Transformaciones de modelos: yendo de UML a un DSL y viceversa. El caso de FlexAB.

Andrade Francisco y Mancino Diego

Temario

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

Temario

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

Motivación (MDD)

- MDD se ha convertido en un nuevo paradigma de desarrollo software.
- MDD promete mejorar el proceso de construcción de software basándose en un proceso guiado por modelos.
- Asigna a los modelos un rol central y activo.
- La transformación entre modelos constituye el motor de MDD.

Motivación (DSM y DSL)

- Modelos para un dominio con lenguajes especializados.
- Los productos finales son generados desde estas especificaciones de alto nivel.
- Dominios pequeños.

Motivación (FlexAB)

- La herramienta FlexAB permite generar aplicaciones para el manejo de información con orientación a objetos, de una manera sencilla, sin programación, y sólo configurando las reglas entre los objetos que intervienen dentro del proceso.

Motivación (UML)

- UML ventajas:
  - Ambiente de modelado sin costo de implementación y diseño.
  - Estándar abierto y de facto para modelado de software.
  - Durable.
  - Conocido por la mayoría de las personas relacionadas a la ingeniería de software.
  - Es soportado por muchas herramientas maduras.

Motivación

- Dadas las ventajas de UML con respecto a FlexAB seria deseable contar con la posibilidad de realizar una traducción automática entre modelos UML y modelos de FlexAB y viceversa.

Objetivos

- Implementar una herramienta que automatice las transformaciones entre los correspondientes elementos de modelado de UML y FlexAB.

Temario

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

Definición de lenguajes de modelado

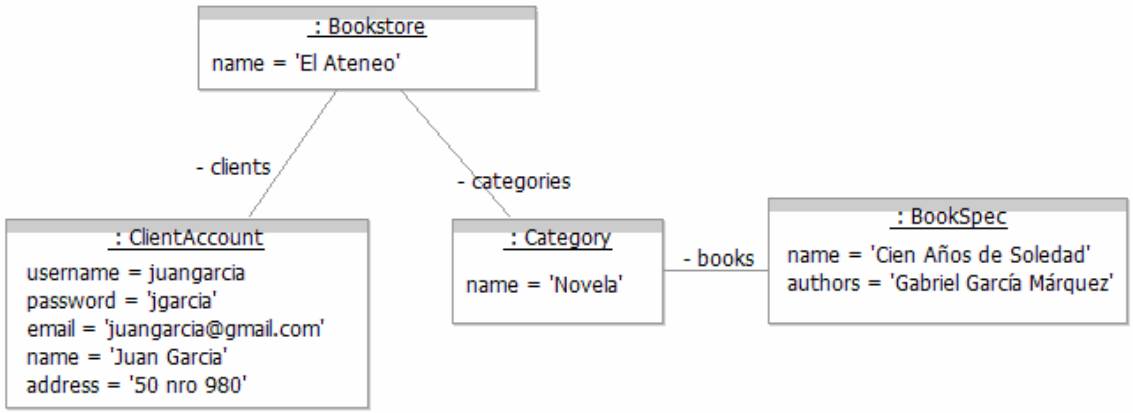
- Backus Naur Form (BNF) era la meta sintaxis usada para definir lenguajes, mediante reglas de derivación.
- En los últimos años se desarrollaron técnicas específicas para facilitar la definición de los lenguajes gráficos, llamada “metamodelado”.

Arquitectura de 4 capas de OMG

- La Arquitectura de cuatro capas de modelado es la propuesta del OMG orientada a estandarizar conceptos relacionados al modelado, desde los más abstractos a los más concretos.

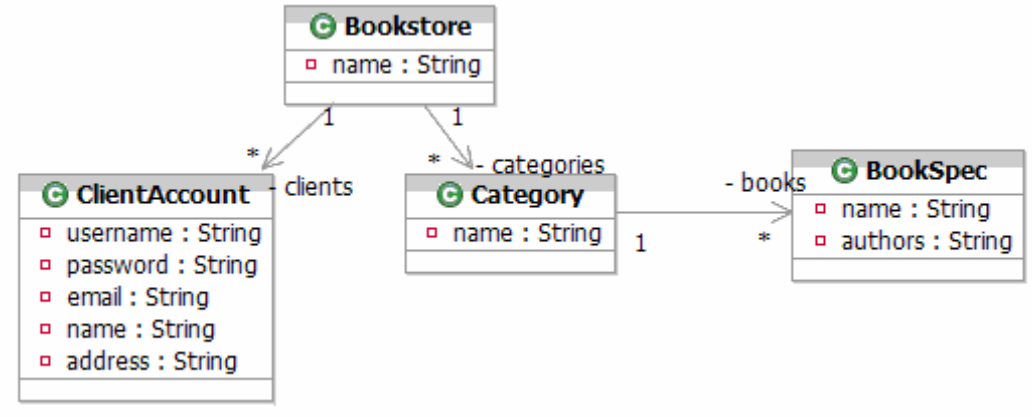
Arquitectura de 4 capas de OMG

- Nivel M0: Instancias “reales” del sistema, objetos de aplicación.



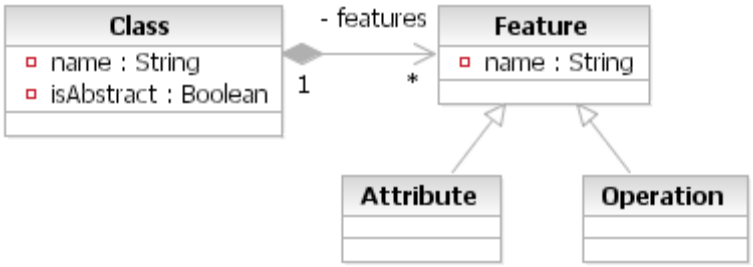
Arquitectura de 4 capas de OMG

- Nivel M1: Modelo de un sistema de software. Cada elemento de M0 es una instancia de un elemento de M1.



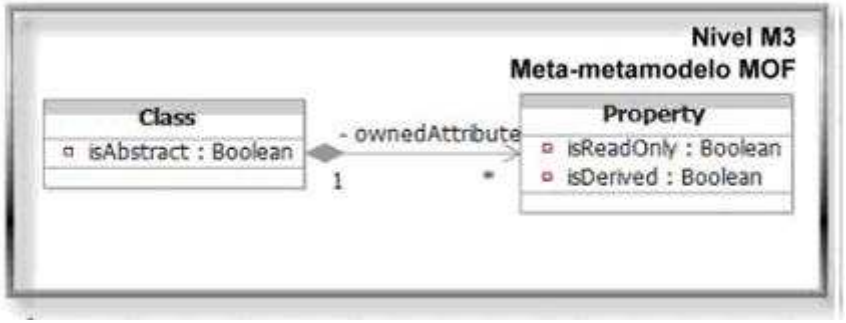
Arquitectura de 4 capas de OMG

- Nivel M2: Esta capa recibe el nombre de metamodelo. Los elementos del nivel M1 son instancias del nivel M2.



Arquitectura de 4 capas de OMG

- Nivel M3: Capa de meta-metamodelo. Un meta-metamodelo es un modelo que define el lenguaje para representar un metamodelo. Los elementos del nivel M2 son instancias del nivel M3.



Metamodelado

- Definir lenguajes de modelado sin ambigüedades.
- Las reglas de transformación se definen usando los metamodelos de los lenguajes fuentes y destino.
- Sintaxis de los lenguajes en los cuales se expresan las reglas de transformación.

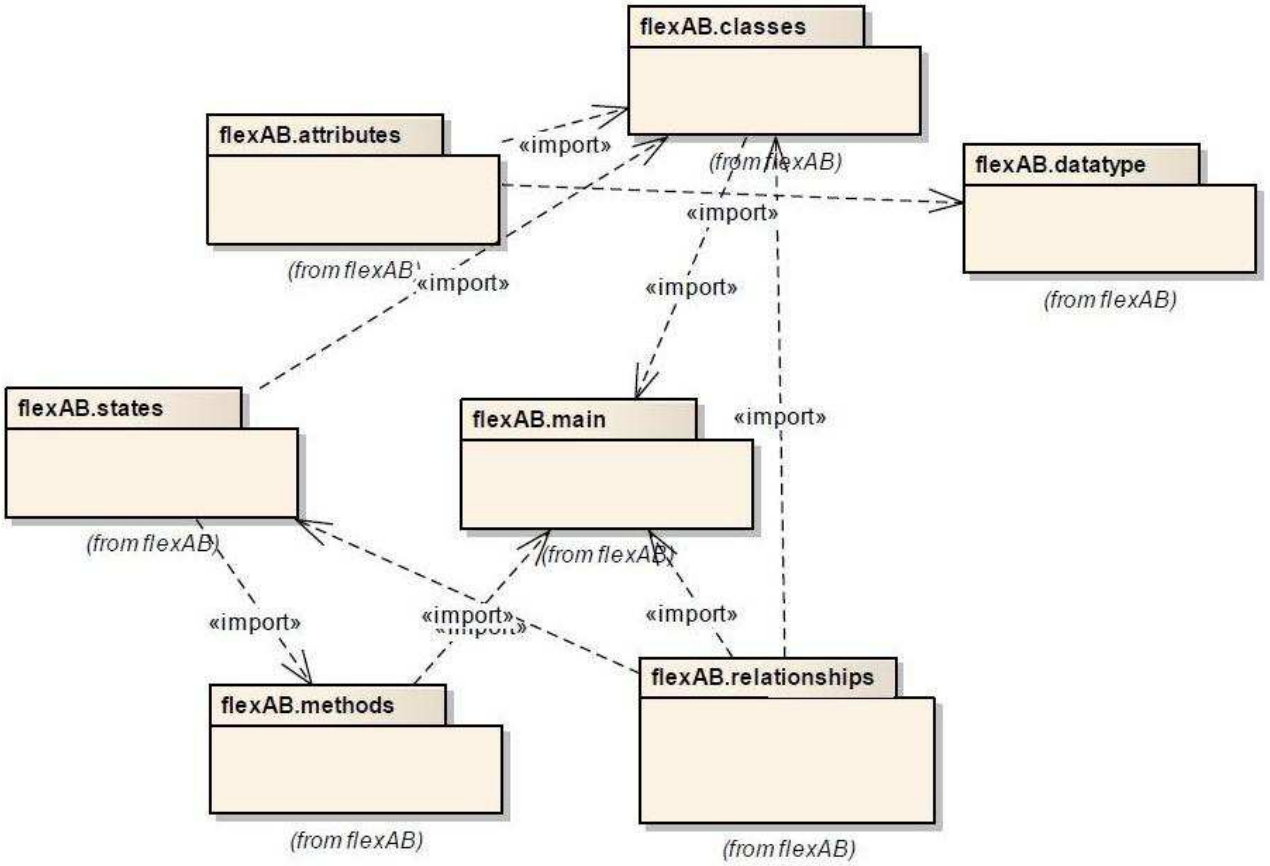
El lenguaje de modelado FlexAB

- FlexAB es una herramienta para la creación de aplicaciones sin programación, con la flexibilidad de ser orientada a objetos, y utilizando robustas bases de datos relacionales.
- El desarrollador interactúa solo con los objetos, los datos y su estructura relacional son transparentes, haciendo que la creación de aplicaciones tenga una reducción en su tiempo de desarrollo.
- FlexAB es una combinación de una herramienta que brinda el marco para desarrollar una aplicación totalmente a medida, pero sobre una plataforma que no requiere programación y por lo tanto estandariza la creación de software.

Como funciona FlexAB

- Definir objetos relacionados con los procesos que necesitamos informatizar, y como interactúan estos objetos con su entorno sin importarnos como es su almacenamiento.
- No es necesaria la generación de código para el manejo de objetos, cada uno tiene la capacidad de definir como interactúan sus propios datos, y como interactúa él con los demás objetos mediante funciones FQL.
- De manera similar se cargan las validaciones de integridad de cada objeto.
- Permite la configuración de mensajes y la ejecución de programas externos mediante la utilización de las funciones COM.

Metamodelo de FlexAB



Transformaciones

- Una regla de transformación es una descripción de como una o más construcciones en el lenguaje fuente pueden ser transformadas en una o más construcciones en el lenguaje destino.
- Una transformación será aplicada muchas veces, independientemente del modelo fuente al que será aplicada.
- Para simplificar la tarea de codificar transformaciones se han desarrollado lenguajes de más alto nivel específicos del dominio de las transformaciones como ATL y QVT.

Temario

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

Marco tecnológico

- Eclipse.
- EMF (Eclipse Modeling Framework).
- Transformaciones ATL.

Temario

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

Solución propuesta

- **Arquitectura:** La arquitectura consta de un proyecto Java que contendrá todo lo relacionados a los modelos, diagramas, instancias de los modelos y código generado a partir de los modelos y otro proyecto ATL que contendrá las transformaciones. El proyecto ATL consume datos y vuelca los datos producidos en el proyecto Java.
- **Transformación FlexAB a UML:** En esta transformación se tomará como entrada un conjunto de instancias de las clases pertenecientes al modelo de FlexAB y se obtendrán como salida las instancias del modelo de UML correspondientes.

Solución propuesta

- **Transformación UML a FlexAB:** En esta transformación se tomará como entrada un conjunto de instancias de las clases pertenecientes al modelo de UML y se obtendrán como salida las instancias del modelo de FlexAB correspondientes.

Temario

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

Implementación de la solución

- Proyecto Java.

Implementación de la solución

- Proyecto ATL

Temario

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

Casos de estudio

- Ver video

Temario

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

Conclusiones

- Se ha desarrollado un software complejo que logra la transformación bi-direccional de ambos lenguajes (FlexAB y UML).
- El resultado es una herramienta interesante que permite cambiar el enfoque del modelador de acuerdo a sus preferencias y/o habilidades.
- Las transformaciones permiten la migración de modelos escritos en FlexAB para poder presentárselos a modeladores que no conocen este lenguaje y si poseen conocimiento de UML.

Conclusiones

- En ciertas ocasiones, es beneficioso tener el modelo en UML porque muchas de las herramientas utilizadas para el modelado soportan el lenguaje.
- Una persona con conocimientos en UML podría trabajar con el modelo en dicho lenguaje y una vez realizadas las modificaciones se puede transformar a FlexAB.

Trabajos futuros

- Completar el análisis de correspondencia de las clases de FlexAB faltantes con sus correspondientes en UML.
- Extender la herramienta para las clases que quedaron sin mapear.
- Mejorar usabilidad de la herramienta implementada.
- Adaptar alguna herramienta para la instanciación de las clases de los distintos modelos.