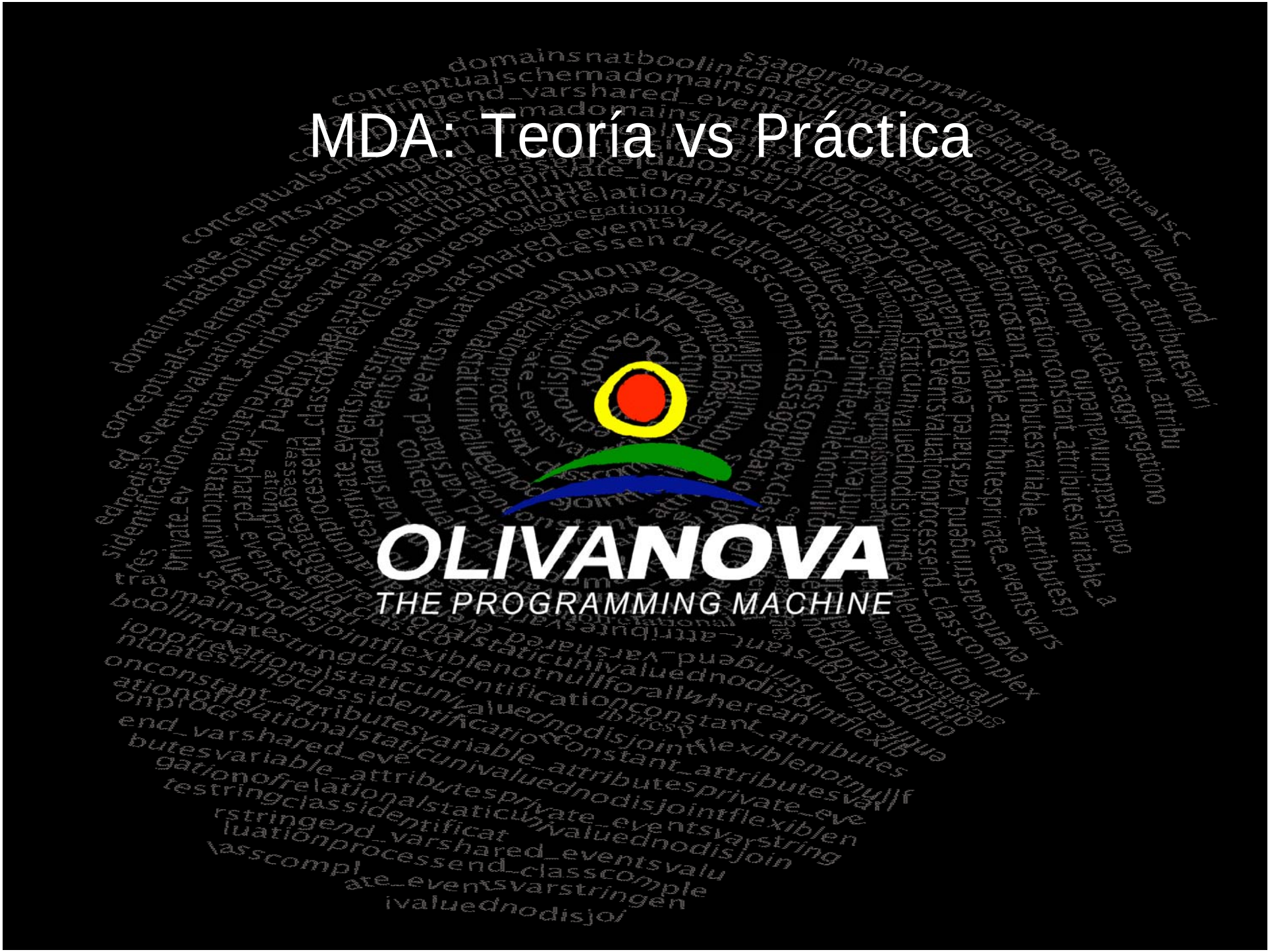
MDA: Teoría vs Práctica



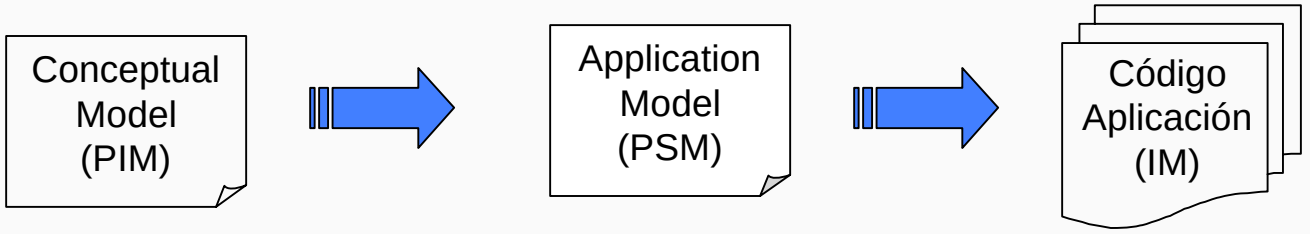
CONTENIDO

- ¿QUÉ ES MDA?
  - Conceptos
- *OLIVA**NOVA***
  - MDA *& OLIVA**NOVA***
  - *OLIVA**NOVA*** en acción
- *OLIVA**NOVA*** Productos
- Modelo Conceptual *OLIVA**NOVA***
  - Modelo de objetos, Modelo Dinámico, Modelo Funcional, Modelo de Presentación
  - Validación
- *OLIVA**NOVA*** Transformation Engines

MDA

- Model Driven Architecture (MDA)
- Iniciativa promovida por OMG
- Separa la definición del problema de la plataforma tecnológica en la cual será implementada
- Cambio en el Paradigma de Desarrollo: enfocado en los modelos, no en el código

CONCEPTOS MDA



- **PIM**
  - Modelo Independiente de Plataforma. Es una vista de un sistema independiente de la plataforma en la que se instala.
- **PSM**
  - Modelo Específico de Plataforma. Es una vista de un sistema para una plataforma específica.
- **IM**
  - Modelo de Implementación. Implementación de un sistema en un plataforma y lenguaje específicos.

Promesa-Realidad-Consecuencia

- Los modelos representan la funcionalidad del negocio y su comportamiento.
  - Con frecuencia, los modelos no son lo suficientemente expresivos como para capturar toda la funcionalidad
    - Modelos incompletos → No documentan completamente el problema → Documentación “por código” → Vuelta al código.
    - Modelos no ejecutables → Generación de código limitada a “esqueletos” → Fill-in-the-gaps → Vuelta al código.
    - Modelos mezclan especificación y código fuente (java, C#) → Sincronización entre modelo y código → Trabajo duro en el código → Vuelta al código

Promesa-Realidad-Consecuencia

- Los modelos pueden transformarse en código a cualquier plataforma
  - La falta de expresividad se suple mediante soluciones que nos atan a plataformas concretas
    - Modelos obsoletos por avances en las tecnologías

¿Qué necesitamos para hacer MDA?

- Un lenguaje de modelado para crear PIMs que sea:
  - Abstracto → Para independizar mis modelos de la plataforma de implementación.
  - Preciso → Para representar completamente la funcionalidad de mis sistemas.
- Un conjunto de PSMs para las plataformas en las que deseo implementar mis sistemas.
- Herramientas de modelado.
- Un conjunto de reglas de transformación de PIM a PSM y de PSM a Código.
- Idealmente, herramientas que automaticen estas transformaciones.

¿Qué necesita un equipo de desarrollo de Software?

- ¿Definir su propio PIM, PSM o IM?
  - Proceso complejo y costoso
- ¿Crear y mantener las reglas de transformación?
  - Alta complejidad
  - Variable por la evolución de la tecnología
  - Dependiente de la expresividad del PIM

¿Desarrollar un compilador de Modelos?

Utilizar herramientas que implementen MDA

*OLIVA**NOVA***

- Conjunto de herramientas desarrolladas por CARE Technologies.
- Separan la definición de una aplicación de los detalles de implementación de la misma.
- Cambio en el paradigma de desarrollo: las aplicaciones se obtienen automáticamente a partir de los modelos.

Un compilador de Modelos Conceptuales

MDA & *OLIVA**NOVA***

- Una misma definición de aplicación puede implementarse de múltiples maneras:

- PIM
- PSM
- IM

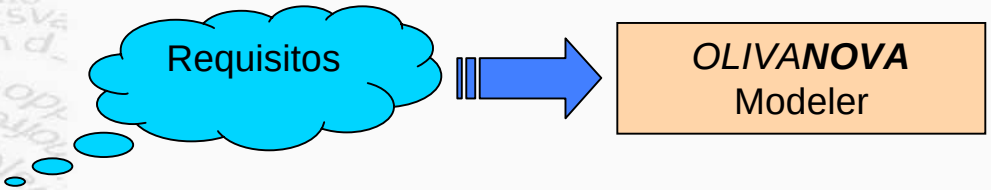
OLIVANOVA
THE PROGRAMMING MACHINE

- Modelo Conceptual
- Modelo Ejecución
- Código Fuente

*OLIVA**NOVA*** **EN ACCIÓN**

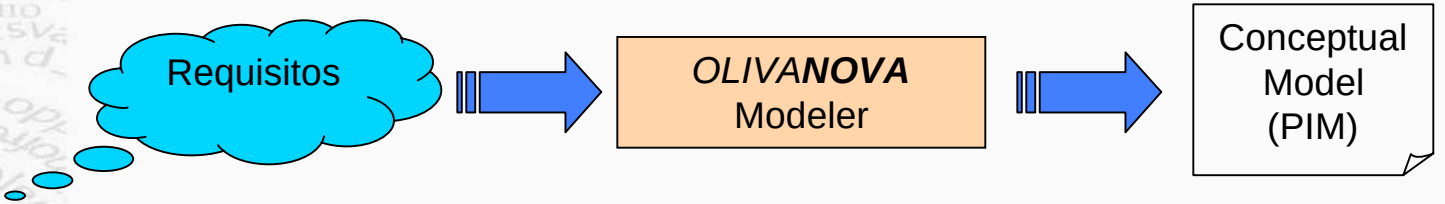
- El cliente o usuario de la aplicación expone los requisitos sobre la aplicación que desea

*OLIVA**NOVA*** **EN ACCIÓN**



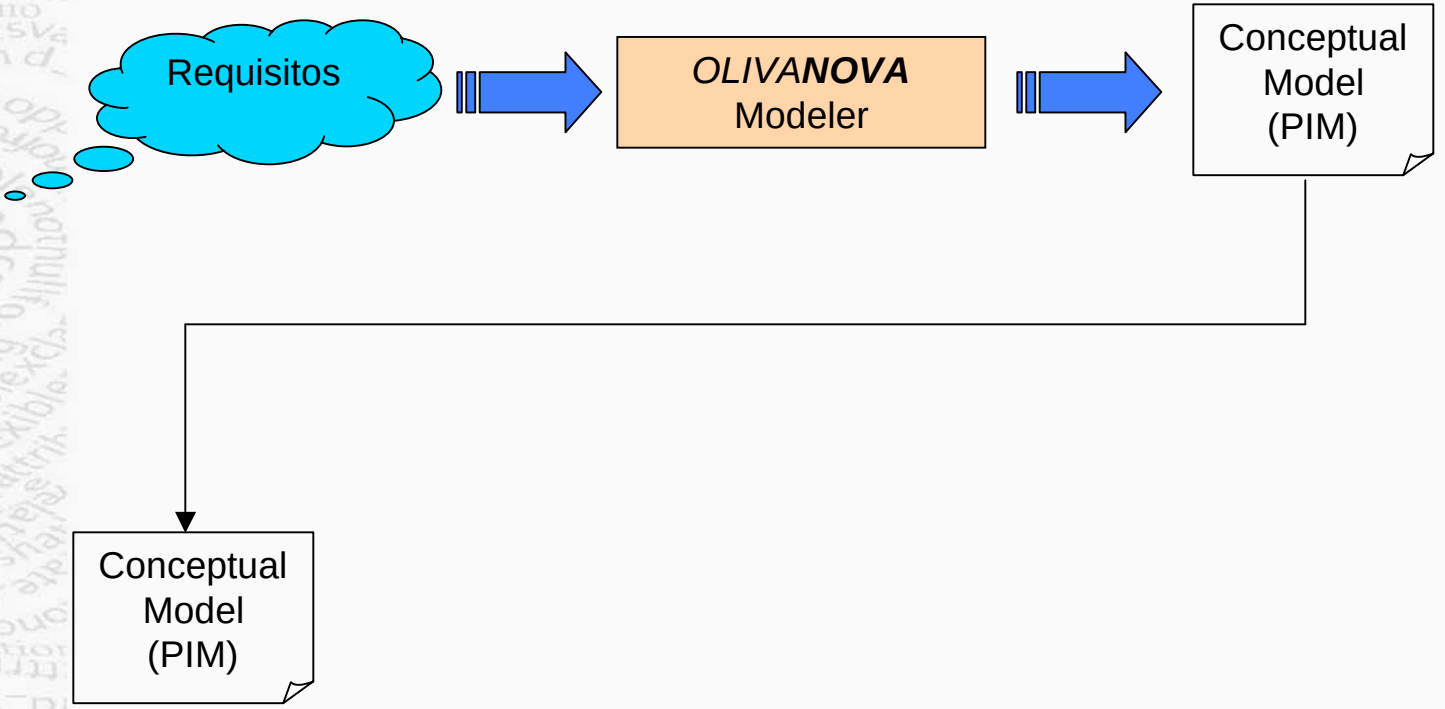
- A partir de los requisitos, el analista utiliza *OLIVA**NOVA*** Modeler para crear ...

*OLIVA**NOVA*** **EN ACCIÓN**



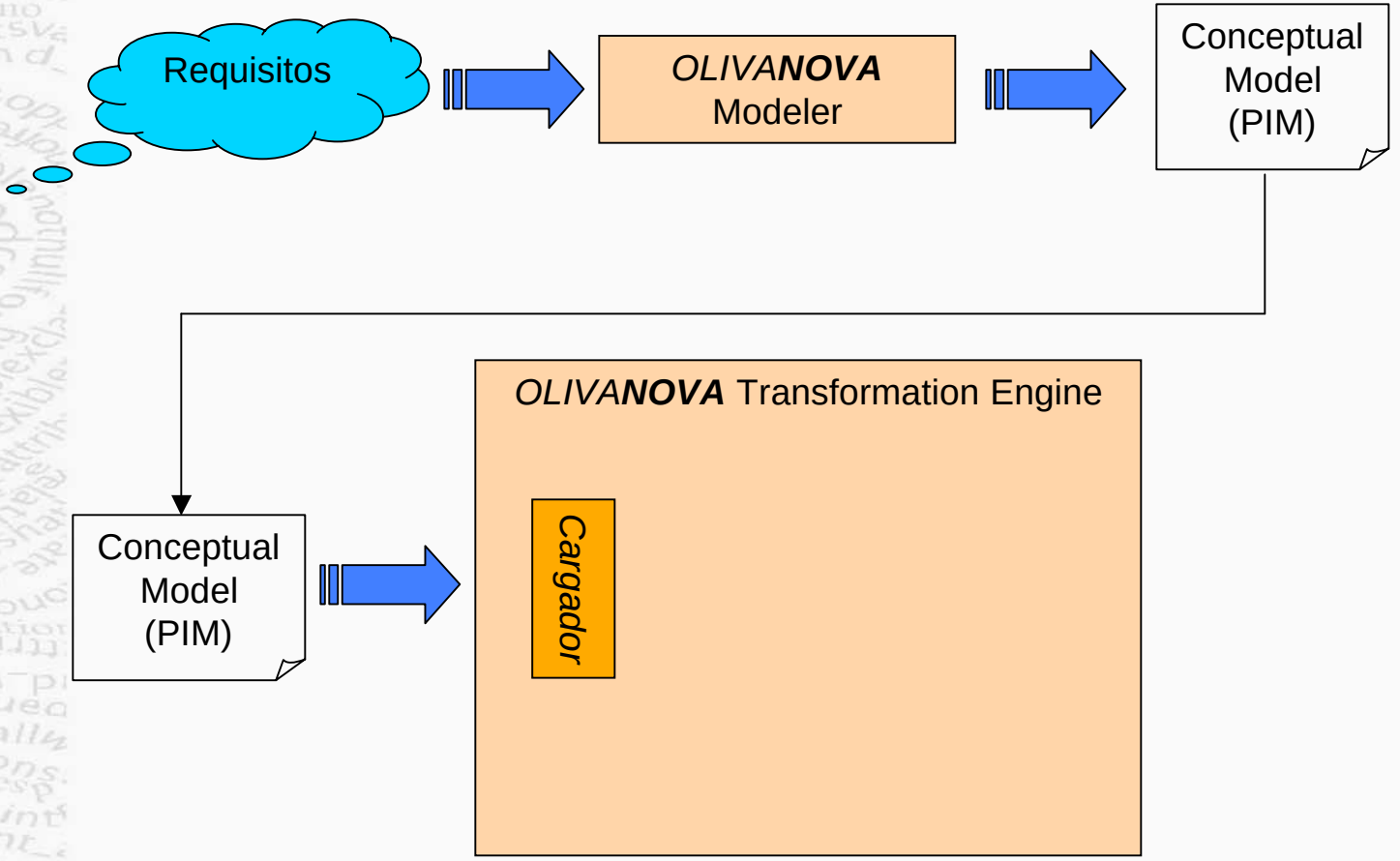
- ... un modelo conceptual de la aplicación sin detalles de implementación

*OLIVA**NOVA*** **EN ACCIÓN**



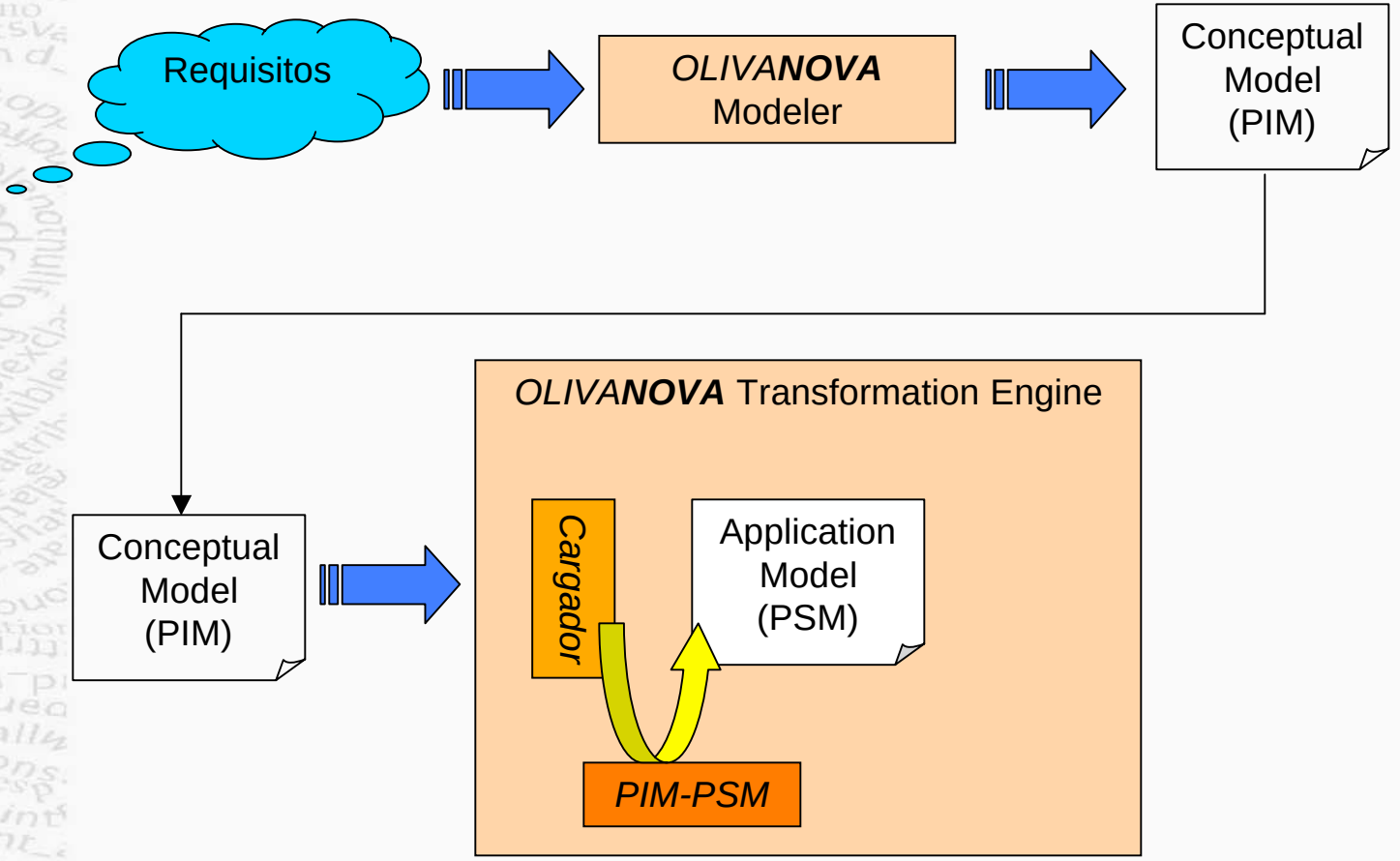
- Dicho modelo conceptual, a su vez ...

*OLIVA**NOVA*** **EN ACCIÓN**



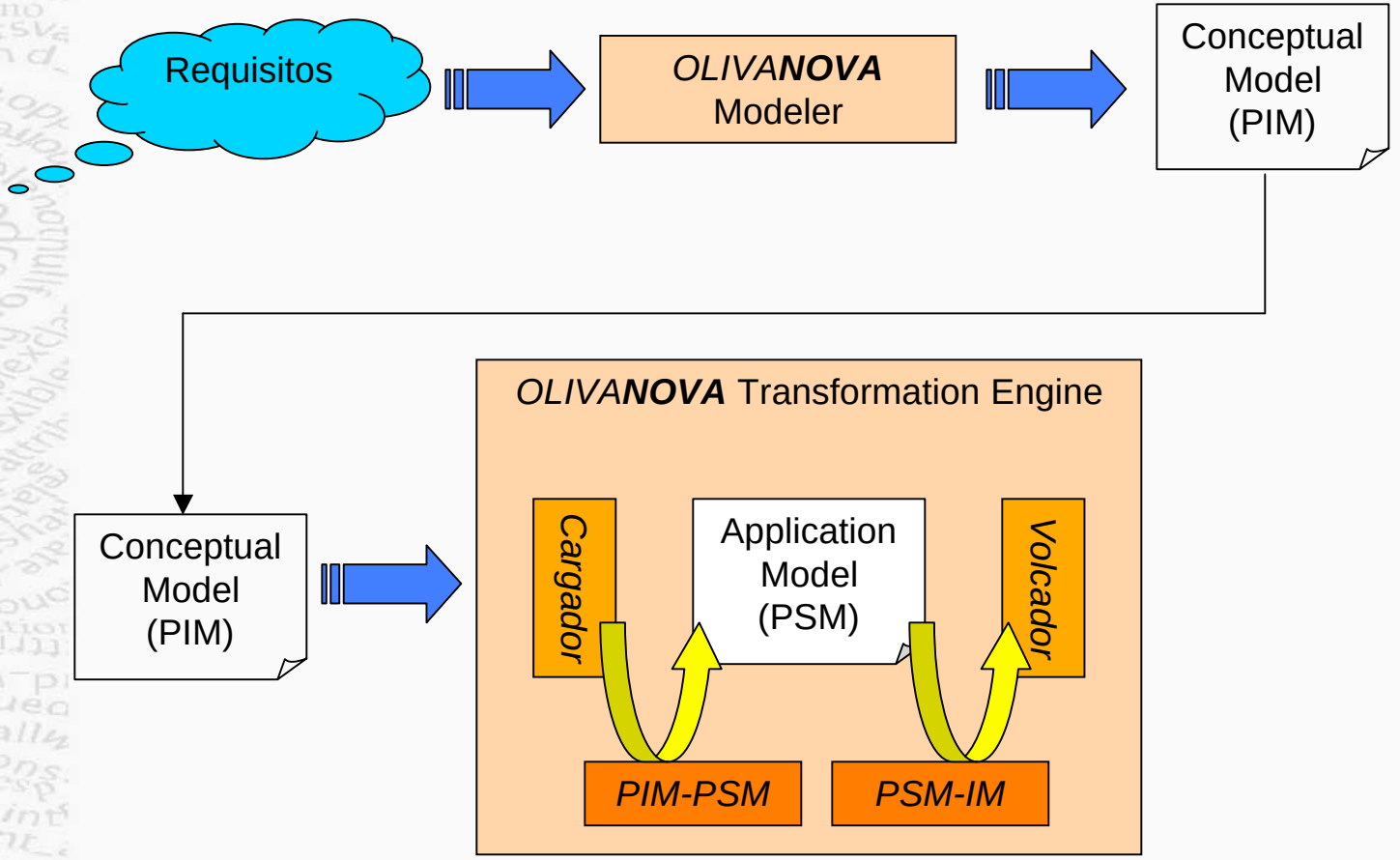
- ... Es cargado por un *OLIVA**NOVA*** Transformation Engine

*OLIVA**NOVA*** **EN ACCIÓN**



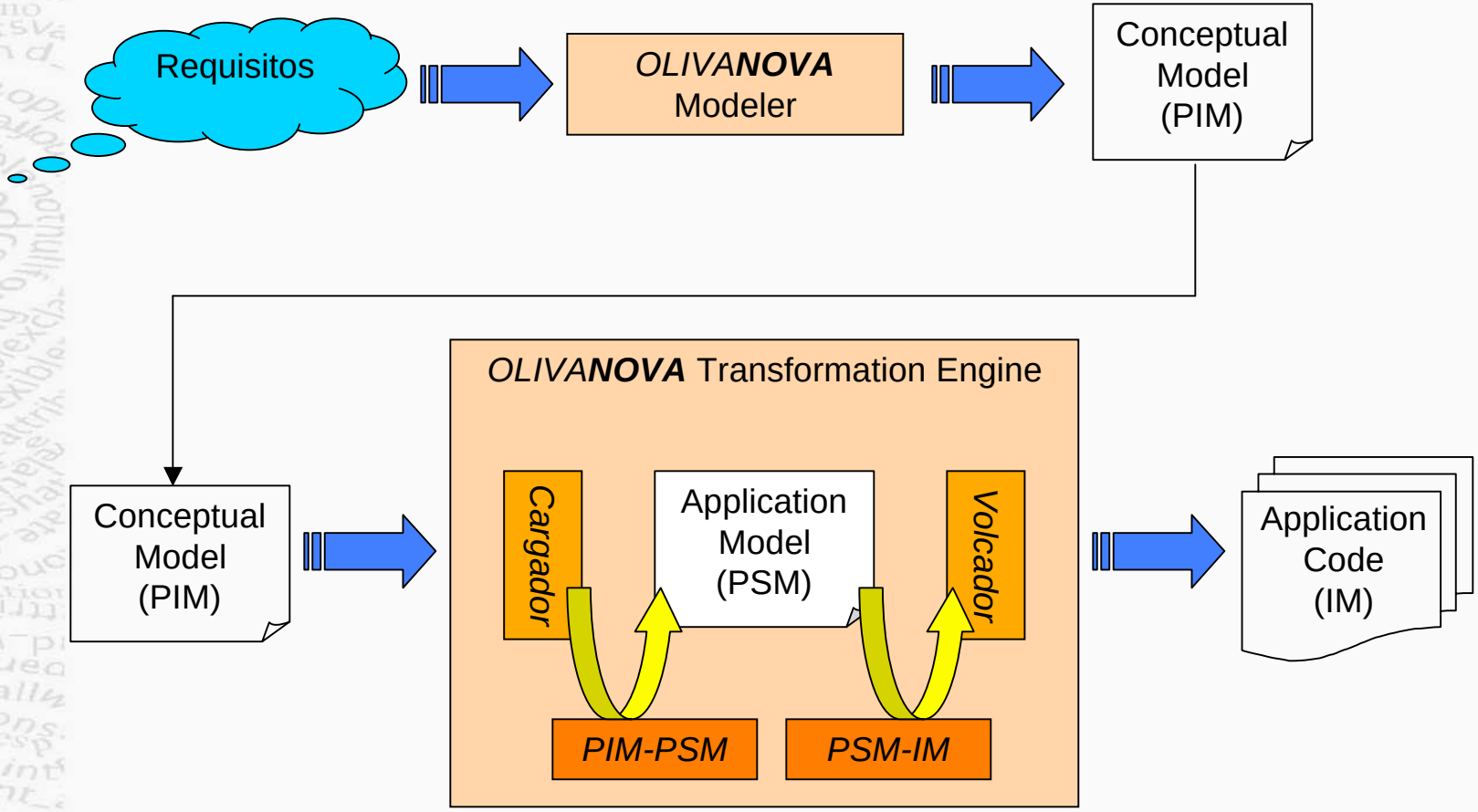
- Un módulo del transformation engine implementa las transformaciones de PIM a PSM para obtener el Application Model (PSM)

*OLIVA**NOVA*** **EN ACCIÓN**



- Otro módulo implementa las transformaciones de PSM a IM para obtener la aplicación ...

*OLIVA**NOVA*** **EN ACCIÓN**

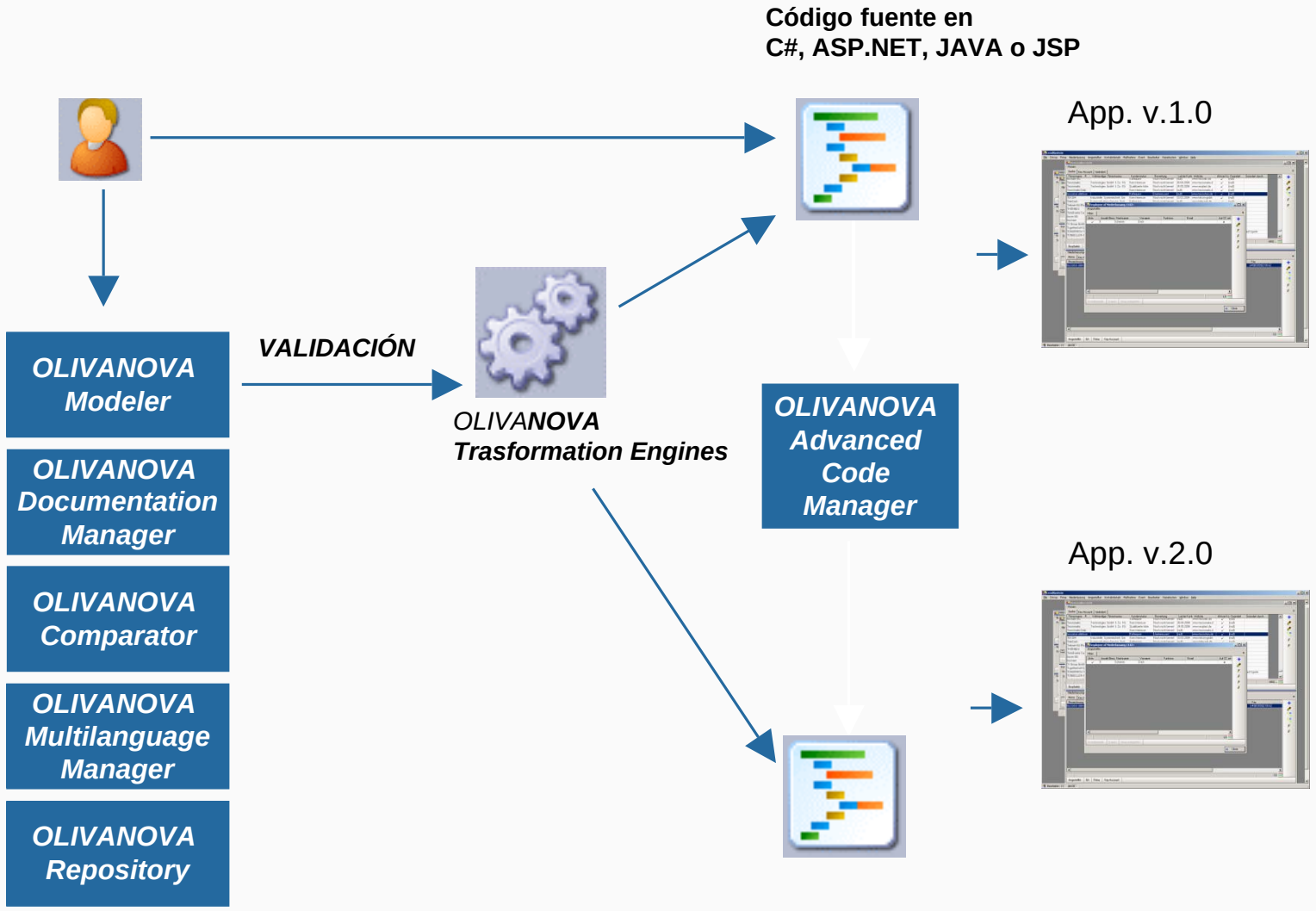


- ... cuyo código es volcado a ficheros.

*OLIVA**NOVA*** **ES MDA**

- *OLIVA**NOVA*** Modeler le permite crear, editar y validar sus PIMs
- *OLIVA**NOVA*** le ofrece Transformation Engines para diferentes PSMs que:
  - Convierten su PIM en un PSM automáticamente
  - Convierten su PSM en código automáticamente

*OLIVA**NOVA*** **PRODUCTOS (1)**



*OLIVA**NOVA*** **PRODUCTOS (2)**

- Los modelos pasan de artefactos de diseño a ser artefactos de desarrollo.
  - Con *OLIVA**NOVA Modeler*** el modelo **ES** el “programa”.
- Basado en Lenguaje de especificación Formal
  - *OLIVA**NOVA Modeler*** valida la corrección y completitud de los modelos eliminando la ambigüedad e impidiendo la contradicción.
- Importación de modelos de otras herramientas
- Trabajo cooperativo y biblioteca de modelos: *OLIVA**NOVA Repository***
- Documentación automática y completa: *OLIVA**NOVA Documentation Manager***

OLIVANOVA PRODUCTOS (3)

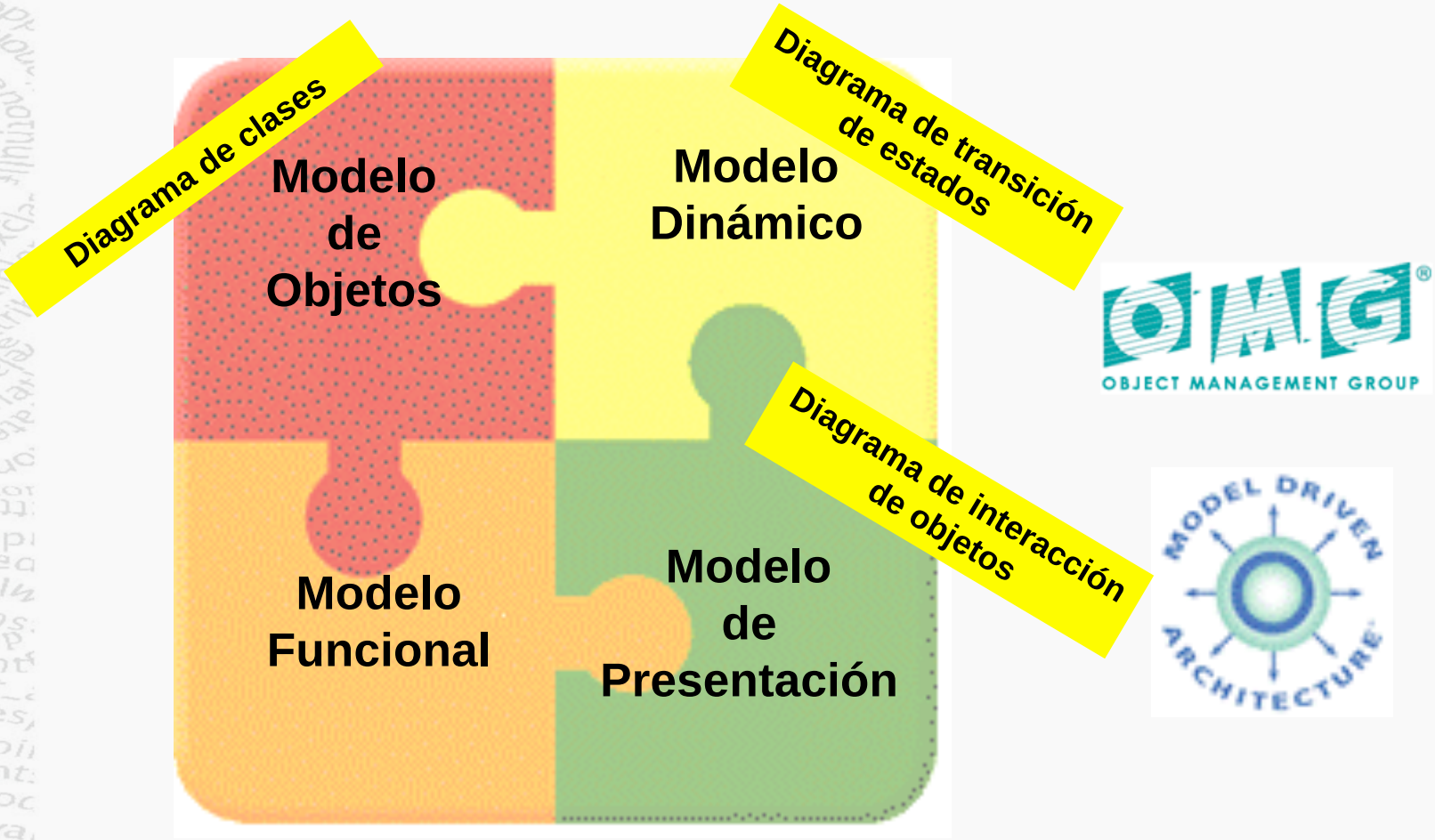
- ***OLIVANOVA* Modeler**
  - Herramienta de modelado para definir el modelo
  - Valida la corrección y completitud de los modelos eliminando la ambigüedad e impidiendo la contradicción
- ***OLIVANOVA* STAR Client**
  - Herramienta utilizada para enviar el modelo al *OLIVANOVA* Transformation Engines
- ***OLIVANOVA* Comparator**
  - Detecta las diferencias entre dos modelos a distintos niveles
- ***OLIVANOVA* Documentation Manager**
  - Genera documentación automática y completa
- ***OLIVANOVA* XMI Exporter**
  - Importación y Exportación de modelos de otras herramientas
- ***OLIVANOVA* Repository**
  - Facilita el trabajo cooperativo y contiene una biblioteca de modelos
- ***OLIVANOVA* Advance Code Management**
  - Registra y aplica los cambios al nuevo fuente para obtener la versión actualizada de la aplicación

MODELO CONCEPTUAL *OLIVANOVA* (1)

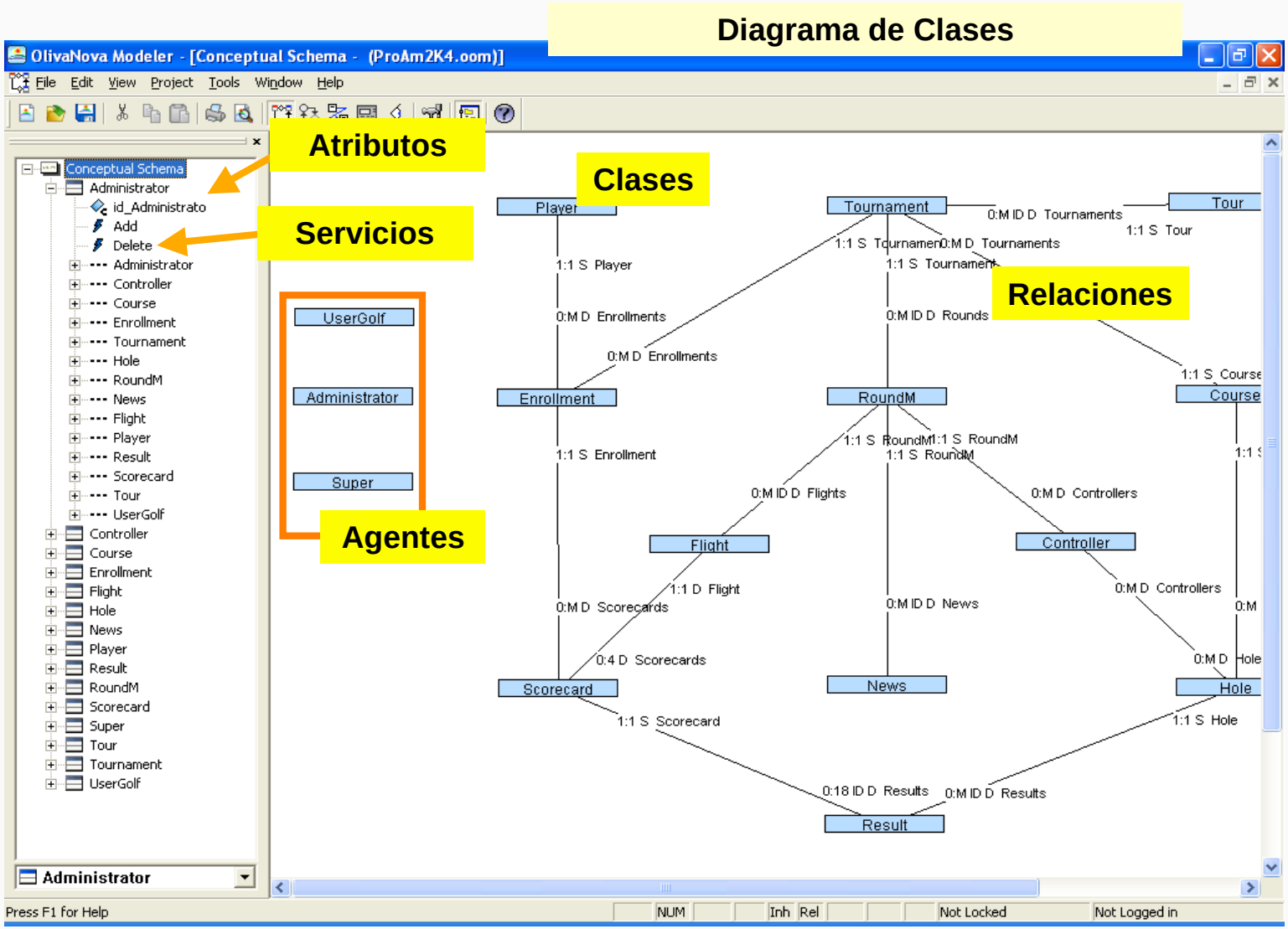
- Representación **correcta**, **completa** y **no ambigua** de los procesos de negocio.
  - Permite la generación de descripciones del sistema semánticamente ricas.
  - Los modelos se vuelcan en formato XML

MODELO CONCEPTUAL
*OLIVA**NOVA*** **(2)**

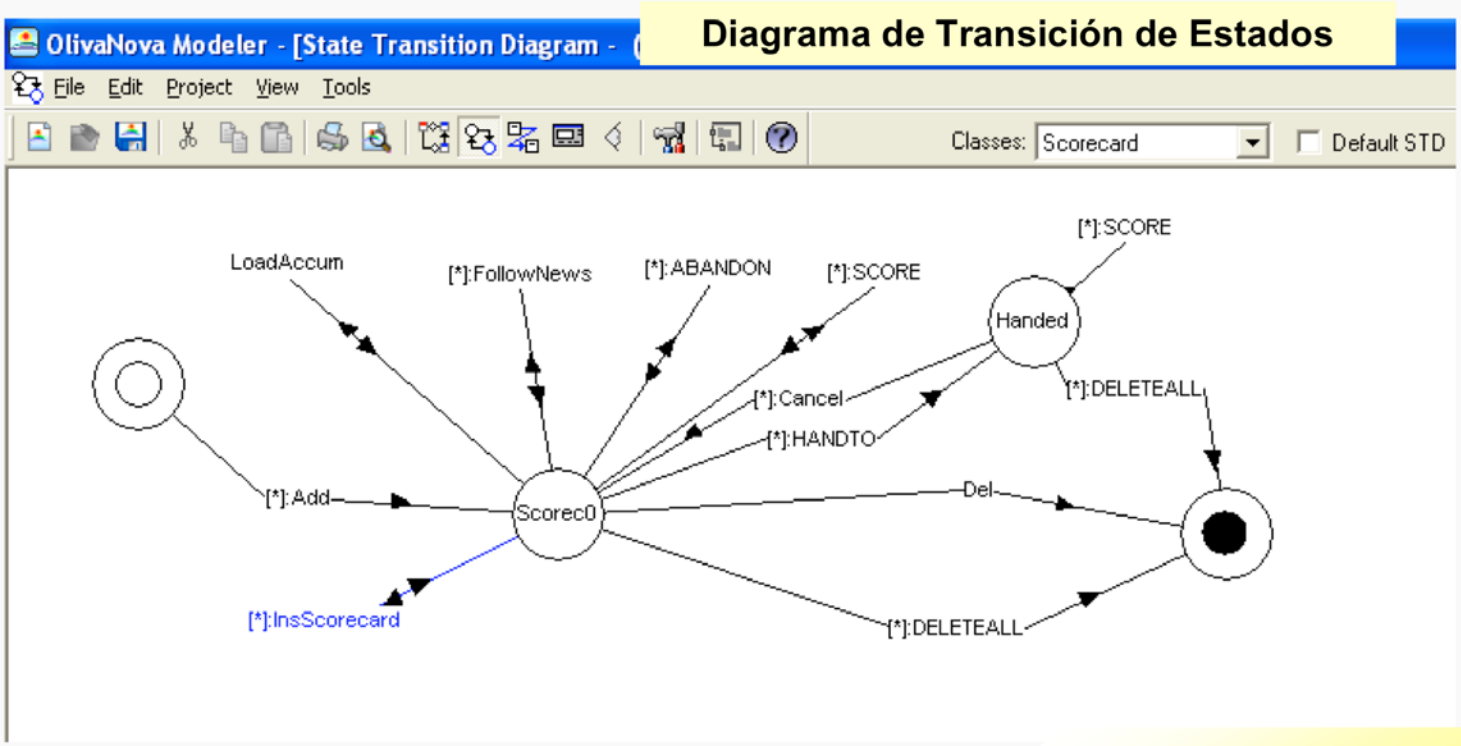
*Contiene **todos** los requisitos funcionales del cliente*



MODELO DE OBJETOS



MODELO DINÁMICO

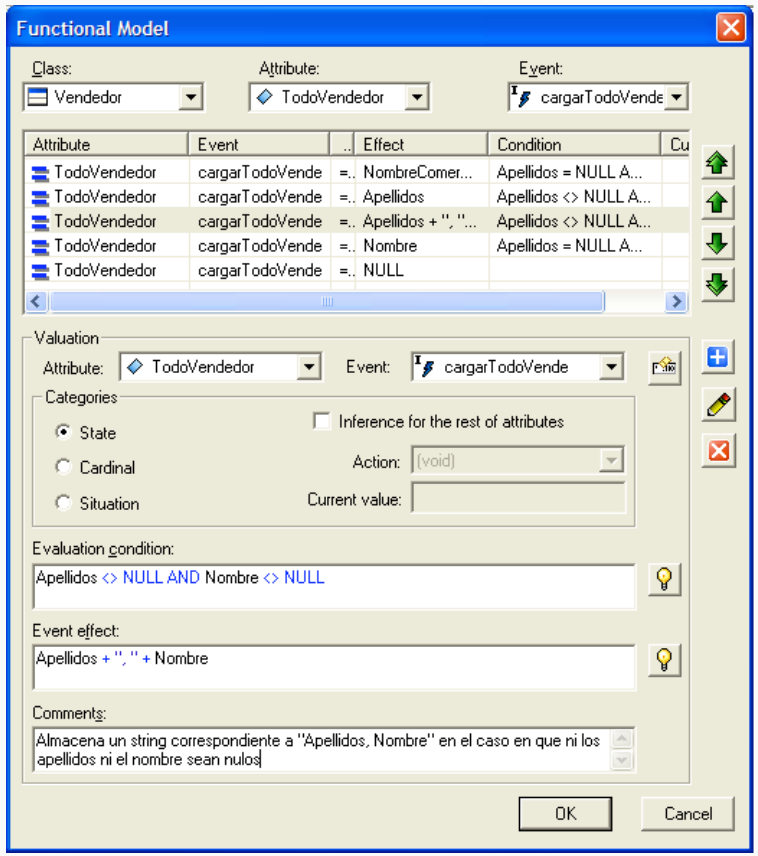


Describe los estados validos en la vida de un objeto

MODELO FUNCIONAL

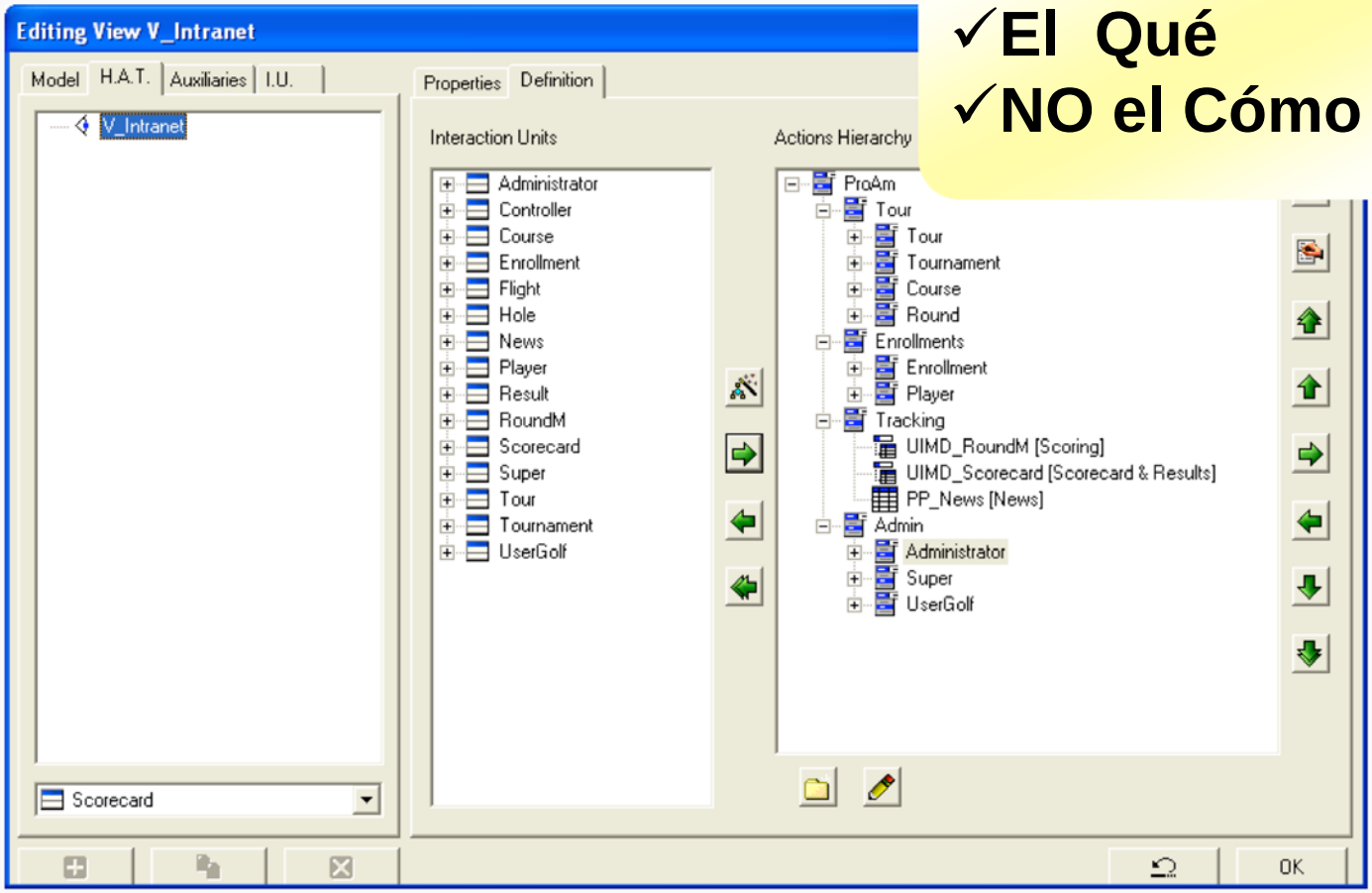
*Especifica la relación entre la **estática** y la **dinámica***

- Recoge la semántica relacionada con los cambios de estado
- Describe cómo la ejecución de eventos cambian el valor de los atributos de las clases



MODELO DE PRESENTACIÓN

*Especificación Abstracta de la **Interfaz de Usuario***

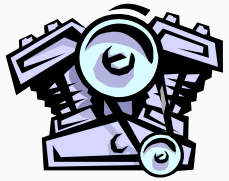


✓El Qué
✓NO el Cómo

VALIDACIÓN FORMAL

- Los Modelos Conceptuales tienen que ser validados para verificar que son **correctos, completos y no ambiguos**
- La pantalla de validación proporciona una **validación formal y completa** de los modelos conceptuales
- Una vez validado el modelo, puede ser guardado en un fichero con formato XML siguiendo un DTD específico que servirá de entrada para otras herramientas *OLIVA**NOVA***

TRANSFORMATION ENGINES



- *OLIVA**NOVA*** **Transformation Engines** crea de forma automática aplicaciones en código fuente listo para compilar e instalar.
- Se obtiene aplicaciones de 3 capas para múltiples arquitecturas

Capa Presentación

Arquitectura Cliente Pesado:
- **Windows-Forms VB**
- **Windows Forms .NET / C#**

Arquitectura Web:
- **JSP**
- **ASP .NET**

Capa Lógica Negocio

Arquitectura Transaccional:
- **COM+ / VB**
- **.NET / C#**

Arquitectura Objetos en memoria:
- **EJB / Java**
  - **WebSphere**
  - **WebLogic**
  - **JBOSS**
  - **Oracle iAS**

Capa Persistencia

Bases de Datos Relacionales:
- **Oracle**
- **SQL Server**
- **DB2**
- **MySQL**

Caso práctico

*OLIVA**NOVA***
La máquina de Programar