Ingeniería de Software Dirigida por Modelos

Procesos y Métodos de Modelado para la Ingeniería Web y Web Semántica

Máster Universitario en Sistemas Inteligentes

Curso 2023-2024

4-15 de marzo de 2024

Dr. Francisco José García Peñalvo
GRupo de investigación en InterAcción y eLearning (GRIAL)
Universidad de Salamanca

fgarcia@usal.es
http://grial.usal.es
http://twitter.com/frangp

Contenidos

1. Desarrollo Dirigido por Modelos (MDD)
2. Metamodelos
3. Model Driven Architecture (MDA)
4. Software Factories
5. Caso de estudio. Dashboards personalizados

1. Desarrollo Dirigido por Modelos (MDD)

Evolución del desarrollo software

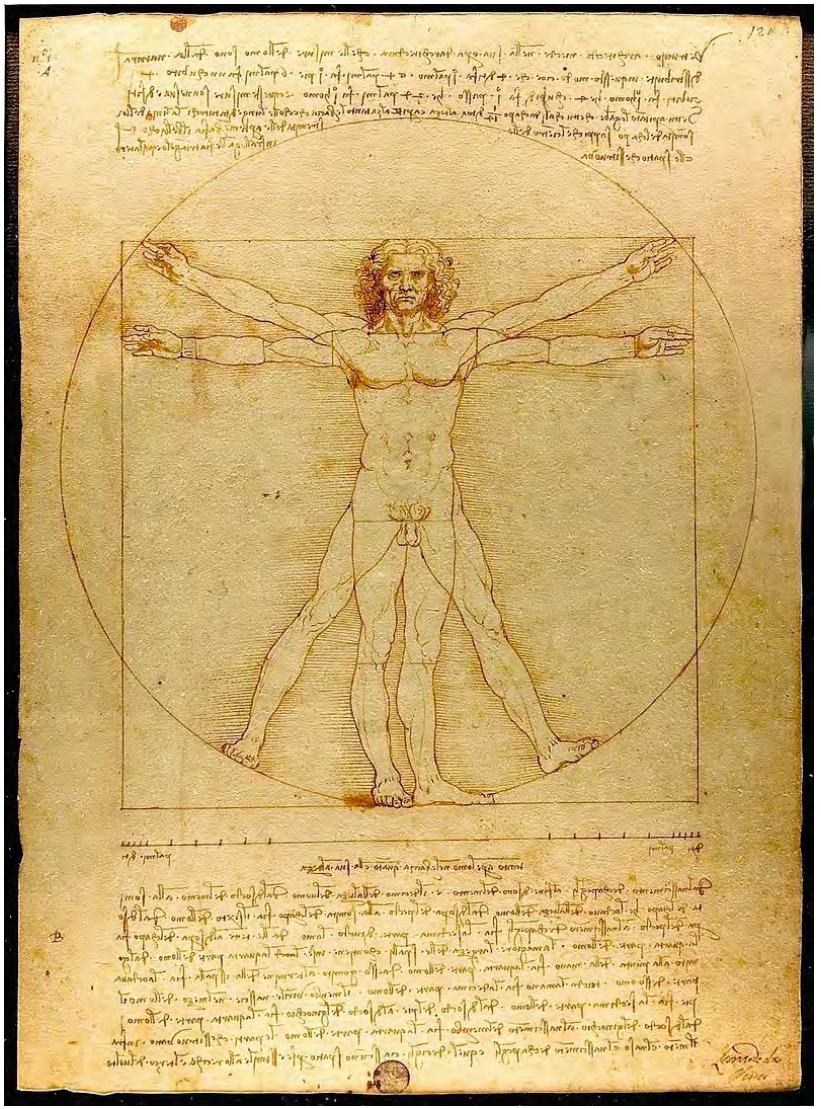
- Programación en Ensamblador
  - X86
- Programación Estructurada
  - Pascal, C
- Orientación a Objetos
  - C++, Java
- Componentes software
  - J2EE, .NET
- Orientación a Aspectos
  - AspectJ, Spring Framework AOP
- Arquitecturas orientadas a Servicios
  - WSDL, SOAP

Modelos en Ingeniería

- Hasta ahora la programación siempre ha sido el centro de atención
- Al igual que en otras ingenierías ¡hay que aumentar el nivel de abstracción!
- Los modelos ayudan a construir sistemas más complejos

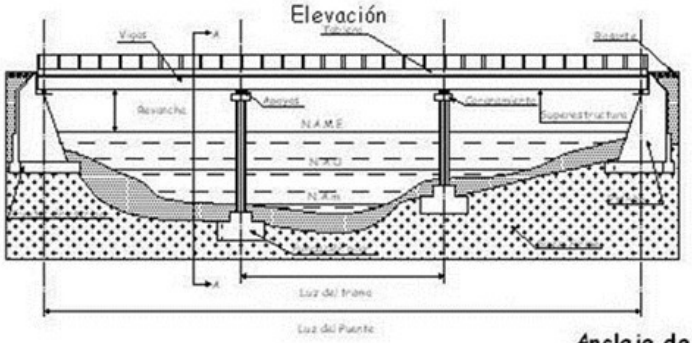
Modelos en Ingeniería

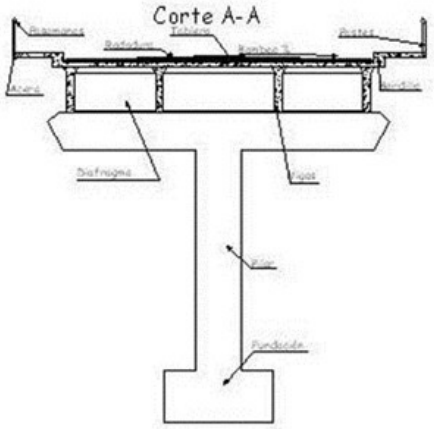
- Tan antiguos como las ingenierías por ejemplo el Hombre de Vitruvio de Leonardo da Vinci

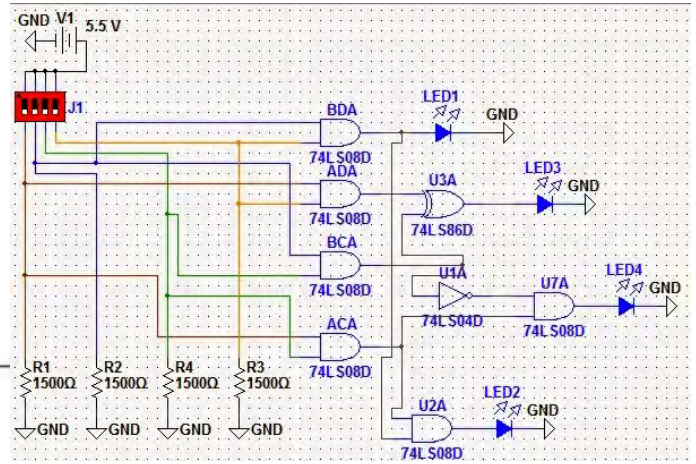


Modelos en Ingeniería

- Los ingenieros “tradicionales” siempre construyen modelos antes de construir sus obras y artefactos

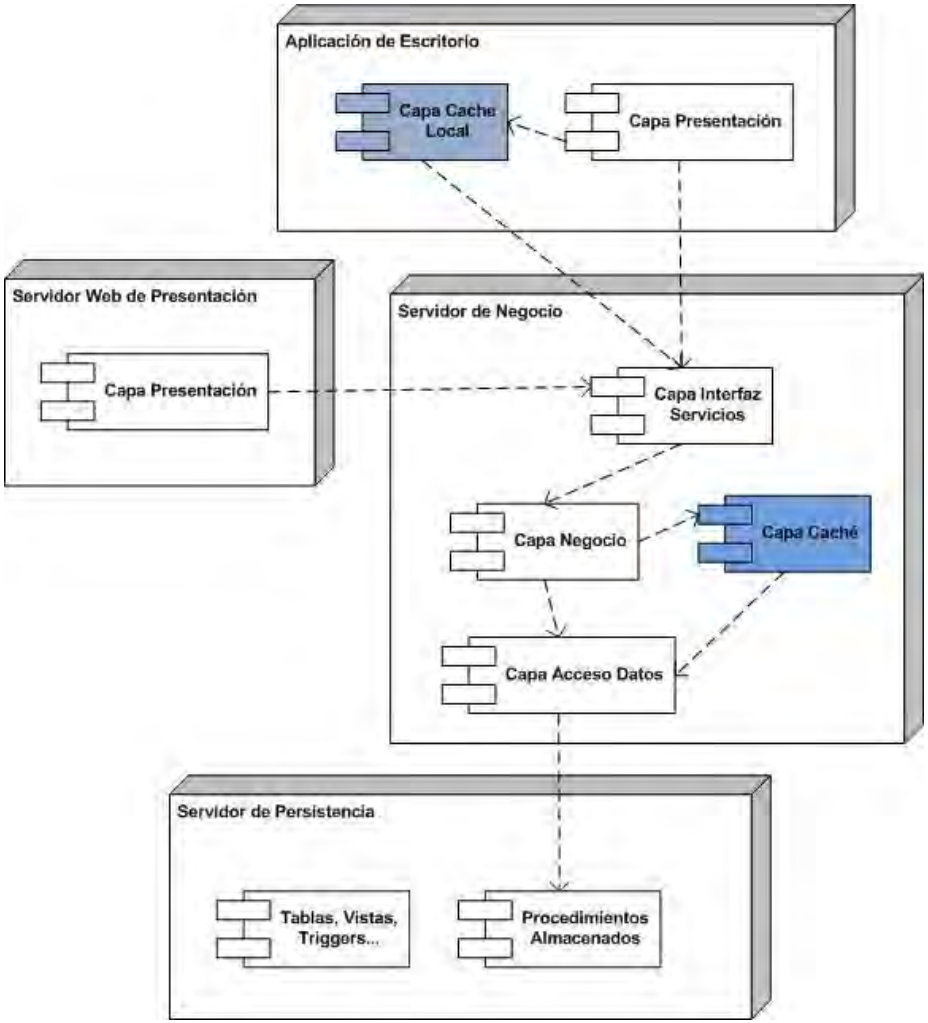






Modelos en Ingeniería

- También en Ingeniería del Software



Los modelos sirven para…

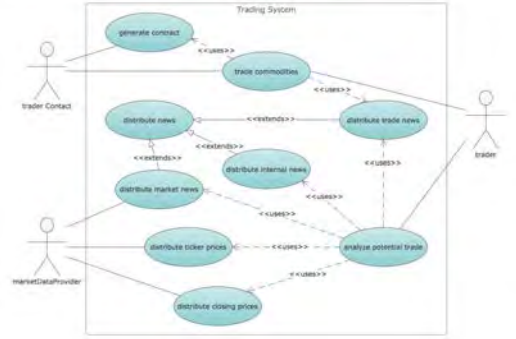
- Especificar el sistema
  - Estructura, comportamiento, etc.
- Comunicarse con los distintos *stakeholders*
- Comprender el sistema (si ya existe)
- Razonar y validar el sistema
  - Detectar errores y omisiones en el diseño
  - Prototipado (ejecutar el modelo)
  - Inferir y demostrar propiedades
- Guiar la implementación

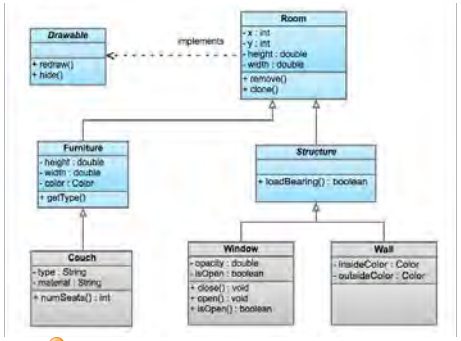
Características de los modelos

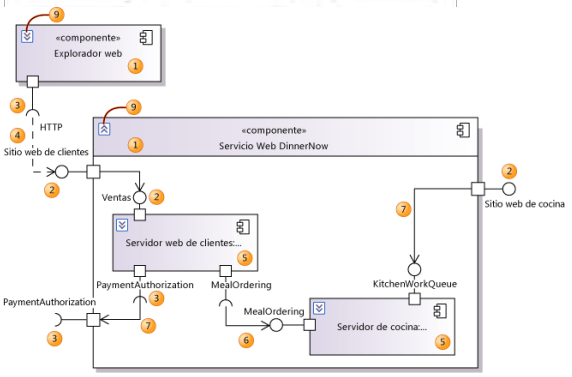
- Abstractos
  - Enfatizan ciertos aspectos, mientras que ocultan otros
- Comprensibles
  - Expresados en un lenguaje comprensible por los usuarios y *stakeholders*
- Precisos
  - Fieles representaciones del objeto o sistema modelado
- Predictivos
  - Deben de poder ser usados para inferir conclusiones correctas
- Baratos
  - Más fáciles y baratos de construir y estudiar que el propio sistema

¿Qué es un modelo de software?

- Descripción o especificación (de parte) de un sistema *software* desde un determinado punto de vista
- *A description of (part of) a system written in a well-defined language (equivalent to specification)* (Kleppe et al., 2003)
- *A representation of a part of the function, structure and/or behavior of a system* (Miller & Mukerji, 2001)
- *A description or specification of the system and its environment for some certain purpose. A model is often presented as a combination of drawings and text* (Object Management Group, 2014)
- *A set of statements about the system* (Seidewitz, 2003) (*Statement: expression about the system that can be considered true or false*)







Limitaciones actuales de los modelos de software

- Solo se usan como documentación
  - ¡Que además no se actualiza!
- *Gap* entre el modelo y la implementación del sistema
  - Grandes diferencias semánticas en los lenguajes respectivos
  - No hay herramientas de propagación automática de cambios
    - Cambios en el modelo no se reflejan en el código
    - Cambios en el código no se reflejan en el modelo (el modelo no vuelve a usarse jamás tras la primera implementación)
- Los distintos modelos del sistema no se armonizan
  - Suponen vistas de un mismo sistema, pero no hay forma de relacionarlas
  - No hay herramientas de integración de modelos
  - Cada lenguaje de vista tiene una semántica distinta del resto
- No hay ni lenguajes ni herramientas para manejar modelos
  - Solo editores, pero no hay compiladores, optimizadores, validadores, transformadores de modelos, etc.
- ¿Se está realmente hablando de Ingeniería del *Software*?

Software y modelos

> Software has the rare property that it allows us to directly evolve models into full-fledged implementations without changing the engineering medium, tools, or methods
>
> Bran Selic and John Hogg

- Esto facilita enormemente garantizar la fiabilidad entre los modelos y los sistemas producidos, puesto que todos viven en el mismo mundo
- **Corolario: El modelo *es* la implementación**
- **Salvedad:** Solo si el modelo contiene toda la información necesaria para producir el sistema

¿Qué es MDD?

- MDD - *Model Driven Development*
- Un enfoque de desarrollo de *software* donde las entidades de primer nivel son los modelos y las transformaciones de modelos
  - Frente a los programas y los compiladores, que constituyeron el paradigma análogo hace treinta años
- MDD implica la generación (casi) automática de implementaciones a partir de modelos
- En MDD son claves los lenguajes, tanto de modelado como de transformación de modelos. Los modelos son conformes a meta-modelos
- MDA es la propuesta para MDD que hace OMG, usando sus estándares
  - MOF, UML, OCL, XMI, QVT
  - MOF y UML permiten definir nuevas familias de lenguajes

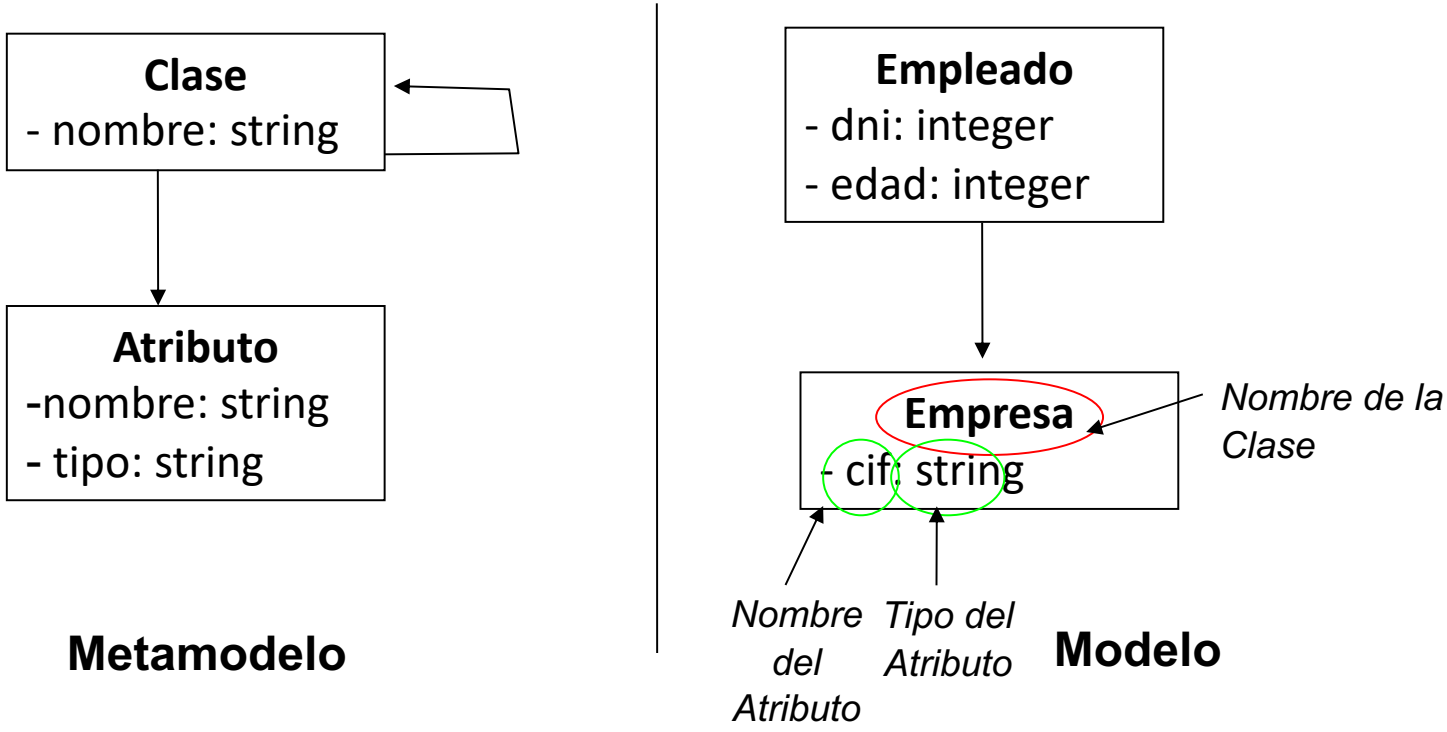
2. Metamodelos

Metamodelado

- Un metamodelo describe la estructura posible de los modelos (o lenguajes) de forma abstracta
  - Define las construcciones o elementos de un lenguaje de modelado y sus relaciones
  - Define las restricciones y las reglas de modelado sobre los elementos
  - No define la sintaxis concreta del lenguaje
- Un metamodelo define la sintaxis abstracta y la semántica estática de un lenguaje de modelado

Metamodelado

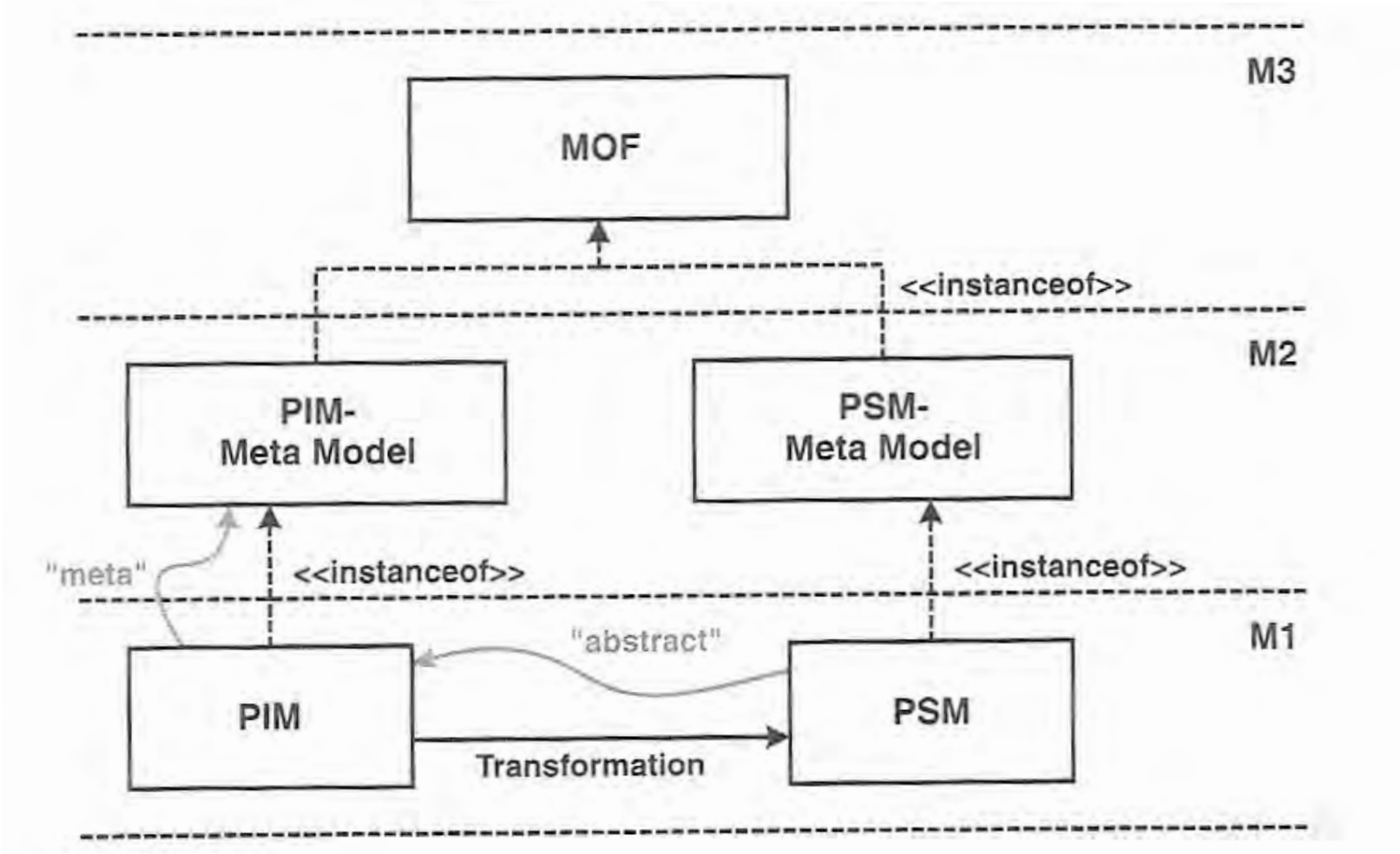
Un modelo para describir un lenguaje de modelado



Metamodelado

- Los metamodelos y los modelos tienen una relación clase-instancia
- Cada modelo es una instancia de un metamodelo
- Para definir un metamodelo se necesita un lenguaje de metamodelado que se describe con un meta metamodelo

Metaniveles vs. Niveles de Abstracción



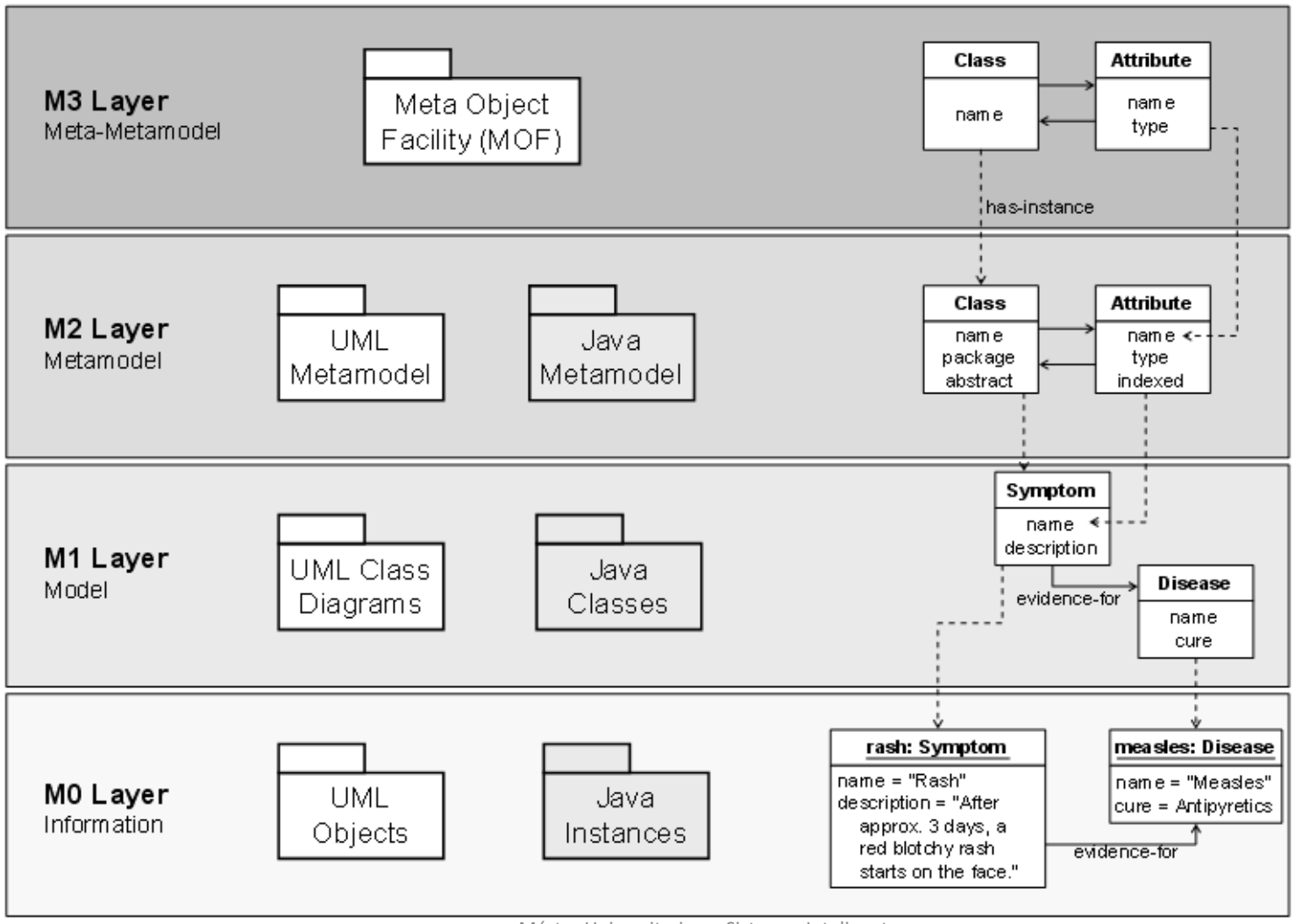
Hacia MOF…

- ¿Cómo definir un lenguaje de metamodelado?
  - Con un lenguaje de meta metamodelado
  - ¿Y cómo definir un lenguaje de meta metamodelado?
    - Con un lenguaje de meta metametamodelado
    - ¿Y cómo definir…?
- ¿Hasta cuándo seguir?
  - Hasta disponer de un lenguaje capaz de describir cualquier otro lenguaje (¡¡incluso a él mismo!!)

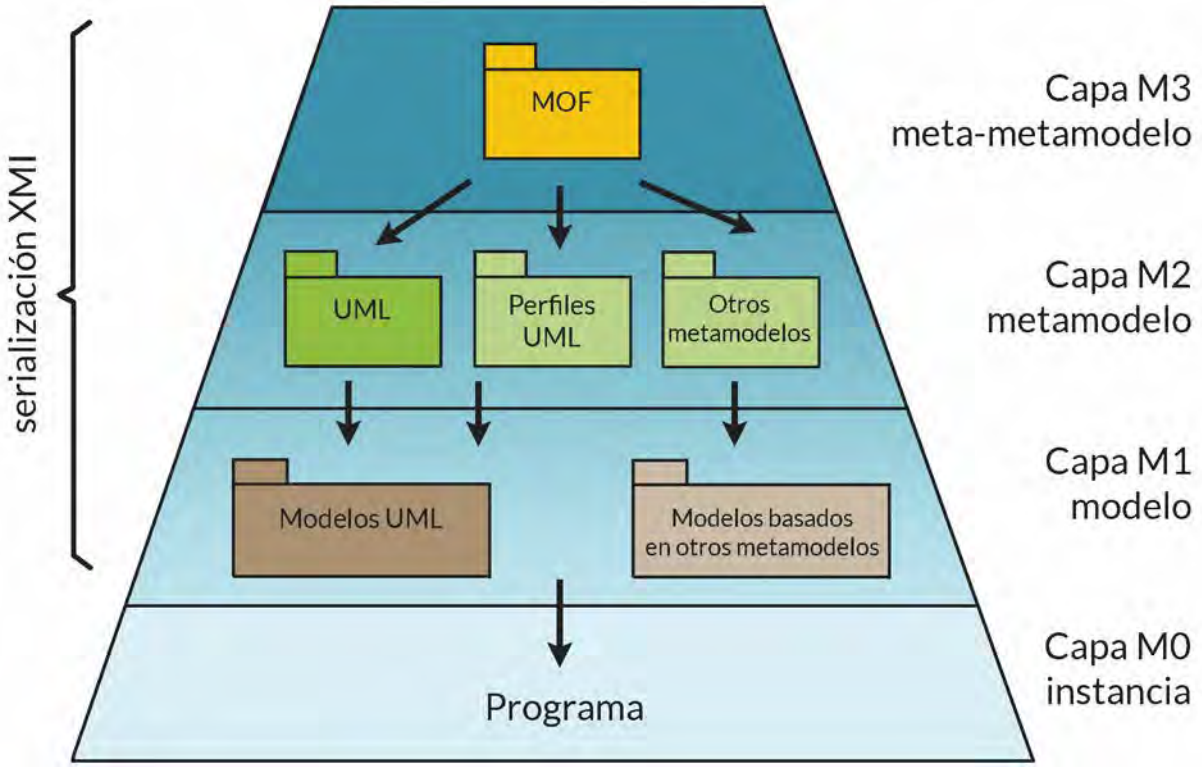
Meta Object Facility (MOF)

- Objetivo: Lenguaje “estándar” para especificar metamodelos
- Subconjunto de UML
- Se utiliza para definir los metamodelos de OMG (como por ejemplo el de UML)
- Versión actual: 2.5.1 (basado en UML 2)
  - http://www.omg.org/spec/MOF/2.5.1/
  - (Object Management Group, 2017)

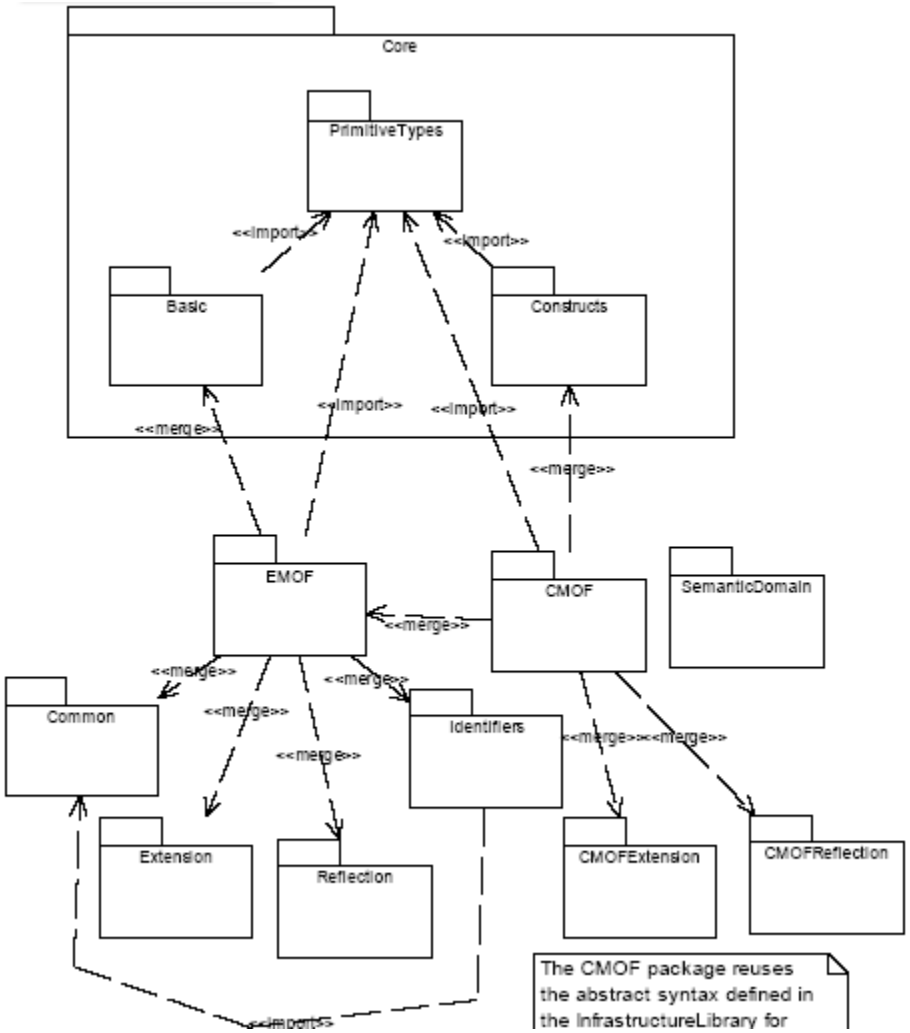
Meta Object Facility (MOF)



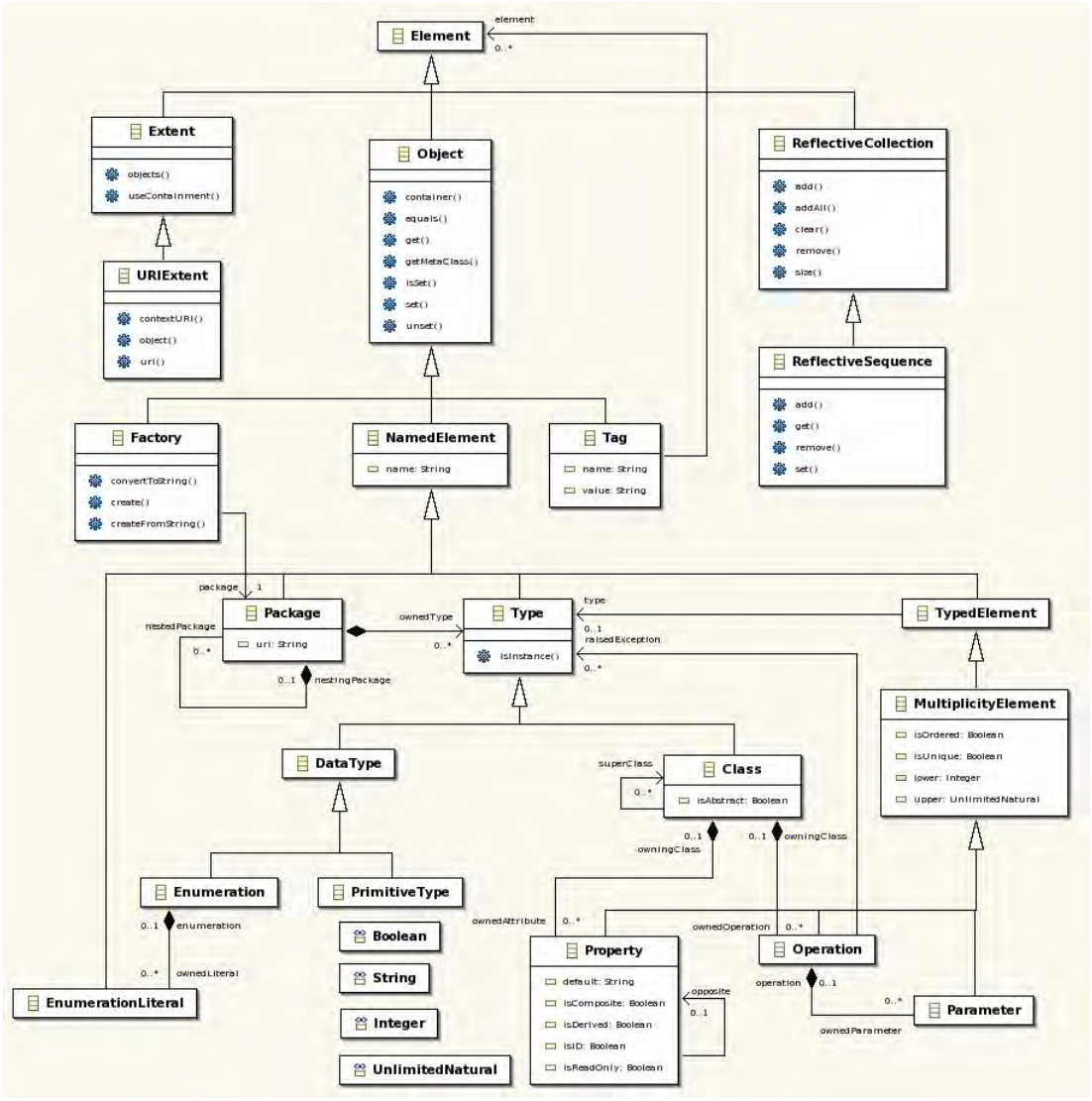
Meta Object Facility (MOF)



Meta Object Facility (MOF)



Meta Object Facility (MOF)



MOF

EMOF (Essential MOF) y CMOF (Complete MOF)

- MOF 2.5.1 está construido en base a un subconjunto de la infraestructura de UML 2.0 (el paquete `Core::Basic`) que le proporciona los conceptos y la notación gráfica para sus modelos
- MOF 2.5.1 se divide en dos paquetes principales
  - Essential MOF (EMOF)
  - Complete MOF (CMOF)
- Según la propia especificación de MOF, el objetivo de EMOF es permitir la definición de metamodelos simples utilizando conceptos sencillos, sin que se pierda la posibilidad en MOF de expresar modelos más complejos
- Para expresar modelos complejos se proporciona CMOF
  - CMOF se construye a partir de EMOF y el paquete `Core::Constructs` de UML 2

MOF

EMOF (Essential MOF) y CMOF (Complete MOF)

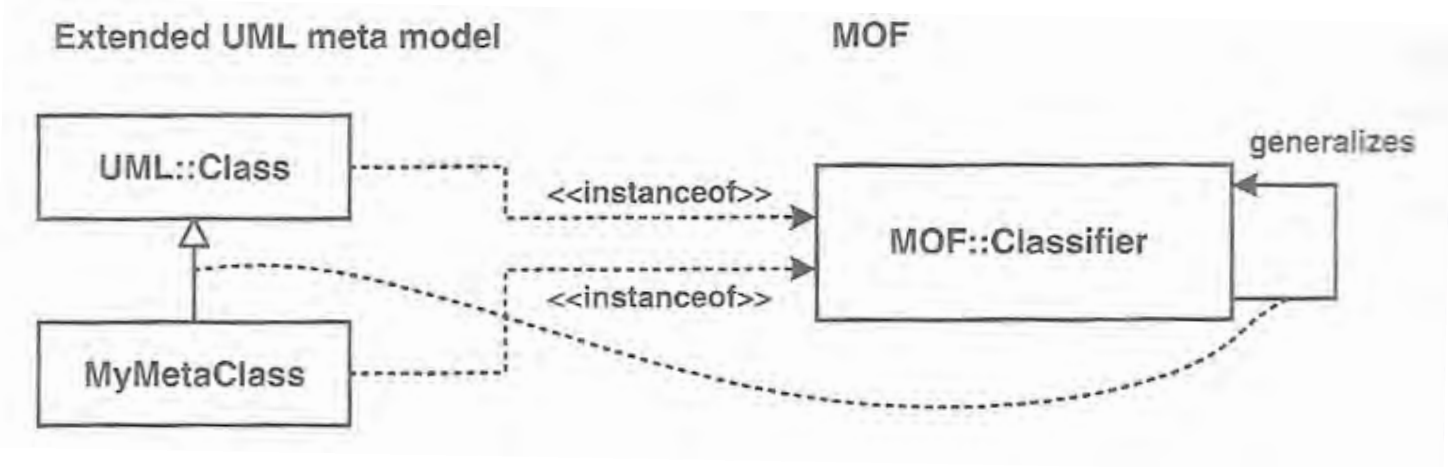
- CMOF extiende o redefine la semántica de algunos elementos de los paquetes `Identity`, `Reflection` y `Extension`
- Estos paquetes se encargan de
  - **Reflection**: proporciona la capacidad de autodescribirse
  - **Identity**: proporciona la capacidad de identificar los elementos sin depender de modelos externos
  - **Extensión**: proporciona la capacidad de extender los elementos con pares nombre/valor

Extensión de UML

- Extensión basada en el Metamodelo
- Extensión mediante Perfiles UML
  - Mediante UML 1.X
  - Mediante UML 2.X

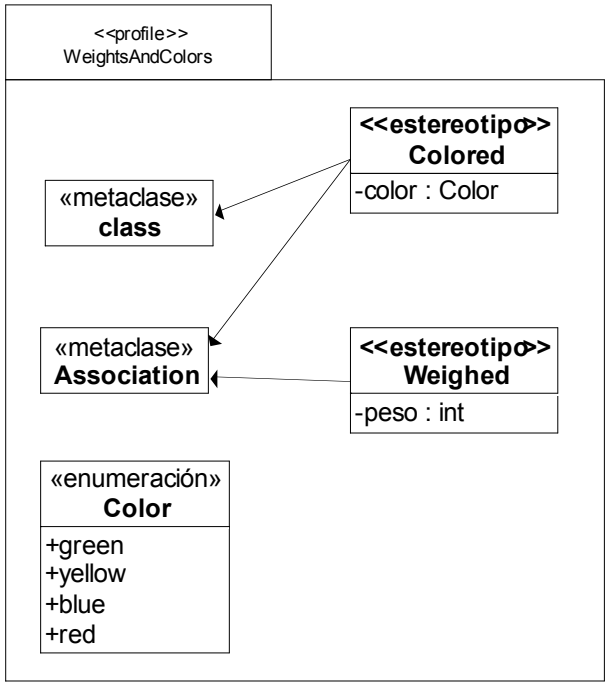
Extensión basada en el metamodelo

- Este tipo de extensión extiende el metamodelo UML
- La herramienta de soporte debe poder manipular un metamodelo de UML basado en MOF



Perfiles UML

- Un **perfil UML** es una agrupación de elementos de modelado (UML) que han sido **adaptados** para un propósito específico



Perfiles UML

- El **objetivo** de los perfiles es
  - Proporcionar un mecanismo directo para **adaptar** un metamodelo existente con constructores que son específicos para un dominio, plataforma o método particular. Dichas adaptaciones se agrupan en un perfil

- **No es posible eliminar** ninguna restricción de los metamodelos de UML, pero **es posible añadir** nuevas restricciones que son específicas para el perfil

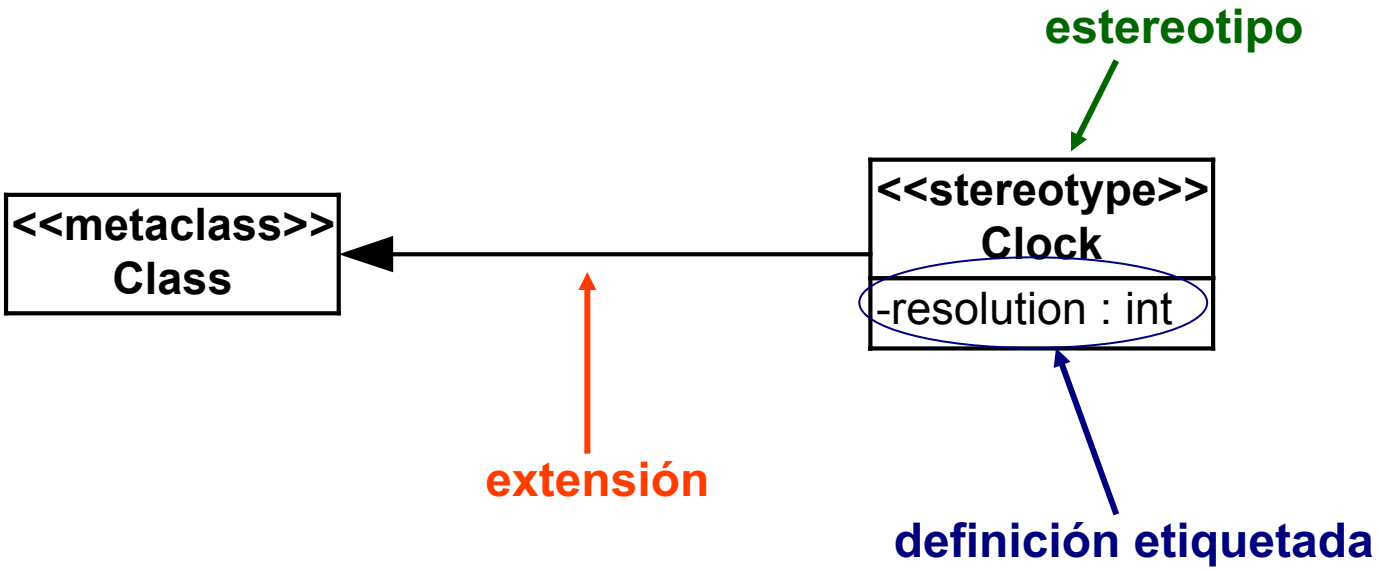
- El **paquete <<*Profiles*>>** de UML contiene mecanismos que permiten extender las metaclases de metamodelos existentes para adaptarlos a diferentes propósitos. En este paquete se definen los elementos
  - Estereotipo
  - Extensión

Perfiles UML

- Elementos que se utilizan en la definición de perfiles
  - **Estereotipo**. Define cómo se puede extender una metaclase de UML
  - **Extensión**. Se utiliza para indicar que las propiedades de una metaclase se extienden a través de un estereotipo
  - **Definición Etiquetada**. Especifican nuevos tipos de propiedades que pueden asociarse a elementos de modelado
  - **Restricción**. Pueden asociarse a cualquier elemento de modelado para redefinir su semántica

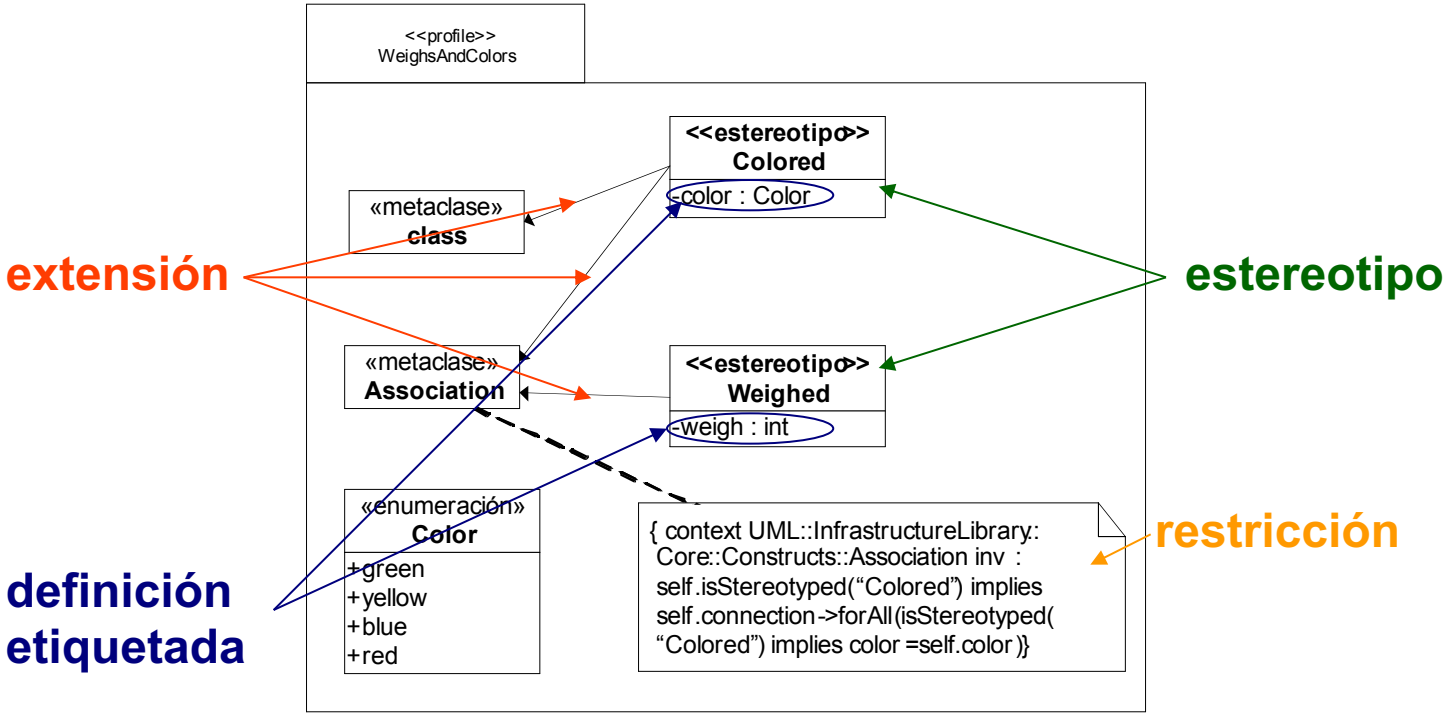
Perfiles UML

- Ejemplo sencillo de un Perfil UML



Perfiles UML

- Un ejemplo más complejo

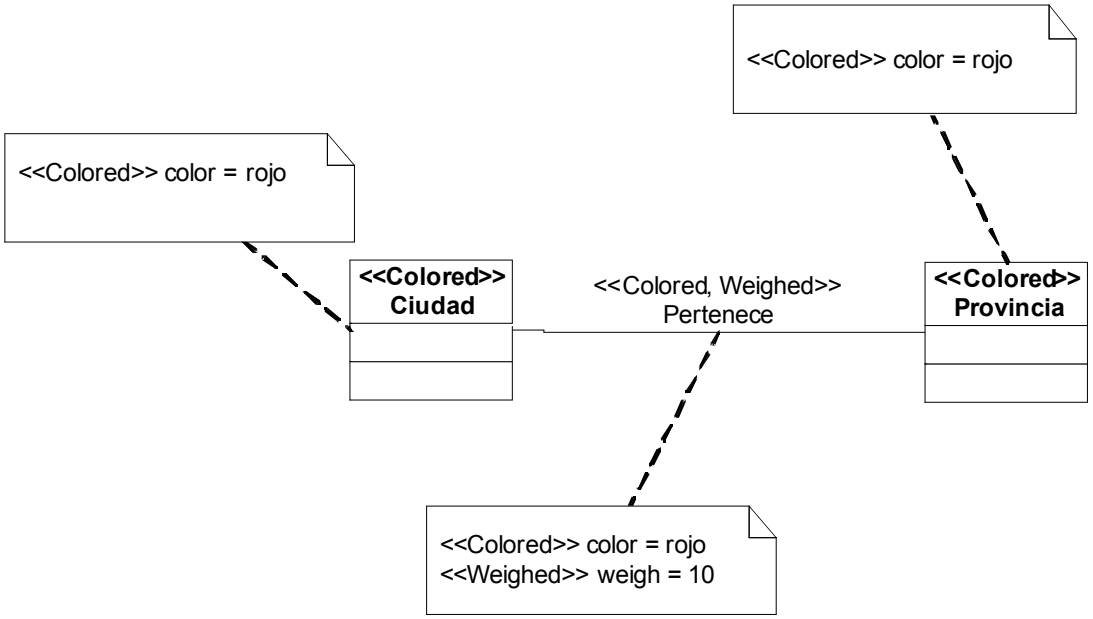


Perfiles UML

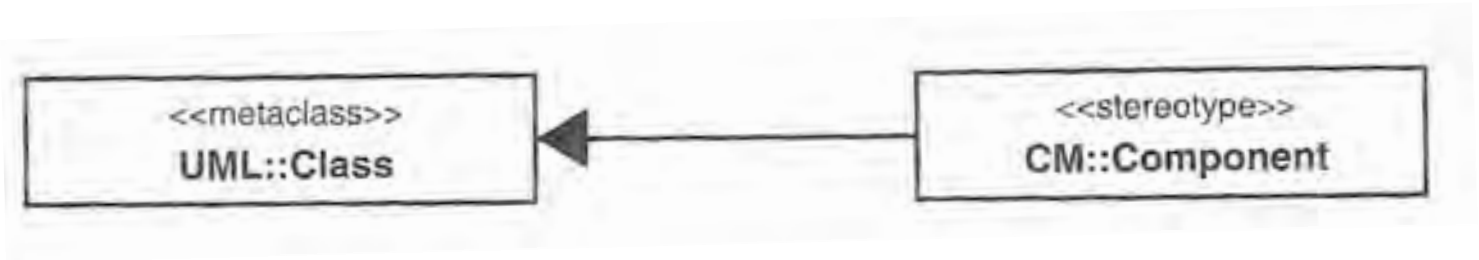
- Estereotipos
  - **<<Colored>>** Proporcionan color a un elemento UML. Solo las clases y las asociaciones pueden colorearse
    - **Definición Etiquetada**: **color**, de tipo Color (tipo Enumerado definido en el perfil). Indica el color de cada clase o asociación que haya sido etiquetada como Colored
  - **<<Weighed>>** Proporcionan peso a un elemento UML. Solo las asociaciones pueden tener asociado un peso
    - **Definición Etiquetada**: **weigh**, de tipo integer e indica el peso de cada asociación que haya sido estereotipada como Weighted
- Restricción sobre la Asociación
  - *"Si dos o más clases están unidas por una asociación coloreada, el color de las clases debe coincidir con el de la asociación"*

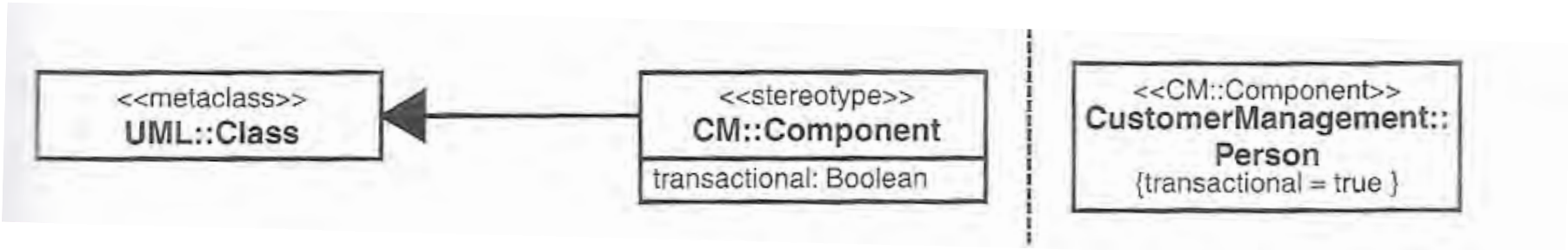
Perfiles UML

- Instanciación del Perfil *WeighsAndColors*



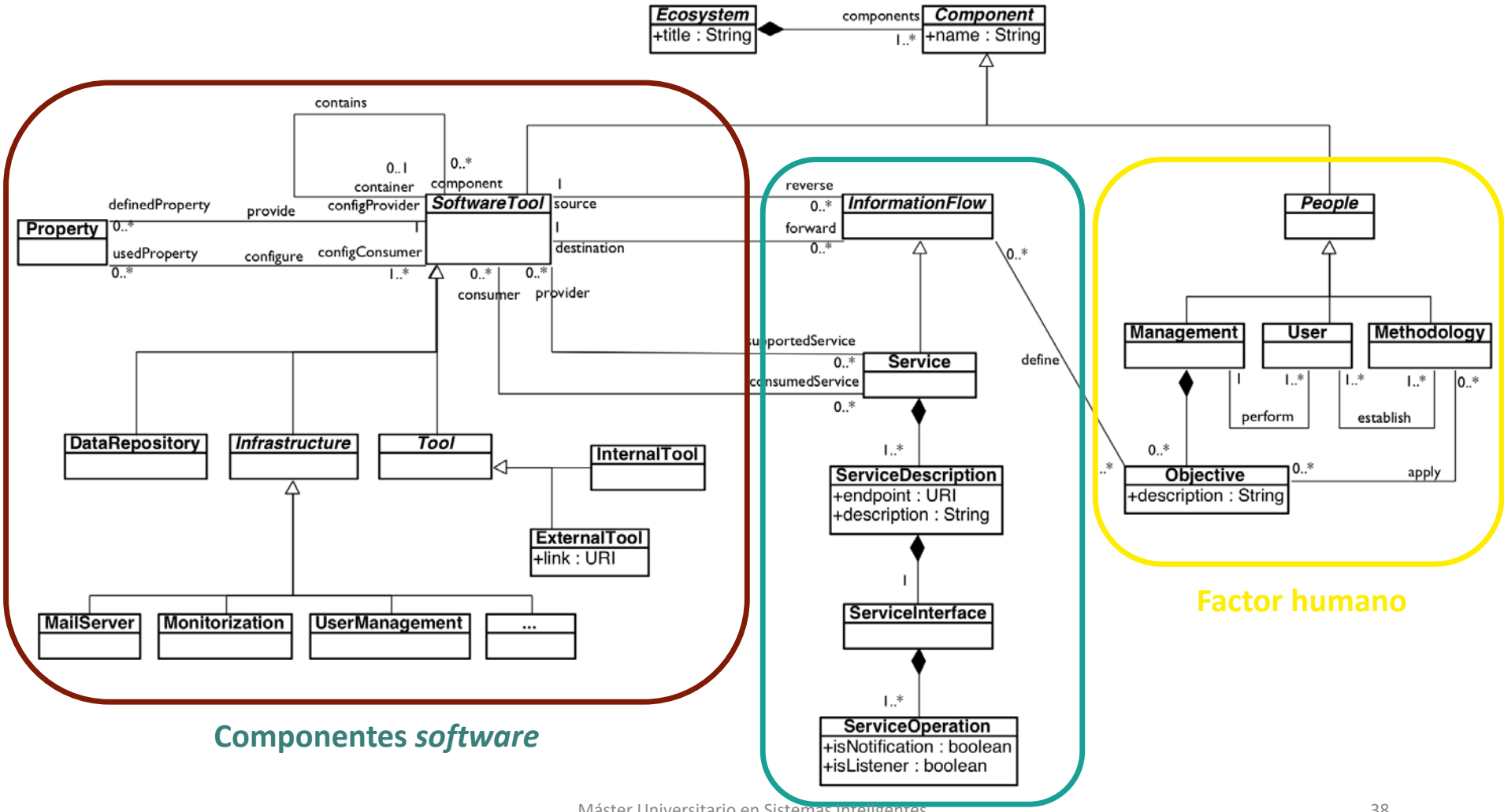
Perfiles UML 2.0



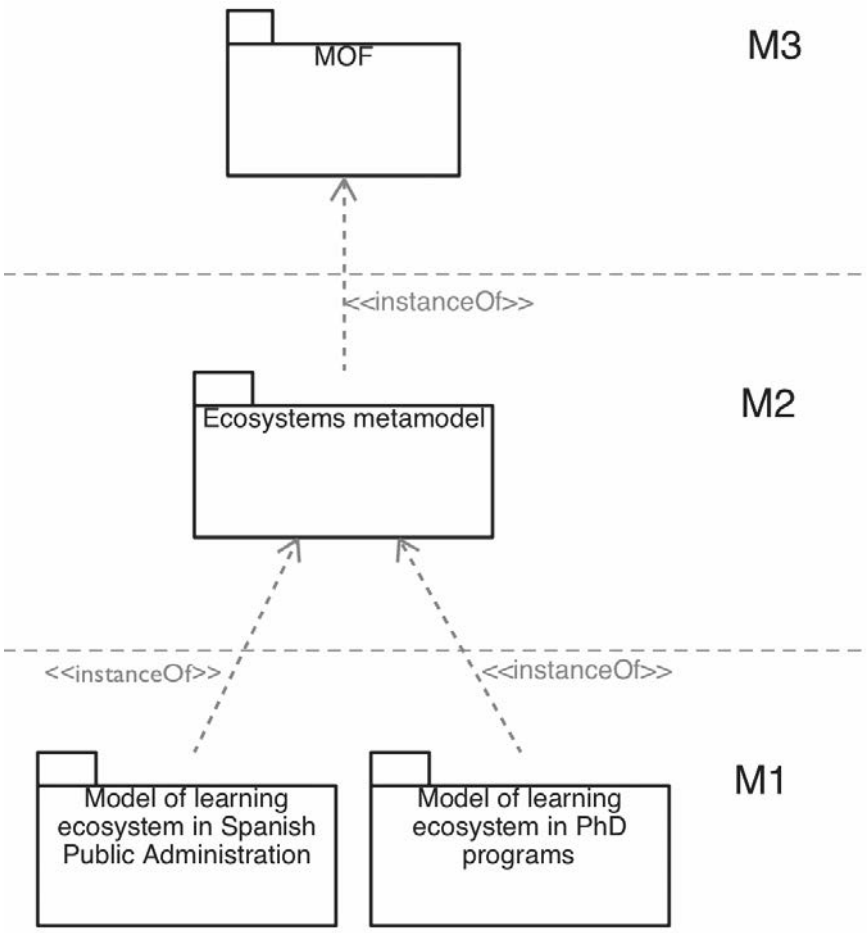


Ejemplo de metamodelo de un ecosistema tecnológico

(García-Holgado, 2018; García-Holgado & García-Peñalvo, 2016, 2017a, 2017b, 2018a, 2018b, 2018c, 2019)



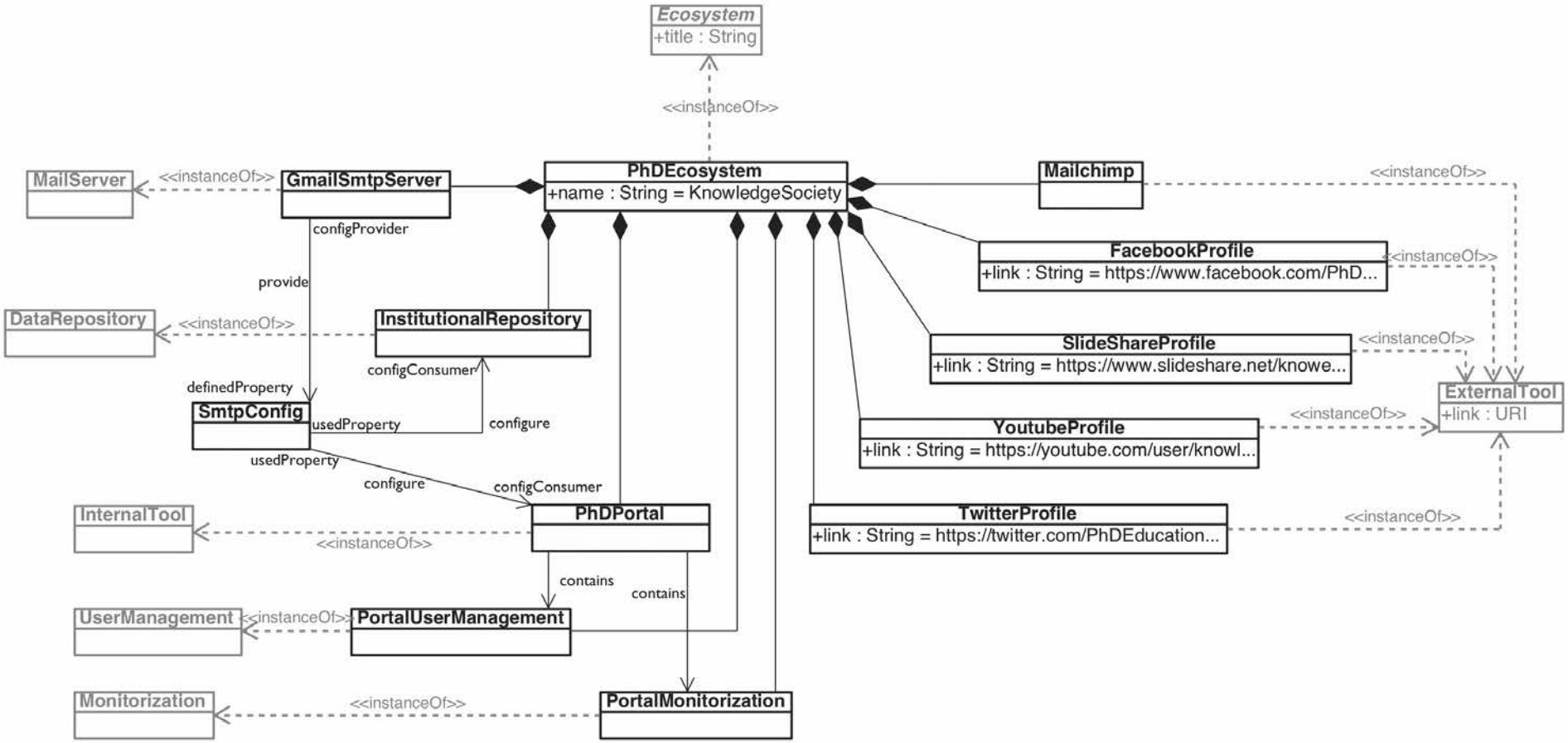
Ejemplo de metamodelo de un ecosistema tecnológico



- El metamodelo de ecosistema de aprendizaje se ha probado en dos casos de estudio con el objetivo de comprobar que permite definir modelos de ecosistemas de aprendizaje reales
- Se han tomado dos de los ecosistemas de aprendizaje utilizados para validar el patrón arquitectónico y se ha definido su correspondiente modelo a partir del metamodelo

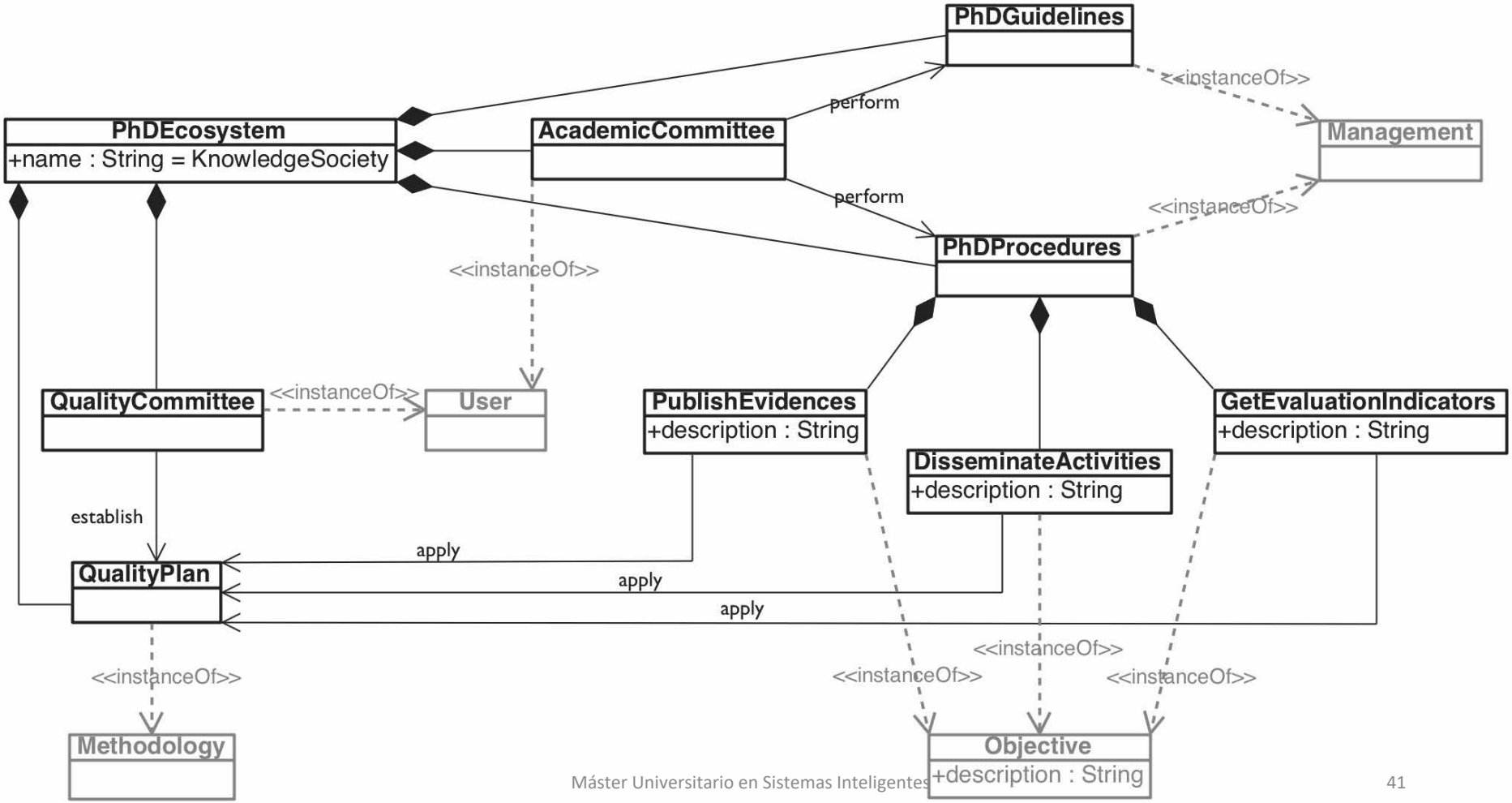
Ejemplo de metamodelo de un ecosistema tecnológico

Ecosistema para la gestión del conocimiento en un Programa de Doctorado: Vista componentes *software*



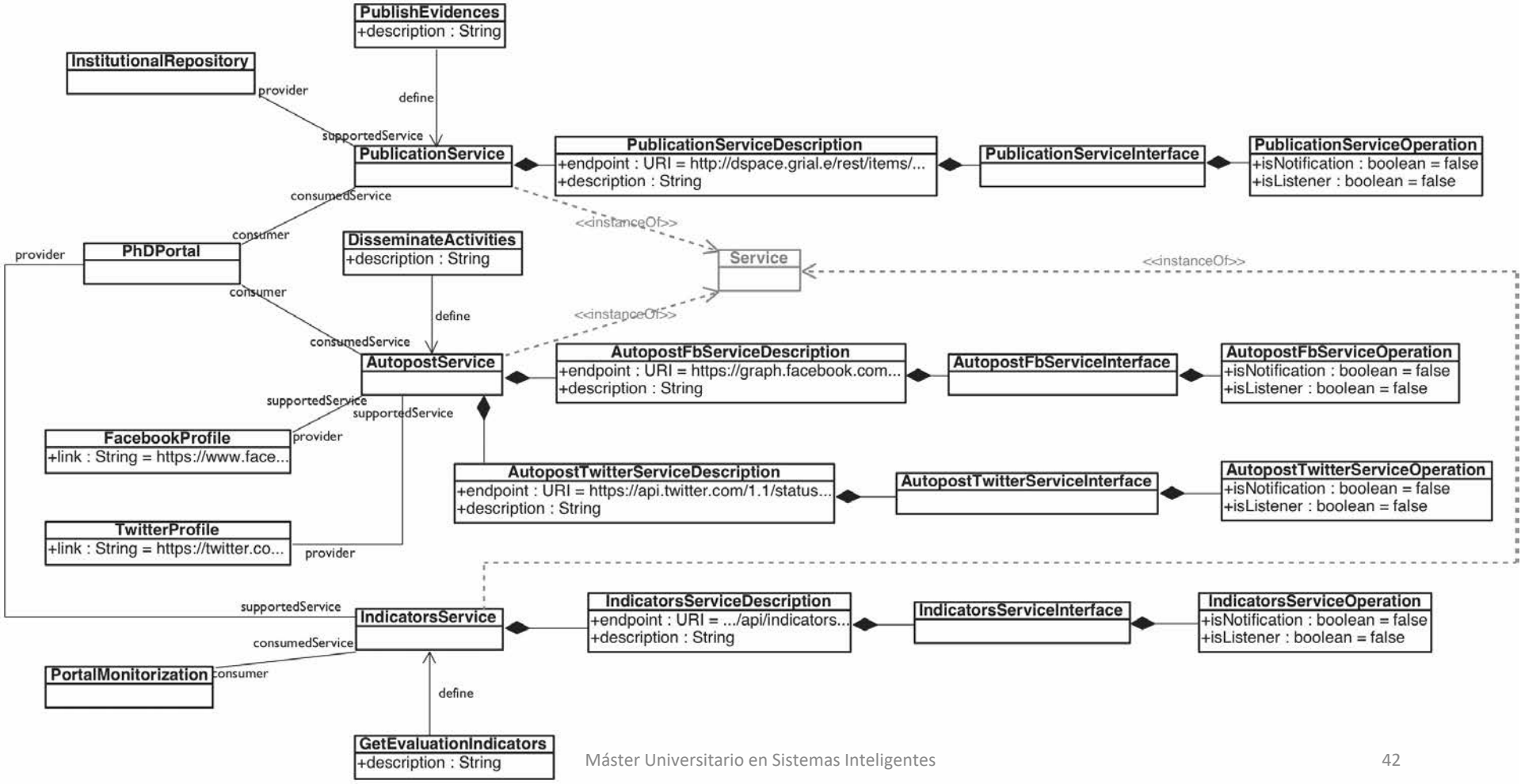
Ejemplo de metamodelo de un ecosistema tecnológico

Ecosistema para la gestión del conocimiento en un Programa de Doctorado: Vista factor humano



Ejemplo de metamodelo de un ecosistema tecnológico

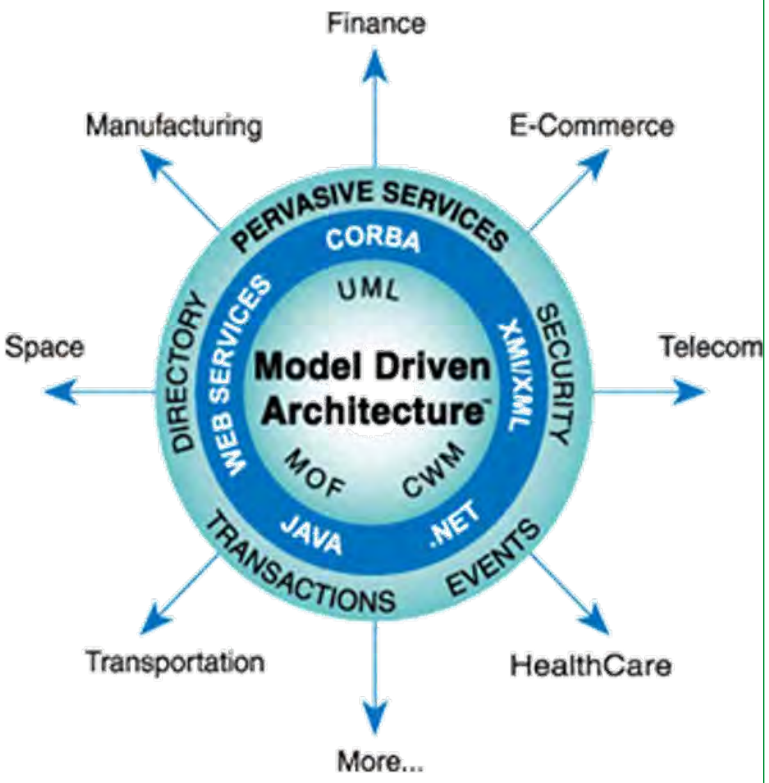
Ecosistema para la gestión del conocimiento en un Programa de Doctorado: Vista relaciones entre componentes



3. Model Driven Architecture (MDA)

Model Driven Architecture

- MDA es una iniciativa de la OMG
  - Anunciada en el 2000
  - 10 años de plazo para madurar
  - Debe durar al menos 20 años
- Extiende OMA
  - Las plataformas *middleware* pasan a un segundo plano
  - La clave son los modelos
- MDA aboga por la separación de la especificación de la funcionalidad de un sistema, independiente de su implementación en cualquier plataforma tecnológica concreta
- http://www.omg.org/mda



Model Driven Architecture

- Propuesta de OMG (IBM, Borland, Hewlett-Packard, Boeing, etc.)
- *MDA is an approach to use models in software development*
- Modelos = recurso principal para el desarrollo de *software*

MDA: Más allá de la tecnología

- **Ahora mismo hay demasiadas plataformas y tecnologías**
  - Objetos distribuidos, componentes, servicios web, etc.
  - Realmente no son interoperables entre sí
  - ¿Cuál es la mejor tecnología para usar hoy?
- **Además, la evolución es demasiado rápida**
  - Las tecnologías evolucionan y se quedan obsoletas muy pronto
  - Se conoce la tecnología de hoy pero, ¿cuál va a salir mañana?
  - ¿Y cuánto durará?
  - ¿Cómo protejo mi inversión en TI frente a estos cambios?
- **Sería bueno separar los modelos y lógica de negocio de la tecnología concreta en la que están implementados a día de hoy**
  - De forma que puedan evolucionar de forma independiente
  - Sin tener que tirarlo todo y comenzar de nuevo cada vez
  - Y de forma que pueda proteger la inversión en cada uno por separado

Ventajas (esperadas) de MDA

- **Protege la inversión ante los continuos cambios en las tecnologías**
  - Conserva los PIM (*Platform-Independent Model*) de una empresa (su modelo de negocio) cuando aparece nuevo *middleware*
- **Permite abordar mejor sistemas más complejos**
  - Mediante la separación de diferentes aspectos en diferentes modelos
- **Permite la simulación y la implementación automática de los modelos**
- **Permite la integración de sistemas existentes (COTS, *legacy systems*)**
  - ADM: *Architecture Driven Modernization*
- **Permite la especificación de los requisitos del sistema independientemente de las plataformas de implementación**
  - MBA: *Model-Based Adquisition*

Conceptos básicos

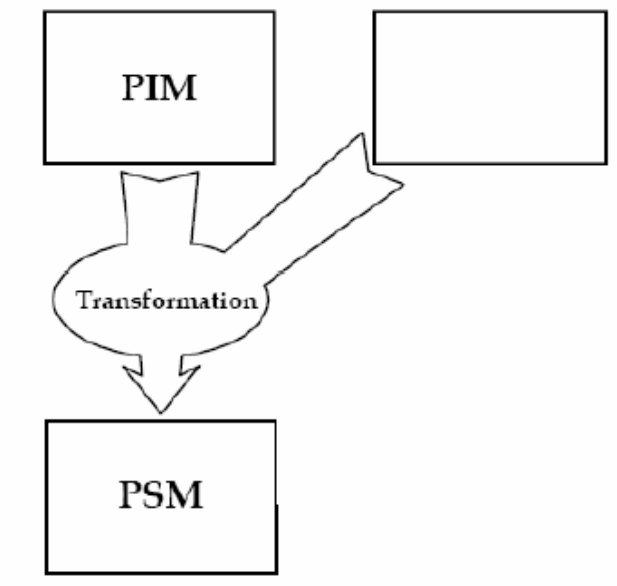
- ***Computational Independent Model (CIM)***
  - *A model of the system and its environment, that describes the system requirements but hides the details of its structure and internals*
- ***Platform Independent Model (PIM)***
  - *A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it*
- ***Platform Specific Model (PSM)***
  - *A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform*
- ***Platform***
  - *A set of subsystems/technologies that provide functionality through interfaces and specified usage patterns. It is specified so that any subsystem can use it without being aware of how the functionality provided by the platform is implemented*

Ejemplos de Modelos MDA

- **CIM**
  - Modelos de casos de uso que capturan los requisitos del sistema
- **PIM**
  - La descripción de la arquitectura *software* del sistema, que especifica cómo se descompone la funcionalidad básica del sistema en términos de componentes (arquitectónicos) y conectores
- **PSM**
  - Un modelo de la implementación J2EE del sistema, expresado usando el perfil UML para EJB para representar cómo los componentes (arquitectónicos) del sistema se han de implementar como EJBs
- **Código**
  - Los propios componentes EJBs, sus ficheros de configuración, y toda la información necesaria para realizar el despliegue en las máquinas concretas

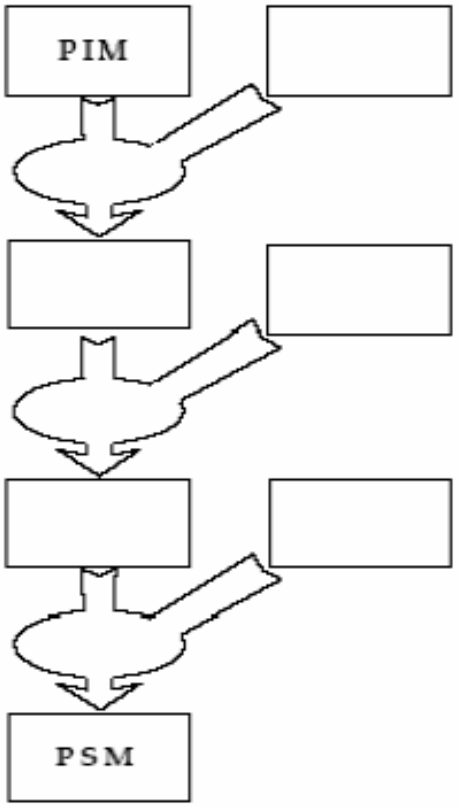
Transformaciones de modelos

- **Una transformación de modelos especifica el proceso de conversión de un modelo a otro**
- **El patrón MDA incluye (al menos)**
  - Un PIM
  - Un modelo de Plataforma
  - Una transformación
  - Un PSM

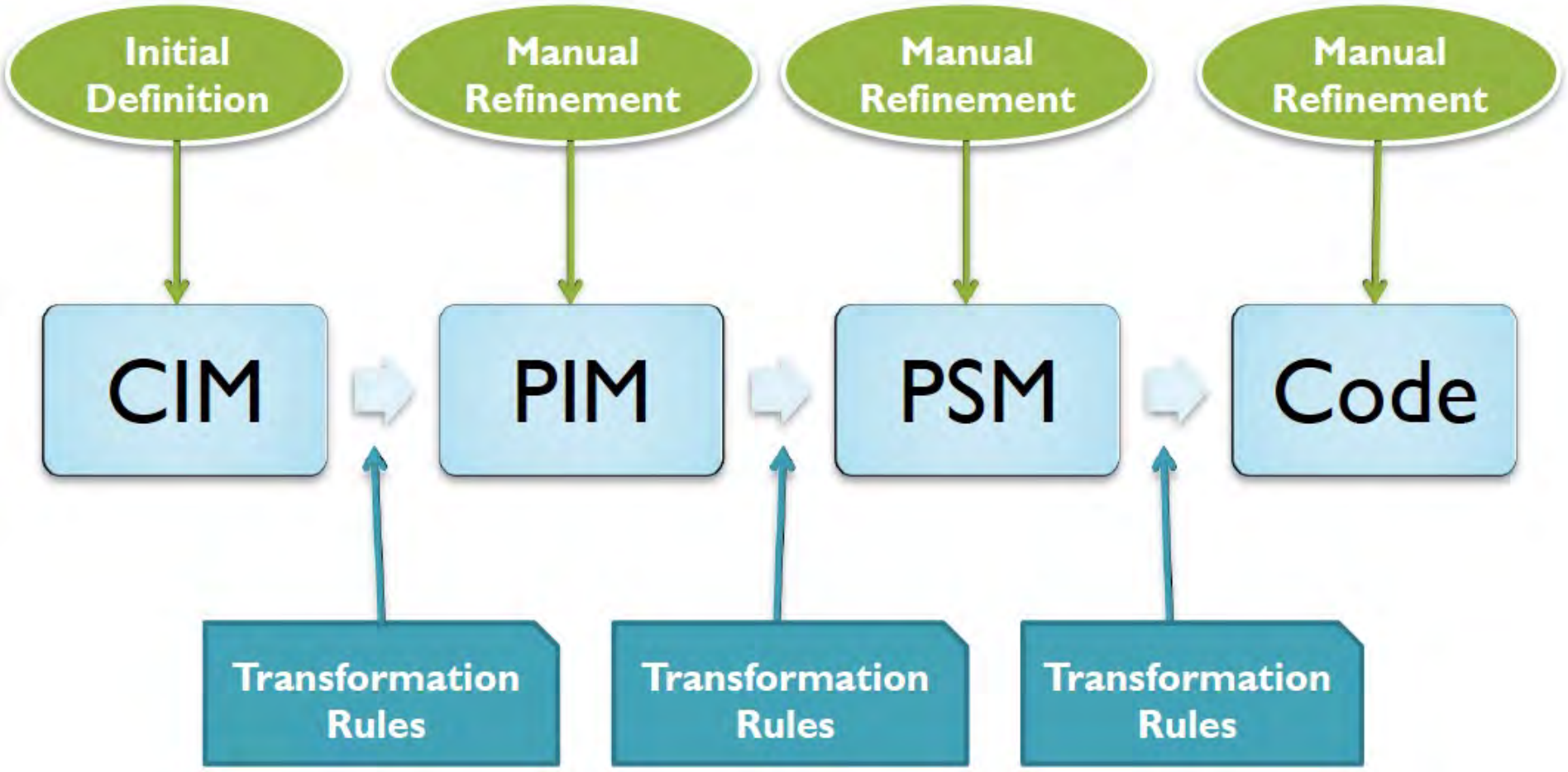


Aplicaciones sucesivas

- **El patrón MDA se utiliza normalmente sucesivas veces para producir una sucesión de cambios**
  - El PSM resultante de una transformación se convierte en el PIM de la siguiente
  - De esta forma, cada plataforma se concentra en un aspecto diferente del sistema y se aplica ordenadamente
  - Este proceso es modular y ordenado

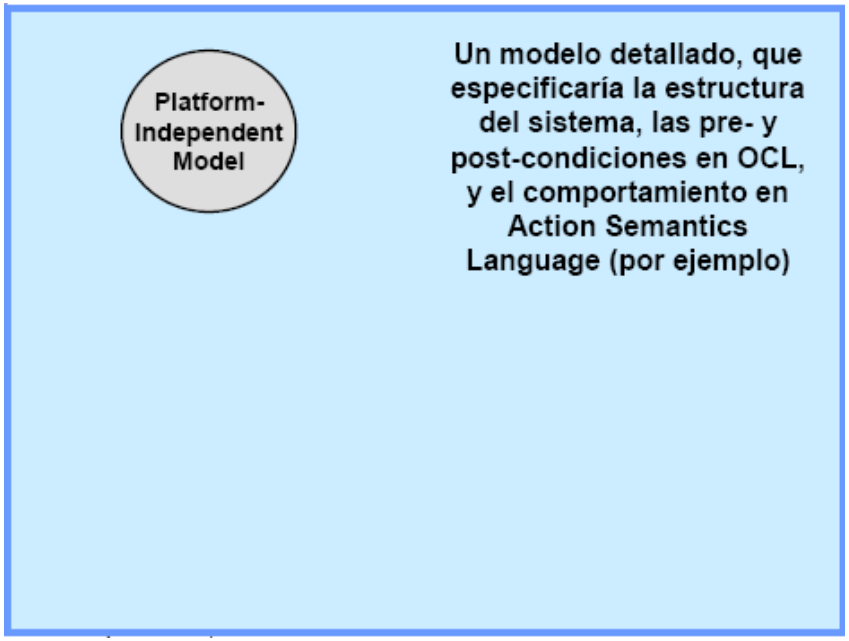


Proceso de desarrollo MDA



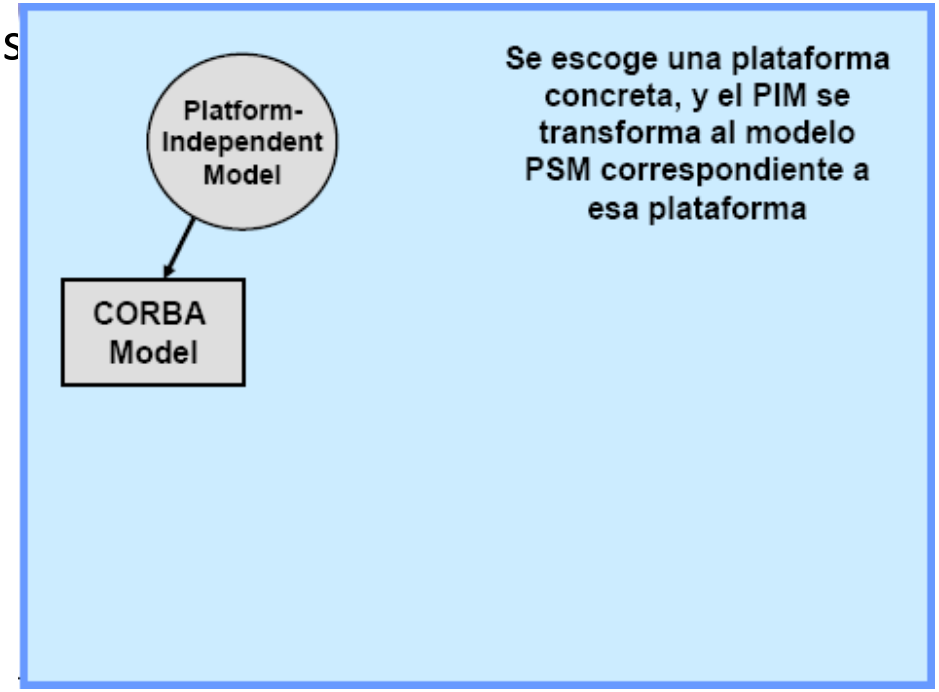
Cómo se construye una aplicación usando MDA

- Se comienza con el *Platform-Independent Model* (PIM) que representa la lógica del negocio y su funcionalidad, independiente de los detalles de la implementación

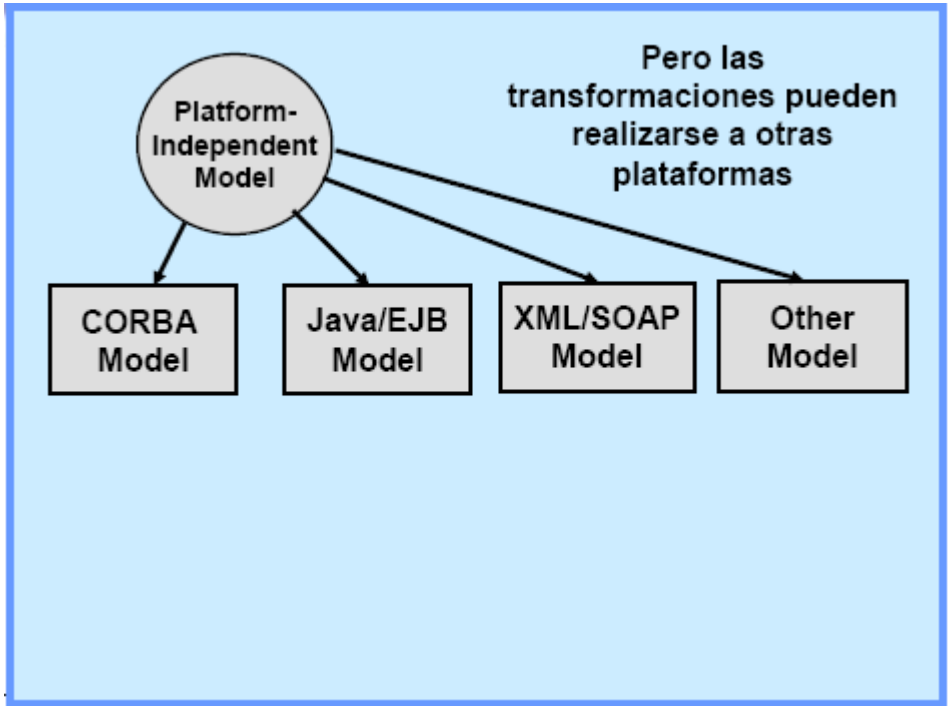


Se genera el PSM

- Las transformaciones pueden ser definidas con QVT (*Query/View/Transformation*), entre los metamodelos origen y destino
- Las transformaciones pueden ser parcial o completamente automatizadas

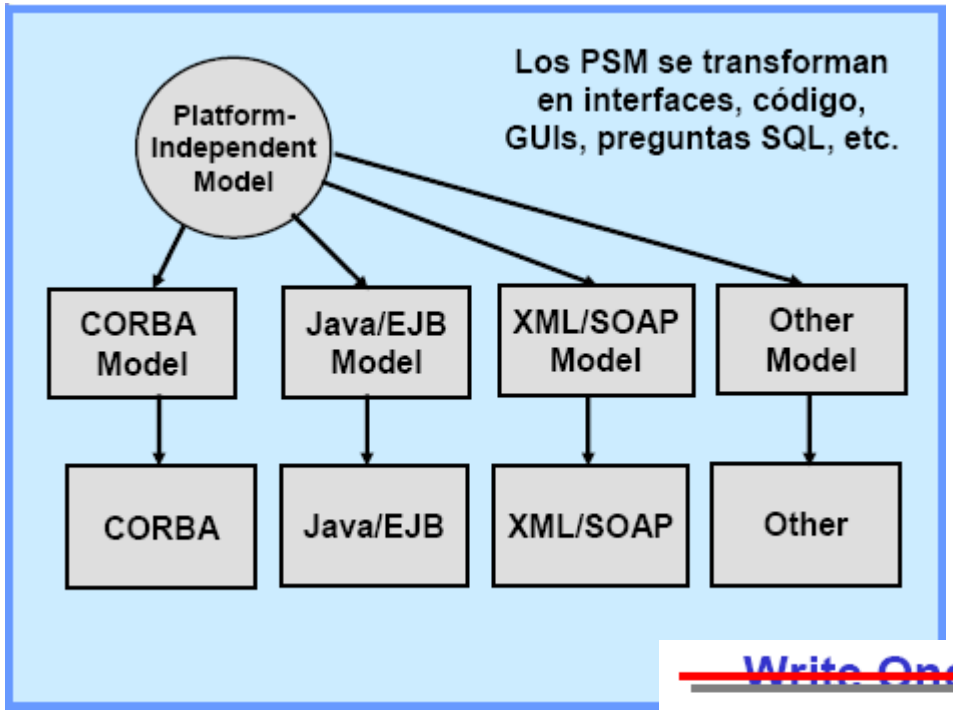


Generación a múltiples tecnologías



Generación de implementaciones

- Es fácil contar con implementadores automáticos a partir de modelos específicos, pues son de bajo nivel

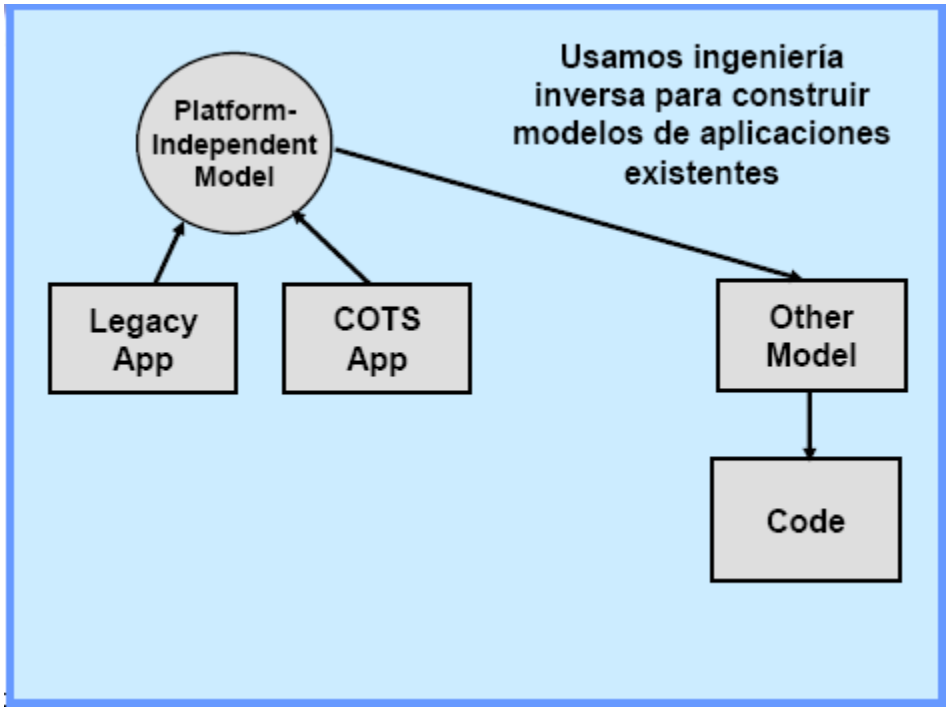


~~Write Once, Run Everywhere~~

Model Once, Generate Everywhere!

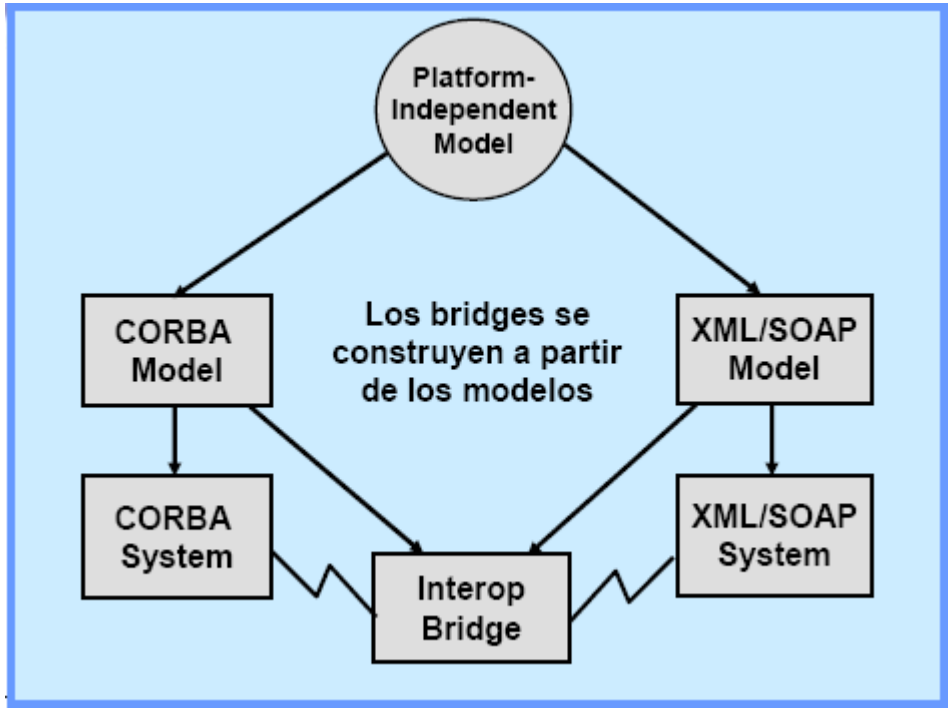
ADM e integración de sistemas

- Muy útil para
  - **(1) Integración** en nuestra aplicación de COTS, sistemas de terceros y sistemas heredados
  - **(2) *Architecture Driven Modernization***: modernización de sistemas actuales



Generación de puentes

- Los puentes pueden generarse de forma automática en la mayoría de los casos, tanto dentro de la propia empresa, como para lograr interoperabilidad entre sistemas de diferentes compañías



Estándares

- **MOF (meta-modelado)**
- **UML (modelado de *software*)**
- **OCL (restricciones)**
- **Otros modelos MOF, como por ejemplo CWM: *Common Warehouse Metamodel***
- **XMI (serialización en XML)**
- **QVT (transformaciones)**
- **Perfiles UML**
- **Plataformas como CORBA**

Ventajas

- **Cada modelo es independiente del resto**
  - Se definen de forma separada
  - Cada modelo define sus propias "entidades", reside en un nivel de abstracción adecuado, y se expresa en un lenguaje apropiado para el tipo de *stakeholders* interesados en ese tipo de modelo
- **El proceso de desarrollo *software* se convierte en transformación de modelos**
  - Cada paso selecciona una plataforma y transforma uno o más PIM del sistema en uno (o más) PSM del mismo hasta que se llegue a la implementación final del sistema
  - Las transformaciones pueden automatizarse
- **Se gana modularidad, flexibilidad y facilidad de evolución**
- **Los modelos de la aplicación que capturan la lógica del negocio y la propiedad intelectual se convierten en los principales activos de la empresa y son independientes de la(s) tecnología(s) en las que serán implementados**

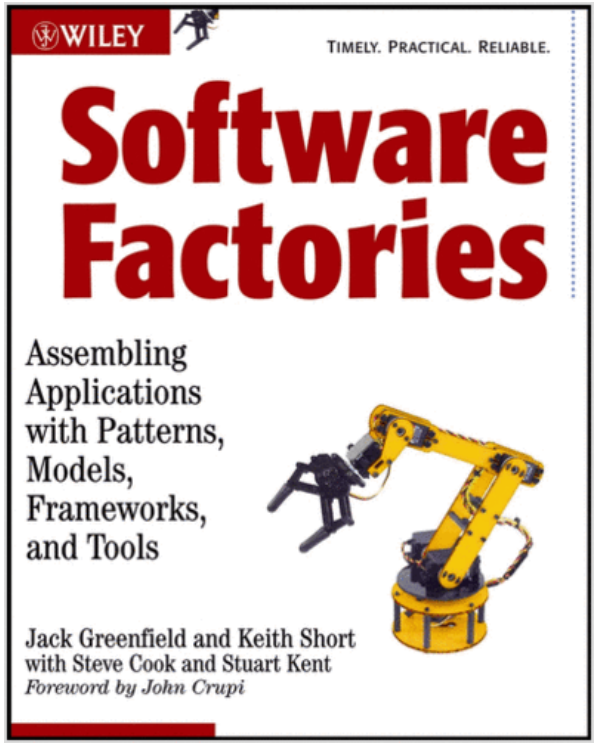
MDA vs. MDD

- Diferenciar MDD y MDA
  - ***Model Driven Development*** (MDD): Aproximación al desarrollo de *software* basado en el modelado del sistema *software* y su generación a partir de los modelos
  - ***Model Driven Architecture*** (MDA): MDD + lenguajes de OMG para el modelado (UML, CWM, MOF, QVT)
  - MDA = Infraestructura para MDD

4. Software Factories

Fábricas de software

*"Es una **línea de productos** software que configura **herramientas** extensibles, procesos y contenido [...] para automatizar el desarrollo y mantenimiento de variantes de un **producto arquetípico** mediante la adaptación, ensamblaje y configuración de componentes basados en **frameworks**"*



"a Software Factory is a development environment configured to support the rapid development of a specific type of application" Jack Greenfield

Fábricas de software

- Objetivo
  - Incrementar el nivel de automatización en la producción de *software*
- ¿No se está haciendo?
- ¿Por qué falla?
  - Problemas crónicos en el desarrollo de *software*

Fábricas de Software

Problemas crónicos en el desarrollo de software

- Construcción monolítica
- Generalidad gratuita
- Desarrollo único (sin reutilización)
- Inmadurez del proceso

- ¿Cómo se puede dar un paso adelante?
  - ✓ Reutilización sistemática
  - ✓ Desarrollo dirigido por modelos
  - ✓ Desarrollo mediante ensamblado
  - ✓ *Frameworks* de proceso

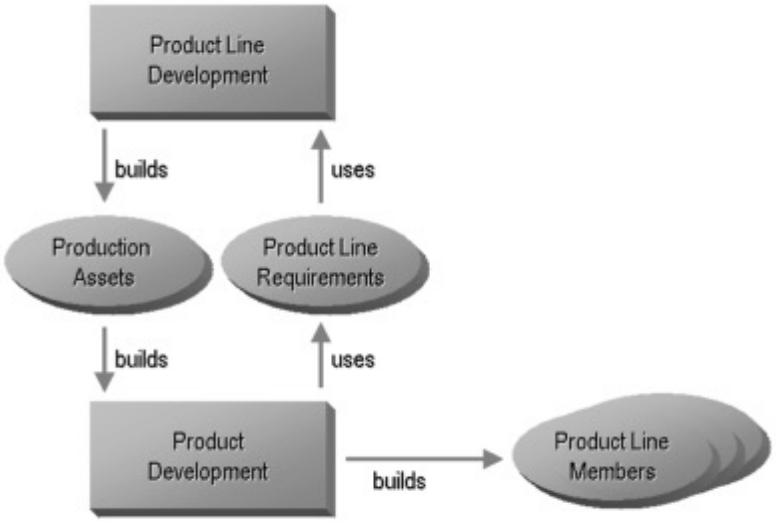
Reutilización sistemática

What is a Software Product Line?

A software product line (SPL) is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way

Carnegie Mellon
Software Engineering Institute

Una **SPL** produce una ***Familia de Productos Software*** en un dominio específico



Reutilización sistemática

- Características de las **familias de productos *software***
  - Proporcionan un contexto en el que los miembros comparten muchas características comunes que pueden desarrollarse de forma colectiva, facilitando la reutilización sistemática
  - Pueden consistir en componentes o productos completos
  - Creadas por
    - **Integradores de Sistemas**, cuando migran una aplicación de un cliente a otro, o construyen nuevas aplicaciones modificando algunas existentes
    - **Vendedores de *Software* Independientes**, cuando desarrollan múltiples aplicaciones en un dominio específico, o múltiples versiones de una aplicación
    - **Organizaciones TI**, cuando personalizan aplicaciones, desarrollan múltiples aplicaciones relacionadas, o múltiples versiones de una aplicación

Desarrollo dirigido por modelos

- El objetivo del MDD es ***elevar el nivel de abstracción*** (*mediante herramientas*) para proporcionar ***altos*** niveles de ***automatización*** (*en dominios específicos*)
- MDD **utiliza modelos** para capturar información de **alto nivel** y **automatizar su implementación** compilando modelos para ***producir ejecutables*** o usándolos para facilitar/guiar el desarrollo manual
- MDD puede incluso facilitar la ***automatización*** de actividades como la ***depuración*** y la ***configuración*** automática del *software*

Desarrollo dirigido por modelos

Lenguajes específicos del dominio (DSL/DSM)

- Desafortunadamente los ***modelos*** se han utilizado principalmente ***como documentación*** para consumo humano y no para ser procesables por máquinas
- Las **Fábricas de *Software*** están interesadas en modelos que puedan ser procesados por herramientas y proponen usarlos de la misma forma en la que se usa el código fuente. Estos modelos deben ser ***precisos***
- Para elevar el nivel de abstracción, un lenguaje de modelado debe estar ***orientado a dominios más reducidos*** que aquellos que permite modelar un lenguaje de propósito general

Desarrollo dirigido por modelos

Lenguajes específicos del dominio (DSL/DSM)

- El **propósito** para el que el lenguaje se ha diseñado debe **explicitarse**, de forma que un observador familiar con el dominio pueda evaluar el lenguaje
- El lenguaje debe usar **términos familiares** a la gente que trabaja con el **dominio**
- La **notación** del lenguaje, sea gráfica o textual, debe ser **fácil de usar**
- El lenguaje debe **tener una gramática** que gobierne la forma en la que se pueden combinar los conceptos **para construir expresiones**
- El significado de cada expresión bien formada debe estar bien definido, de forma que los usuarios puedan **construir modelos** que otros usuarios puedan **entender** y que las herramientas puedan **generar implementaciones** válidas a partir de los modelos
- Un lenguaje que cumple estos criterios se llama ***Domain-Specific Language* (DSL),** porque modela conceptos de un dominio específico. Un **DSL** se define con **mayor rigor** que un lenguaje de propósito general

MDA vs. Fábricas de Software

MDA	***Fábricas de Software***
(Ok) Proporciona técnicas	(X) No concreta
(X) Solo estrategia (PIM->PSM)	(Ok) Más guías (DSL, Frameworks, etc.)
(Ok) Más tiempo	(X) Más nueva (2004)
(Ok) Más herramientas (Arcstyler, OptimalJ, EMF, etc.)	(X) Herramientas no maduras, poco conocidas (DSL, MetaEdit+)
(_) Promovido por OMG	(_) Promovido por Microsoft

MDA vs. Fábricas de Software

- Integración ¿Por qué no?
  - Desarrollo de una línea de producto
  - *Framework* de implementación
  - Construcción de un DSL
    - Primitivas conceptuales independientes de plataforma (PIM)
    - MOF o profile UML
  - Generación de código mediante transformaciones “DSL to Framework”

MDA vs. Fábricas de Software

- Son enfoques compatibles
  - Pero desaconsejado por
    - OMG: No al “Babel de Lenguajes”
    - Microsoft: Estándares de OMG imprecisos
- Elección según circunstancias
- El éxito final puede depender factores comerciales

5. Caso de estudio. Dashboards personalizados

Introducción

- Incremento del volumen de datos generados
- Necesidad de analizar los datos para sacarles partido
- **Visión:** obtener un proceso de toma de decisiones estratégicas arropado por una consistente base informativa
- Necesidad de herramientas que faciliten la comprensión de los resultados obtenidos del análisis de datos por parte de todos los perfiles involucrados

(Vázquez-Ingelmo, 2022; Vázquez-Ingelmo et al., 2017, 2018a, 2018b, 2018c, 2018d)

Introducción

Dashboards

- Constituyen uno de los productos *software* más empleados para la explotación y extracción de conocimiento de conjuntos de datos
- Recursos gráficos interactivos
- Permiten la identificación de patrones y relaciones en los conjuntos de datos de manera visual

Introducción

Dashboards

- Su diseño y desarrollo no es trivial
- La evolución e incremento del volumen de datos llama a la necesidad de que los *dashboards* sean flexibles y puedan adaptarse a cambios
- La diversidad de requisitos de cada perfil de usuario puede hacer el mantenimiento de los *dashboards* muy complejo

Introducción

- Ciertos paradigmas de la ingeniería del software ofrecen soluciones viables para abordar la variedad de requisitos de forma flexible
- Concretamente, el enfoque de las familias software (SPL) constituye una buena estrategia para la generación de *dashboards* personalizados dada una serie de requisitos
- Decremento de tiempo de desarrollo
- Aumento de mantenibilidad, escalabilidad, flexibilidad y personalización

Contexto

Observatorio de Empleabilidad y Empleo Universitarios

- Organización de investigadores y técnicos con el objetivo de producir, analizar y difundir información sobre la empleabilidad y empleo de titulados/as universitarios en España
- Visión de convertirse en una referencia para entender las variables que afectan a la empleabilidad y al empleo

https://oeeu.org/

(Michavila et al., 2015, 2016, 2018a, 2018b)

Contexto

Observatorio de Empleabilidad y Empleo Universitarios

Observatorio de Empleabilidad
y Empleo Universitarios

- Gran cantidad de datos recolectados
- Diversos usuarios con diversas necesidades (administradores, usuarios generales, universidades, etc.)
- Flexibilidad necesaria de cara a nuevas ediciones de sus estudios

https://oeeu.org/

(Michavila et al., 2015, 2016, 2018a, 2018b)

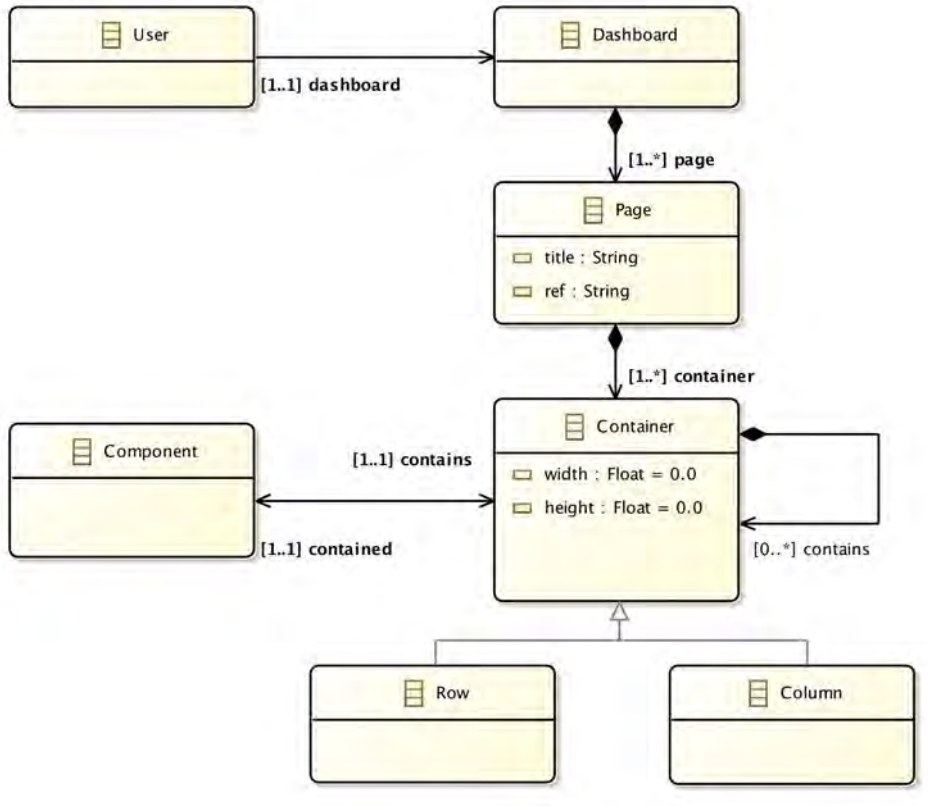
Metodología

- Meta-modelado del dominio
- Aproximación de SPL con generación automática de código
- Tres niveles de personalización
  - Funcionalidad
  - Fuentes de datos
  - Diseño

Metodología

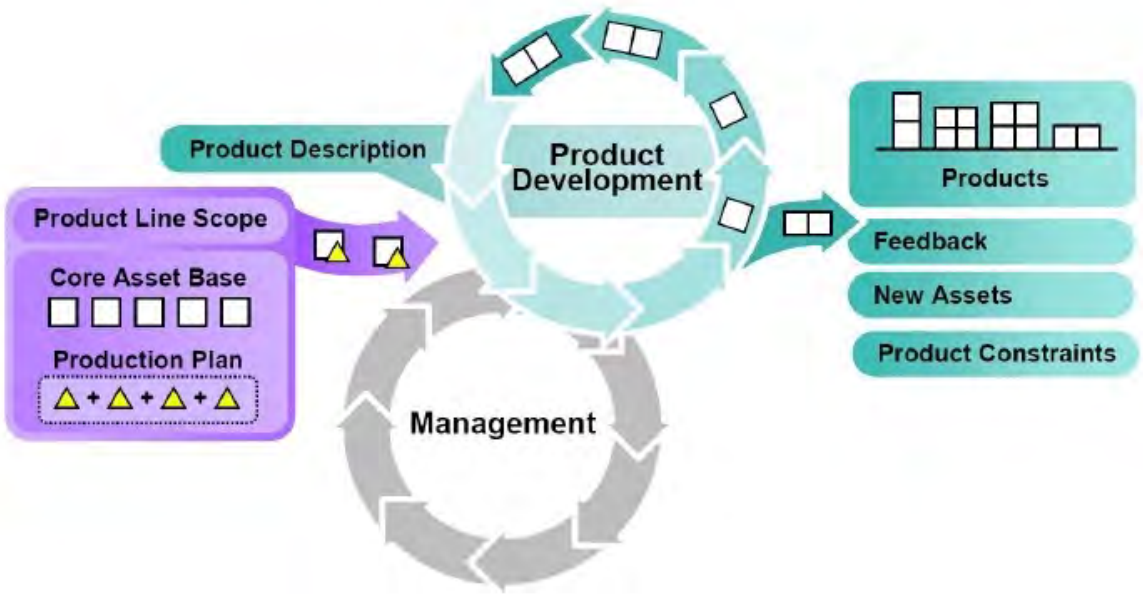
Meta-modelado

Especificación a alto nivel el dominio del problema



Metodología

Líneas de productos software

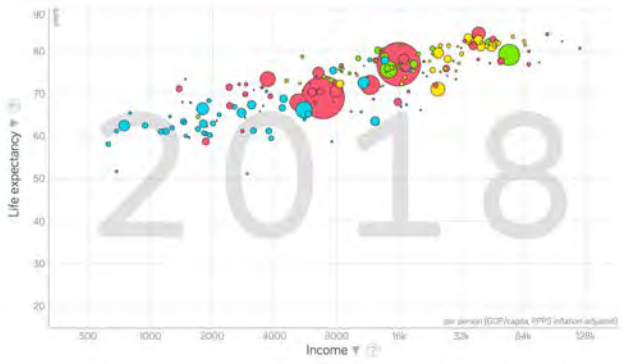


https://www.slideshare.net/pagsousa/software-product-lines

Propuesta

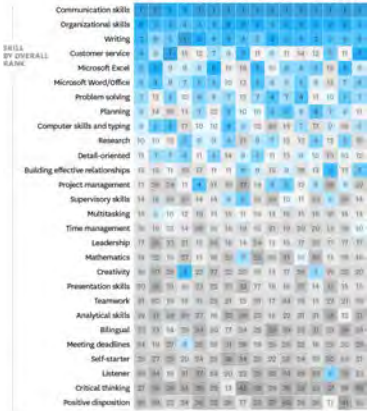
Core assets **de la propuesta desarrollada**

Diagrama de dispersión



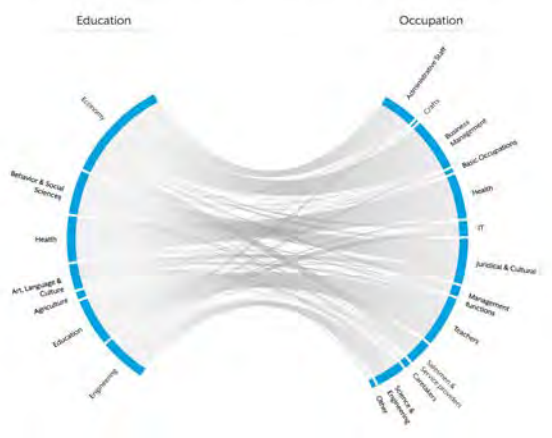
Exploración de la relación de hasta tres variables numéricas simultáneamente categorizadas por variables categóricas

Mapa de calor



Exploración de conjuntos de variables numéricas de forma simultánea

Diagrama de cuerdas

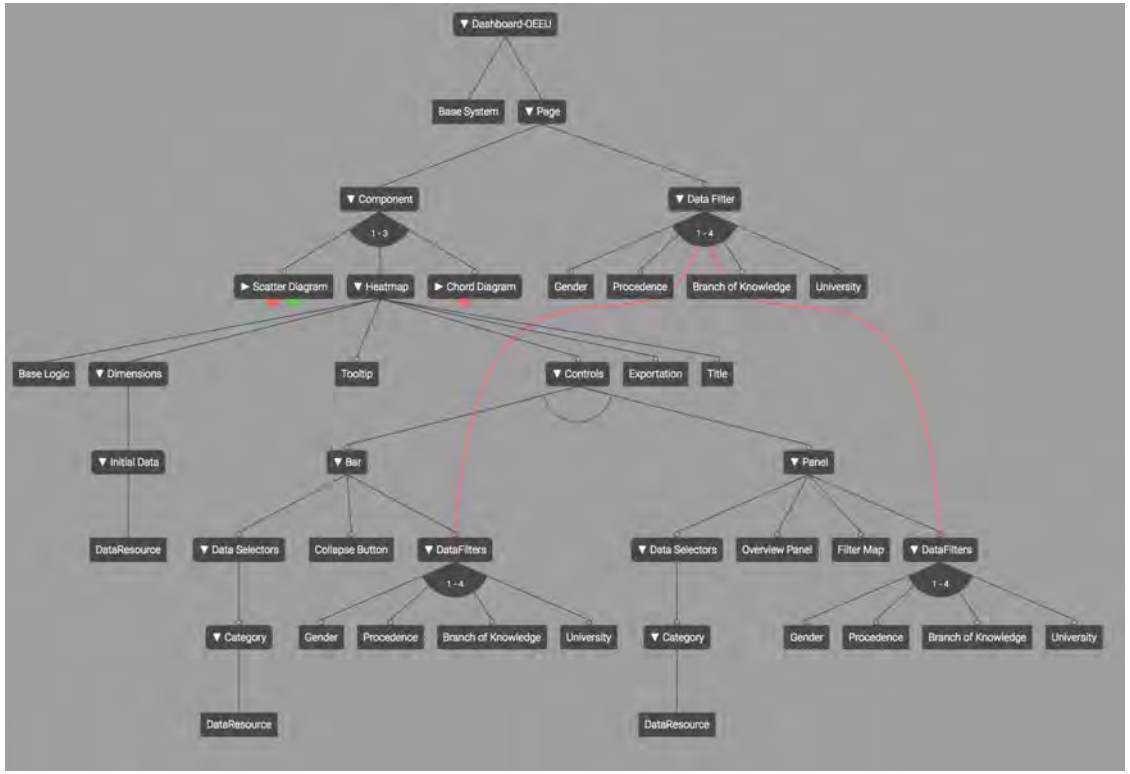


Exploración de relaciones entre variables categóricas

Propuesta

Diagrama de características

- Especificación de las características de cada componente de la línea y restricciones de la misma



Propuesta

Lenguaje de dominio específico (DSL)

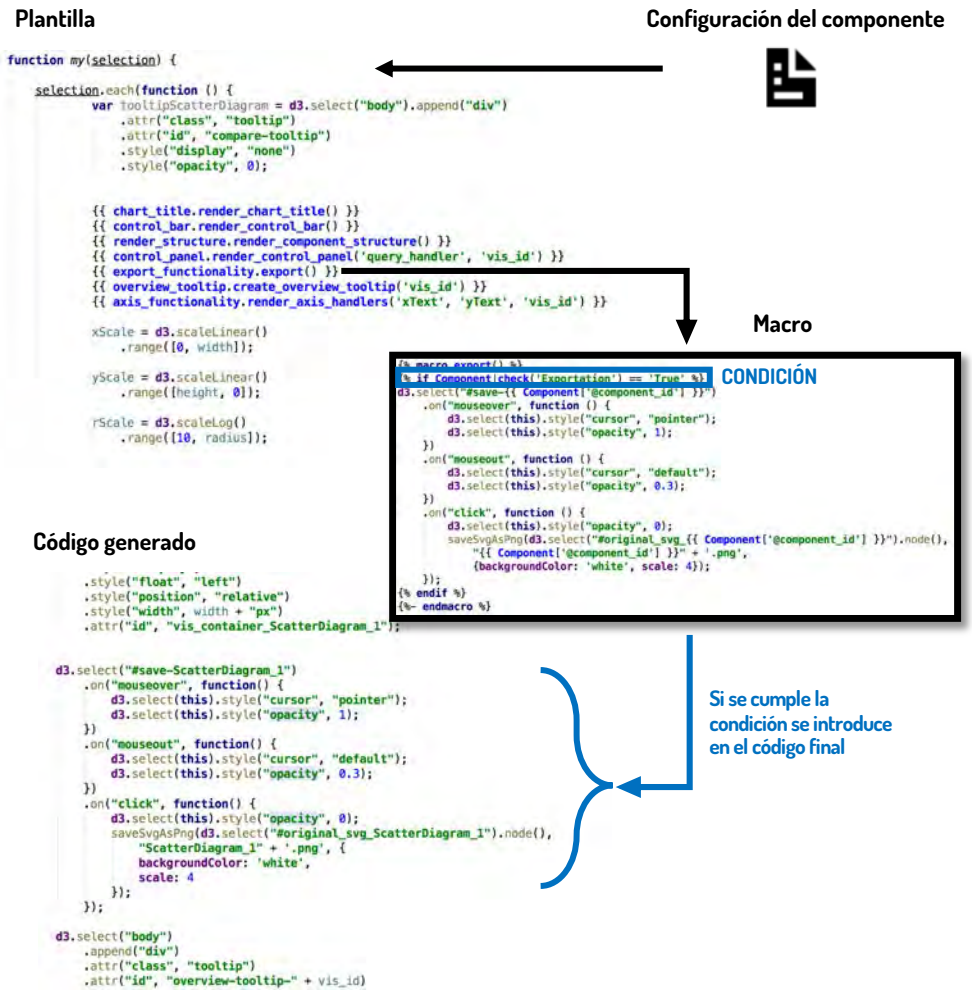
- Basado en el meta-modelo y diagrama de características
- Implementado mediante XML
- Validado mediante un esquema XML (XSD) y un script de Python para cumplir las restricciones de la SPL

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLS
    <xs:element name="Dashboard">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Title" type="xs:string" minOccurs="0" />
                <xs:element name="NavigationBar" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Brand" type="xs:string" />
                            <xs:choice>
                                <xs:element name="Link"...>
                                <xs:element name="LinkGroup"...>
                            </xs:choice>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="ScreensConfig"...>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" />
            <xs:attribute name="user" type="xs:int" />
        </xs:complexType>
    </xs:element>
    <xs:complexType name="LayoutType"...>
</xs:schema>
```

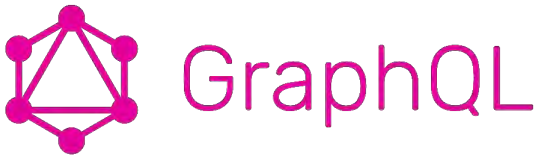
Propuesta

Generación de código

- Plantillas de código (Jinja2)
- Uso de macros y sentencias
- Inyección de funcionalidades mediante la configuración XML de cada usuario



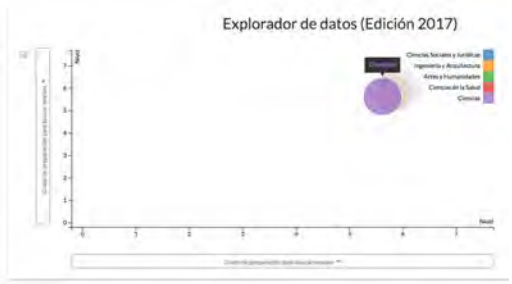
Propuesta

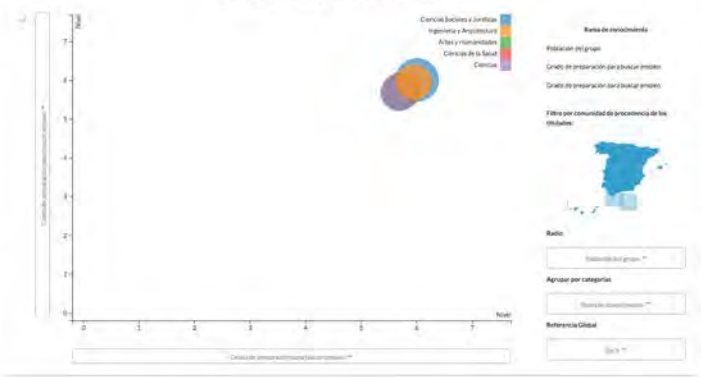


API GraphQL

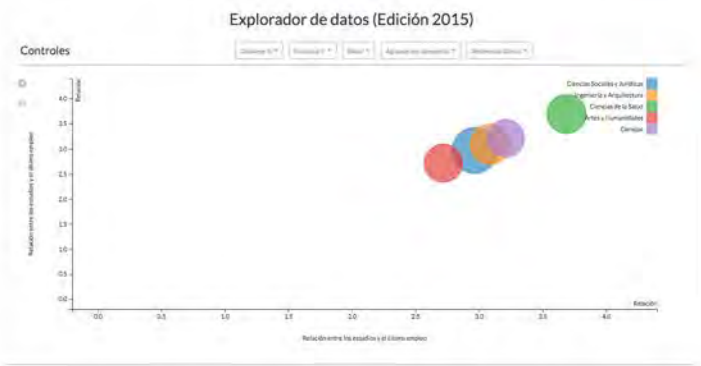
- Necesidad de desacoplar los componentes de presentación de las fuentes de datos
- Implementación de una API GraphQL
  - Lenguaje de consulta
  - Consultas parametrizables
  - Recolección de los campos explícitamente especificados en la consulta
  - Varias consultas en una misma llamada
- Cálculo de métricas a demanda del usuario

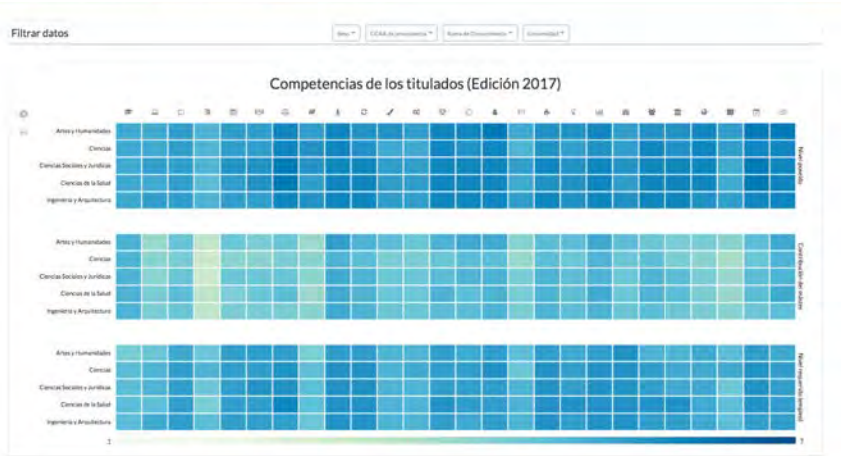
Resultados





https://oeeu-infovis.grial.eu/dashboard/





Discusión

- Primera aproximación para la consecución de un *framework* generativo flexible en el dominio de los *dashboards*
- Este enfoque permite centrarse y guiar el desarrollo en base a los requisitos de los usuarios
- El DSL permite un nexo entre la especificación formal de la SPL y la implementación final
- La generación de código por plantillas se ha constituido como una solución simple y efectiva para la inyección de características en el código

Discusión

Personalización a nivel funcional

- Nivel de personalización más simple
- Añadir funcionalidades en un componente solamente implica la inyección del código asociado a dicha funcionalidad
- Es posible tener componentes con diversas características funcionales: filtros, selectores de datos, información detallada, etc.

Discusión

Personalización a nivel de datos

- Requisitos de información variados: ciertos usuarios pueden considerar una serie de datos muy relevantes, mientras que otros pueden considerarlos irrelevantes
- La construcción de una API GraphQL ha conseguido dotar al Observatorio de un método para que los clientes puedan recuperar información de su banco de conocimiento.
- La fuente de datos utilizada es interna al Observatorio
- Introducir fuente externas de datos traería más dificultades, al no estar bajo el control de los desarrolladores y necesitar mucho más mantenimiento

Discusión

Personalización a nivel de diseño

- El nivel más complejo de modelar
- Dificultad de automatización del proceso
- Las aproximaciones propuestas en la literatura apuntan a la necesidad de introducir procesos manuales o semiautomáticos teniendo en cuenta los requisitos de diseño de cada usuario específico
- Principal reto en la generación de *dashboards*: han de ser usables para que la experiencia de usuario no se vea comprometida.

Conclusiones

- Enfoque beneficioso para la gestión de diferentes requisitos dinámicos
- Incremento de mantenibilidad y trazabilidad de requisitos
- Esfuerzo inicial en la construcción de *core assets*
- Propagación de cambios, actualizaciones y correcciones en todos los productos

Referencias

- García-Holgado, A. (2018). *Análisis de integración de soluciones basadas en software como servicio para la implantación de ecosistemas tecnológicos educativos.* (PhD), Universidad de Salamanca, Salamanca. Retrieved from https://goo.gl/LToHcq
- García-Holgado, A., & García-Peñalvo, F. J. (2016). Architectural pattern to improve the definition and implementation of eLearning ecosystems. *Science of Computer Programming, 129*, 20-34. doi:10.1016/j.scico.2016.03.010
- García-Holgado, A., & García-Peñalvo, F. J. (2017a). A metamodel proposal for developing learning ecosystems. In P. Zaphiris & A. Ioannou (Eds.), *Learning and Collaboration Technologies. Novel Learning Ecosystems. 4th International Conference, LCT 2017. Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9–14, 2017. Proceedings, Part I* (pp. 100-109). Switzerland: Springer International Publishing.
- García-Holgado, A., & García-Peñalvo, F. J. (2017b). Preliminary validation of the metamodel for developing learning ecosystems. In J. M. Dodero, M. S. Ibarra Sáiz, & I. Ruiz Rube (Eds.), *Fifth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'17) (Cádiz, Spain, October 18-20, 2017)* New York, NY, USA: ACM.
- García-Holgado, A., & García-Peñalvo, F. J. (2018a). Gestión del conocimiento abierto mediante ecosistemas tecnológicos basados en soluciones Open Source. In J. A. Merlo Vega (Ed.), *Ecosistemas del Conocimiento Abierto* (pp. 147-160). Salamanca, España: Ediciones Universidad de Salamanca.
- García-Holgado, A., & García-Peñalvo, F. J. (2018b). Human interaction in learning ecosystems based on open source solutions. In P. Zaphiris & A. Ioannou (Eds.), *Learning and Collaboration Technologies. Design, Development and Technological Innovation. 5th International Conference, LCT 2018, Held as Part of HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings, Part I* (pp. 218-232). Cham, Switzerland: Springer.
- García-Holgado, A., & García-Peñalvo, F. J. (2018c). Learning ecosystem metamodel quality assurance. In Á. Rocha, H. Adeli, L. P. Reis, & S. Costanzo (Eds.), *Trends and Advances in Information Systems and Technologies* (Vol. 1, pp. 787-796). Cham: Springer.
- García-Holgado, A., & García-Peñalvo, F. J. (2019). Validation of the learning ecosystem metamodel using transformation rules. *Future Generation Computer Systems, 91*, 300-310. doi:10.1016/j.future.2018.09.011
- Greenfield, J., & Short, K. (2004). *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Indianapolis, IN, USA: Wiley Publishing Inc.

Referencias

- Kleppe, A., Warmer, J., & Bast, W. (2003). *MDA Explained: The Model Driven Architecture: Practice and Promise*. Boston, USA: Addison-Wesley.
- Michavila, F., Martín-González, M., Martínez, J. M., García-Peñalvo, F. J., & Cruz-Benito, J. (2015). Analyzing the employability and employment factors of graduate students in Spain: The OEEU Information System. In G. R. Alves & M. C. Felgueiras (Eds.), *Proceedings of the Third International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'15) (Porto, Portugal, October 7-9, 2015)* (pp. 277-283). New York, USA: ACM.
- Michavila, F., Martínez, J. M., Martín-González, M., García-Peñalvo, F. J., & Cruz Benito, J. (2018). Empleabilidad de los titulados universitarios en España. Proyecto OEEU. *Education in the Knowledge Society, 19*(1), 21-39. doi:10.14201/eks20181912139
- Michavila, F., Martínez, J. M., Martín-González, M., García-Peñalvo, F. J., & Cruz-Benito, J. (2016). *Barómetro de empleabilidad y empleo de los universitarios en España, 2015 (Primer informe de resultados)*. Madrid: Observatorio de Empleabilidad y Empleo Universitarios.
- Michavila, F., Martínez, J. M., Martín-González, M., García-Peñalvo, F. J., Cruz-Benito, J., & Vázquez-Ingelmo, A. (2018). *Barómetro de empleabilidad y empleo universitarios. Edición Máster 2017*. Madrid, España: Observatorio de Empleabilidad y Empleo Universitarios.
- Miller, J., & Mukerji, J. (2001). *Model Driven Architecture (MDA)* (ormsc/01-07-01). Needham, MA, USA: Object Management Group. Retrieved from https://goo.gl/CRLSnH
- Object Management Group. (2014). *Model Driven Architecture (MDA). MDA Guide rev. 2.0* (ormsc/2014-06-01). Needham, MA, USA: Object Management Group. Retrieved from https://goo.gl/wXwCTo
- Object Management Group. (2017). *Unified Modeling Language specification version 2.5.1* (formal/17-12-05). Needham, MA, USA: Object Management Group. Retrieved from https://goo.gl/kaE82a
- Seidewitz, E. (2003). What models mean. *IEEE Software, 20*(5), 26-32. doi:10.1109/MS.2003.1231147
- Vázquez-Ingelmo, A. (2022). *Automatic generation of software interfaces for supporting decision-making processes. An application of domain engineering & machine learning.* Universidad de Salamanca]. Salamanca. https://bit.ly/3UpojXj
- Vázquez-Ingelmo, A., Cruz-Benito, J., & García-Peñalvo, F. J. (2017). Improving the OEEU's data-driven technological ecosystem's interoperability with GraphQL. In J. M. Dodero, M. S. Ibarra Sáiz, & I. Ruiz Rube (Eds.), *Fifth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'17) (Cádiz, Spain, October 18-20, 2017)* New York, NY, USA: ACM.
- Vázquez-Ingelmo, A., Cruz-Benito, J., García-Peñalvo, F. J., & Martín-González, M. (2018). Scaffolding the OEEU's Data-Driven Ecosystem to Analyze the Employability of Spanish Graduates. In F. J. García-Peñalvo (Ed.), *Global Implications of Emerging Technology Trends* (pp. 236-255). Hershey PA, USA: IGI Global.

Referencias

- Vázquez-Ingelmo, A., García-Peñalvo, F. J., & Therón, R. (2018a). Application of domain engineering to generate customized information dashboards *Learning and Collaboration Technologies. Design, Development and Technological Innovation. 5th International Conference, LCT 2018, Held as Part of HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings, Part II* (pp. 518-529). Cham, Switzerland: Springer.
- Vázquez-Ingelmo, A., García-Peñalvo, F. J., & Therón, R. (2018b). Domain engineering for generating dashboards to analyze employment and employability in the academic context. In F. J. García-Peñalvo (Ed.), *TEEM'18 Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality (Salamanca, Spain, October 24th-26th, 2018)* (pp. 896-901). New York, NY, USA: ACM.
- Vázquez-Ingelmo, A., García-Peñalvo, F. J., & Therón, R. (2018c). Generation of Customized Dashboards Through Software Product Line Paradigms to Analyse University Employment and Employability Data. In M. Á. Conde, C. Fernández-Llamas, Á. M. Guerrero-Higueras, F. J. Rodríguez-Sedano, Á. Hernández-García, & F. J. García-Peñalvo (Eds.), *Proceedings of the Learning Analytics Summer Institute Spain 2018 – LASI-SPAIN 2018, (León, Spain, June 18-19, 2018)* (pp. 113-124). Aachen, Germany: CEUR-WS.org.

Ingeniería de Software Dirigida por Modelos

Procesos y Métodos de Modelado para la Ingeniería Web y Web Semántica

Máster Universitario en Sistemas Inteligentes

Curso 2023-2024

4-15 de marzo de 2024

Dr. Francisco José García Peñalvo
GRupo de investigación en InterAcción y eLearning (GRIAL)
Universidad de Salamanca

fgarcia@usal.es
http://grial.usal.es
http://twitter.com/frangp