Procesadores de Lenguajes 2

Desarrollo de metamodelos con EMF

Curso 2013-2014

Iván Ruiz Rube
Departamento de Ingeniería Informática
Escuela Superior de Ingeniería
Universidad de Cádiz

En la clase anterior…

- Eclipse es una de las mejores plataformas de desarrollo de código abierto.
- Es altamente extensible mediante plugins.
- Existen muchos proyectos alrededor de una importante comunidad de usuarios y desarrolladores.
- El workbench de Eclipse se compone de: workspace, asistentes, editores, vistas y perspectivas.

Contenidos

- Introducción
- El meta-metamodelo Ecore
- Creación de metamodelos
- Un primer ejemplo

DESARROLLO DE METAMODELOS CON EMF

INTRODUCCIÓN

Introducción

- Eclipse Modeling Framework (EMF) es el núcleo de la plataforma Ecipse para el desarrollo dirigido por modelos.
- Framework para el desarrollo de metamodelos (sintaxis abstracta).
- Permite generar automáticamente clases de implementación en Java para los elementos de nuestros metamodelos.

Funcionalidades de EMF

- Diseñar metamodelos Ecore
  - Editor basado en una estructura tipo árbol
  - Editor visual similar al modelado UML
- Construir editores de modelos basados en estructura tipo árbol
  - Generación de clases Java de soporte al metamodelo:
    - Factorías, interfaces, listeners, etc.
  - Generación de casos de prueba en Junit

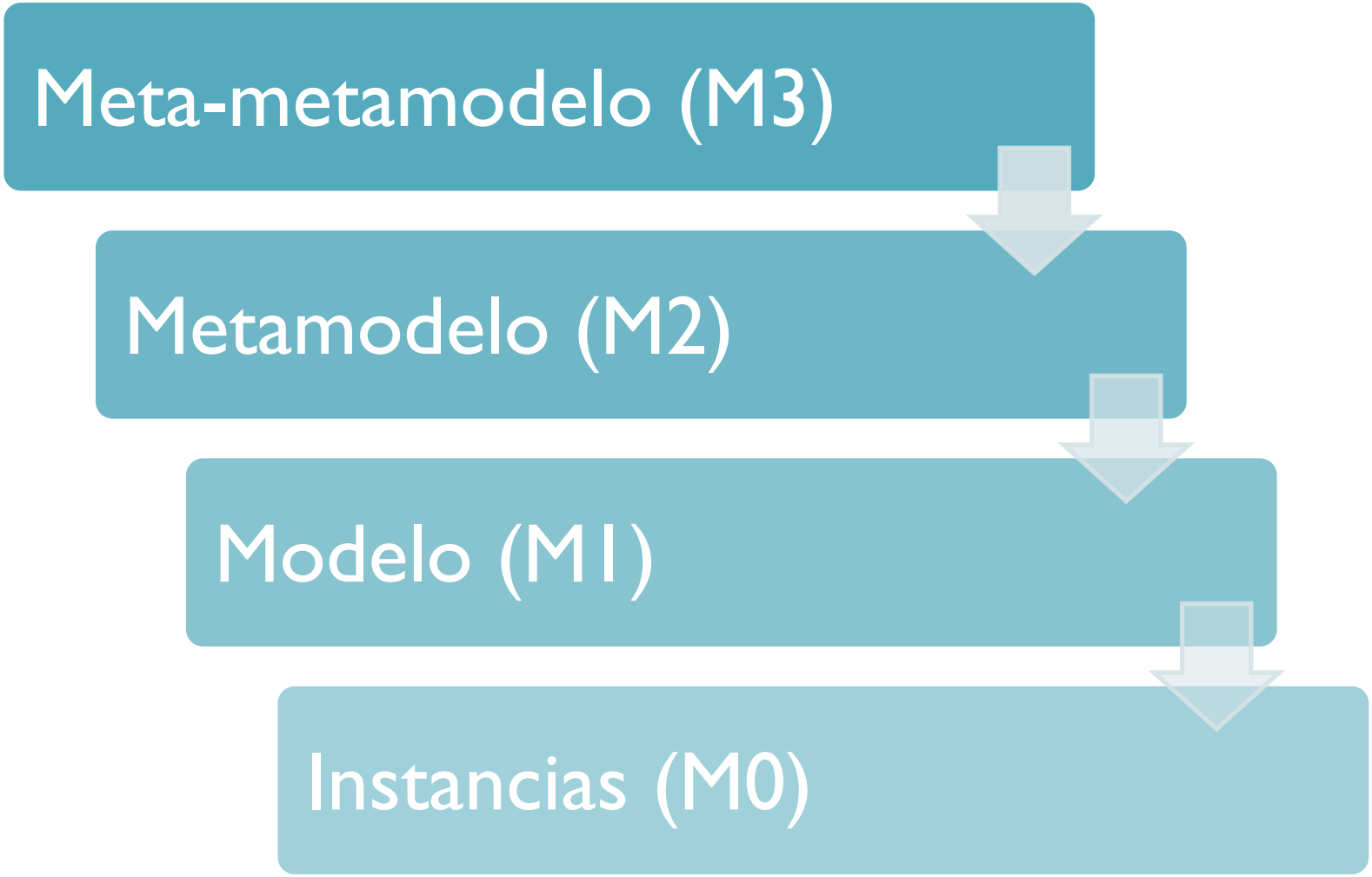
Subproyectos EMF

- CDO
- EMF Compare
- Model Query
- Model Transaction
- Net4j
- SDO
- Teneo
- Validation Framework
- B3
- Ecore Tools
- Mint
- EMFatic
- EMF Search
- EEF
- EFG
- Modeling Workflow
- Temporality

DESARROLLO DE METAMODELOS CON EMF

EL META-METAMODELO ECORE

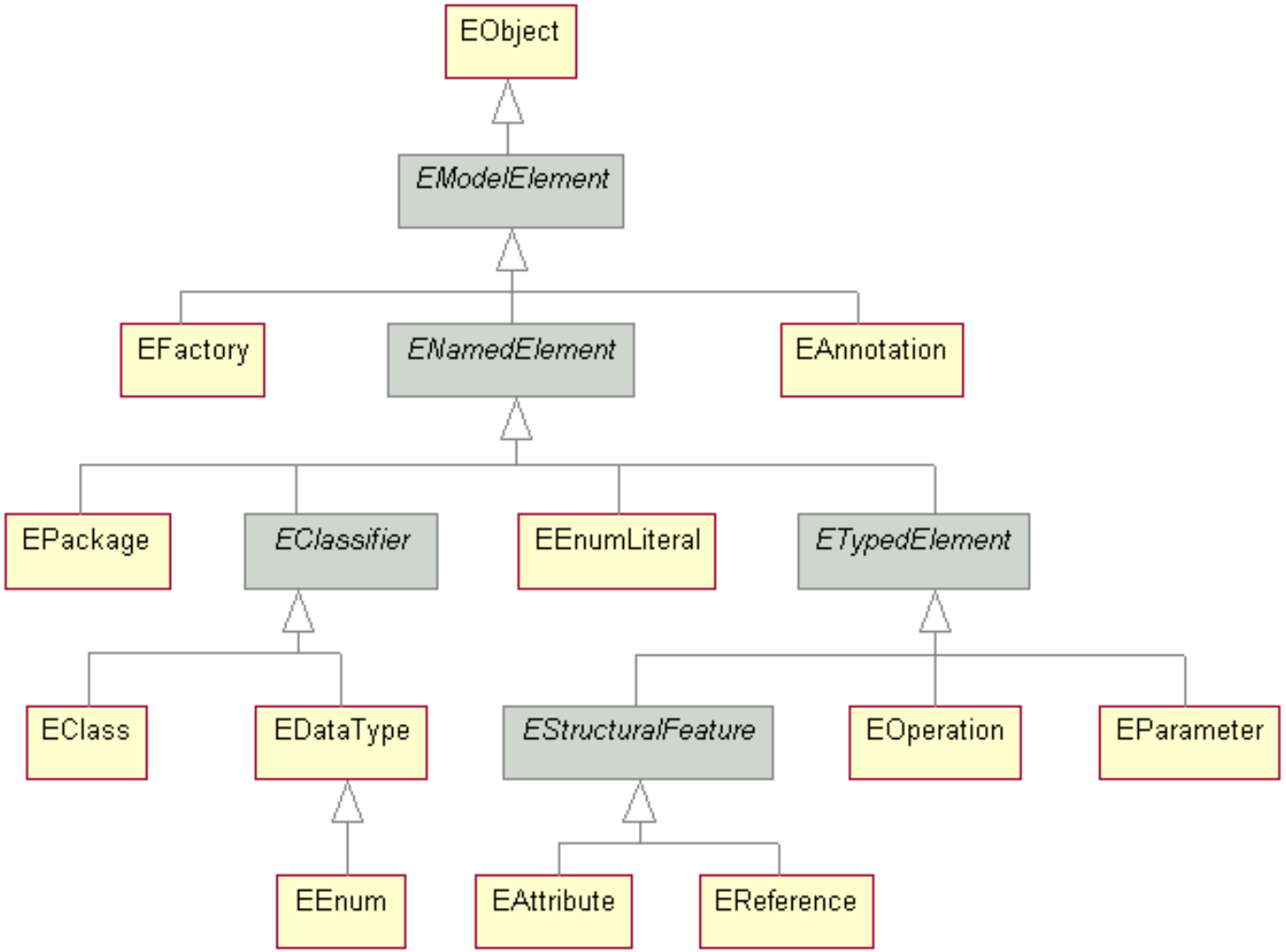
Arquitectura de metamodelado



Ecore

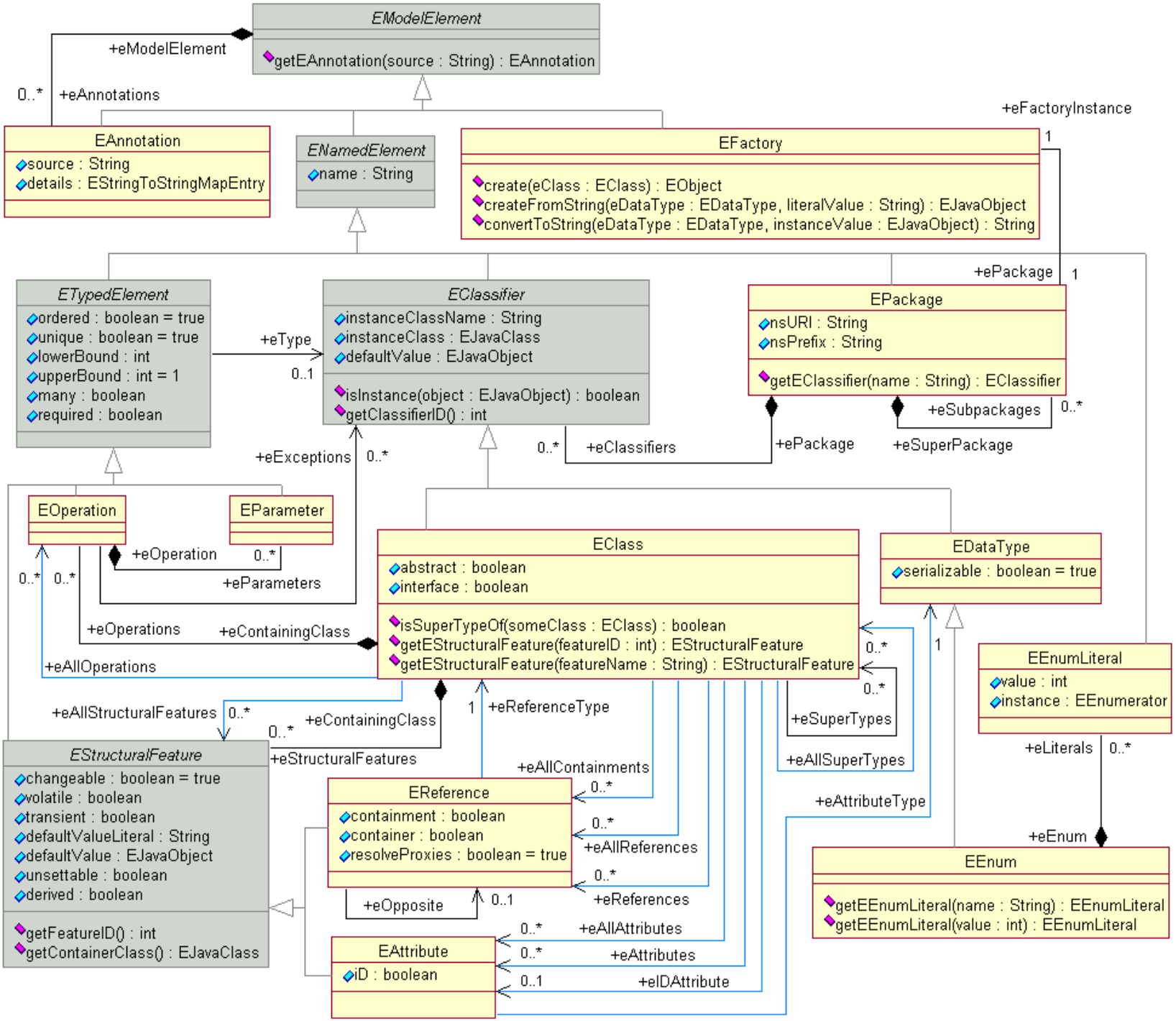
- Es el meta-metamodelo propuesto por la comunidad Eclipse.
- Versión simplificada de MOF.
- Los metamodelos se serializan en ficheros XML con extensión *.ecore*
- El diagrama visual del metamodelo se almacena en el fichero *.ecorediag*
- Permite el intercambio de (meta)modelos

Componentes de Ecore



Principales componentes de Ecore

- *EPackage:* componente que permite organizar clases y tipos de datos.
- *EClass*: conceptos en el metamodelo
- *EReference*: asociación entre conceptos
- *EAttribute*: propiedades de los conceptos
- *EDataType*: tipo de un atributo.


EModelElement
getEAnnotation(source : String) : EAnnotation
+eModelElement
0..*
+eAnnotations
EAnnotation
source : String
details : EStringToStringMapEntry
ENamedElement
name : String
EFactory
create(eClass : EClass) : EObject
createFromString(eDataType : EDataType, literalValue : String) : EJavaObject
convertToString(eDataType : EDataType, instanceValue : EJavaObject) : String
+eFactoryInstance
1
ETypedElement
ordered : boolean = true
unique : boolean = true
lowerBound : int
upperBound : int = 1
many : boolean
required : boolean
+eType
0..1
EClassifier
instanceClassName : String
instanceClass : EJavaClass
defaultValue : EJavaObject
isInstance(object : EJavaObject) : boolean
getClassifierID() : int
+ePackage
1
EPackage
nsURI : String
nsPrefix : String
getEClassifier(name : String) : EClassifier
+eSubpackages
0..*
+eSuperPackage
+eClassifiers
0..*
+ePackage
+eExceptions
0..*
EOperation
EParameter
+eOperation
0..*
+eParameters
0..*
0..*
+eOperations
+eContainingClass
+eAllOperations
EClass
abstract : boolean
interface : boolean
isSuperTypeOf(someClass : EClass) : boolean
getEStructuralFeature(featureID : int) : EStructuralFeature
getEStructuralFeature(featureName : String) : EStructuralFeature
EDataType
serializable : boolean = true
0..*
+eSuperTypes
0..*
+eAllSuperTypes
1
EEnumLiteral
value : int
instance : EEnumerator
+eLiterals
0..*
+eEnum
EEnum
getEEnumLiteral(name : String) : EEnumLiteral
getEEnumLiteral(value : int) : EEnumLiteral
+eAttributeType
+eAllStructuralFeatures
0..*
0..*
+eContainingClass
+eStructuralFeatures
1
+eReferenceType
EStructuralFeature
changeable : boolean = true
volatile : boolean
transient : boolean
defaultValueLiteral : String
defaultValue : EJavaObject
unsettable : boolean
derived : boolean
getFeatureID() : int
getContainerClass() : EJavaClass
+eAllContainments
0..*
EReference
containment : boolean
container : boolean
resolveProxies : boolean = true
0..*
+eAllReferences
0..*
+eReferences
+eOpposite
0..1
0..*
+eAllAttributes
0..*
+eAttributes
0..1
+eIDAttribute
EAttribute
iD : boolean

DESARROLLO DE METAMODELOS CON EMF

CREACIÓN DE METAMODELOS

Pasos para crear un metamodelo

1. Crear un proyecto EMF
2. Diseñar el metamodelo con Ecore
3. Validar el metamodelo

Creación de un proyecto EMF

- Crear un proyecto EMF:
  - Importando un metamodelo creado con alguna herramienta compatible con UML 2.0.
  - Importando un XML Schema.
  - Importando una clase Java, convenientemente anotada.
- Crear un proyecto EMF vacio:
  - Utilizaremos los editores de metamodelos Ecore que ofrece EMF.

Diseño de metamodelos Ecore

- El diseño de los metamodelos basados en Ecore será parecido al diseño de un diagrama de clases en UML.
- Las metaclases tendrán metaatributos y metaasociaciones.
- Podremos utilizar herencia múltiple entre nuestras metaclases.
- Para cada metaasociación se definirán los roles, multiplicidades, navegabilidad y tipo (asociación o composición).

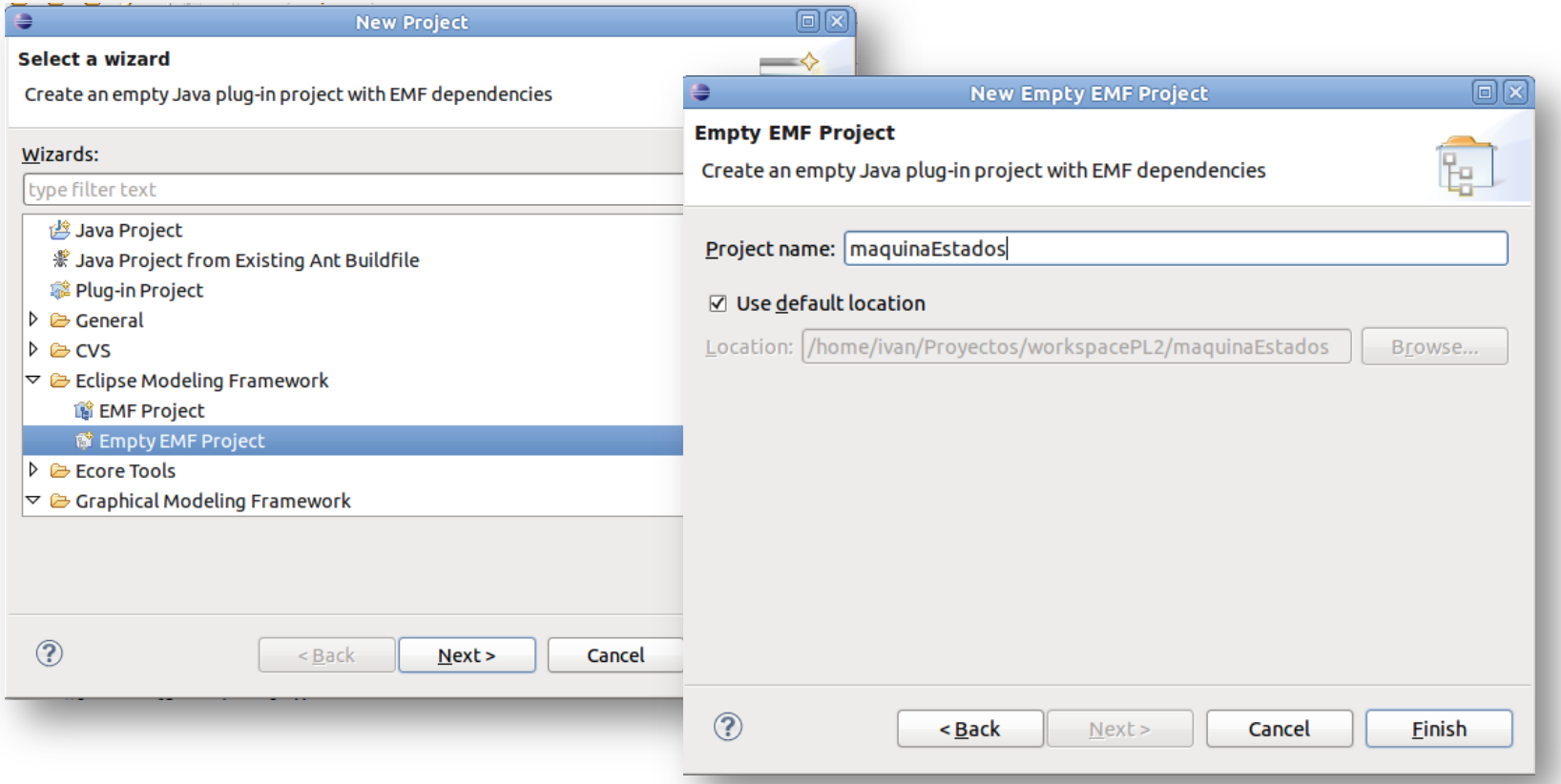
Validación del metamodelo

- La complejidad en el desarrollo de los editores y las reglas de transformación depende de la calidad del diseño del metamodelo.
- Todo metamodelo debe tener una metaclase que actúe de contenedor raíz. Su nombre debe ser distinto al nombre del fichero del metamodelo
- Todas las metaclases deben de estar conectadas directa o indirectamente a la metaclase raiz, mediante una composición.
- Las metaclases deben tener un atributo identificador único.
- Asegurar que las propiedades del EPackage estén definidas.

DESARROLLO DE METAMODELOS CON EMF

UN PRIMER EJEMPLO

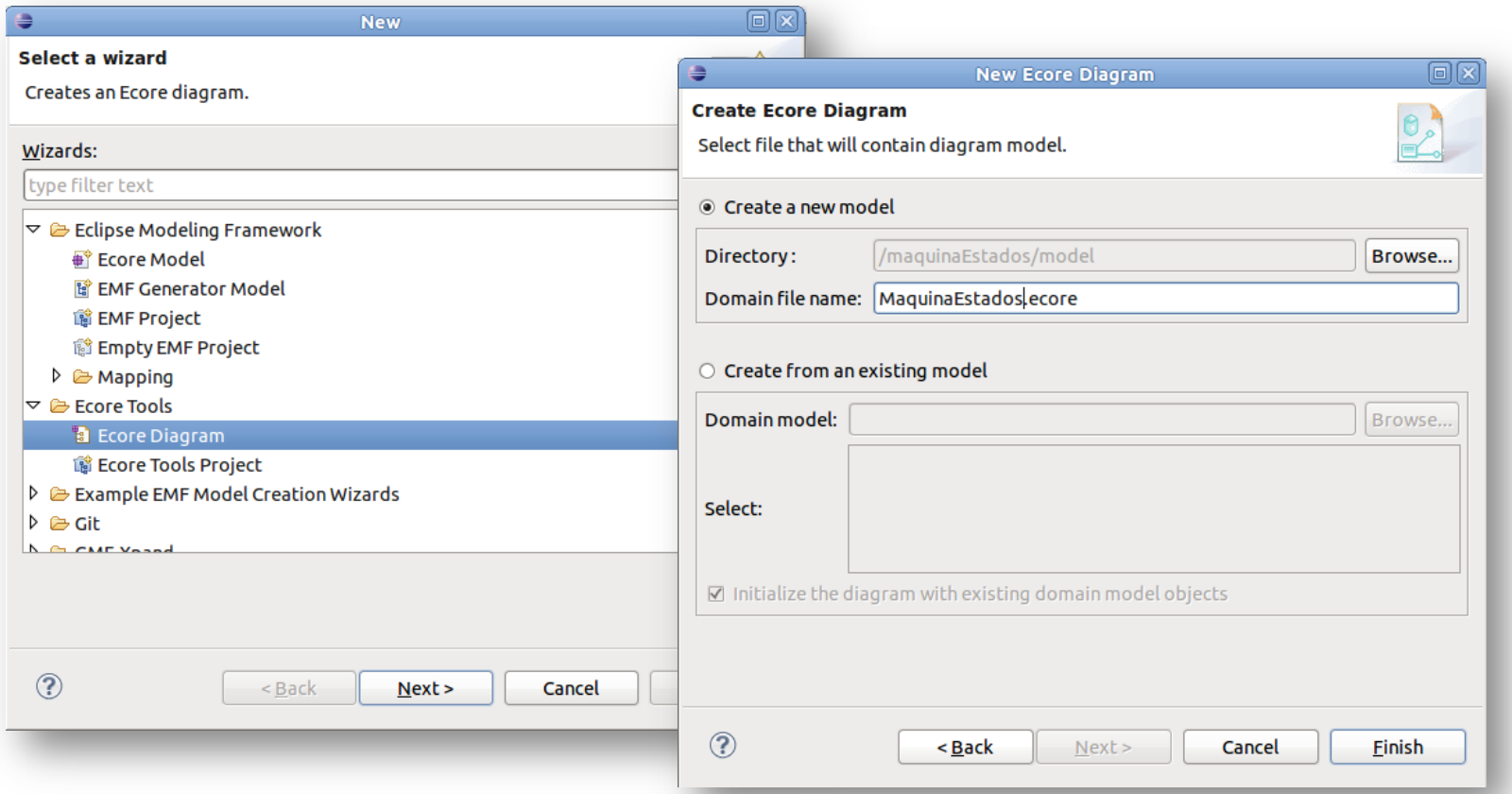
Creación de un proyecto EMF



`File → New Project → Empty EMF Project`

Los metamodelos los crearemos dentro de un proyecto EMF

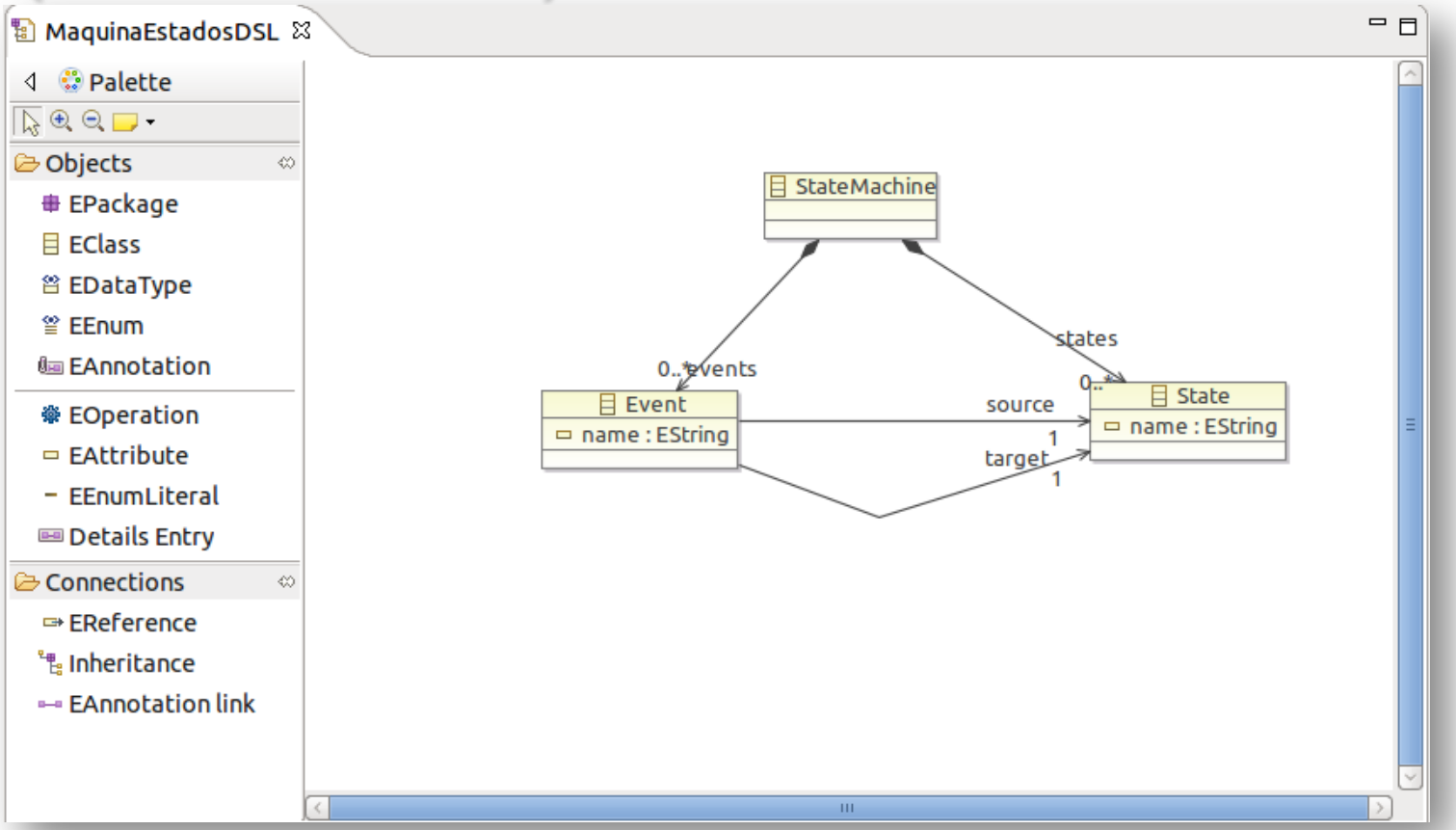
Creación de un metamodelo Ecore



`File → New → Ecore Diagram`

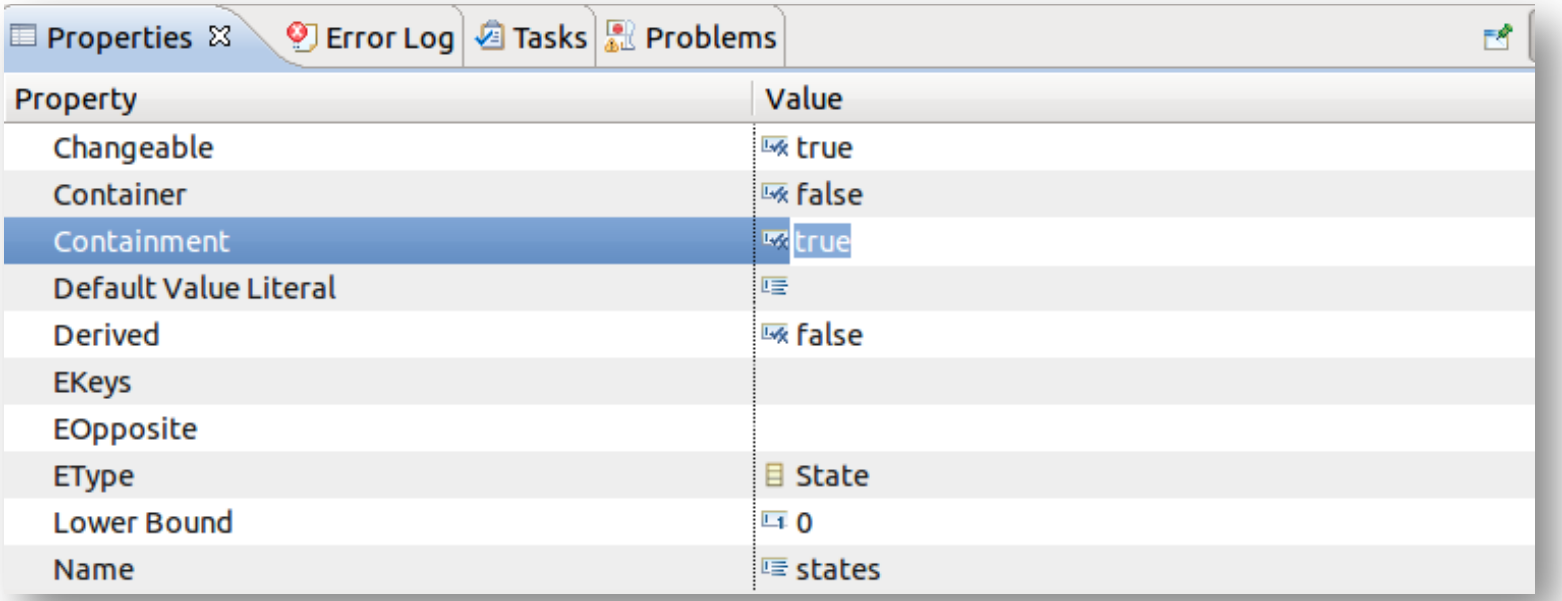
Podemos crear el metamodelo "MaquinaEstados" utilizando el editor visual. Se generará un fichero ".ecore" y ".ecorediag" y lo guardaremos en la carpeta "model"

Edición de un metamodelo Ecore (editor visual)



Para diseñar el metamodelo, utilizaremos los elementos Ecore de la paleta de componentes. Adicionalmente, emplearemos la vista de Propiedades para definir sus características.

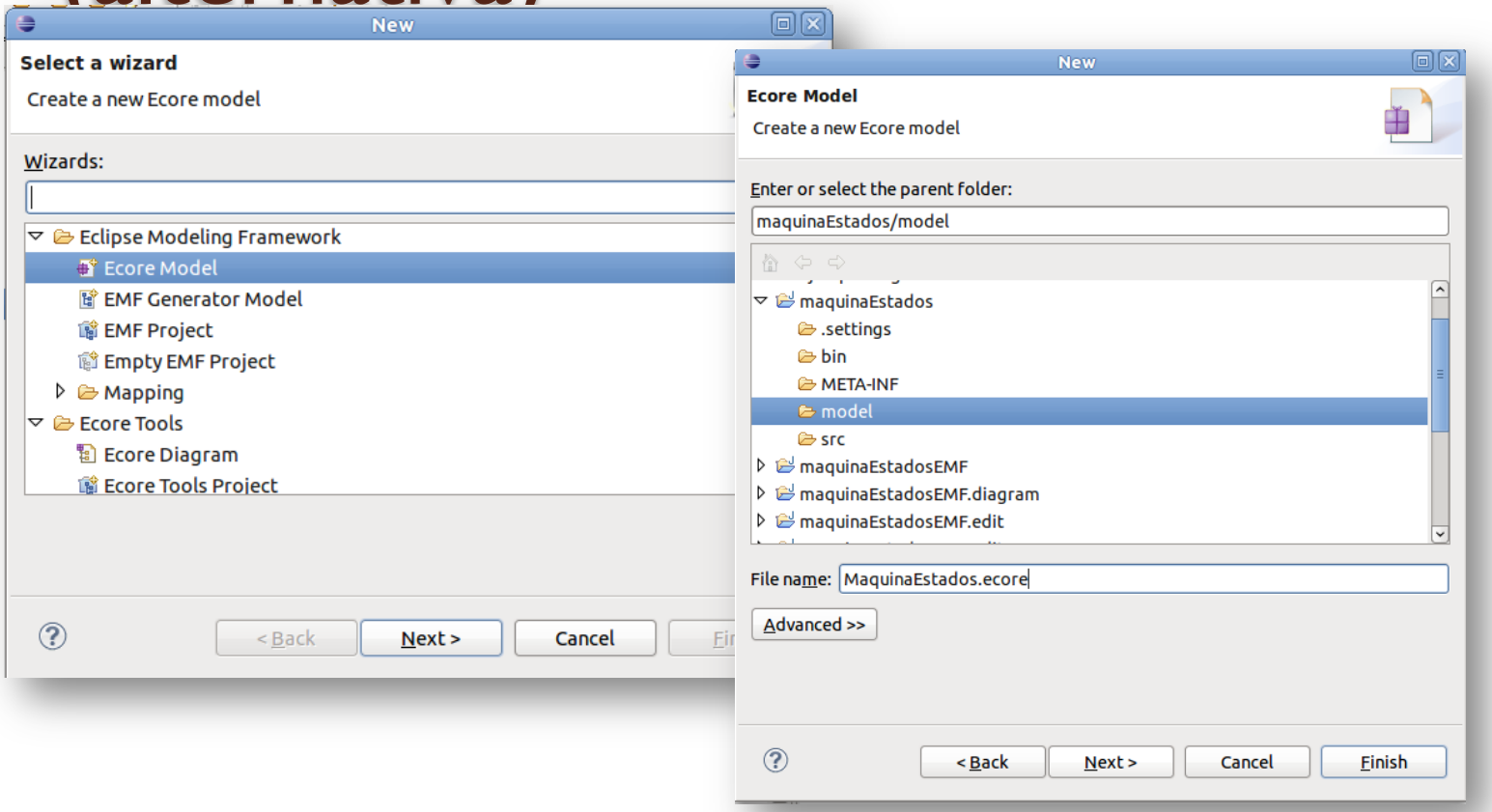
Vista de Propiedades



`Window → Show View → Properties`

Necesitamos la vista de Propiedades, para ajustar las características de cada uno de los elementos de nuestro metamodelo.

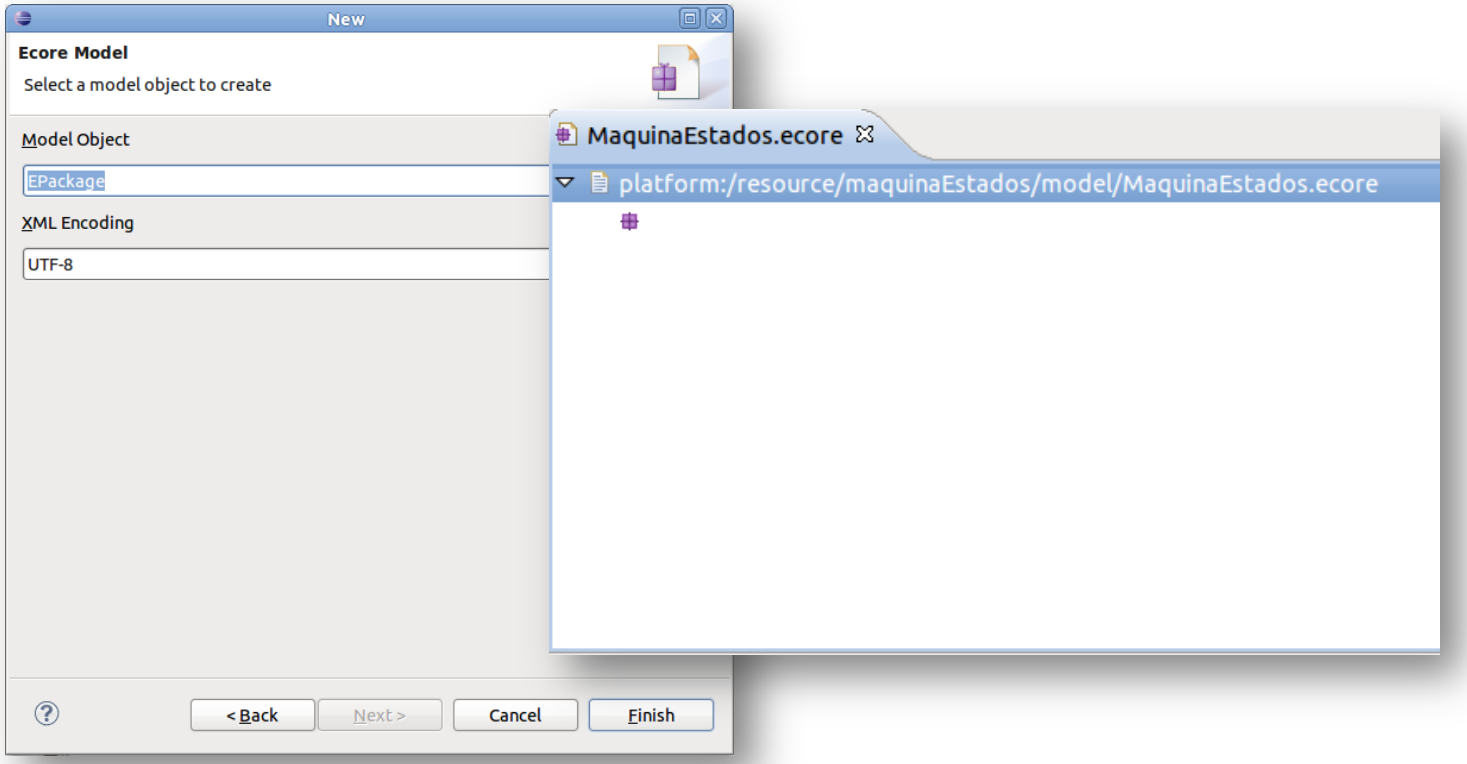
Creación de un metamodelo Ecore (alternativa)



`File → New → Ecore Model`

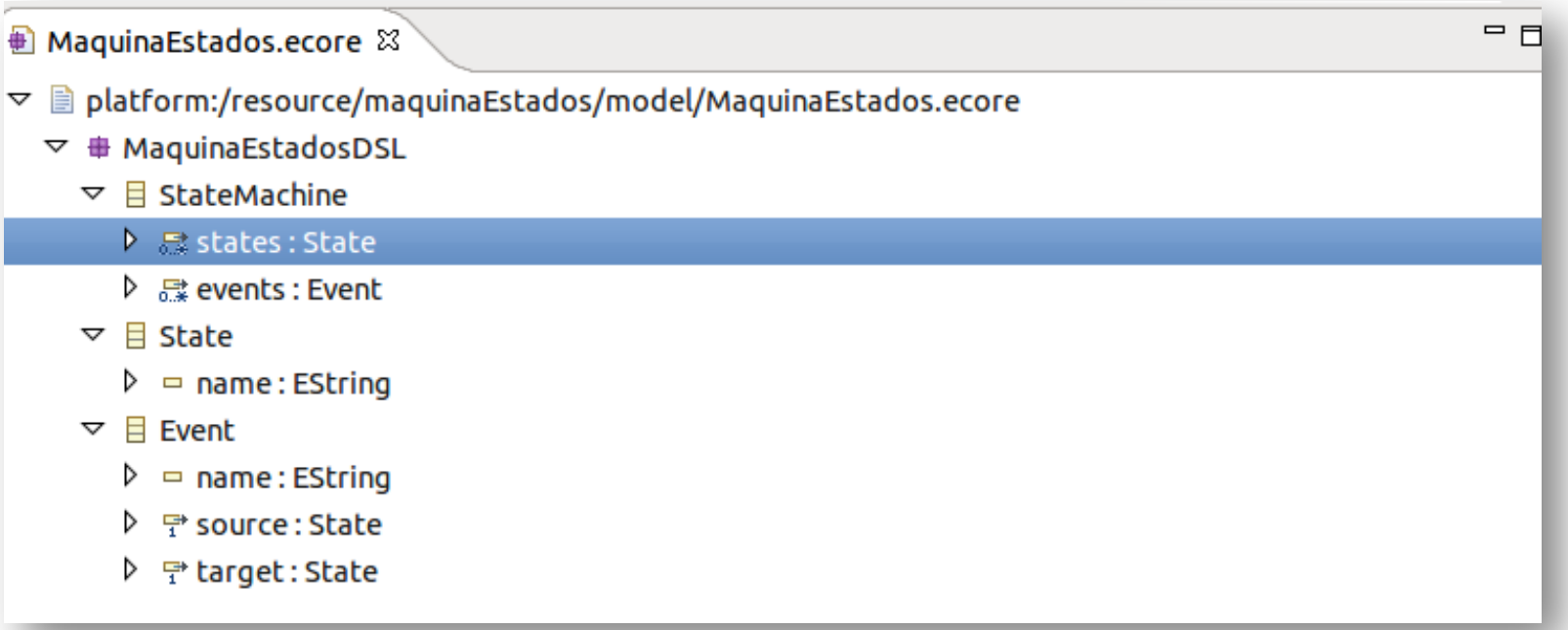
De forma alternativa, podemos crear nuestro metamodelo con el editor basado en árbol

Creación de un metamodelo Ecore



Seleccionamos *EPackage* y *UTF-8* en las propiedades del metamodelo a crear. Nos aparecerá una nueva vista donde editar nuestro metamodelo.

Edición de un metamodelo Ecore (tree-editor)



Utilizando el menú contextual podemos añadir nuevos elementos (clases, atributos, etc.) al metamodelo mediante "New child" o "New sibling"

Edición de un metamodelo Ecore (XML)

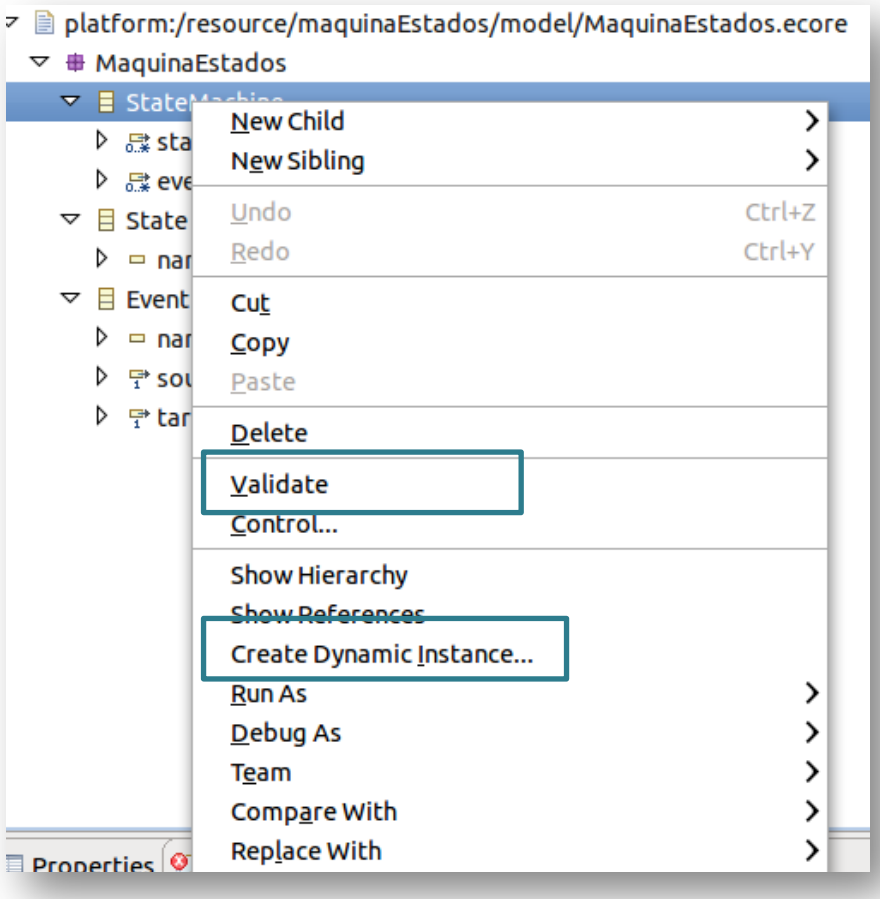
MaquinaEstados.ecore | MaquinaEstados.ecore

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="MaquinaEstadosDSL"
    nsURI="http://spi-fm.uca.es/dsl/maquinaestados/1.0" nsPrefix="MaquinaEstadosDSL">
  <eClassifiers xsi:type="ecore:EClass" name="StateMachine">
    <eStructuralFeatures xsi:type="ecore:EReference" name="states" upperBound="-1"
        eType="#//State" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="events" upperBound="-1"
        eType="#//Event" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="State">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType http://www.eclipse
        iD="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Event">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" eType="ecore:EDataType http://www.eclipse
        iD="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="source" lowerBound="1"
        eType="#//State"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="target" lowerBound="1"
        eType="#//State"/>
  </eClassifiers>
```

[fichero] Open With → Text Editor

Podemos editar directamente el fichero del metamodelo, haciendo uso de la sintaxis XML de Ecore. No es sencillo.

Validar el metamodelo



- Eclipse permite validar la corrección de nuestros metamodelos.
- Además, podremos crear modelos como instancias dinámicas de los metamodelos en XMI.
- Estos modelos serán manipulados con el tree-editor y podrán ser también validados.

DESARROLLO DE METAMODELOS CON EMF

RESUMEN

¿Qué hemos aprendido hoy?

- Características de Eclipse Modeling Framework.
- Los elementos del lenguaje de metamodelado Ecore.
- Utilizar el entorno EMF para crear metamodelos Ecore mediante un editor basado en árbol un editor visual.

Procesadores de Lenguajes 2

Desarrollo de metamodelos con EMF

Curso 2013-2014

Iván Ruiz Rube
ivan.ruiz@uca.es