



Herramientas Eclipse para Desarrollo de Software Dirigido por Modelos

Cristina Vicente Chicote

Teléfono: (+34) 968 32 6448

E-mail: Cristina.Vicente@upct.es

Diego Alonso Cáceres

Teléfono: (+34) 968 32 5341

E-mail: Diego.Alonso@upct.es



División de Sistemas e Ingeniería Electrónica (DSiE)
Departamento de Tecnologías de la Información y Comunicaciones
Escuela Técnica Superior de Ingeniería de Telecomunicación
Edificio Antigones, Plza. del Hospital N° 1, 30202 Cartagena
Universidad Politécnica de Cartagena



Tabla de contenidos

- Breve introducción al DSDM
- La plataforma Eclipse
- Herramientas Eclipse para DSDM
 - Herramientas de meta-modelado
 - Editores gráficos de modelos
 - Transformaciones M2M
 - Transformaciones M2T
 - Herramientas auxiliares
 - Entornos integrados
 - Herramientas de modelado UML
- Otras herramientas para DSDM
- Lecciones aprendidas

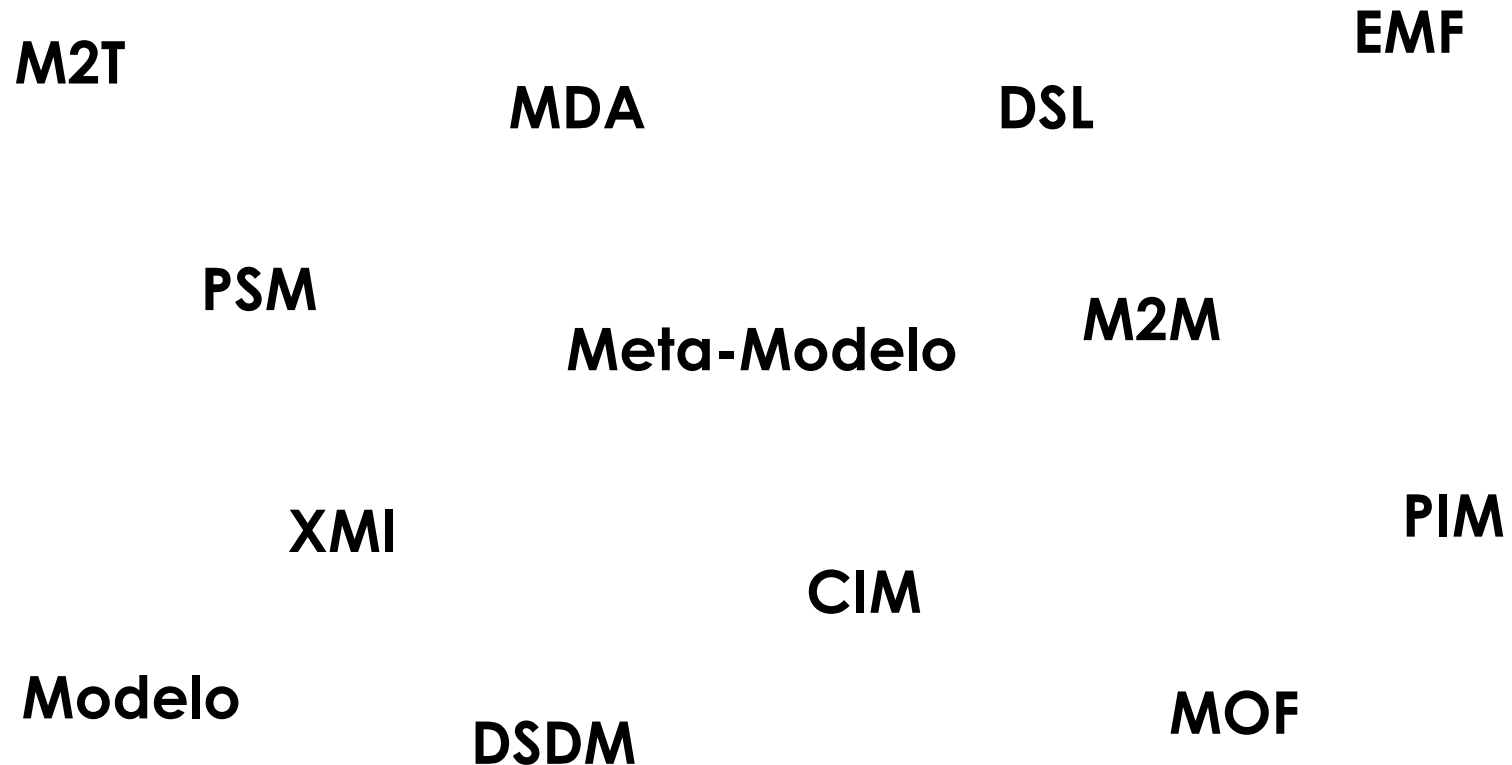


Disclaimer

- Las herramientas que se comentan en este tutorial...
 - **No son todas... aún hay más!!**
 - **No somos expertos en todas...**

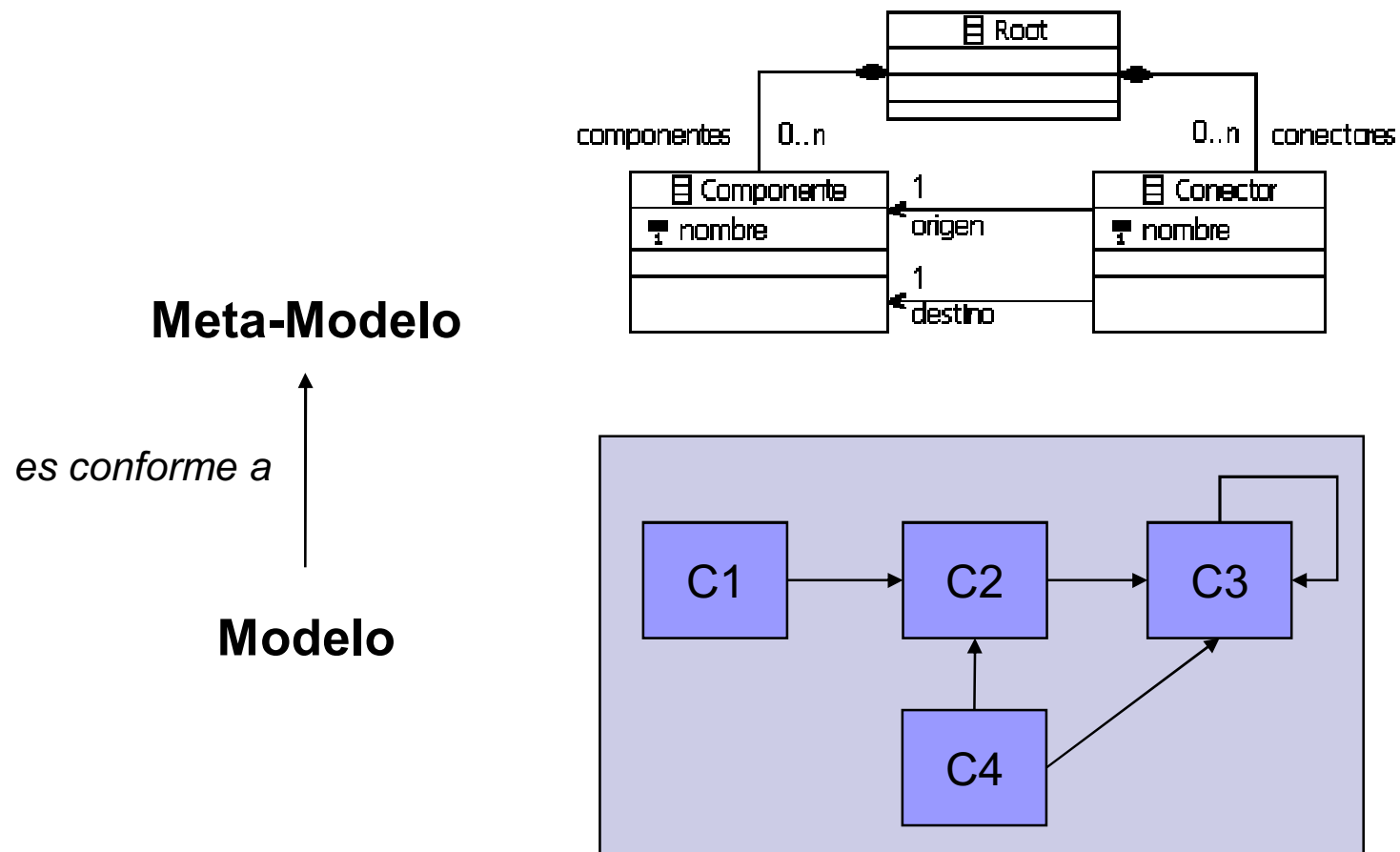


Introducción al DSDM

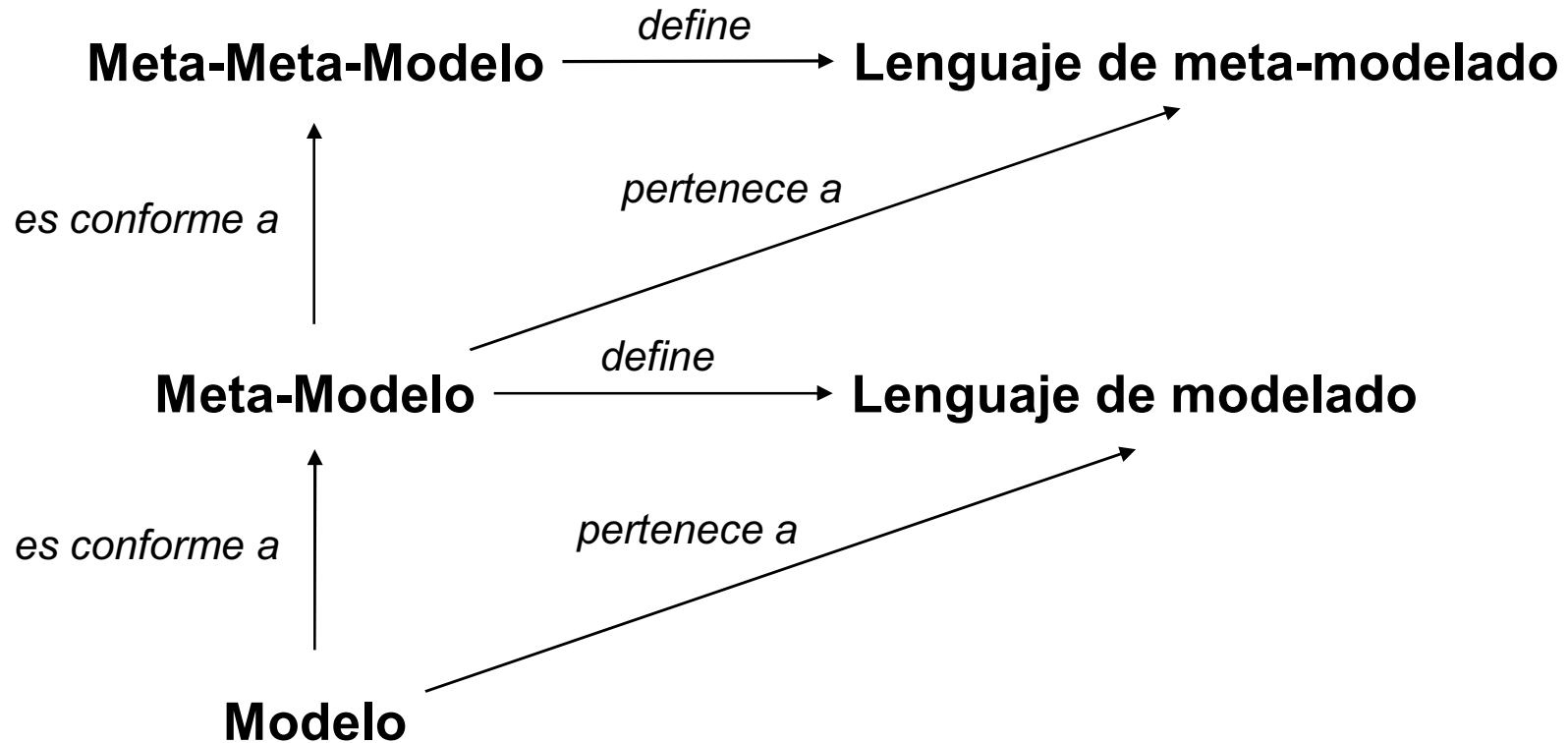


¡¡ Un “meta-infierno” de siglas !!

Introducción al DSDM

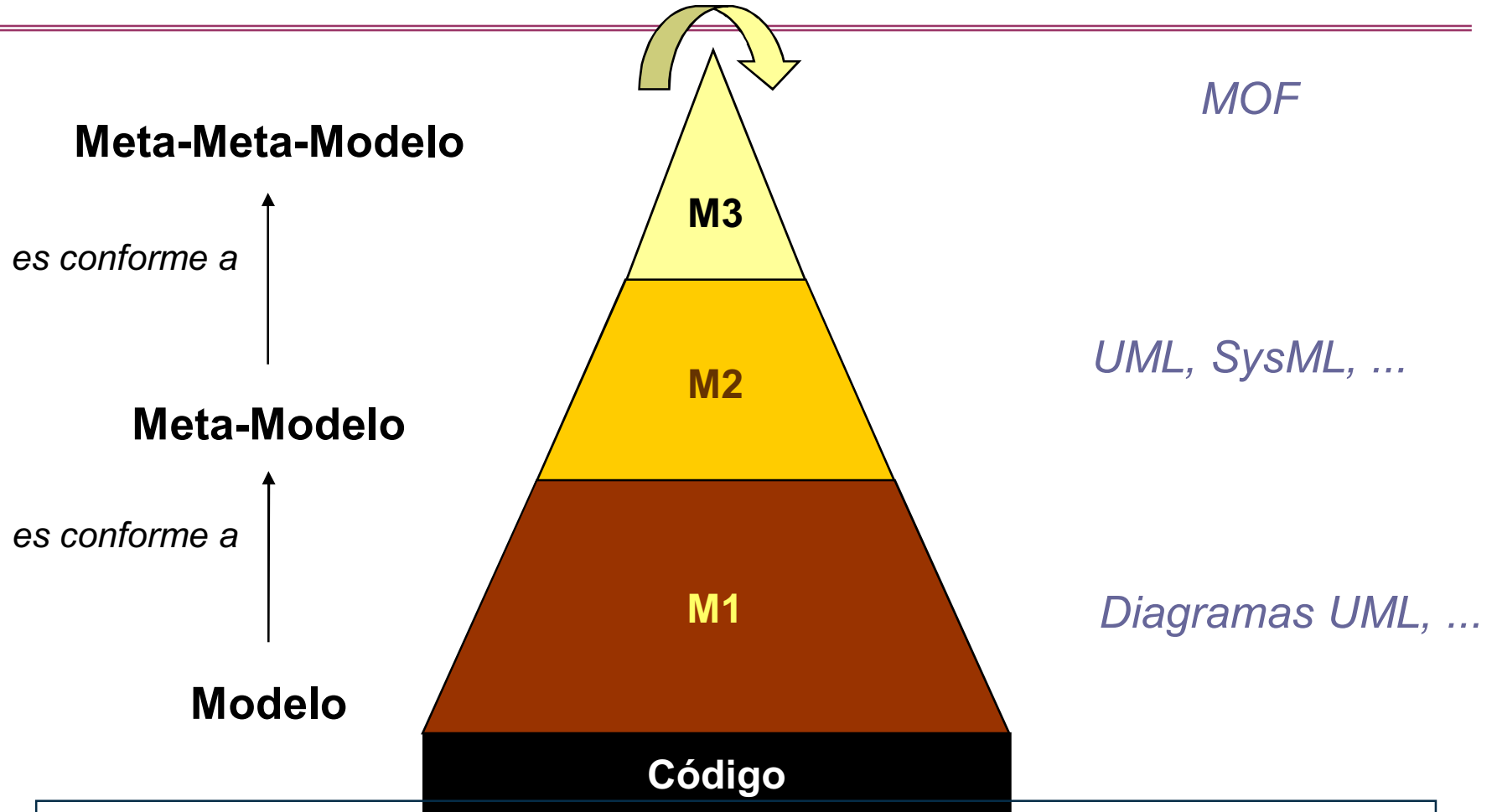


Introducción al DSDM



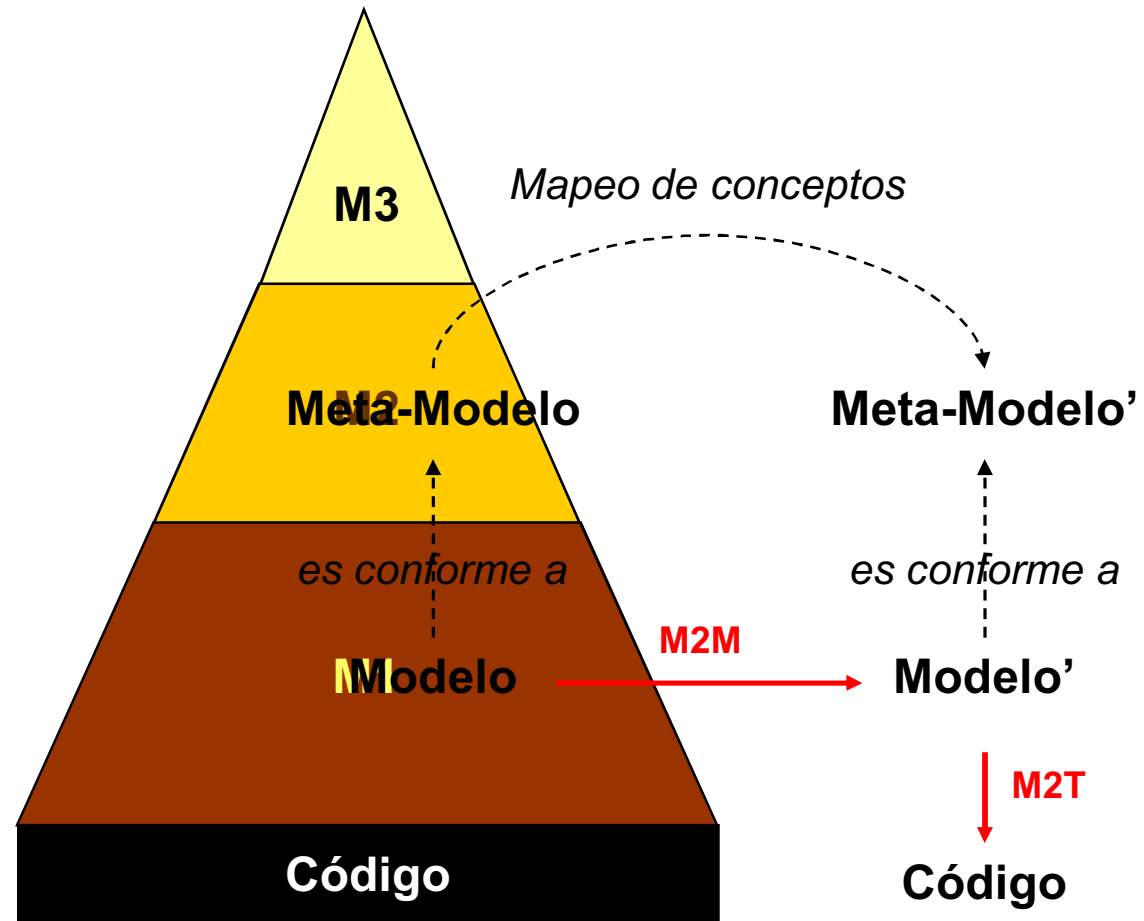
**J. M. Favre, *Foundations of Meta-Pyramids: Languages vs. Metamodels*
Episode II: Story of Thotus the Baboon**

Introducción al DSDM

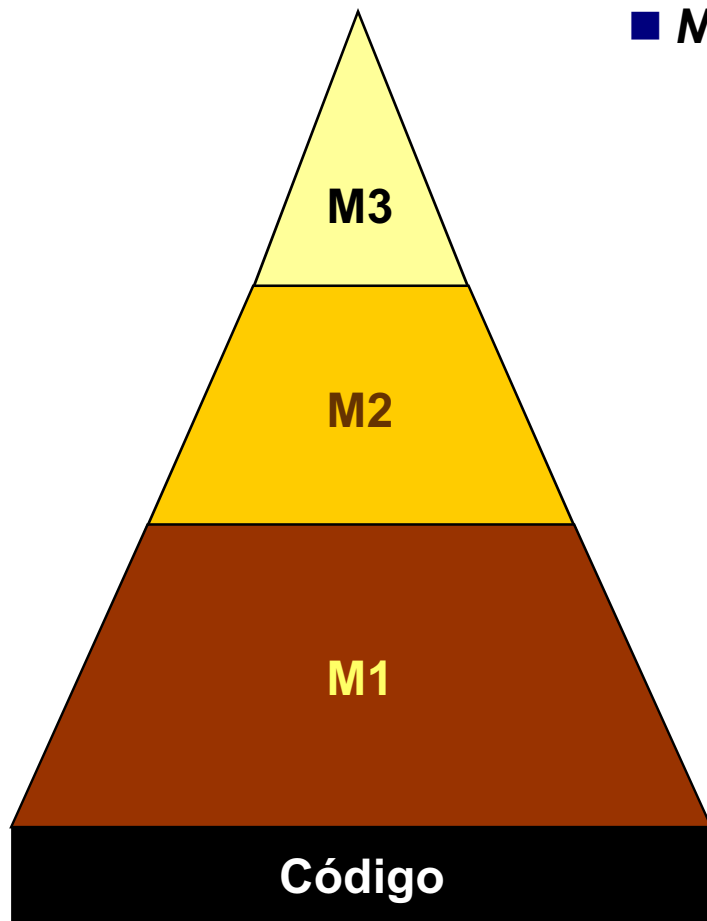


**J. M. Favre, *Foundations of Meta-Pyramids: Languages vs. Metamodels*
Episode II: Story of Thotus the Baboon**

Introducción al DSDM

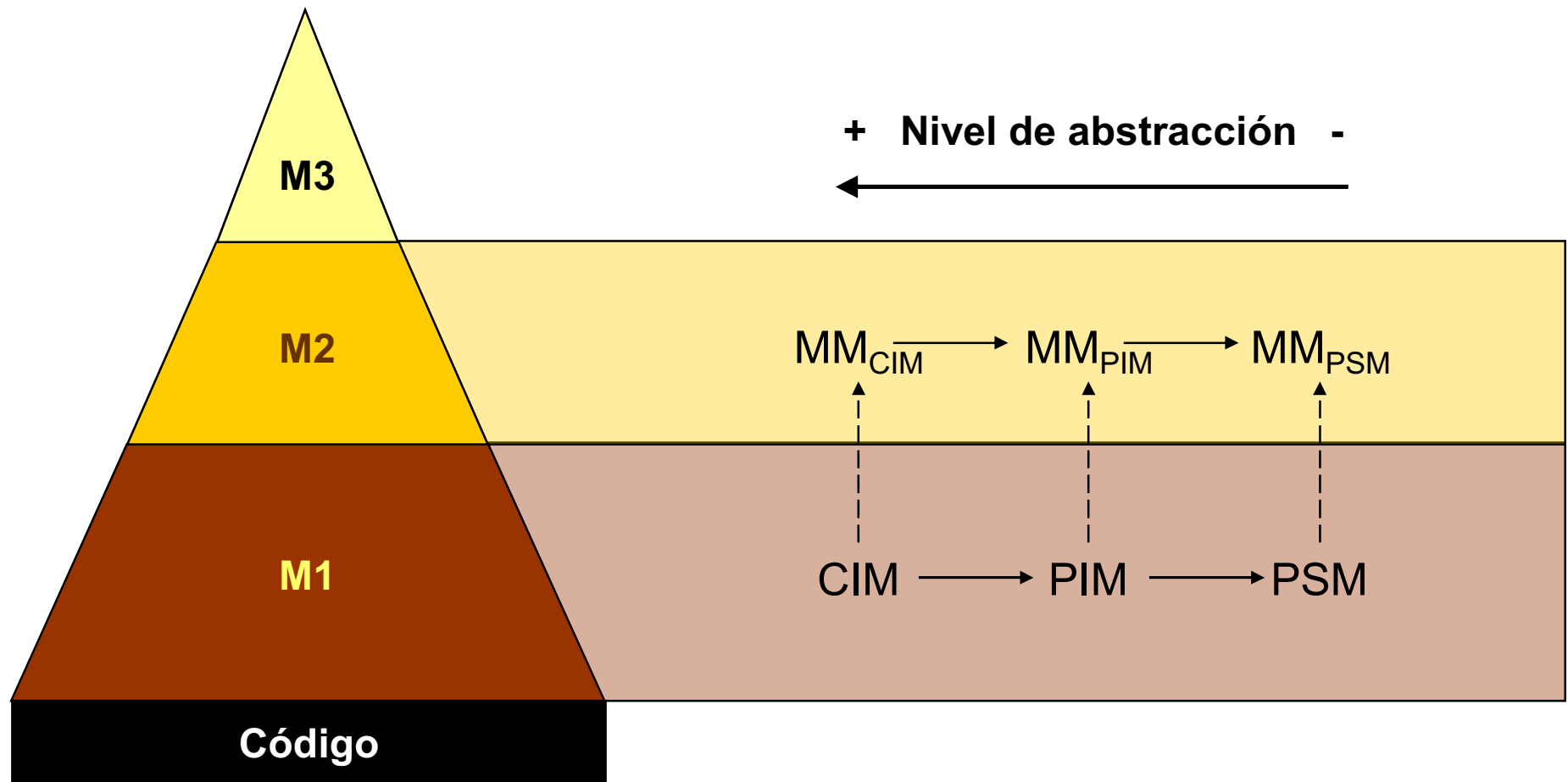


MDE y MDA



- **MDA** clasifica los modelos en tres categorías:
 - **CIM** (*Computational Independent Models*)
 - **PIM** (*Platform Independent Models*)
 - **PSM** (*Platform Specific Models*)

MDE y MDA



La plataforma Eclipse

- Eclipse es una plataforma abierta y de libre distribución

<http://www.eclipse.org>



Enterprise Development



Embedded + Device Development



Rich Client Platform



Language IDE

- En su desarrollo participan importantes empresas como Borland, IBM, Intel, Motorola, etc.
- Actualmente, la comunidad Eclipse se organiza en torno a múltiples proyectos que evolucionan en paralelo de manera independiente o cooperativa.
- Actualmente, los proyectos relacionados con DSDM se encuentran entre los más activos. Entre ellos, cabe destacar: **EMF, GMF, M2M, M2T, TMF, GMT...**

<http://www.eclipse.org/modeling/>



La plataforma Eclipse

- Necesita run-time de Java (*jre*)
 - Determinados plug-ins sólo funcionan con JRE 1.6
- No necesita instalación (se descomprime en cualquier carpeta)
- Fácil de extender con distintos plug-ins:
 - Se descargan y descomprimen directamente en `\eclipse`
 - También utilizando el menú “[Help](#) → [Software Updates](#)”
- Entorno de trabajo:
 - **Workspace**: directorio donde se almacenan todos los proyectos relacionados. Mantienen sus propias propiedades.
 - **View**: ventanas de utilidad, como gestor de proyectos, pestaña de propiedades, consola, etc.
 - **Perspective**: agrupación de vistas (*views*) que facilitan alguna tarea concreta, e.g. desarrollo Java.

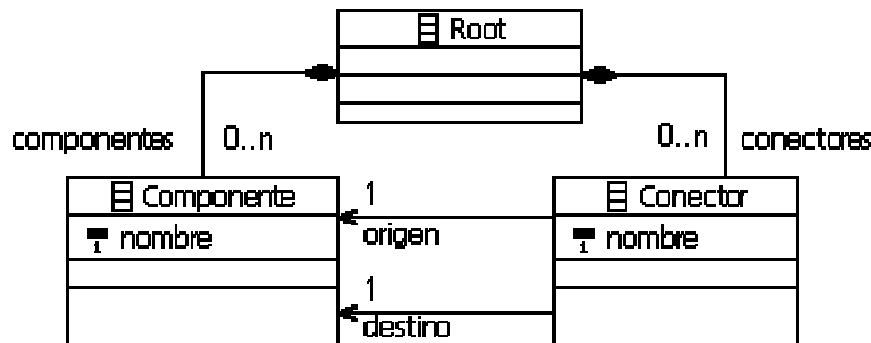
Eclipse Modeling Framework



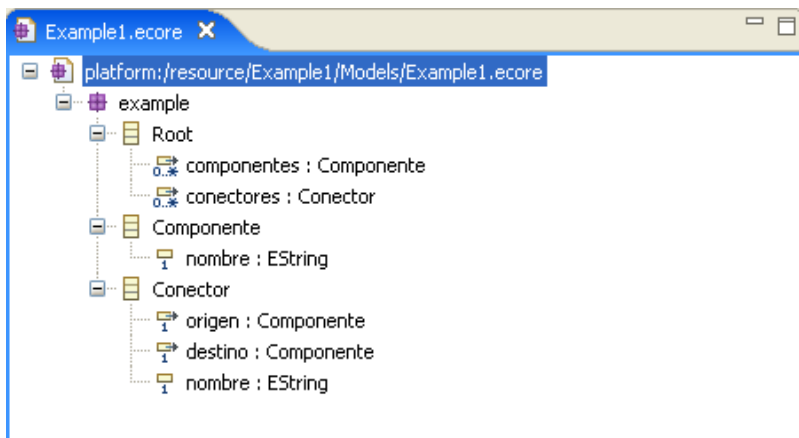
Implementación de Essential-MOF

- Soporte básico para DSDM en Eclipse
- EMF agrupa otros plug-ins que permiten:
 - Realizar consultas sobre modelos EMF (*Query*)
 - Verificar la conformidad de los modelos respecto a sus meta-modelos (*Validation*)
 - Generar una implementación Java de los meta-modelos
 - Generar editores básicos de modelos en forma de árbol (tree-editor)

Un ejemplo de meta-modelo



Example1.ecore_diagram



Example1.ecore

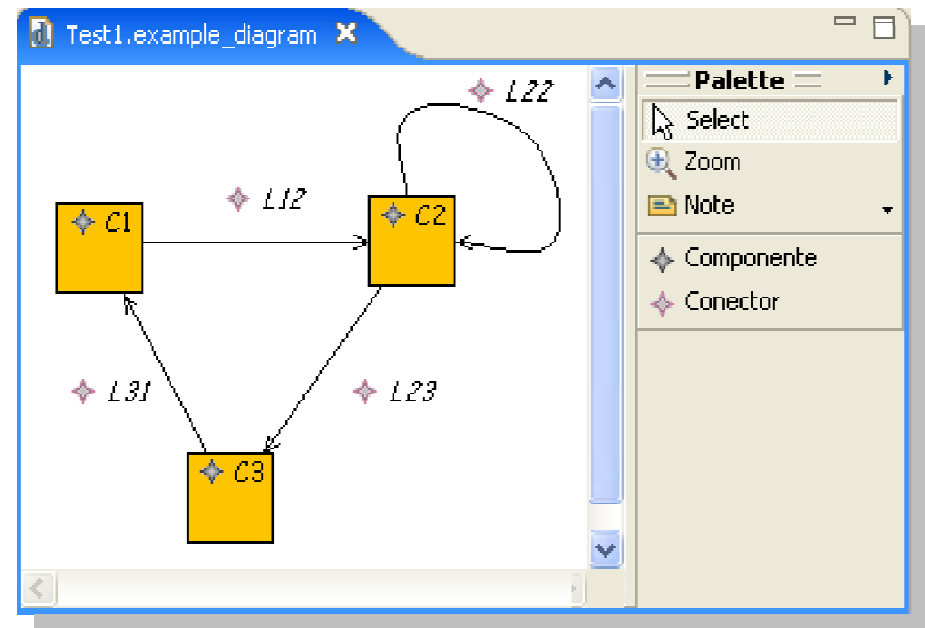
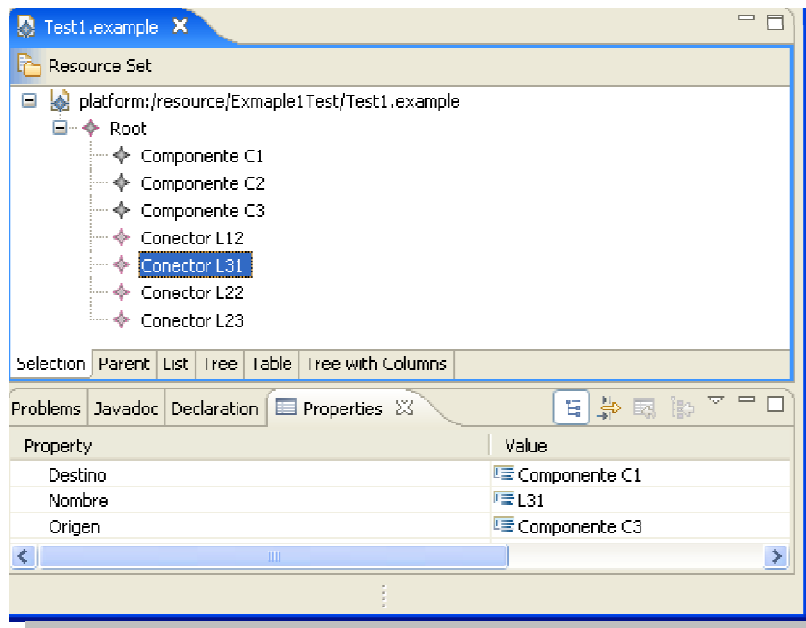
```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
...
<eClassifiers xsi:type="ecore:EClass" name="Root">
  <eStructuralFeatures xsi:type="ecore:EReference"
    name="componentes" upperBound="-1"
    eType="#//Componente" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
    name="conectores" upperBound="-1"
    eType="#//Conector" containment="true"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Componente">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="nombre" lowerBound="1"
    eType="ecore:EDatatype"
  </eStructuralFeatures>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="Conector">
  <eStructuralFeatures xsi:type="ecore:EReference"
    name="origen" lowerBound="1"
    eType="#//Componente"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
    name="destino" lowerBound="1"
    eType="#//Componente"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute"
    name="nombre" lowerBound="1"
    eType="ecore:EDatatype"
  </eStructuralFeatures>
</eClassifiers>
</ecore:EPackage>
```



Herramientas de modelado

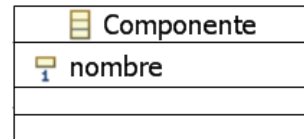
- Editores reflexivos **EMF** (tree-editors)
- Plug-ins para la generación de editores gráficos:
 - **GMF**: *Graphical Modeling Framework*. Proporciona un plug-in generativo y la infraestructura de soporte necesaria
 - **TOPCASED**: *Toolkit In OPen source for Critical Applications & SystEms Development*. Es un entorno integrado y abierto para modelado de sistemas embebidos, desarrollado y mantenido por un consorcio de universidades, centros de investigación y empresas del sector
- Plug-ins para la generación de editores textuales:
 - **Xtext**: parte de openArchitectureWare
 - **TCS**: desarrollado por el grupo ATLAS (como AMMA, ATL, etc.)

Un ejemplo de modelo

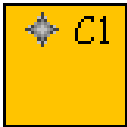
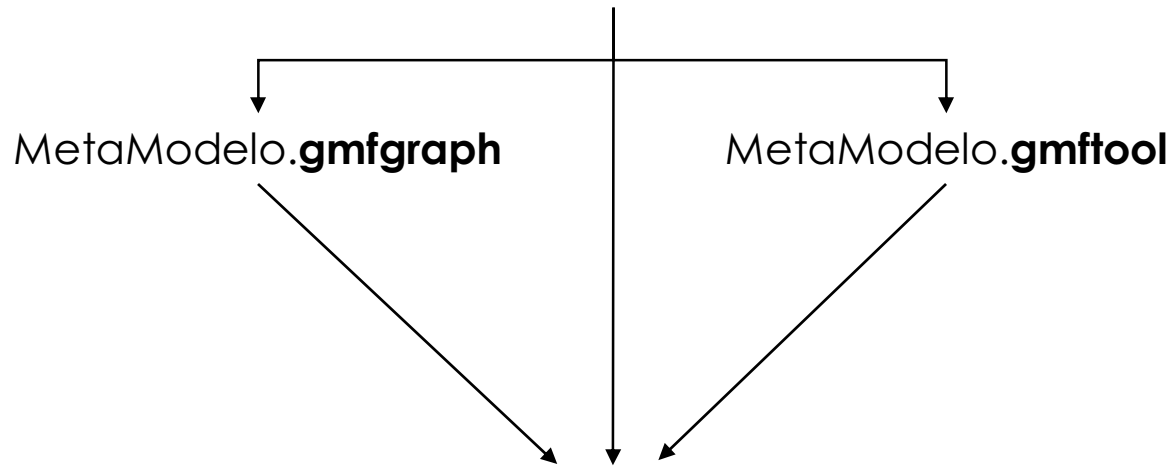


```
<?xml version="1.0" encoding="UTF-8"?>
<example:Root xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:example="example">
  <componentes nombre="C1"/>
  <componentes nombre="C2"/>
  <componentes nombre="C3"/>
  <conectores origen="//@componentes.0" destino="//@componentes.1" nombre="L12"/>
  <conectores origen="//@componentes.2" destino="//@componentes.0" nombre="L31"/>
  <conectores origen="//@componentes.1" destino="//@componentes.1" nombre="L22"/>
  <conectores origen="//@componentes.1" destino="//@componentes.2" nombre="L23"/>
</example:Root>
```

Graphical Modeling Framework

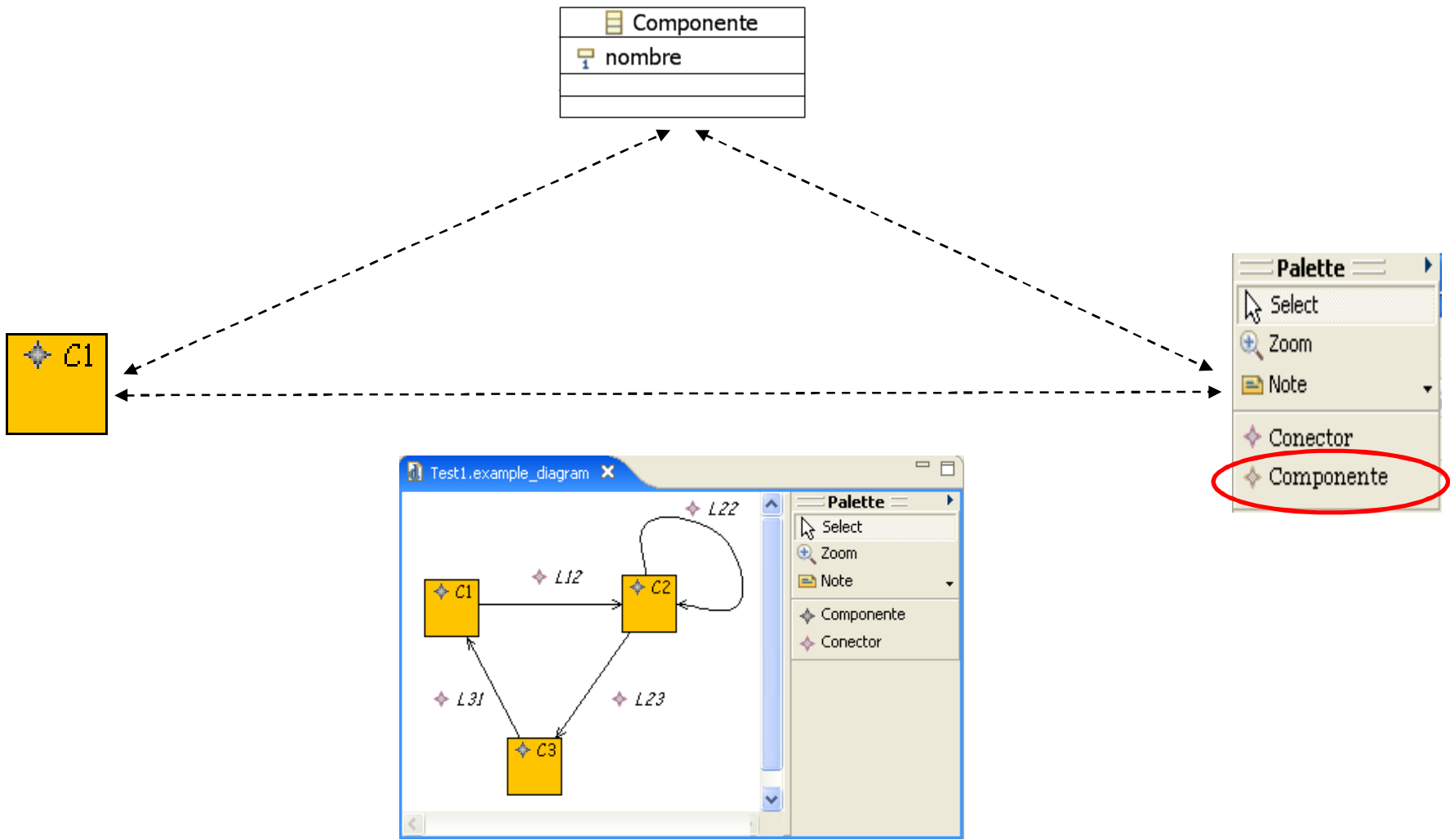


MetaModelo.ecore
MetaModelo.ecore_diagram



MetaModelo.gmfmap
(permite añadir restricciones OCL)

Graphical Modeling Framework





Transformaciones M2M

■ Basados en reglas:

- **QVT**: Operational QVT, Medini QVT (QVT Relational), SmartQVT (QVT Operational), ...
- **Otros**: ATL, RubyTL, Tefkat, Epsilon, oAW...

■ Basados en grafos: AGG, Viatra, Moment, ...

“Lenguajes de Transformación”:

<http://www.slideshare.net/jesus.sanchez/lenguajes-de-transformacin-presentation>

ATLAS Transformation Language

- Lenguaje híbrido basado en reglas
- Implementa la mayoría de operadores y tipos OCL y algunos propios
- ATL permite transformaciones MIMO
- Tiene “drivers” para meta-modelos UML y MDR

```
create mr : mrMM from er : erMM;
```

```
rule entidad2tabla {
```

```
  from f : erMM!Entidad
```

```
  to   t : mrMM!Tabla (nombre<-f.nombre, columnas<-f.atributos,  
                      clavePrimaria<-f.atributos->any(i | i.clavePpal = true))
```

```
  do {
```

```
    if (t.clavePrimaria = OclUndefined) {
```

```
      'No hay clave primaria en '.concat(f.nombre).println();
```

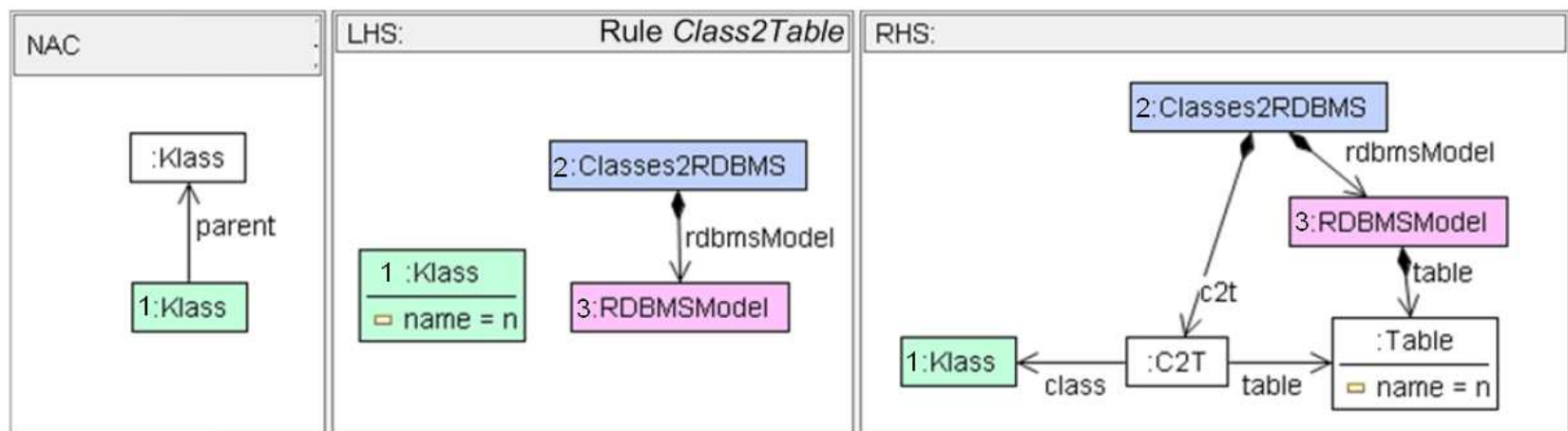
```
    }
```

```
  }
```

```
}
```

AGG (plug-in Tiger EMT)

- Reglas representadas con diagramas (visualmente)
- Aproximación más simple e intuitiva que los lenguajes de transformación textuales
- Reglas de transformación
 - Precondiciones (LHS, *Left Hand Side*)
 - Efecto de la regla (RHS, *Right Hand Side*)
 - Condiciones de aplicación negativa (NAC)





Transformaciones M2T

- **Basados en plantillas:** Acceleo, JET, Xpand, Epsilon, ...
- **Basados en lenguajes:** MOFScript, RubyTL, Kermet, ...

Java Emitter Templates (JET)

- Tecnología basada en plantillas, similar a JSP, que son traducidas a clases Java
- Permite añadir código Java
- Parte de EMF

```
<c:iterate select="//ES[@type!='PLC_Mark']" var="i" delimiter=";">
  <c:choose select="$i/@type">
    <c:when test="PLC_Input">
      <c:get select="$i/@name"/> : IN STD_LOGIC
    </c:when>
    <c:when test="PLC_Output">
      <c:get select="$i/@name"/> : OUT STD_LOGIC
    </c:when>
  </c:choose>
</c:iterate> );
```



MOFScript

- Admite múltiples modelos de entrada
- Lenguaje mixto: declarativo/imperativo

```
MetaModel.StateMachine :: main ( ){  
    file sm_ads (self.name + ".ads")  
    var stateNames:List = getStateNames ( )  
    var transitionNames:List = getTransitionNames ( )  
    sm_ads.println("package " + self.name + " is\n")  
    sm_ads.print("\t type T_State is ("  
    stateNames -> forEach ( s : String ) {  
        if (position() != 0) sm_ads.print (" , "  
        sm_ads.print ( s )  
    }  
}
```



Herramientas auxiliares

- **Service Data Objects (SDO)**: framework para aplicaciones SOA que integra patrones J2EE
- **EMF Compare**: comparación entre modelos EMF. El delta es también un modelo
- **Connected Data Objects (CDO)**: tecnología para distribuir modelos y meta-modelos EMF, basado en una arquitectura en 3 capas con repositorio central (BBDD relacional, BBDD OO, sistema de ficheros)



Herramientas auxiliares

- **Object Constraint Language (OCL)**
Permite realizar consultas y verificar restricciones OCL en modelos
- **Unified Modeling Language 2.x (UML2):**
Proporciona una implementación EMF del meta-modelo de UML 2.x
- **EMF Ontology Definition Metamodel (EODM)**
Proporciona una implementación del meta-modelo RDF(S)/OWL y un conjunto de herramientas de inferencia, transformación y análisis

Entornos integrados

- **MOMENT**: *MOdel manageMENT*, framework para la gestión de modelos basado en Maude (lenguaje algebraico). Permite:
 - Creación y gestión de modelos
 - Transformación de modelos
 - Verificación formal: restricciones OCL + model checking de invariantes

<http://moment.dsic.upv.es/>

MOMENT

A framework for MOdel manageMENT

Entornos integrados

- **AMMA**: *ATLAS Model Management Architecture*
 - **ATL**: lenguaje de transformación de modelos
 - **AMW**: model weaver
 - **AM3**: permite la gestión (i.e. creación, visualización, acceso, modificación y almacenamiento) de modelos.
 - **ATP**: integración con otros espacios tecnológicos, e.g. XML, DBMS, etc.
 - **TCS**: generación de editores textuales de modelos

<http://www.eclipse.org/m2m/atl/>



Entornos integrados

- **openArchitectureWare**: framework modular para DSDM. Proporciona un entorno unificado para:
 - Realizar transformaciones M2T, M2M y T2M
 - Verificar restricciones sobre modelos
 - Realizar model *weaving*
 - Soporta distintos tipos de modelos: EMF, UML2, XML, ...
 - Motor de ejecución de trabajos (*workflow engine*)

<http://www.openarchitectureware.org/>



Entornos integrados

- **Epsilon**: familia de lenguajes independientes de la tecnología de meta-modelado basado en el Epsilon Object Language (EOL). Proporciona:
 - Transformaciones M2M y M2T
 - Validación, comparación y fusión de modelos
 - Model weaving
 - Motor de ejecución de trabajos (*workflow engine*)

<http://www.eclipse.org/gmt/epsilon/>



Entornos integrados

- **Cadena:** entorno unificado para DSDM, orientado al diseño de LPS basadas en componentes. Define su propio ADL (CALM) y utiliza CORBA
 1. Definición del modelo de componentes (soporta CCM, EJB, etc.)
 2. Modelado de LPS y derivación de productos
 3. Generación del código del producto

<http://cadena.projects.cis.ksu.edu/>



Entornos integrados

- **Kermeta**: lenguaje orientado a modelos y aspectos. Permite:
 - Definición de la semántica estática (OCL) y dinámica (comportamiento) de los meta-modelos
 - Definición de transformaciones M2M, M2T y T2M
 - Model Weaving

<http://www.kermeta.org/>





Entornos integrados

- **RubyTL:** conjunto de DSLs embebidos basados en el lenguaje Ruby (lenguaje dinámico e interpretado). Proporciona:
 - Lenguaje para definir transformaciones M2M y M2T
 - Lenguaje de validación similar a OCL
 - Permite importar/exportar EMOF y ECore
 - Permite definir cadenas de transformaciones

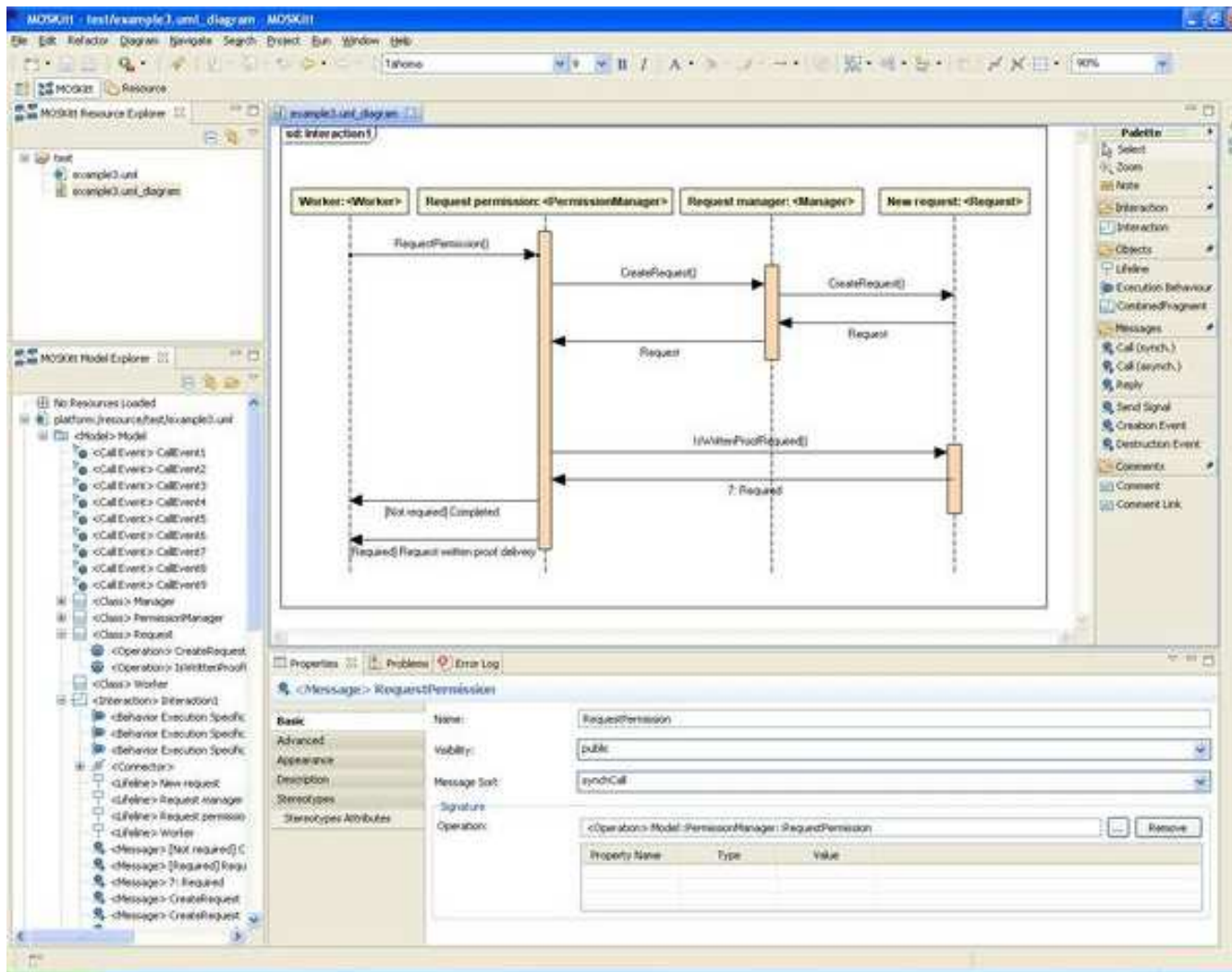
<http://rubytl.rubyforge.org/>



Herramientas para UML

- **UML2Tools:** casos de uso, clases, componentes, estructuras compuestas, despliegue, actividades, interacción, máquina de estados y profiles
- **Papyrus:** proporciona soporte para algunos profiles, como MARTE, SysML, East-ADL y CWM. Extensible
- **MOSkitt:** desarrollado por la Consellería de Infraestructuras y Transporte CV. Casos de uso, clases, actividad, secuencia, máquina de estados
- **Omondo:** plug-in propietario con versión gratuita. Casos de uso, clases, paquetes, objetos, secuencia, actividades, máquina de estados. Generación de código Java y aplicaciones web

MOSKitt



Otras herramientas para DSDM

- **Meta-Edit+**: desarrollado por Metacase



- **DSL Tools**: desarrollado por Microsoft

Domain-Specific Development
with Visual Studio DSL Tools



Lecciones aprendidas

■ Restricciones OCL:

- Actualmente sólo pueden incorporarse a los meta-modelos EMF como meras anotaciones. Posteriormente, se pueden modificar las transformaciones JET que generan los editores para que las tengan en cuenta.
- La opción más común (aunque no muy razonable) es incorporar estas restricciones directamente a las especificaciones GMF

■ GMF:

- Cuando la funcionalidad que debe proporcionar el editor no es la contemplada por GMF, es necesario retocar el código generado automáticamente por GMF...





Lecciones aprendidas

■ ATL:

- Los mensajes de error del compilador son bastante “crípticos”
- Mejor adoptar un enfoque de desarrollo incremental: *“Escribe regla, valida regla”*

■ MOFScript:

- Existe una nueva versión (v1.3.4) para Eclipse 3.3, disponible desde marzo de 2009
- Cuando las transformaciones son de cierto tamaño deja de funcionar correctamente

■ JET:

- Es bastante rápido y eficiente y permite ejecutar código Java. Sin embargo, es bastante farragoso de usar y proporciona un control muy primitivo de la salida generada



Lecciones aprendidas

- Existen numerosas dependencias e incompatibilidades entre las distintas versiones de los plug-ins.
 - Las versiones distribuidas como parte de los “bundles” de modelado de Eclipse, han supuesto un gran avance en este sentido.
- Las News de Eclipse son la mejor (o la única) fuente de ayuda fiable/actualizada para muchas de las herramientas comentadas!!