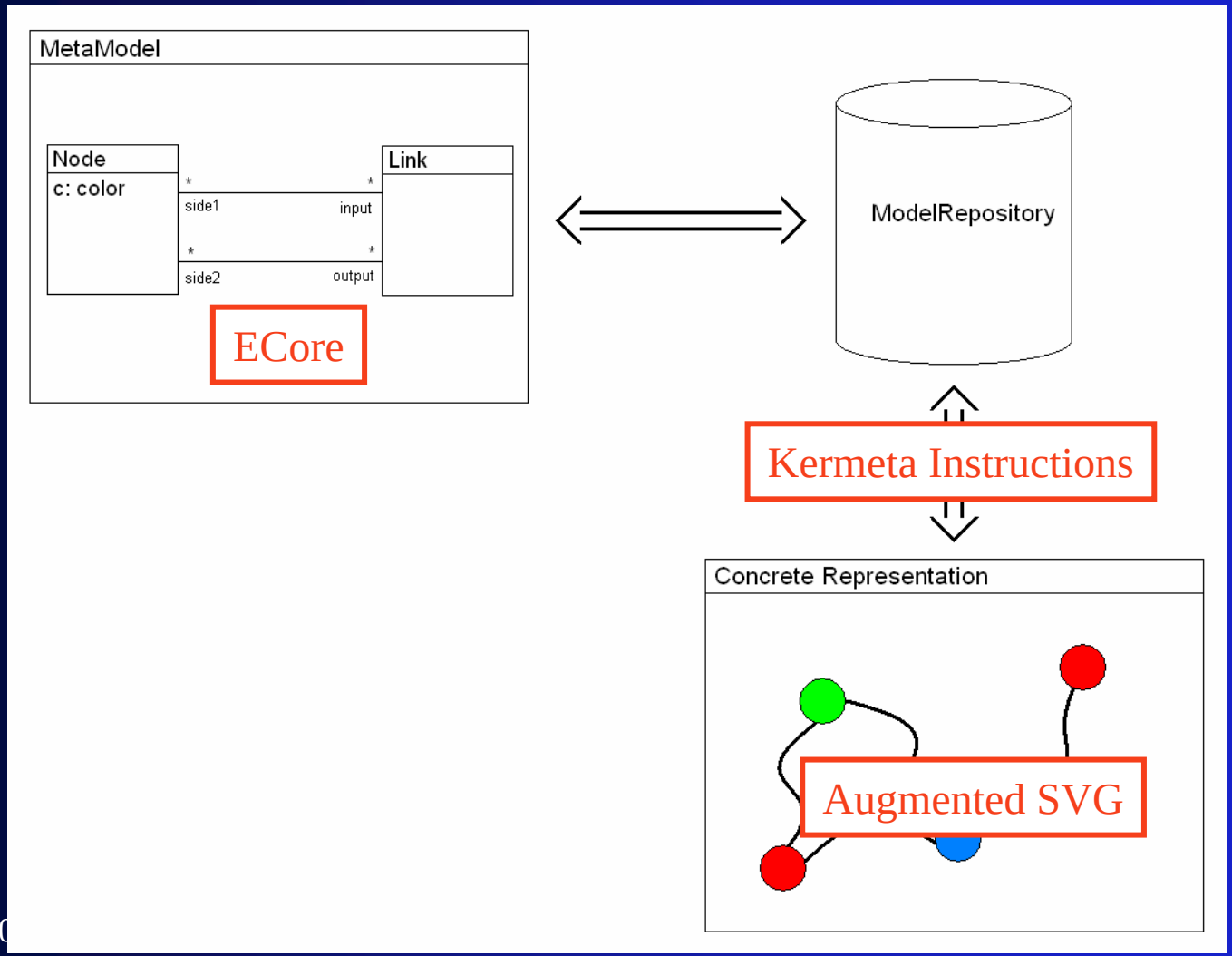
=> Our task now (rising synchronization) is to imforme the model that the name changed in the graphical representation.

Reminder
Problematic

- Need to keep synchronized display objects and representation templates



Reminder
Chosen technologies



Reminder
KerMeta

- What is KerMeta?
  - Metamodeling language
  - Defines
    - Structure of metamodel
    - Behavior of metamodel
  - High level of abstraction
    - We don’t have to deal with the concept of repository
    - We just handle KerMeta’s object and operations.

Switch

- Fabien will continue the presentation…

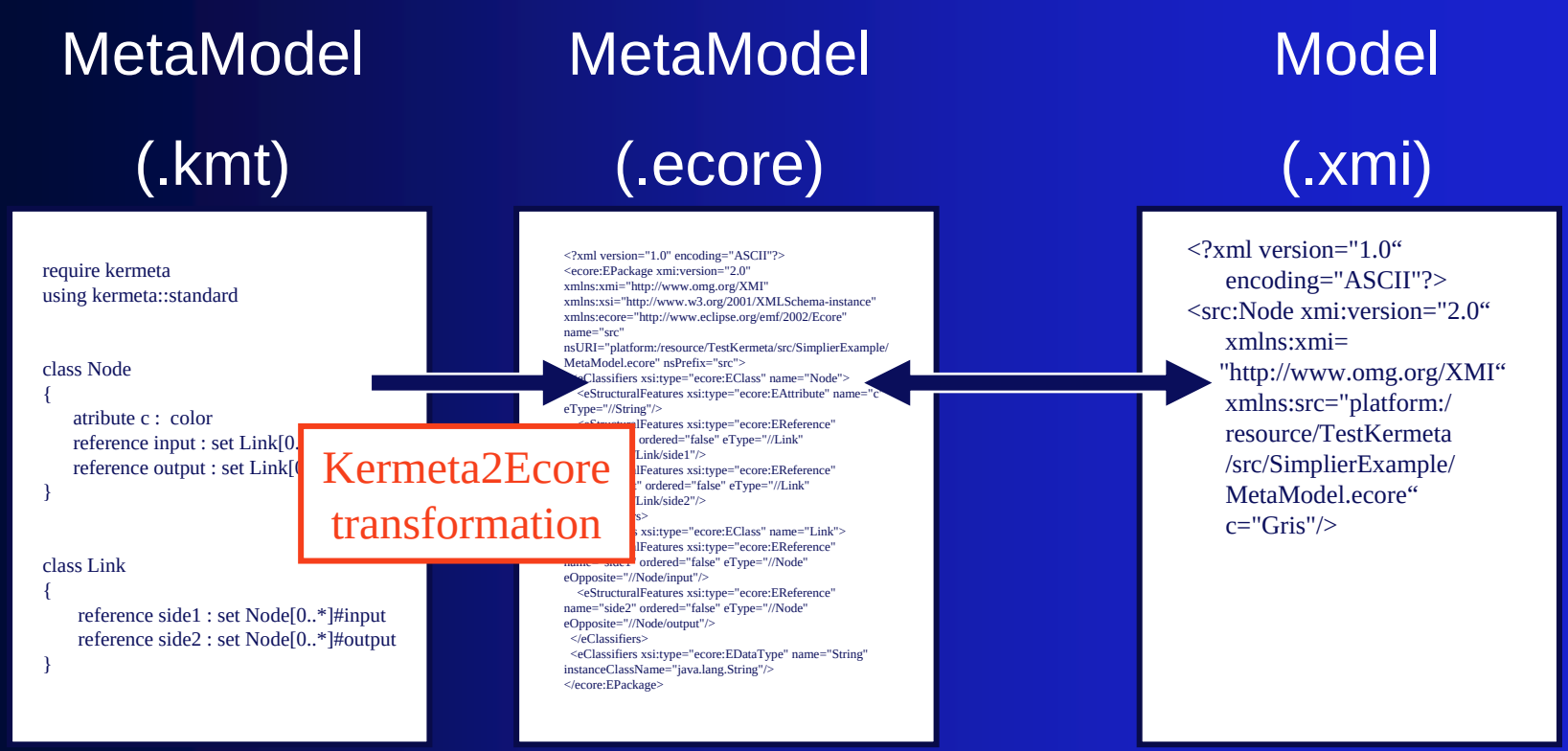
Plan

- Reminder
- What we have already done
- What we're working on
- Problems
- To do

Defining a new language

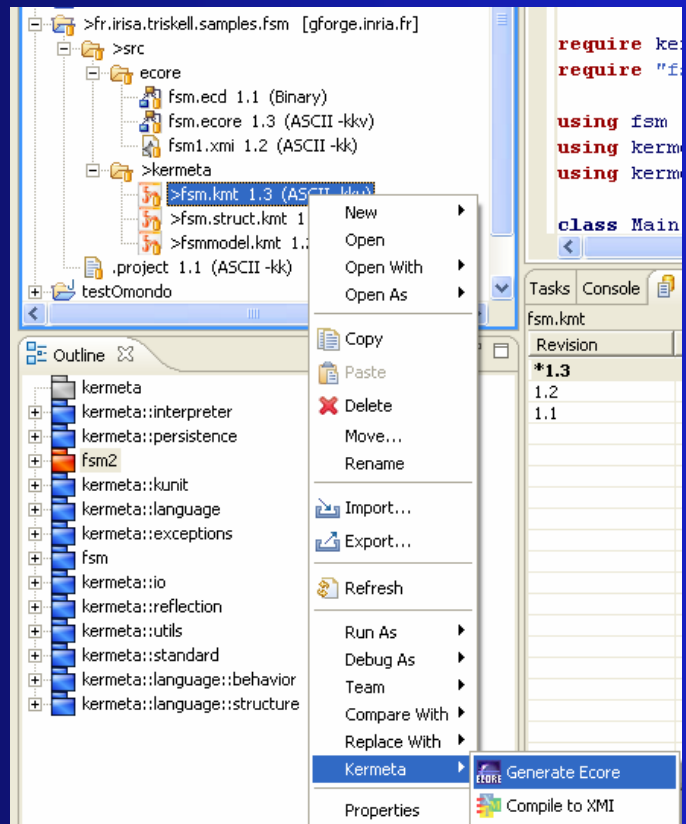
- Definition of the language
  - MetaModel => Ecore
  - Templates => SVG+ Templates
- Instancitation
  - Abstract representation => XMI
  - Concrete representation => SVG+

Creating a new language

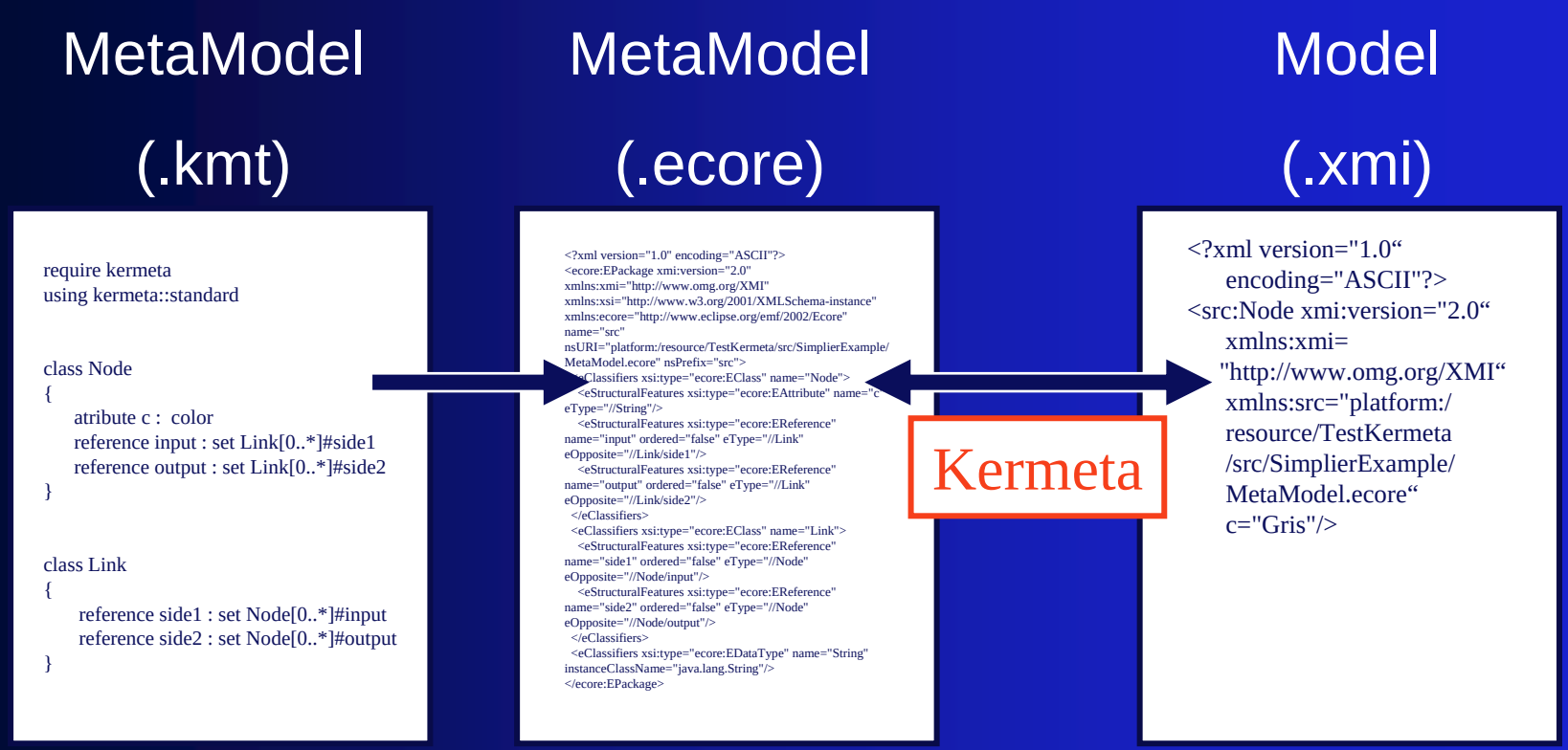


Kermeta2Ecore transformation

- Supported by Kermeta
- In eclipse environnement

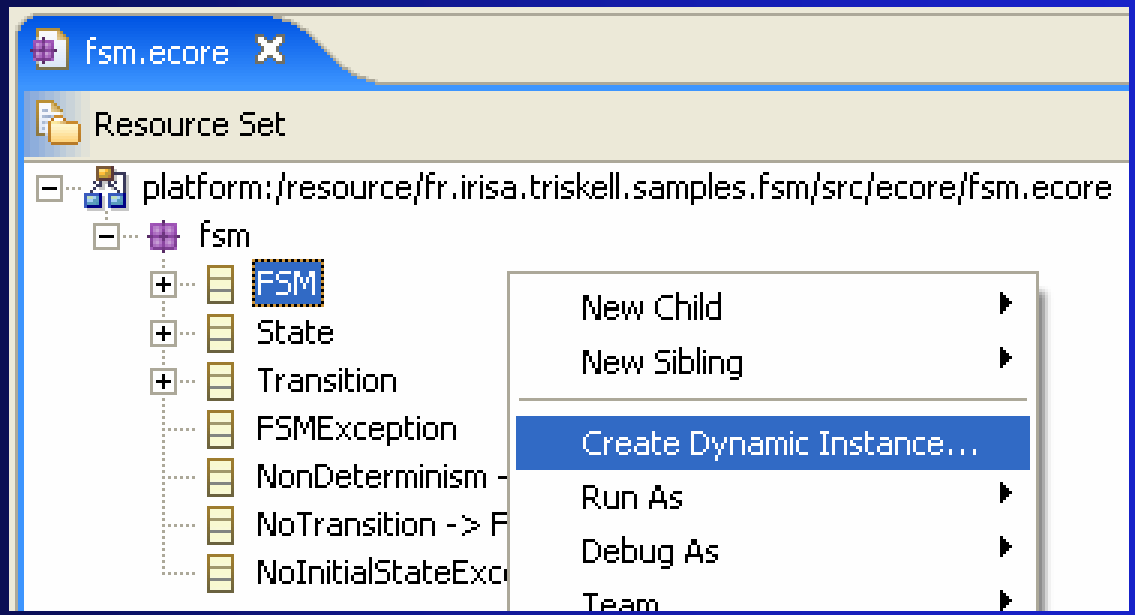


Creating a new language

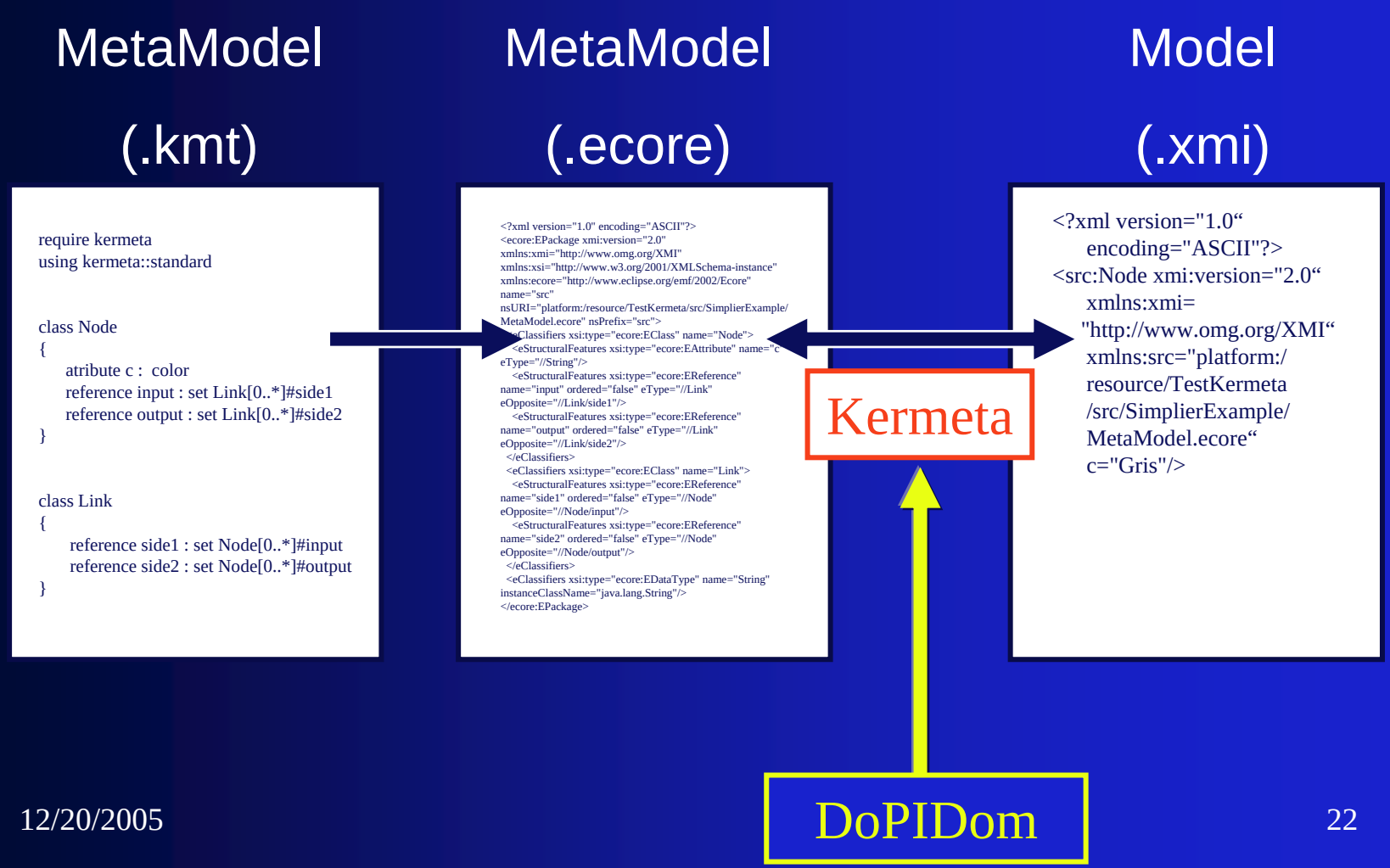


Creating a dynamic instance

- Ecore allow to create a dynamic instance of one specific element of the metamodel
- In eclipse environnement



Creating a new language



Handle the Model with Kermeta

- One can handle the resource
  - Load it
  - Save it
  - Manipulate its elements (like a Dom tree)
    - Creating
    - Removing
    - Moving

Plan

- Reminder
- What we have already done
- What we're working on
- Problems
- To do

Rising synchronization
Interaction between DoPIDom & KerMeta

- Call the KerMeta interpretor from DoPIDom
- Handle KerMeta operations from DoPIDom
- Keep synchronized graphical elements with their abstract representation

Rising synchronization

- Add Kermeta instructions into SVG templates

Allow multilanguage

Recover tag and attributes

- **<c:content query="{OCL | self.name.text}“ update(s:String)= "{KerMeta| self.name.text := s}"/>**

Identification of « self »

Plan

- Reminder
- What we have already done
- What we're working on
- Problems
- To do

Difficulties
Intrinsic problem with the language

- Kermeta is a young language
  - Some annoying bugs
    - Problems in the transition kermeta2ecore
    - Problems when we want to save the model
    - Bad error localization
  - Obsolete tutorials
  - Lack of documentation
- But good contact with the developpers

Difficulties

- Understand DoPIDom architecture
  - We need to know some details of implementation because we have to interact and add new features to the project.

Plan

- Reminder
- What we have already done
- Problems
- To do

TO DO : Further step

- Descendent synchronization!

Time for questions!