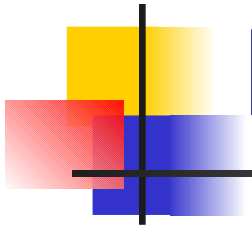
Consideraciones para integrar los métodos formales como práctica en el desarrollo de software

Raquel Anaya
Líder del grupo de Ingeniería de Software
ranaya@eafit.edu.co



Motivación

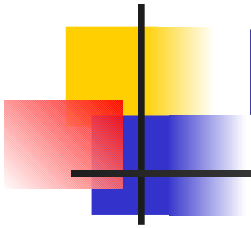
¿Por qué los métodos formales no son utilizados en el desarrollo de software industrial?

No es único que garantiza la calidad

Mayor énfasis en el tiempo de puesta en el mercado que la calidad del producto

Ámbito limitado de los métodos formales

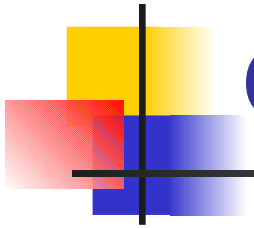
Escalabilidad limitada de los métodos



Motivación

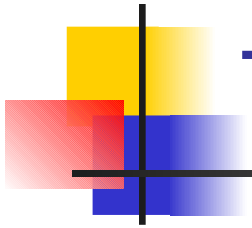
- ¿Qué son los métodos formales?
- ¿Será posible combinar los lenguajes gráficos con lenguajes formales?
- ¿En qué campos de aplicación se ve promisorio el uso de los lenguajes formales?
- Si quisiéramos adoptar un lenguaje formal como estrategia pedagógica para formar nuestros estudiantes, cuál adoptaríamos?

Categorías de los métodos del desarrollo de software



- Métodos heurísticos: aproximaciones informales
- Métodos formales: soportados en formalismos matemáticos
- Métodos de prototipado: aproximaciones basadas en diferentes formas de prototipos
- Estas categorías no son disjuntas: por ejemplo un método orientado a objetos particular puede integrar una técnica formal y utilizar prototipado para verificación y validación

El papel de los métodos formales



- Qué es un método formal
  - Es cualquier actividad relacionada con representaciones matemáticas del software:
    - Especificación formal del sistema
    - Análisis y demostración de la especificación
    - Desarrollo transformacional
    - Verificación de programas

El papel de UML (Unified Modeling Language) en el desarrollo de software

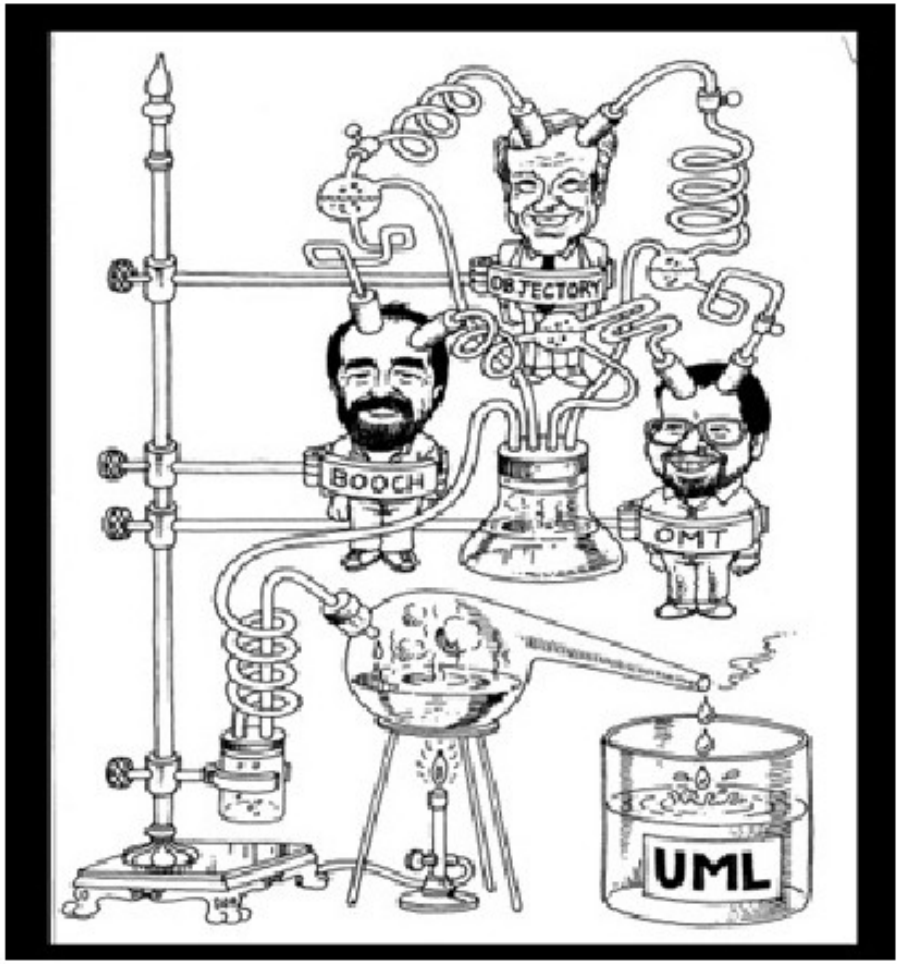
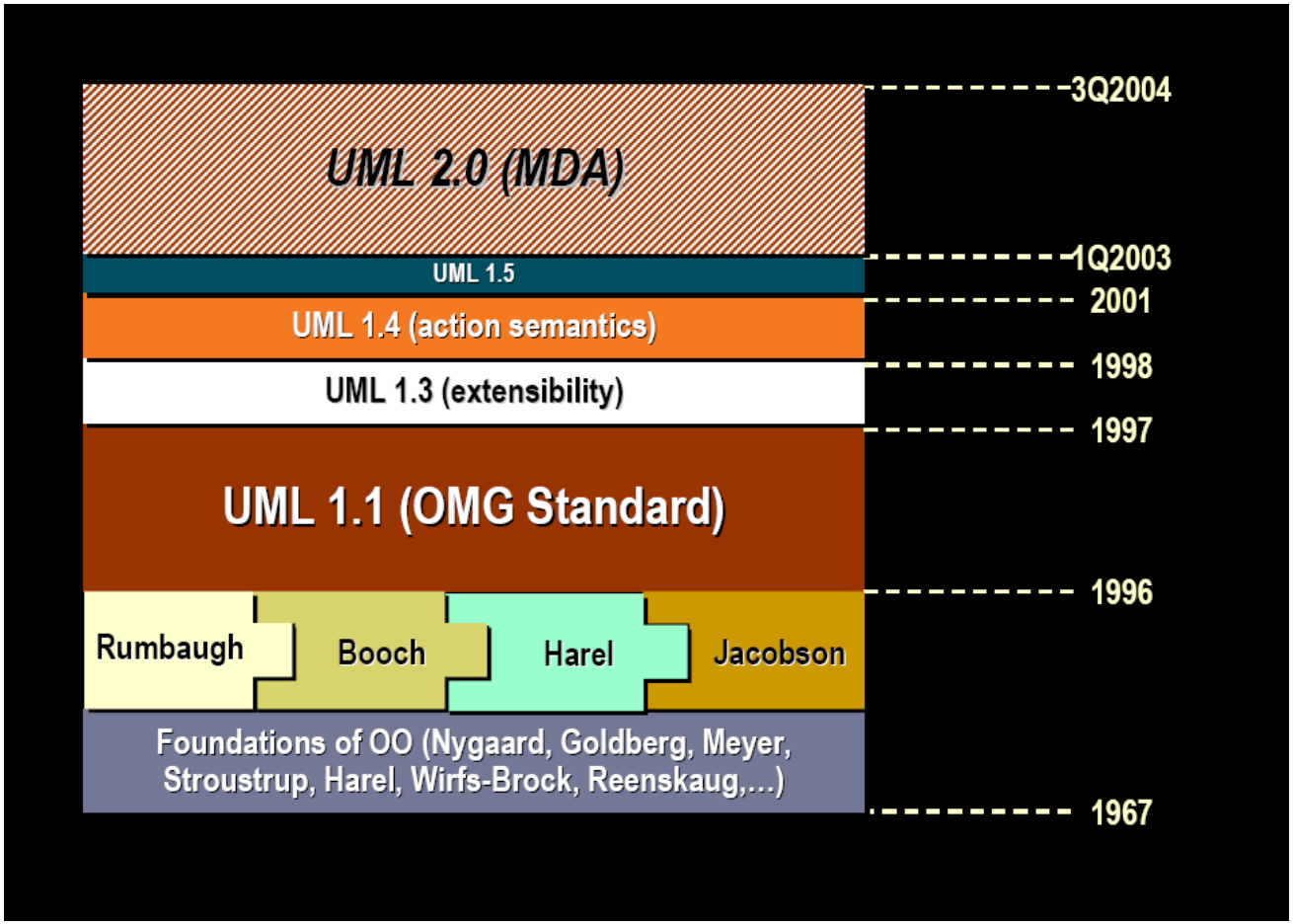
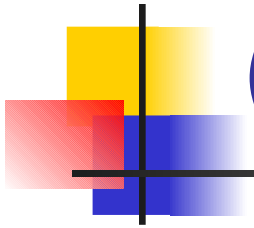


Figure 1: How it all began

El papel de UML



Arquitectura de modelos (OMG)



- **MO: Layer de objetos de usuario**
  - Layer en el que residen las instancias de objetos activas en ejecución. Compuesto de la información que se desea describir. Utilizado para formalizar expresiones específicas referentes a un elemento dado
  - Vista del usuario final
- **M1: Layer de modelo del usuario**
  - Layer donde de desarrollan los modelos que representan la solución de un problema. El modelo del usuario describe información del dominio. Sirve como plantilla para describir la estructura del layer MO
  - Vista del diseñador
- **M2: Layer del metamodelo**
  - Layer donde se define el metamodelo de UML. Conceptos utilizados por el modelador del UML. Incluye conceptos de una aproximación metodológica (OO, CBSE, AOSD, ADL)
  - Vista del metodologista
- **M3: Layer meta del metamodelo**
  - Layer que contiene la definición meta de los elementos sobre los cuales se define un lenguaje de modelado. En la terminología de OMG este layer recibe el nombre de MOF (Meta Object Faciclity)
  - Vista del integrador de meta modelos
- La integración de estos layers dan lugar a herramientas abiertas a nivel de Meta CASE

Arquitectura de Modelos (OMG)

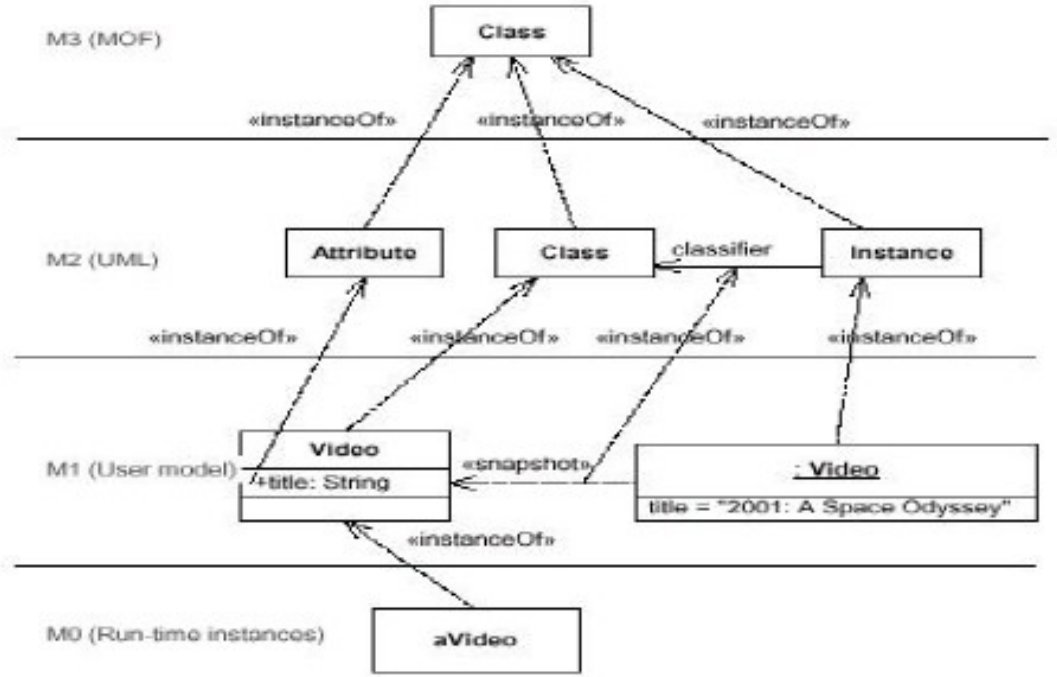
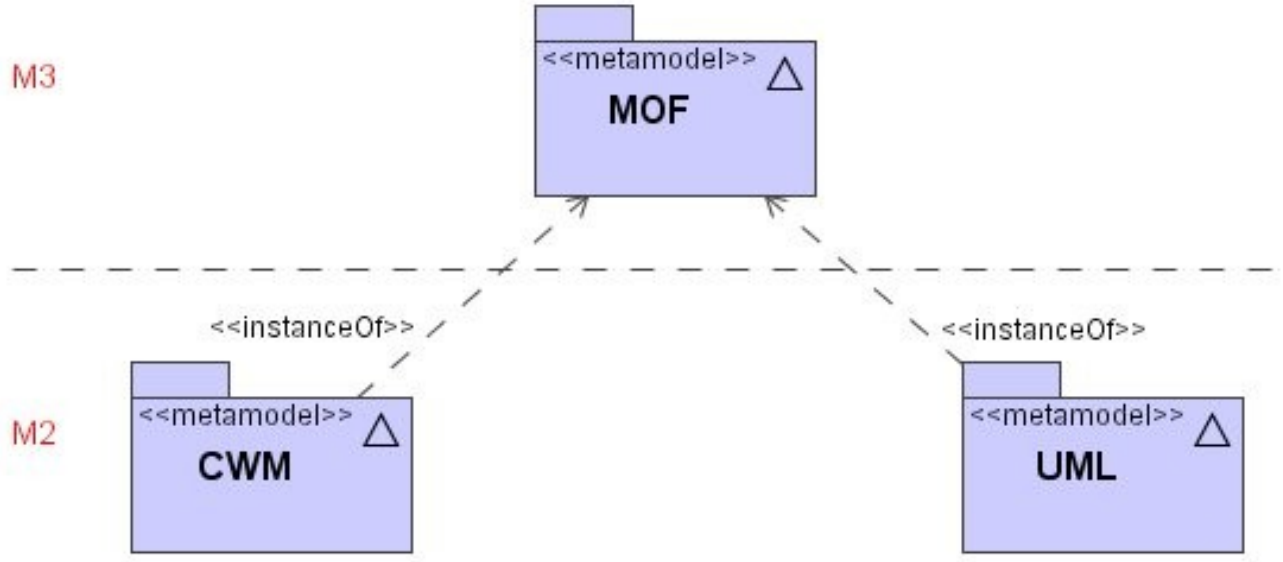
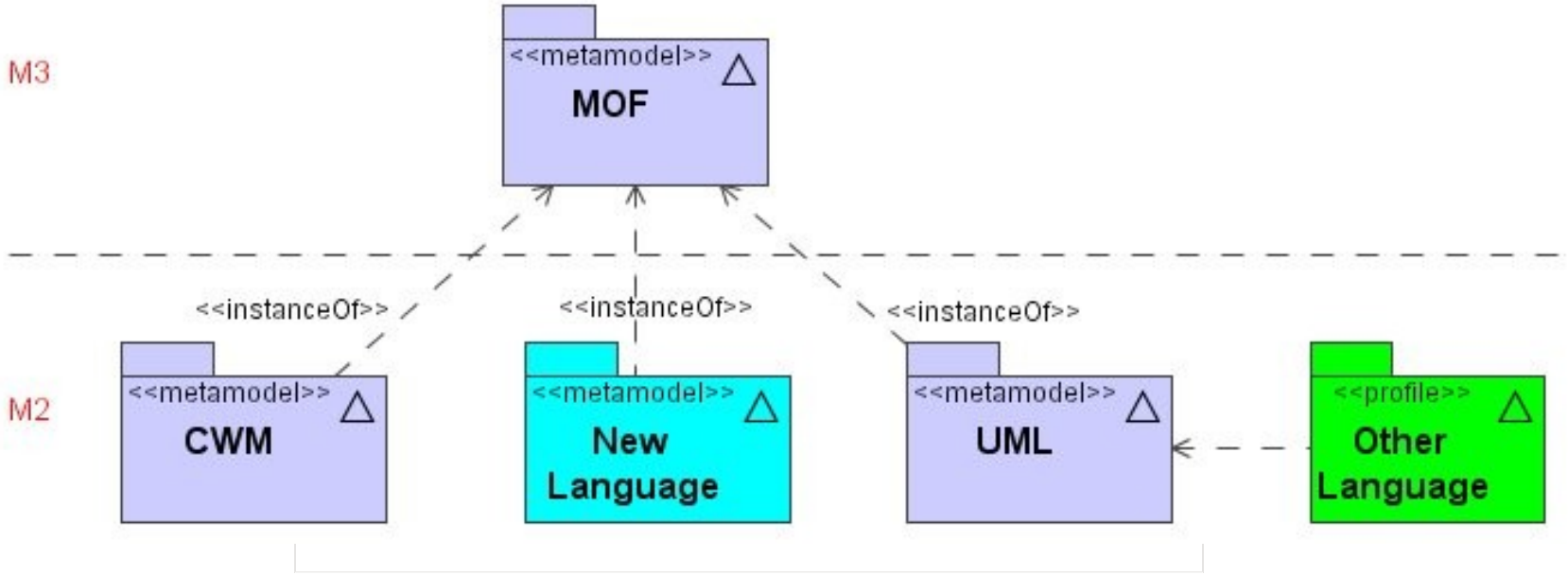


Figure 1: UML Metamodel Layer Architecture

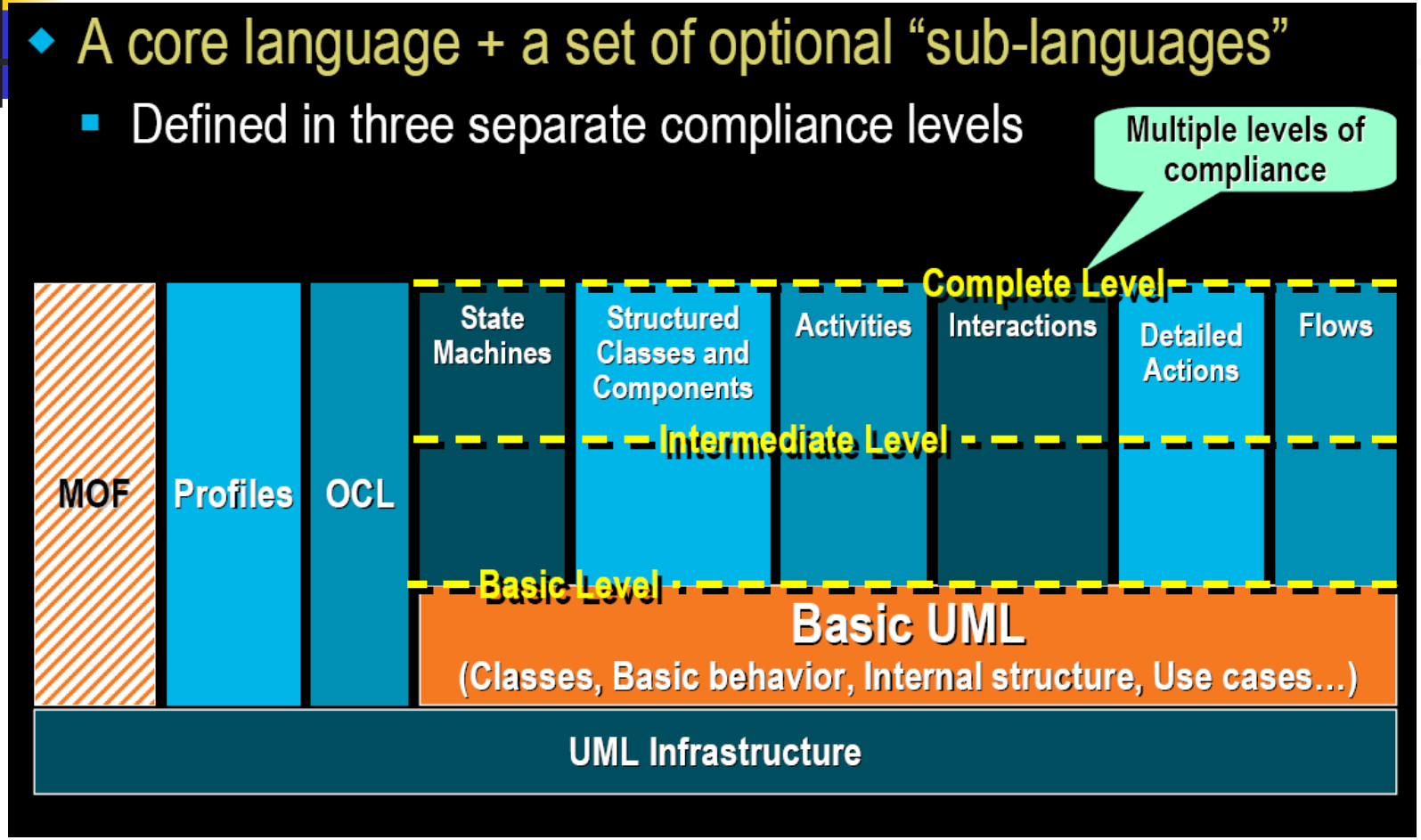
MOF como “librería” para *construír* nuevos lenguajes



Language definition mechanisms

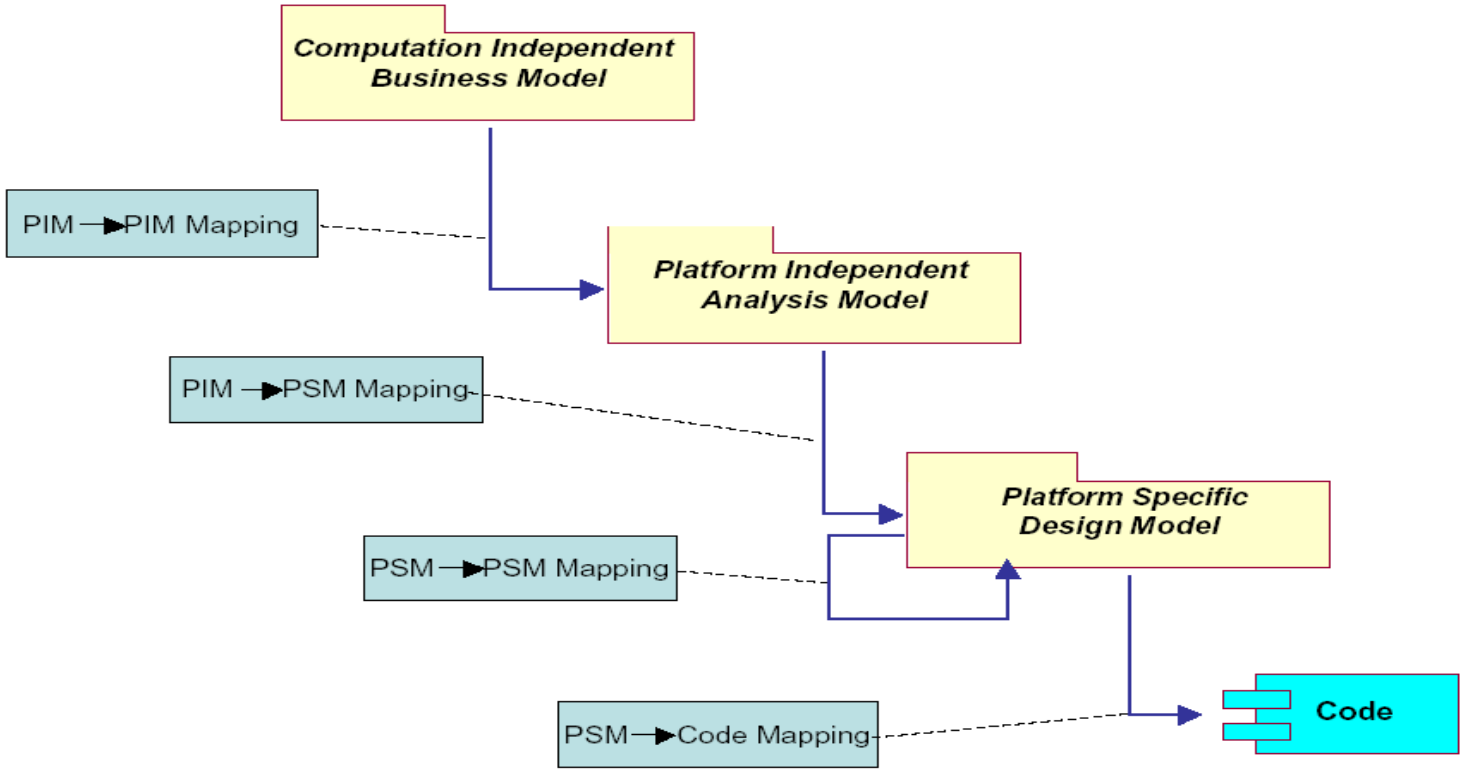


La “tienda de lenguajes de UML”

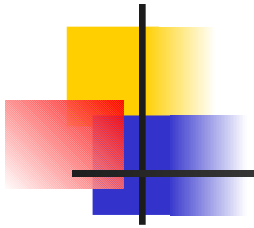


Hasta que nivel de formalización llegar? Cuál es el conjunto mínimo de Notaciones con que se puede representan un sistema complejo?

El proceso de tranformación:

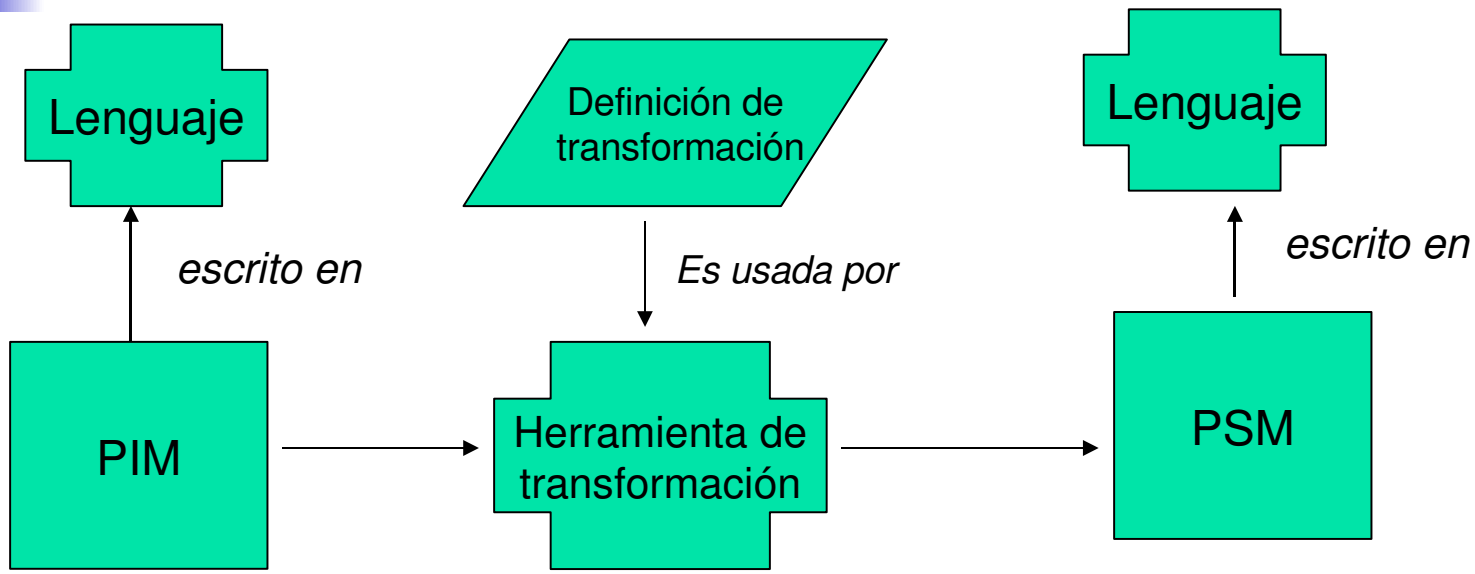


El proceso de Transformación



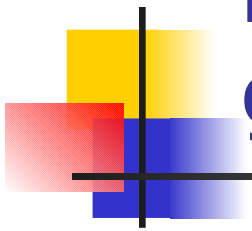
- La transformación es la generación automática de un modelo fuente en un modelo objetivo, de acuerdo a unas definiciones de transformación
- Una definición de transformación esta conformada por un conjunto de reglas que describen como el modelo en el lenguaje fuente puede ser transformado en un modelo en el lenguaje destino
- Una regla de transformación es una descripción de uno o más constructores en el lenguaje fuente que pueden ser transformados en uno o mas constructores en el lenguaje destino

Elementos de la transformación



El lenguaje fuente y el lenguaje destino pueden ser el mismo

Estrategias de acercamiento de los métodos formales con los lenguajes gráficos



- Estrategia arriba-abajo
  - Partir de un lenguaje gráfico y definir un formalismo matemático que los sustente
- Estrategia abajo-arriba
  - Partir de un formalismo existente y proponer un lenguaje gráfico que permita capturar todos los elementos del lenguaje
- Estrategia híbrida
  - Tomar un lenguaje gráfico existente, por ejemplo UML y tomar un lenguaje formal existente y hacer una integración entre ellos

Un enfoque práctico de la integración de métodos formales

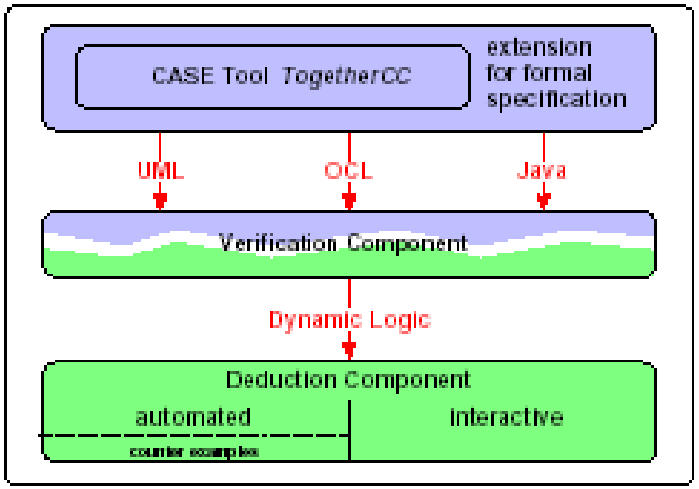
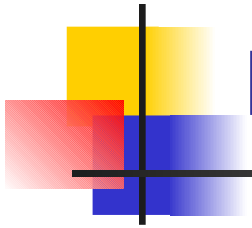


Fig. 1. Structure of the KeY system

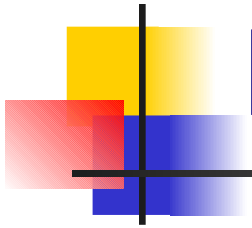
URL: http://i12www.ira.uka.de/~key/

- UML para el modelado + OCL para definir restricciones
- Extensión de una herramienta CASE (TogetherCC) para agregar la especificación formal
- Verificación formal basada en una semántica axiomática de Java (Java CARD)

Métodos formales reconocidos



- De tipo algebraico
  - Secuenciales: Larch, OBJ
  - Concurrente: Lotos
- Basado en modelos
  - Secuencial: Z, VDM, B
  - Concurrentes: CSP, Redes de Petri
- Algunos contactos ya establecidos: RAISE



Necesidades de aplicación

- Para modelar la orquestación de servicios de sistemas basados en SOA
- Para modelar la flexibilidad del proceso software
- Para hacer síntesis y animación de modelos en herramientas

¿Acciones a seguir?