TESTING BASADOS EN MODELOS: ESPECIFICACIÓN GRÁFICA Y DERIVACIÓN DE CÓDIGO FUENTE

Anselmo Abadía – Juan Barisich

Temario

- Introducción
- Conceptos básicos
- Marco tecnológico
- Solución propuesta
- Diseño de la solución
- Caso de estudio
- Conclusiones

Temario

- **Introducción**
- Conceptos básicos
- Marco tecnológico
- Solución propuesta
- Diseño de la solución
- Caso de estudio
- Conclusiones

Motivación

- El software día a día va incrementado su complejidad, concurrencia y dinamismo
- Son necesarios métodos para asegurar los niveles de calidad del software
- Los mecanismos de testing actuales ayudan a esta tarea, pero en general:
  - son poco practicados
  - no están guiados por procesos
  - son administrados por los desarrolladores:
    - implementados manualmente
    - propensos a errores funcionales

Motivación

- Se necesita mejorar la brecha existente entre los desarrolladores y el dominio del problema utilizando una metodología estándar para expresar casos de prueba en una etapa más temprana del ciclo de vida del software
- Se necesita mejorar la productividad del desarrollo de casos de prueba intentando que no sea una tarea desechada cuando los tiempos de desarrollo son críticos
- Los desarrollos guiados por modelos cumplen con esta función pero son escazas las herramientas para la especificación de casos de pruebas

Objetivos

- Realizar una herramienta que permita:
  - **definir modelos de manera gráfica** que reflejen los casos de prueba, permitiendo refinar los aspectos de calidad sobre los requerimientos del sistema en una etapa más temprana
  - **derivar código fuente** a partir de estos modelos y colaborar con el desarrollador en la construcción de los casos de pruebas

Temario

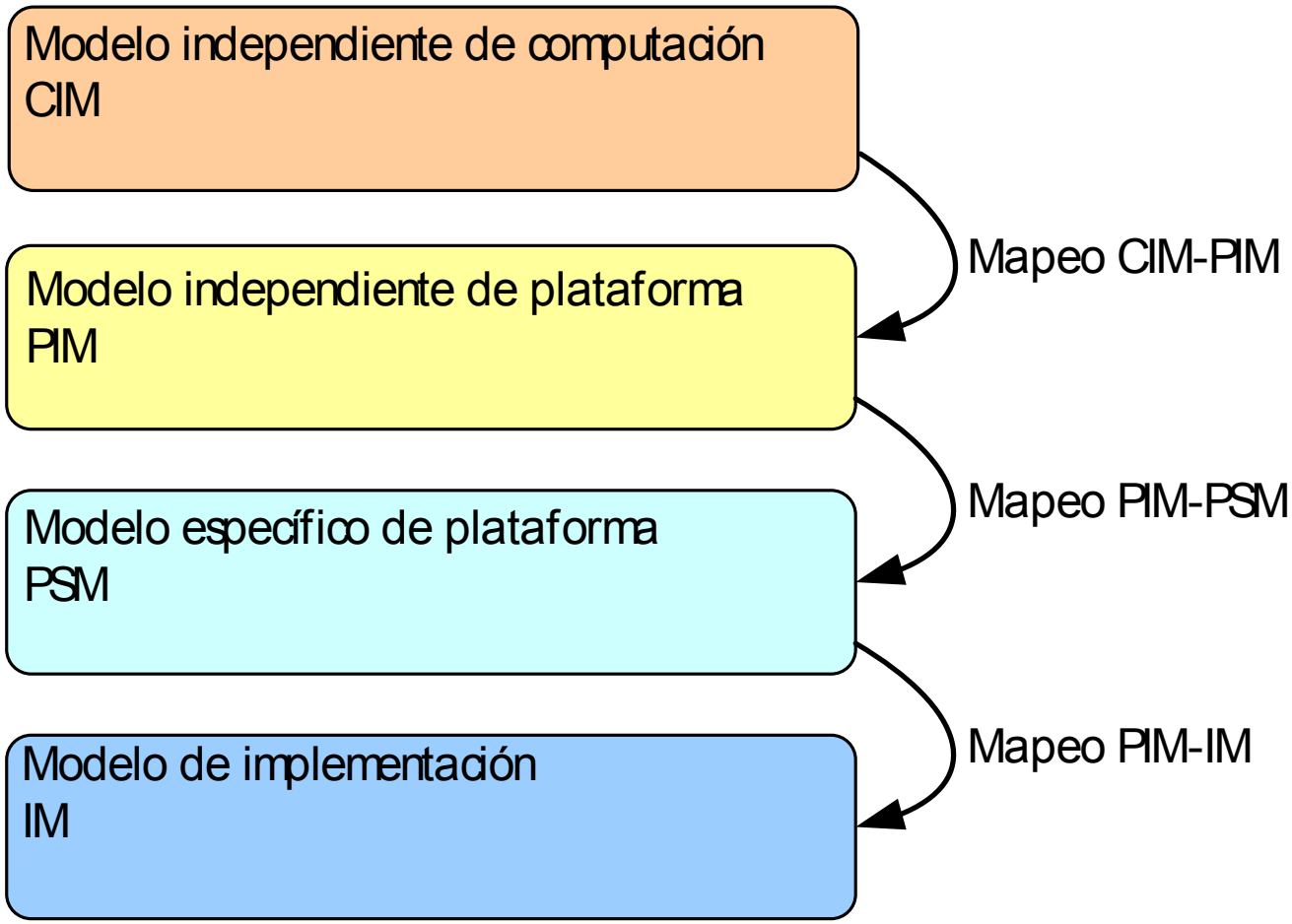
- Introducción
- **Conceptos básicos**
- Marco tecnológico
- Solución propuesta
- Diseño de la solución
- Caso de estudio
- Conclusiones

MDE

- Es un paradigma para la construcción de software
- Los modelos, orientados al dominio, constituyen el foco principal en el desarrollo
- El código, orientado a la plataforma, es generado automáticamente aplicando transformaciones sobre el modelo
- Permite reducir el tiempo y los costos del desarrollo del Software

MDE

- Contempla cuatro tipos de modelos

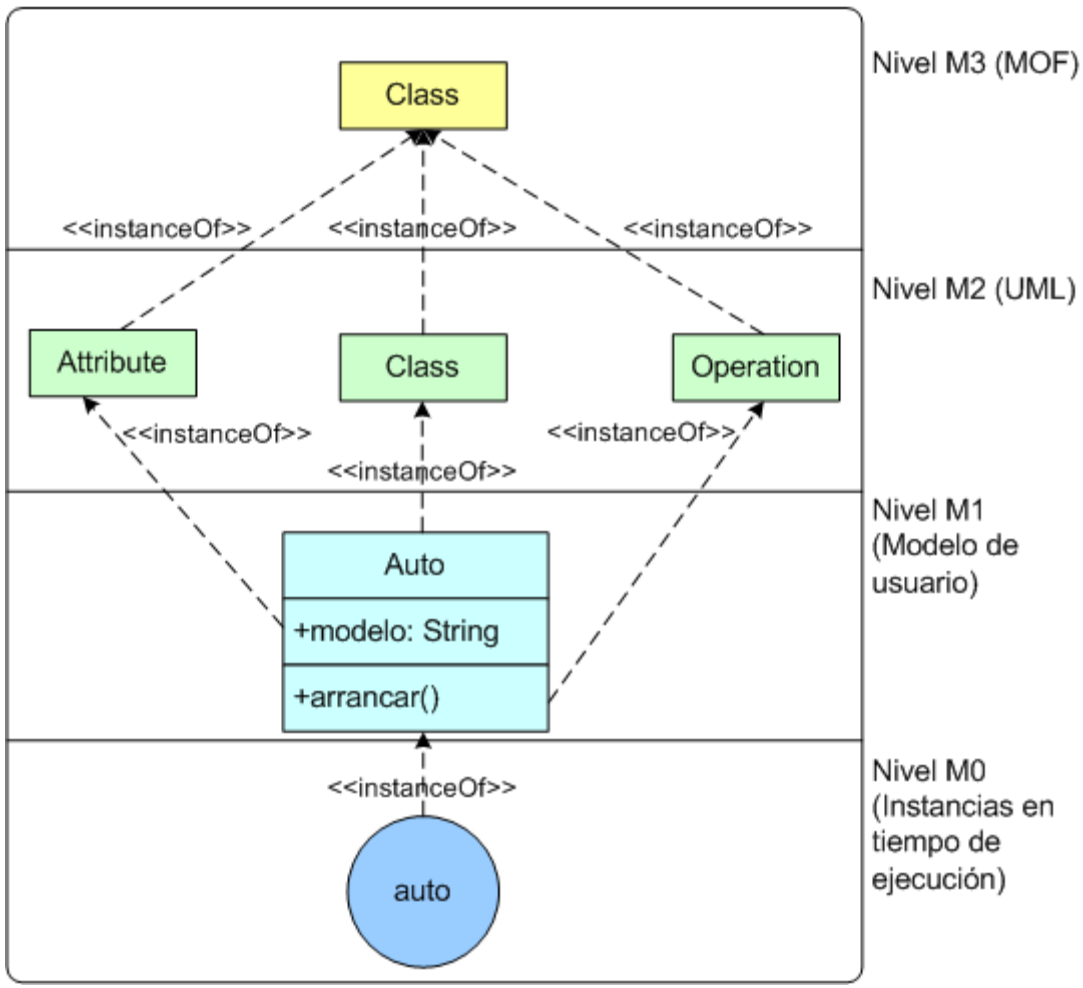


MDA

- Es una propuesta de la OMG para MDE
- Define una sintaxis para la creación y manipulación de modelos: **metamodelado**
- Un metamodelo de un modelo describe qué elementos se pueden usar en el modelo y cómo pueden ser conectados

MDA

- La arquitectura de cuatro capas permite organizar los conceptos de modelado en diferentes niveles de abstracción:
  - Meta-metamodelo
  - Metamodelo
  - Modelo
  - Objetos de usuario



UML

- Es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan a un sistema
- Internamente esta definido utilizando el modelo de cuatro capas de MDA
- Los modelos se especifican utilizando diferentes tipos de diagramas:
  - Estructura
  - Comportamiento
  - Interacción

UML Profile

- Es un mecanismo de extensión provisto por UML para modelar los conceptos de ciertos dominios particulares
- Un perfil es un paquete de UML que agrega nuevos constructores al metamodelo de UML

UML2 Testing Profile

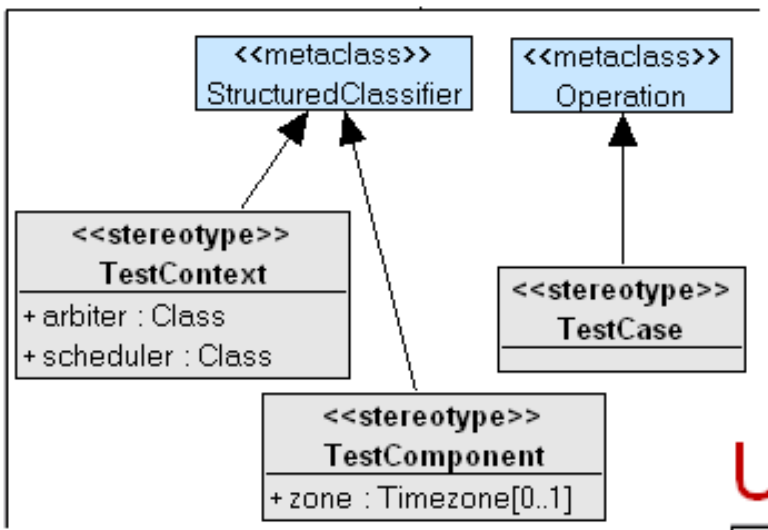
- Es un perfil de UML desarrollado por la OMG para especificar los artefactos de sistemas de prueba de caja negra
- Esta organizado en cuatro grupos conceptuales
  - Test architecture
  - Behavior
  - Data
  - Time

UML2 Testing Profile

- Test Architecture:
  - SUT
  - TestComponents
  - Arbiter
  - Scheduler
  - TestContext
- Test Behavior:
  - TestCase
  - Veredict
- Test Data:
  - DataPool
  - DataPartition
  - DataSelector

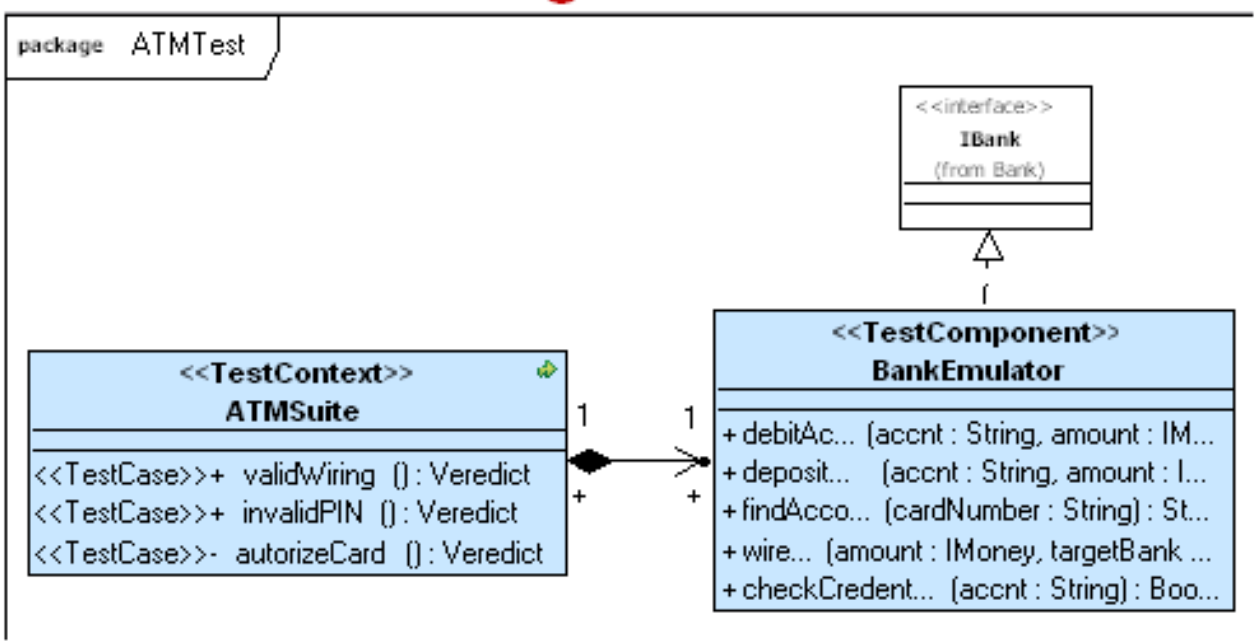
UML2 Testing Profile

U2TP Profile



Aplicación del perfil

UML Class Diagram + U2TP



Temario

- Introducción
- Conceptos básicos
- **Marco tecnológico**
- Solución propuesta
- Diseño de la solución
- Caso de estudio
- Conclusiones

Marco tecnológico

- Eclipse
- Eclipse Modeling Framework (EMF)
- Eclipse Model to Model (M2M)
- Eclipse UML2
- TopCased
- Acceleo

Temario

- Introducción
- Conceptos básicos
- Marco tecnológico
- **Solución propuesta**
- Diseño de la solución
- Caso de estudio
- Conclusiones

Solución propuesta

- Agregar a Eclipse una implementación del perfil U2TP
- Construir un framework basado en Eclipse que transforme modelos UML+U2TP en código fuente para múltiples lenguajes de programación y frameworks de testing
- Extender el framework anterior para crear una implementación que derive código fuente de Java y JUnit
- Crear un caso de estudio que demuestre la utilidad de la herramienta

Temario

- Introducción
- Conceptos básicos
- Marco tecnológico
- Solución propuesta
- **Diseño de la solución**
- Caso de estudio
- Conclusiones

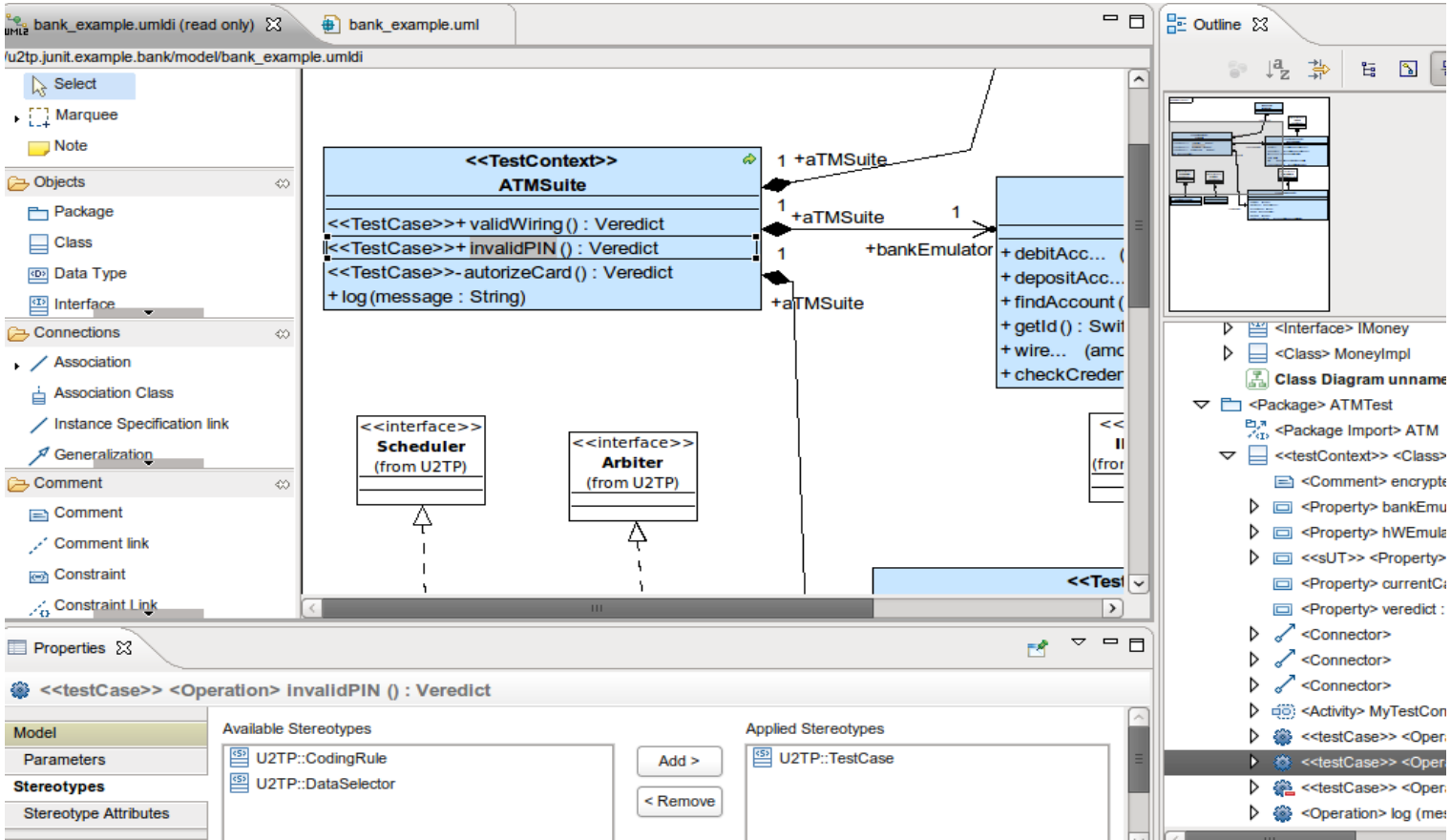
Diseño de la solución

El esquema de la solución se divide en 5 etapas:

- Especificación gráfica de modelos UML+U2TP
- Definición del metamodelo U2TP
- Transformación PIM a PSM
- Derivación de código fuente para clases del dominio
- Derivación de código fuente para los casos de prueba

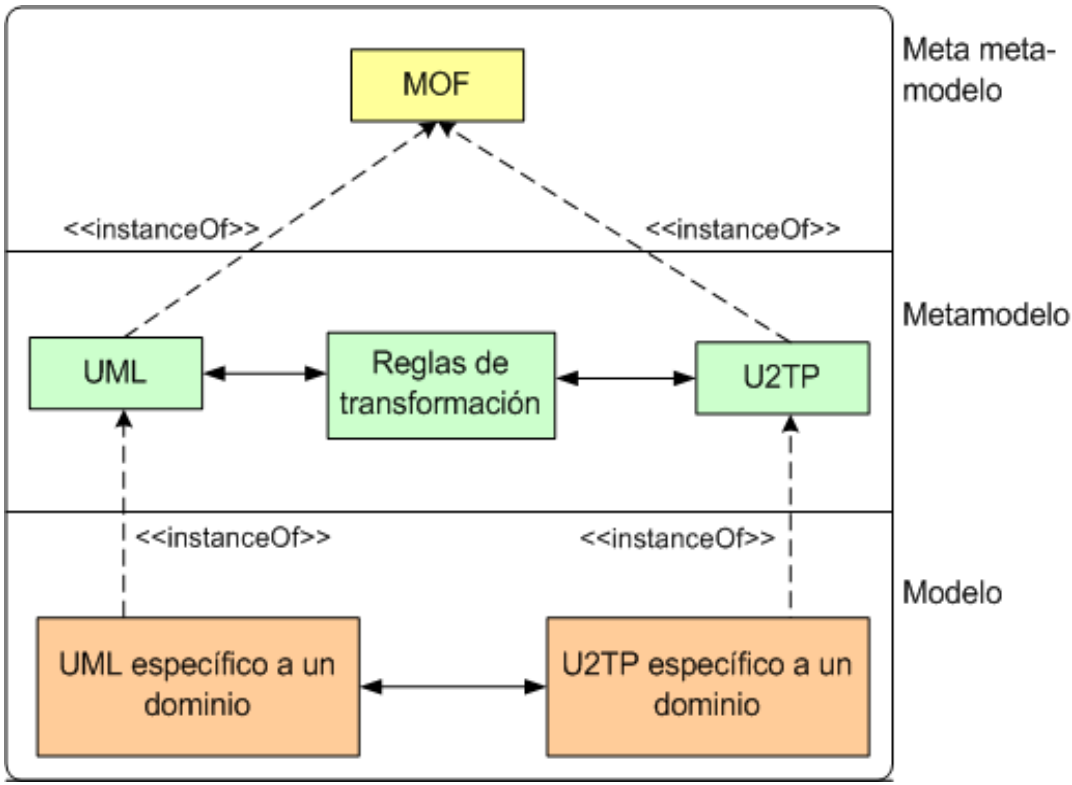
Especificación gráfica

- Agregar a Eclipse una implementación del perfil U2TP para que pueda ser utilizado por las herramientas de modelado gráfico



Metamodelo U2TP

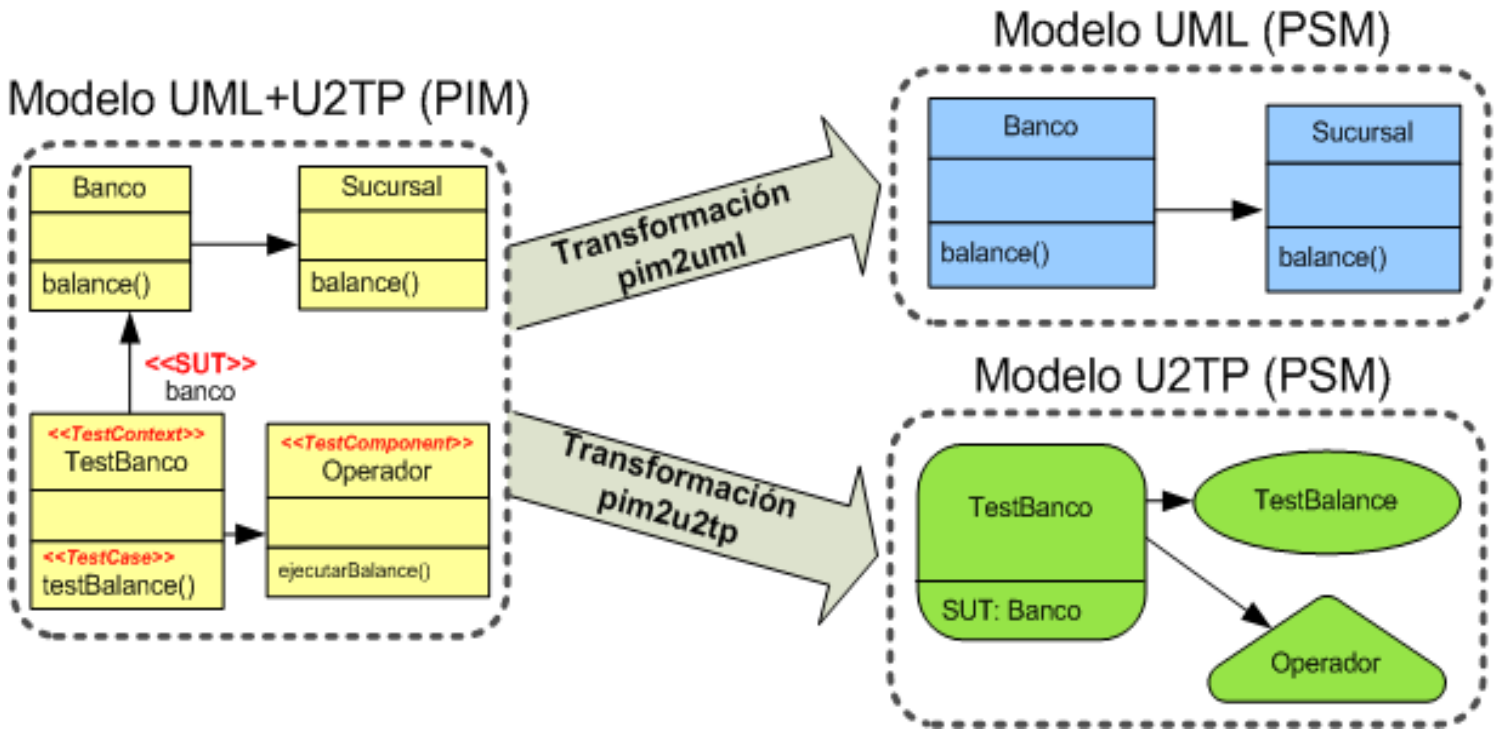
- Está definido dentro de la especificación U2TP
- Es utilizado para modelar sólo los conceptos de testing
- A partir de un modelo UML creado por el usuario se sintetizan los conceptos de testing en un modelo U2TP

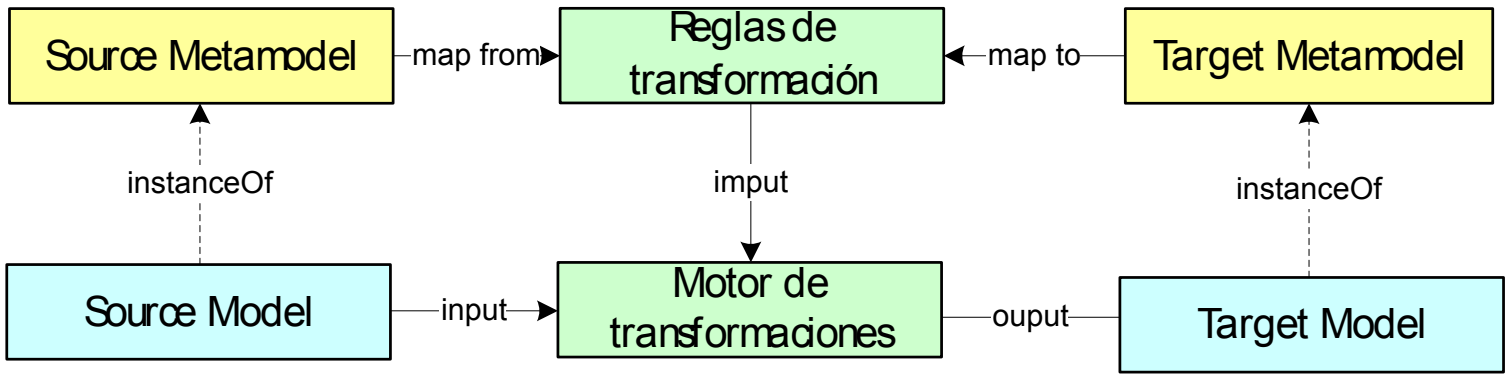


Transformación PIM a PSM

- A partir de un modelo UML+UTP2 creado por el usuario se generan dos modelos intermedios:
  - Modelo UML:
    - instancia del metamodelo UML
    - sólo contiene las clases del dominio
  - Modelo U2TP
    - instancia del metamodelo U2TP
    - sólo contiene los elementos que especifican los casos de prueba
- Para estas transformaciones utilizamos QVTO

Transformación PIM a PSM

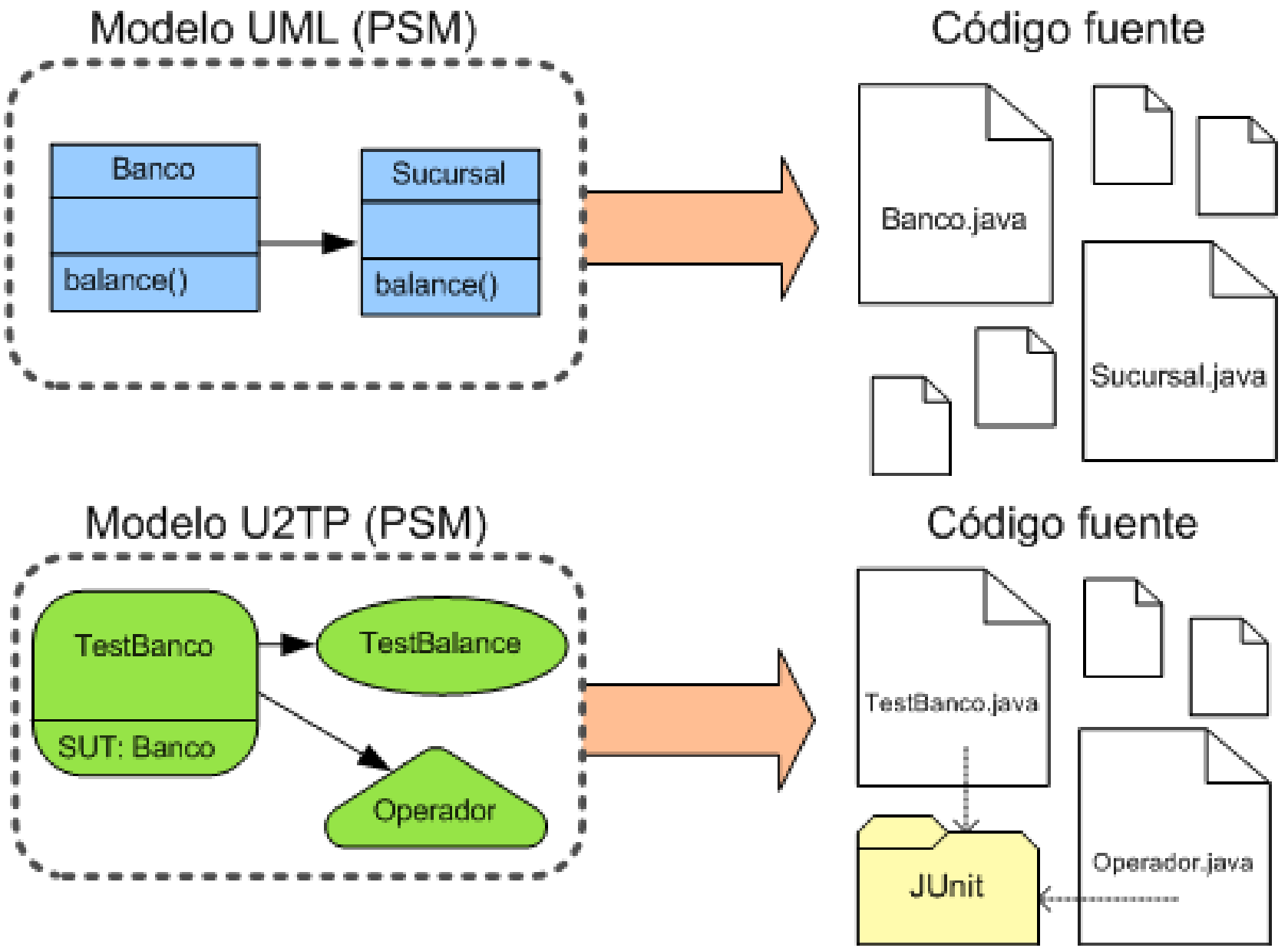




Derivación a código fuente

- A partir de los modelos intermedios creados anteriormente se genera el código fuente para:
  - las clases del dominio
  - los casos de prueba
- Para generar el código fuente se utiliza Acceleo
- En la implementación de esta etapa se definen:
  - lenguaje de programación
  - framework de testing

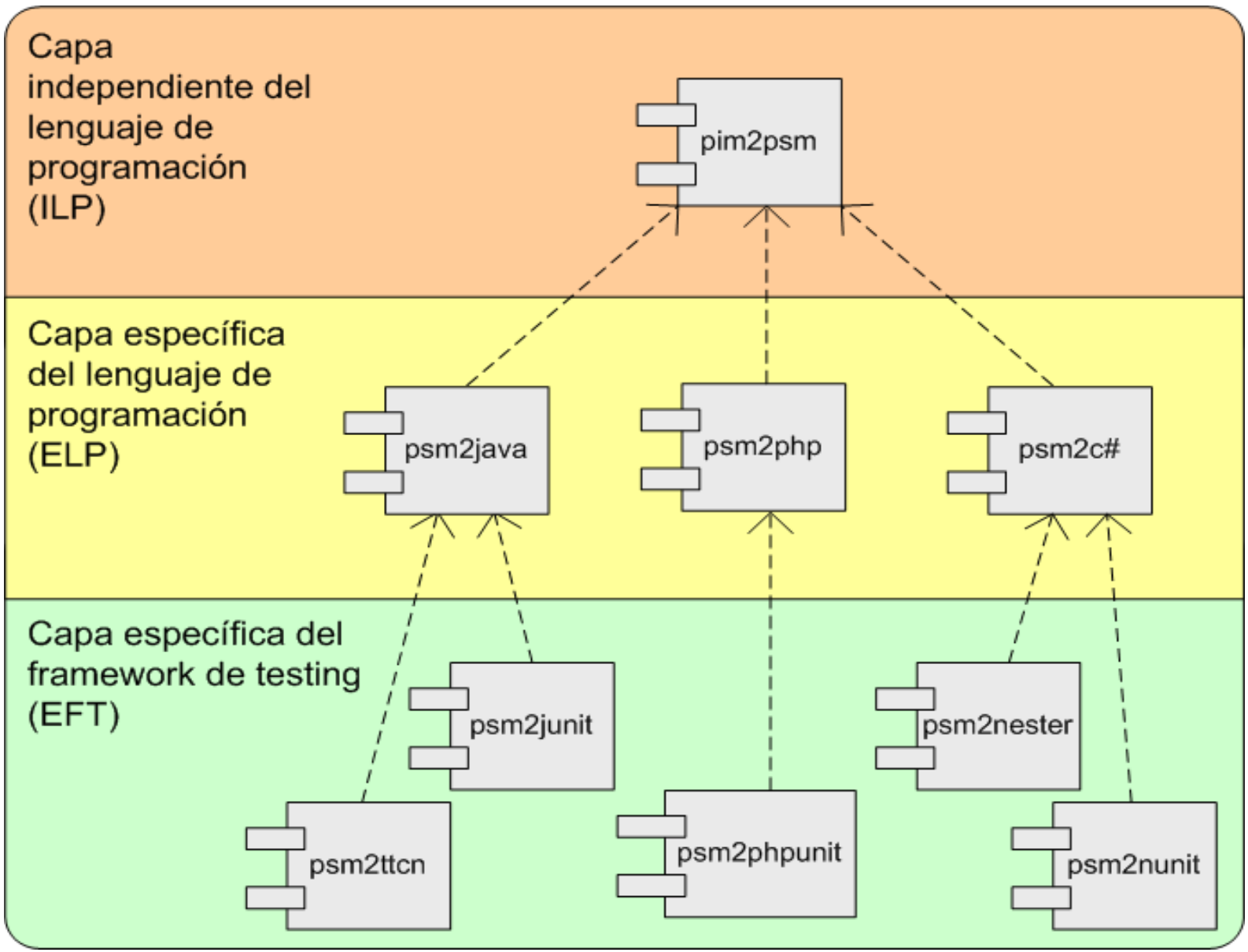
Derivación a código fuente



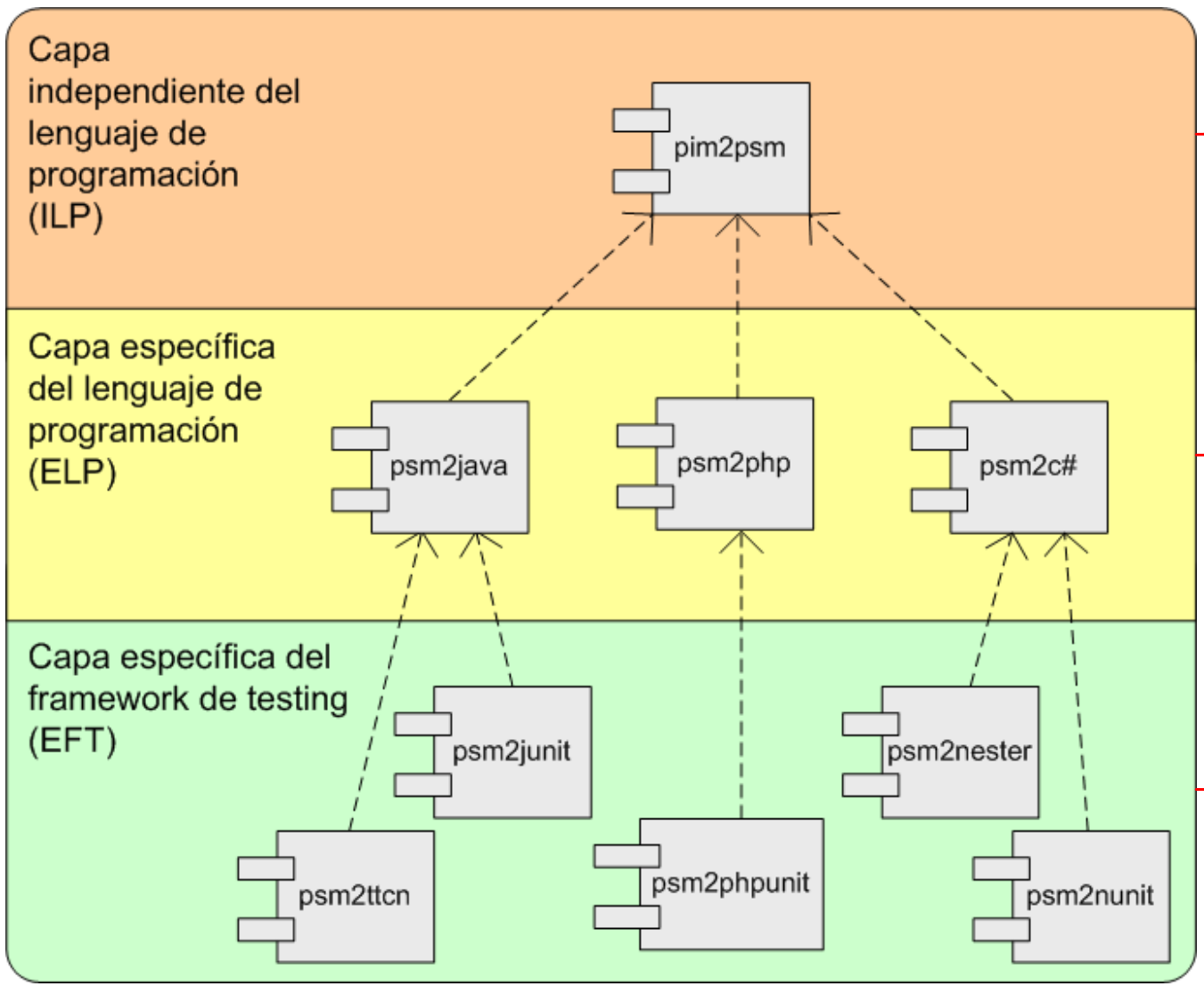
Arquitectura de la solución

- Proporciona un esquema flexible para la generación de código fuente en múltiples lenguajes de programación y frameworks de testing
- Se define a partir de un conjunto de módulos de software donde cada uno tiene una responsabilidad diferente en la derivación de código fuente
- Cada módulo recae en una de las siguiente capas:
  - capa independiente del lenguaje de programación
  - capa específica del lenguaje de programación
  - capa específica del framework de testing

Arquitectura de la solución



Arquitectura de la solución

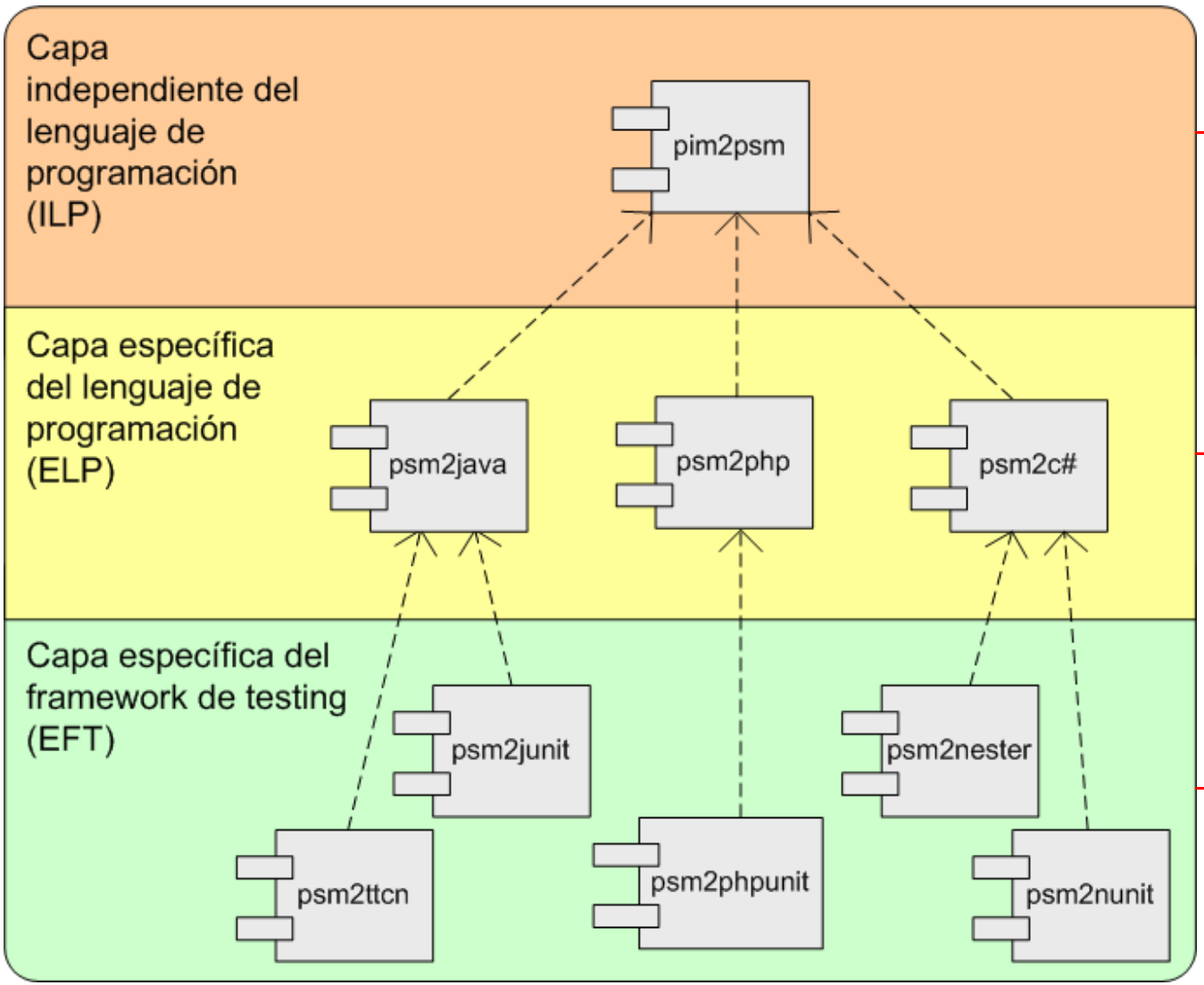


- Especificación gráfica de modelos UML+U2TP
- Definición del metamodelo U2TP
- Transformaciones PIM a PSM

- Generación de código fuente para clases de dominio

- Generación de código fuente para casos de prueba

Arquitectura de la solución

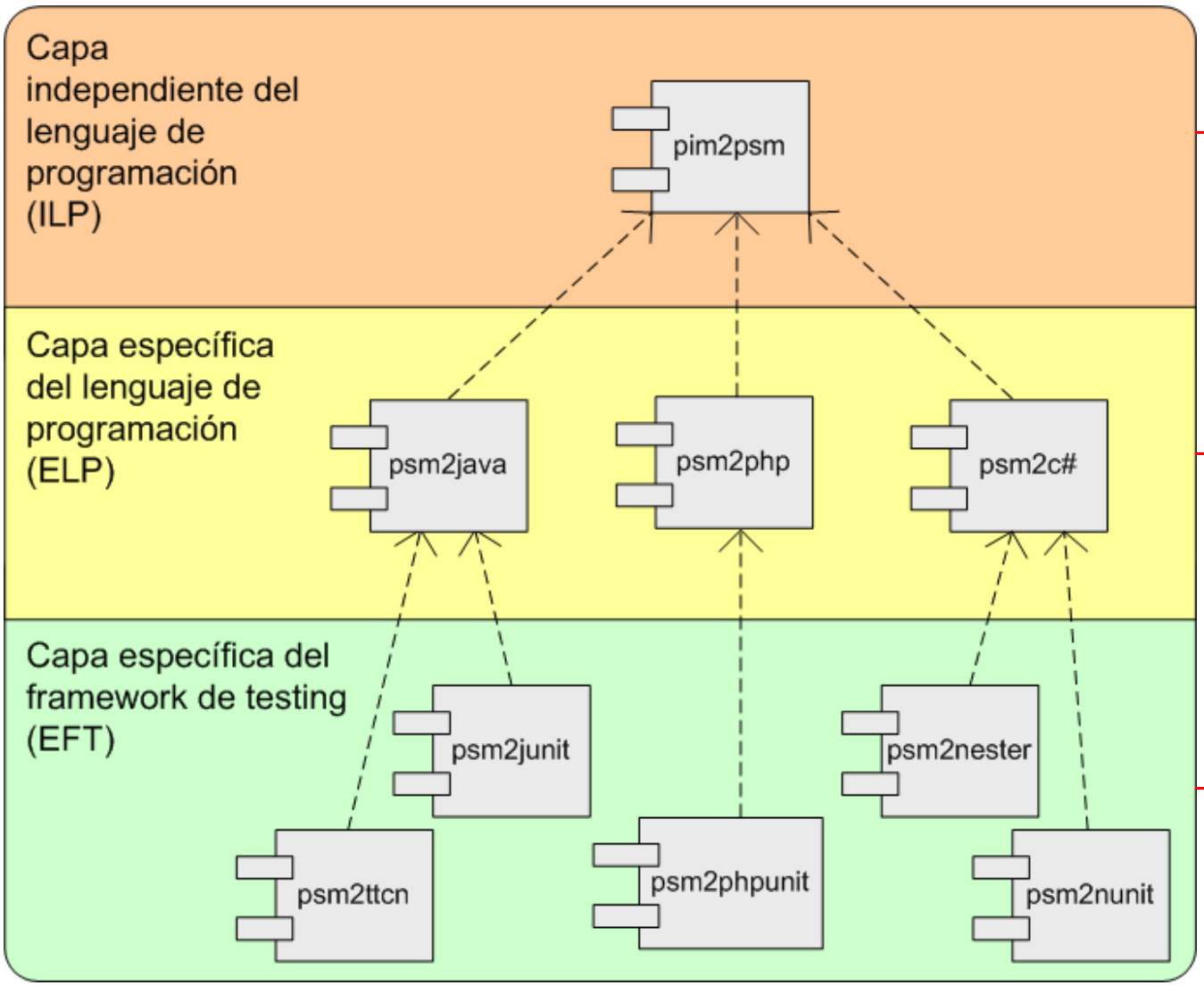


- Especificación gráfica de modelos UML+U2TP
- Definición del metamodelo U2TP
- Transformaciones PIM a PSM

- Generación de código fuente para clases de dominio

- Generación de código fuente para casos de prueba

Arquitectura de la solución

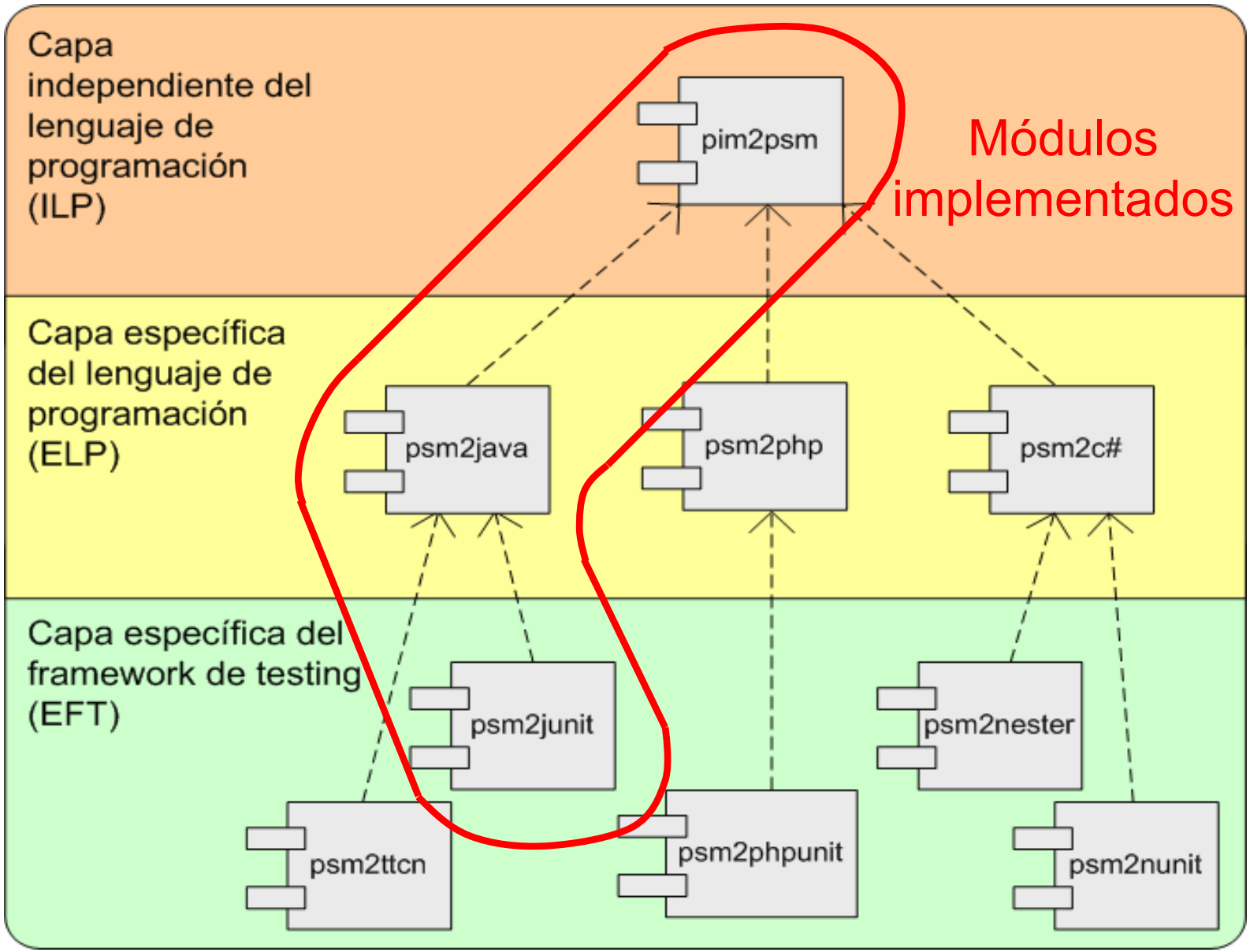


- Especificación gráfica de modelos UML+U2TP
- Definición del metamodelo U2TP
- Transformaciones PIM a PSM

- Generación de código fuente para clases de dominio

- Generación de código fuente para casos de prueba

Arquitectura de la solución



Temario

- Introducción
- Conceptos básicos
- Marco tecnológico
- Solución propuesta
- Diseño de la solución
- **Caso de estudio**
- Conclusiones

Temario

- Introducción
- Conceptos básicos
- Marco tecnológico
- Solución propuesta
- Diseño de la solución
- Caso de estudio
- **Conclusiones**

Resultados obtenidos

- Colaboramos con la comunidad MDA:
  - Implementamos el perfil y el metamodelo completo de U2TP
  - Implementamos un Framework modular que permite:
    - Convertir modelos UML con el perfil U2TP al modelo U2TP
    - Extenderlo para agregar derivaciones de código fuente para diferentes lenguajes de programación
  - Extendimos nuestro propio Framework adicionando una implementación que deriva código fuente para el lenguaje de programación Java y el Framework JUnit a partir de un modelo U2TP (sólo parte estructural)

Conclusiones

- Investigar los los conceptos de MDE y principalmente de MDA nos ha aportado un nuevo paradigma que nos permiten analizar los futuros problemas desde una nueva perspectiva
- La premisa de pensar mayormente en el dominio del problema en lugar de los detalles implementativos permite crear mejores sistemas desde el punto de vista funcional y acelerar los tiempos de desarrollo
- Sin embargo, aún falta invertir mucho esfuerzo en las herramientas de soporte de MDA para que pueda ser utilizado en ambientes productivos

Conclusiones

- La apuesta de la OMG en definir un estándar para especificación de casos de prueba nos parece positivo pero creemos que puede quedar obsoleto si las comunidades de los lenguajes de programación más importantes no lo reconocen
- La distancia semántica entre U2TP y los frameworks de testing más utilizados es grande
- Con respecto a la herramienta desarrollada, creemos haber dado el puntapié inicial, desde el punto de vista implementativo, que puede motivar a otras personas a continuar el desarrollo de nuevas herramientas

Trabajos futuros

- Implementar los conceptos faltantes del perfil U2TP
- Extender la herramienta para dar soporte a nuevos lenguajes de programación y frameworks de testing
- Analizar la viabilidad de realizar ingenería inversa sobre los casos de prueba

¿PREGUNTAS?