Executable meta-modeling in Kermeta with a rpg formalism

Ward Loos

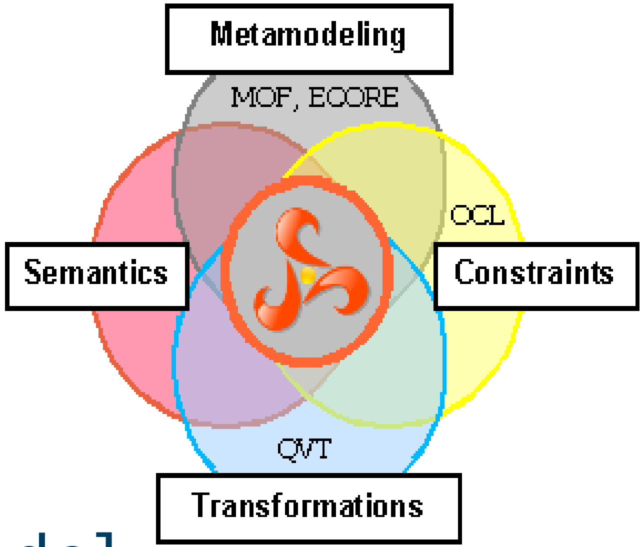
Universiteit Antwerpen

Kermeta overview

- Tries to be common denominator between modeling languages
- Object-oriented and statically typed
- Framework depends on standardized technologies by the OMG
- Uses Eclipse Modeling Framework (EMF)
- Workbench provided as eclipse plugin
- Java interpreter

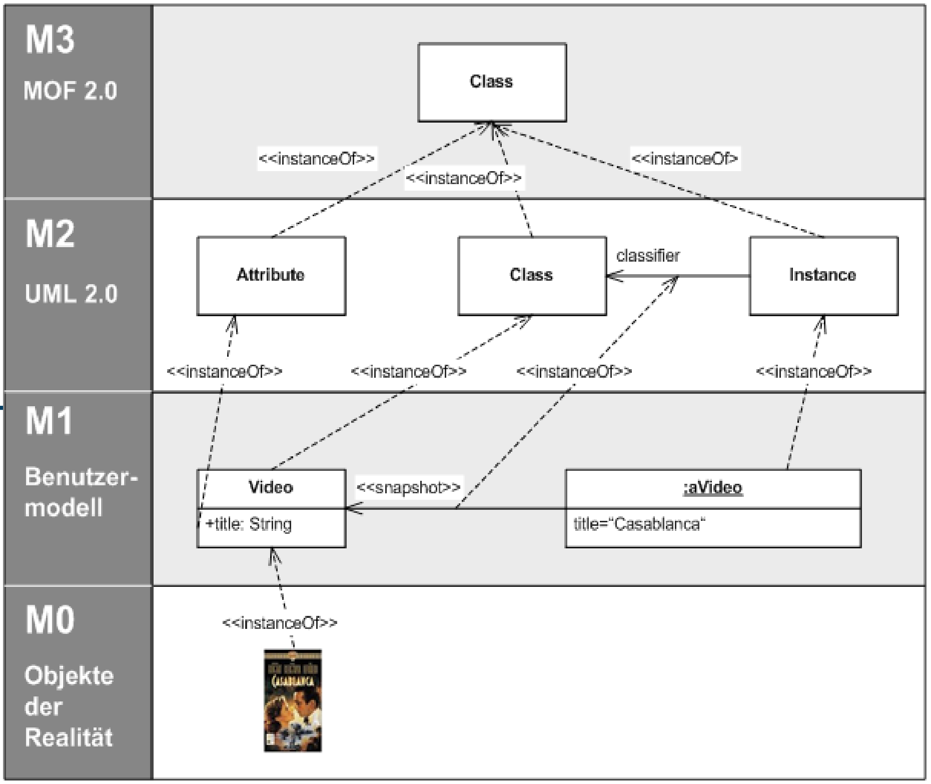
Modeling features

- Compliant with OCL
- Extends EMOF
- Associations with multiplicities
- Both structural and behavioral code in meta-model



Standardized techniques (1)

- Meta-Object Facilities (MOF):
  - Four-layered architecture
  - Provides meta-meta model in M3
  - Meta-model for UML
  - Kermeta uses Ecore variant defined by EMF



Universiteit Antwerpen

Standardized techniques (2)

- Object Constraint Language (OCL) provides contraint and query expressions for MOF models
- XML Metadata Interchange (XMI):
  - Standard for exchanging metadata
  - Metadata whose meta-model can be expressed in MOF

Other features (1)

- Mainly object-oriented:
  - (abstract) classes and methods
  - Properties
  - Multiple inheritance
  - Exceptions
  - Generics
  - Namespaces

Other features (2)

- Design by contract
- Aspect-oriented programming
- Statically typed

Missing:

- Constructors
- Return, break and continue statements

(Meta-)Model creation and storage

- Meta-model defined in Kermeta source file (kmt)
- Converted to ecore meta-model for model creation
- Model creation in EMF
- Models validated with ecore meta-model
- Kermeta needs a root element

Kermeta workbench

- Eclipse plugin
- Syntax highlighting and type checking
- Debugger
- Interpreter
- Conversion from kmt to ecore (and back)

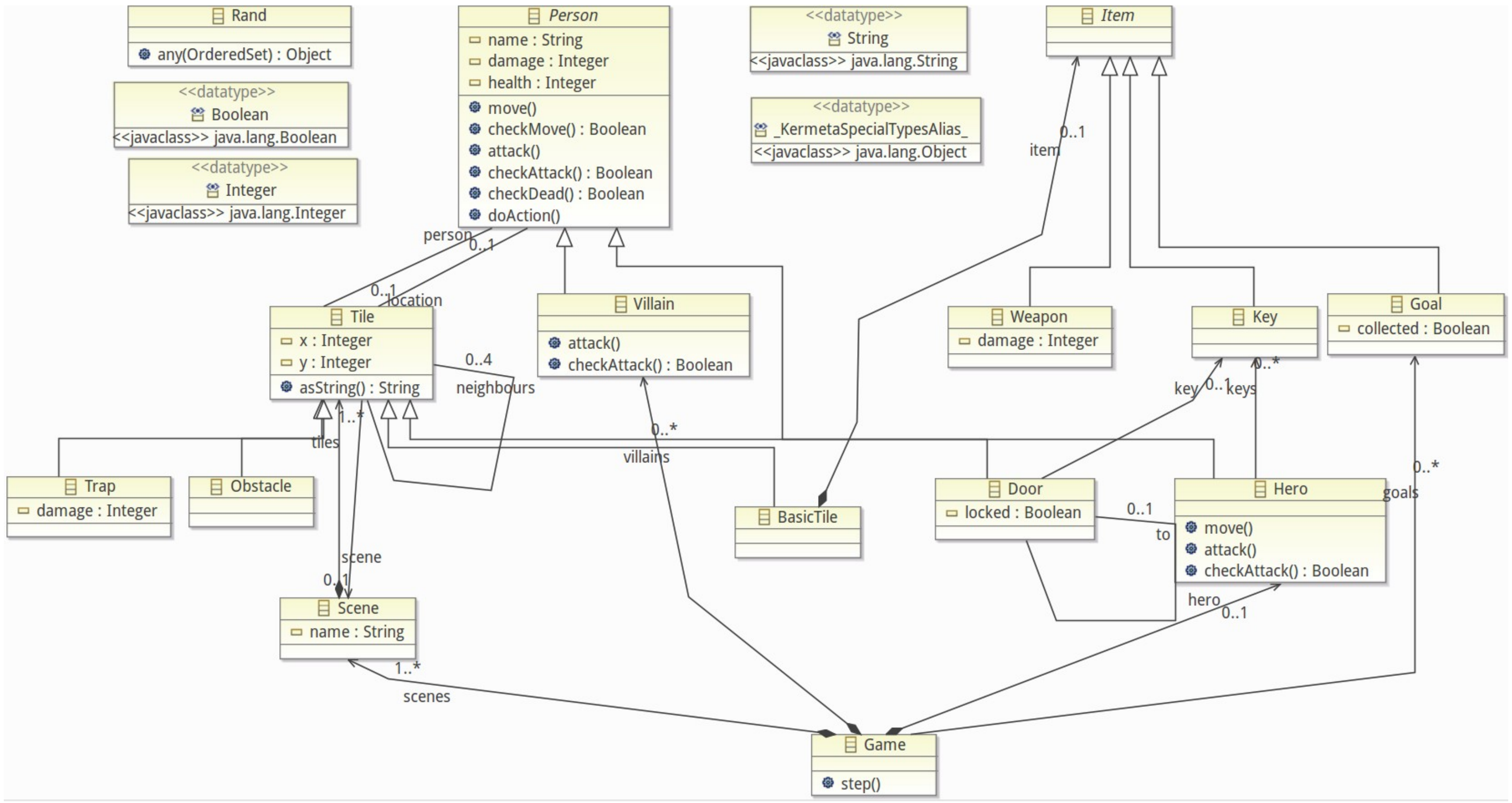
Eclipse Modeling Framework

- Eclipse plugin
- Generate and edit ecore diagrams (variant of UML diagrams)
- Generate ecore meta-model from diagram
- Visual editor for models
- Other tools can be used

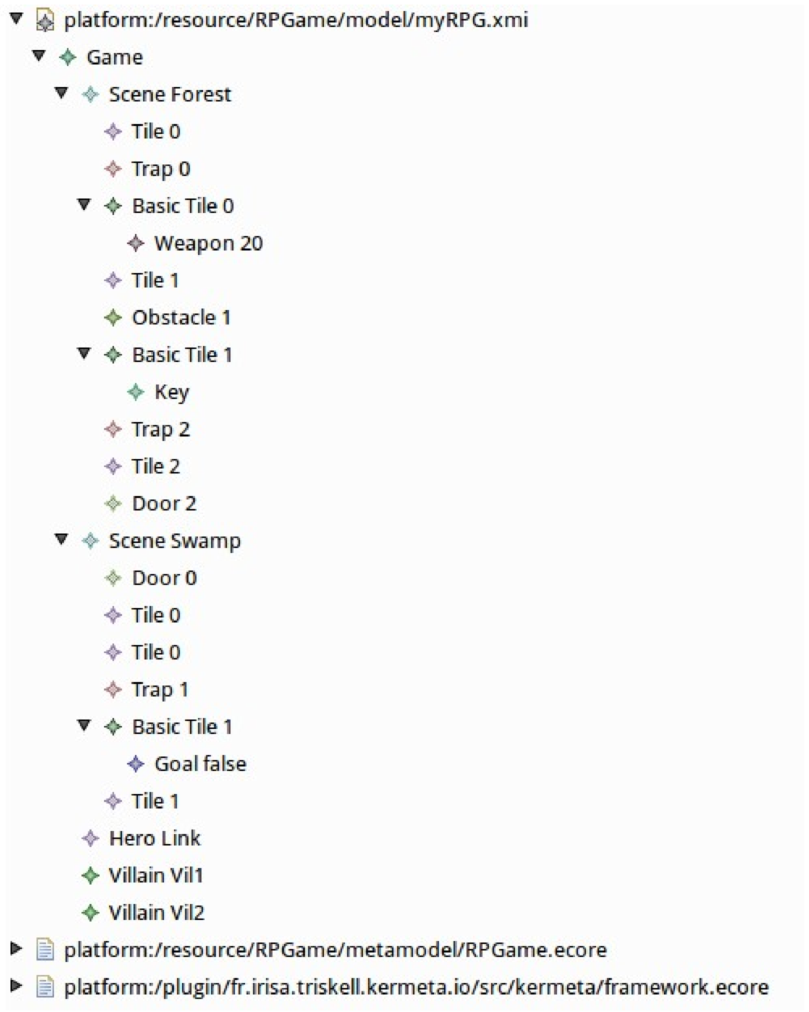
Kermeta 2

- Released in 2012
- Uses Scala instead of Java
- Allows compilation to bytecode for the JVM

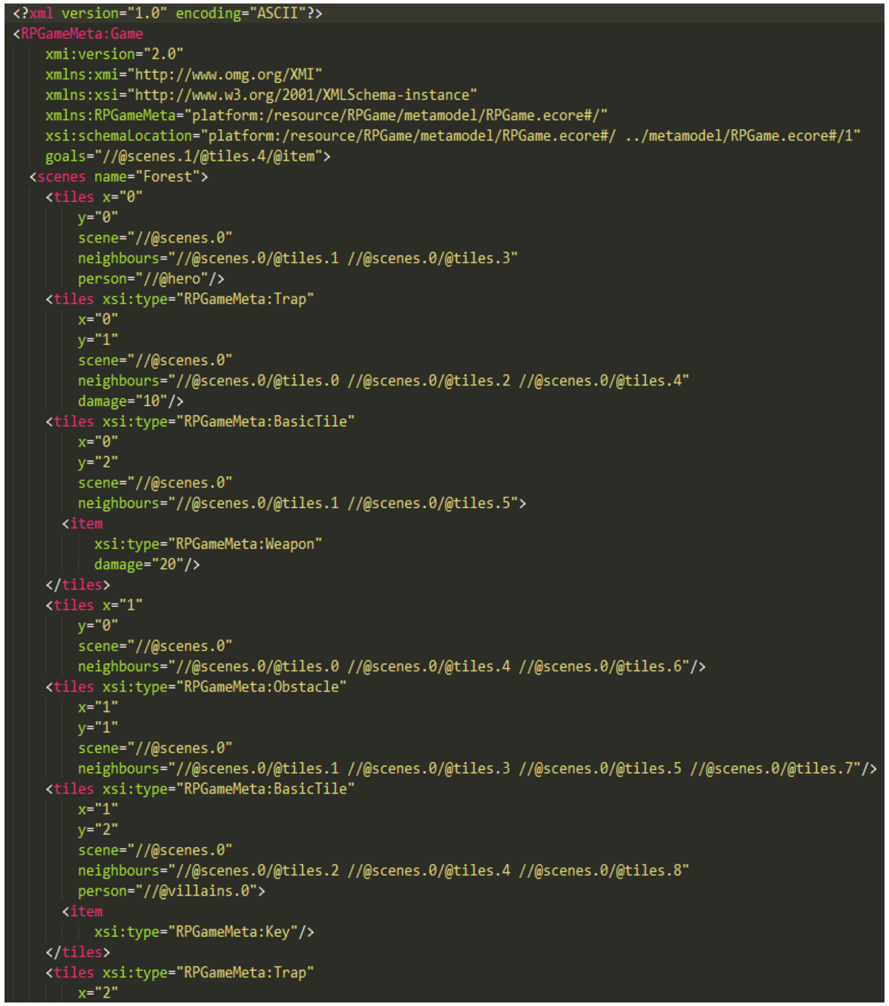
RPGame ecore diagram



RPGame model



```
▼ platform:/resource/RPGame/model/myRPG.xmi
  ▼ Game
    ▼ Scene Forest
        Tile 0
        Trap 0
      ▼ Basic Tile 0
          Weapon 20
        Tile 1
        Obstacle 1
      ▼ Basic Tile 1
          Key
        Trap 2
        Tile 2
        Door 2
    ▼ Scene Swamp
        Door 0
        Tile 0
        Tile 0
        Trap 1
      ▼ Basic Tile 1
          Goal false
        Tile 1
      Hero Link
      Villain Vil1
      Villain Vil2
▶ platform:/resource/RPGame/metamodel/RPGame.ecore
▶ platform:/plugin/fr.irisa.triskell.kermeta.io/src/kermeta/framework.ecore
```



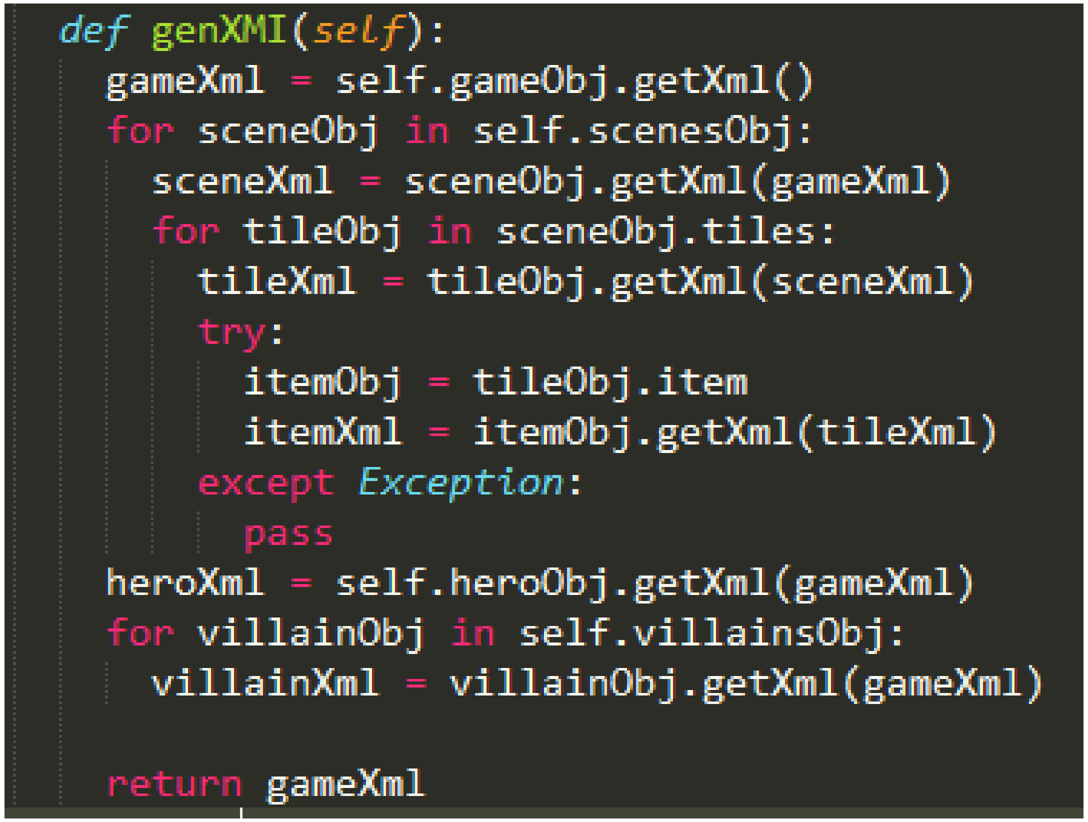
```
<?xml version="1.0" encoding="ASCII"?>
<RPGameMeta:Game
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:RPGameMeta="platform:/resource/RPGame/metamodel/RPGame.ecore#/"
    xsi:schemaLocation="platform:/resource/RPGame/metamodel/RPGame.ecore#/ ../metamodel/RPGame.ecore#/1"
    goals="//@scenes.1/@tiles.4/@item">
  <scenes name="Forest">
    <tiles x="0"
        y="0"
        scene="//@scenes.0"
        neighbours="//@scenes.0/@tiles.1 //@scenes.0/@tiles.3"
        person="//@hero"/>
    <tiles xsi:type="RPGameMeta:Trap"
        x="0"
        y="1"
        scene="//@scenes.0"
        neighbours="//@scenes.0/@tiles.0 //@scenes.0/@tiles.2 //@scenes.0/@tiles.4"
        damage="10"/>
    <tiles xsi:type="RPGameMeta:BasicTile"
        x="0"
        y="2"
        scene="//@scenes.0"
        neighbours="//@scenes.0/@tiles.1 //@scenes.0/@tiles.5">
      <item
          xsi:type="RPGameMeta:Weapon"
          damage="20"/>
    </tiles>
    <tiles x="1"
        y="0"
        scene="//@scenes.0"
        neighbours="//@scenes.0/@tiles.0 //@scenes.0/@tiles.4 //@scenes.0/@tiles.6"/>
    <tiles xsi:type="RPGameMeta:Obstacle"
        x="1"
        y="1"
        scene="//@scenes.0"
        neighbours="//@scenes.0/@tiles.1 //@scenes.0/@tiles.3 //@scenes.0/@tiles.5 //@scenes.0/@tiles.7"/>
    <tiles xsi:type="RPGameMeta:BasicTile"
        x="1"
        y="2"
        scene="//@scenes.0"
        neighbours="//@scenes.0/@tiles.2 //@scenes.0/@tiles.4 //@scenes.0/@tiles.8"
        person="//@villains.0">
      <item
          xsi:type="RPGameMeta:Key"/>
    </tiles>
    <tiles xsi:type="RPGameMeta:Trap"
        x="2"
```

Universiteit Antwerpen

Convert AToM^3 model to XMI (1)

- Button added to buttons model
- General strategy:
  - Parse all elements from ASGroot object
  - Create objects from parsed elements
  - Objects keep track of sub-elements
  - Then parse all links and fill in blank spots in objects
  - Nested for-loops to create XMI file

Convert AToM^3 model to XMI (2)



```
def genXMI(self):
    gameXml = self.gameObj.getXml()
    for sceneObj in self.scenesObj:
        sceneXml = sceneObj.getXml(gameXml)
        for tileObj in sceneObj.tiles:
            tileXml = tileObj.getXml(sceneXml)
            try:
                itemObj = tileObj.item
                itemXml = itemObj.getXml(tileXml)
            except Exception:
                pass
    heroXml = self.heroObj.getXml(gameXml)
    for villainObj in self.villainsObj:
        villainXml = villainObj.getXml(gameXml)

    return gameXml
```

Universiteit Antwerpen

Future work

- Default export function for AToM^3 models to XMI
- Use Kermeta for model transformation from AToM^3 to Kermeta model

Reference

- On Executable Meta-Languages applied to Model Transformations,P Muller, F Fleurey, D Vojtisek, Z Drey, D Pollet, F Fondement, P Studer, and J Jézéquel (2005)

Questions?

Universiteit Antwerpen