Arquitecturas Orientadas por Modelos y Lenguajes Específicos de Dominio

José Mauricio Alvarez H.

Mauricio.Alvarez@Microsoft.com

http://blogs.msdn.microsoft/mauricioalvarez

Arquitecto Soluciones, Microsoft

Tel. 3264709

Bogotá

Objetivos y conclusiones

- Objetivos
  - Explicar que es modelamiento
  - Dar un poco de historia y el estado actual
- El modelamiento ha estado alrededor siempre
- Típicamente se ha enfocado en abstraer la aplicación y de alguna manera como se engancha con la plataforma
- La idea ahora es elevar el nivel de abstracción de la plataforma para disminuir el ***gap*** de abstracción.
- Los lenguajes específicos de dominio son una alternativa para cubrir el ***gap***.

Que no aprenderá

- Productos específicos o características jugosas
- Como ser un excelente modelador
- Sintaxis/meta-modelos
- Detalles de herramientas especificas de modelamiento

Qué es un modelo?

(Del it. *modello*).

1. m. Arquetipo o punto de referencia para imitarlo o reproducirlo.

2. m. En las obras de ingenio y en las acciones morales, ejemplar que por su perfección se debe seguir e imitar.

3. m. Representación en pequeño de alguna cosa.

4. m. Esquema teórico, generalmente en forma matemática, de un sistema o de una realidad compleja, como la evolución económica de un país, que se elabora para facilitar su comprensión y el estudio de su comportamiento.

5. m. Objeto, aparato, construcción, etc., o conjunto de ellos realizados con arreglo a un mismo diseño. *Auto modelo 1976. Lavadora último modelo.*

6. m. Vestido con características únicas, creado por determinado modista, y, en general, cualquier prenda de vestir que esté de moda.

7. m. En empresas, u. en aposición para indicar que lo designado por el nombre anterior ha sido creado como ejemplar o se considera que puede serlo. *Empresa modelo. Granjas modelo.*

8. m. *Esc.* Figura de barro, yeso o cera, que se ha de reproducir en madera, mármol o metal.

9. m. *Cuba.* **impreso** (? hoja con espacios en blanco).

10. com. Persona de buena figura que en las tiendas de modas se pone los vestidos, trajes y otras prendas para que las vean los clientes.

11. com. *Esc.* y *Pint.* Persona u objeto que copia el artista.

~ vivo.

1. com. Persona, por lo común desnuda, que sirve para el estudio en el dibujo.

Diccionario de la Lengua Española, XXII ed.
http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=modelo

Hmmm, OK, pero en la vida real

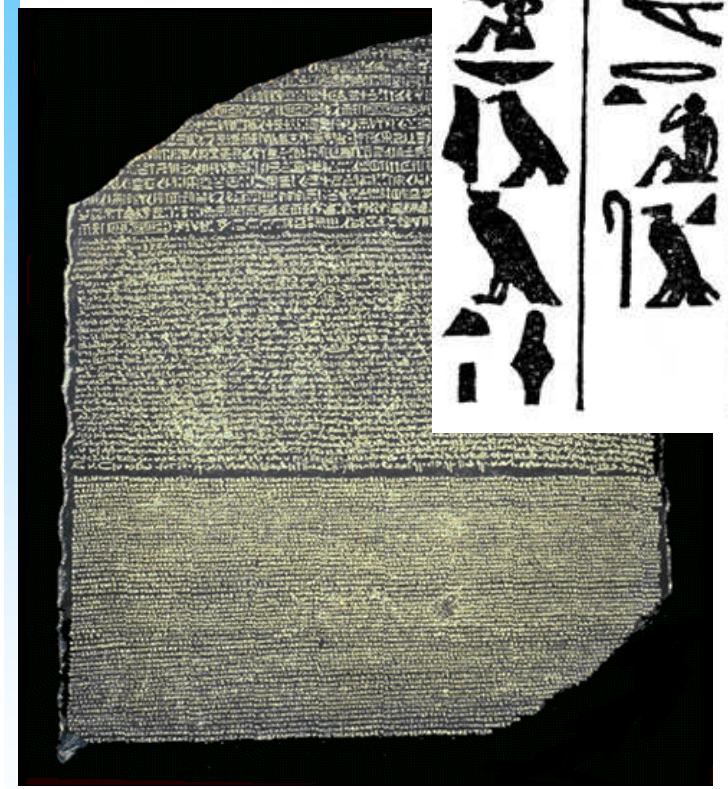
- En computación, generalmente significa:

Una representación de un (aspecto de un) programa en una manera mas abstracta.

Clarificación de modelo

- No son solo para código
- Pueden representar cualquier cosa:
  - Requerimientos
  - Casos de Uso
  - Arquitectura
  - Protocolos
  - Flujos de Proceso
  - Transiciones de estado
  - Topologías de distribución
  - Características/parámetros operacionales
  - La plataforma
  - … Cualquier cosa

Orígenes del Modelamiento… Parte 1



Orígenes del Modelamiento… Parte 2

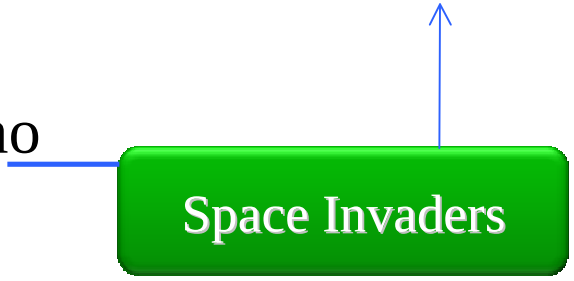
```
00110010 10001110 10101010 11110001
32H 4AH 00H …

MOV AH, 46h; XOR AH, AL

int c = 32; ++c;
class model { int c; int incC (void) { return ++c;} }

Select * from models where type = ‘useful’

[ServiceContract] … [OperationContract] …

<XML>”Is Often Cited”</XML>
```

Por qué Tenemos Modelos?

- El principio esencial es hacer las cosas mas sencillas:
  - Arquitecto, quien puede expresar componentes y conexiones mediante diagramas
  - Desarrollador, quien tiene construcciones de más alto nivel para hacer mas en menos palabras.
  - IT Pro, quien puede conocer mas acerca de los requerimientos operacionales de una aplicación
  - Mantenimiento/Debugger, quien puede mirar un programa y que realmente haga sentido en un tiempo mas corto.
  - Analista de Negocio, quien puede expresar conceptos de negocio de una manera que “ambos lados” puedan entender
  - Herramientas de modelamiento, que pueden ejecutar análisis mas completos de un sistema
- Por supuesto, esto es solamente un ***principio*** …

Qué Significa “Declarativo” y Por qué es Bueno?

- La programación declarativa se diferencia de la imperativa enfocándose en el ***que*** antes que en el ***como***
- Es un nivel de abstracción al cual mucha mas gente puede referenciar
  - Aumentando la audiencia de “programadores”
- Puede ser mas fácil de escribir, leer y mantener
- Es mas fácil para que las herramientas lo analicen
- No implica una representación
  - <SpaceInvaders> es tan declarativo como



Algunas Desventajas del Modelamiento

- Usado para documentar, no ***Diseñar***
  - Muchos sabores
    - Usado al inicio en diseño, pero se des actualiza
    - Usado para diseñar y generar código, pero se modifica el código y el diseño no representa la realidad
    - Usado por algunos pero no todos
    - Etc., etc., etc.
- No hay suficiente fidelidad, (demasiado genérico)
  - No todos los conceptos son capturados
  - Se fuerza a la separación de elementos específicos de la plataforma
- Asunto de generación de código
- Asunto del meta meta meta

Asunto de Generación de Código

- El modelo abstrae la aplicación
- Asume una “plataforma virtual” – Una que no existe
- Las herramientas generan cantidades alarmantes de código para llenar el gap de abstracción
- Este código puede ser extremadamente difícil de:
  - Entender
  - Mantener
  - Depurar
  - Optimizar
  - Personalizar
- No cambie el código
  - Difícil de sincronizar – quien manda a quien?

Meta Death

- Hablemos de meta modelos …
- Si un esquema SQL es un Modelo, entonces el Meta-Modelo SQL dice que puede ir en un esquema SQL – este es el esquema esquema de SQL
- El problema es que dice lo que puede ir en el Meta Modelo de SQL? El esquema esquema esquema de SQL?
- Esto puede parar?

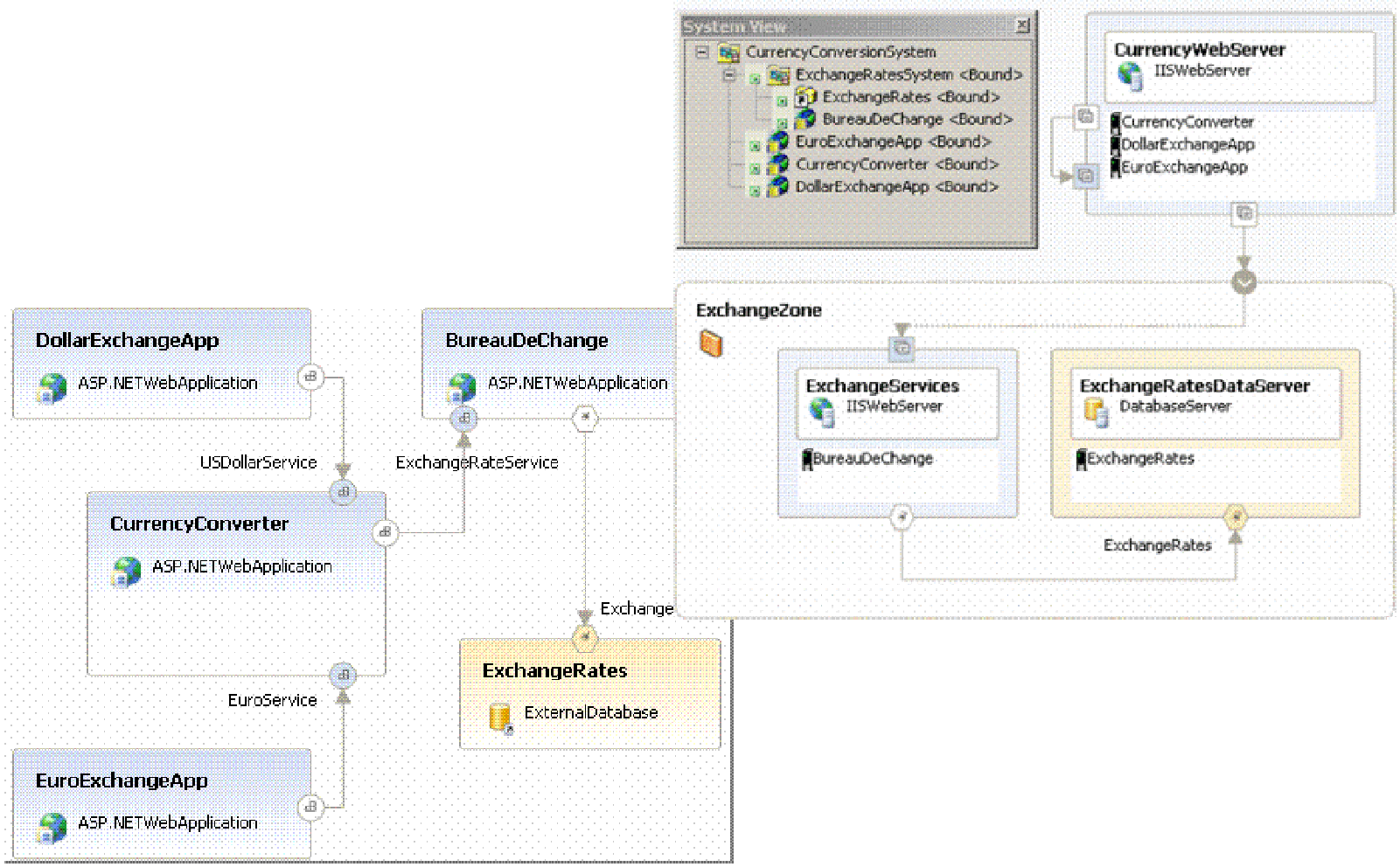
Algunos Alivios

- "Solamente usado para documentar"
  - Haga que los modelos sean artefactos de primera clase, como código fuente
  - Esto significa que escribir modelos, *es* escribir aplicaciones
- "No hay suficiente fidelidad"
  - Cree lenguajes de modelamiento para su propósito especifico
  - Lenguajes Específicos de Dominio (DSLs)
- "El asunto de generación de código"
  - No genere código?
  - Lleve la plataforma hasta el modelo
    - En lugar del revés

Cómo son los modelos?

- Dibujos y diagramas
  - Cajas y líneas
  - Usan notación comúnmente aceptada
- Representaciones textuales
  - XSD para XML
  - XMI para UML
  - XML para configuración de aplicaciones
  - C# para IL
  - Java para Bytecode
- Otros
  - Si, algunos modelos son binarios

Un ejemplo


System View
CurrencyConversionSystem
ExchangeRatesSystem <Bound>
ExchangeRates <Bound>
BureauDeChange <Bound>
EuroExchangeApp <Bound>
CurrencyConverter <Bound>
DollarExchangeApp <Bound>
CurrencyWebServer
IISWebServer
CurrencyConverter
DollarExchangeApp
EuroExchangeApp
ExchangeZone
ExchangeServices
IISWebServer
BureauDeChange
ExchangeRatesDataServer
DatabaseServer
ExchangeRates
ExchangeRates
DollarExchangeApp
ASP.NETWebApplication
BureauDeChange
ASP.NETWebApplication
USDollarService
ExchangeRateService
CurrencyConverter
ASP.NETWebApplication
Exchange
EuroService
ExchangeRates
ExternalDatabase
EuroExchangeApp
ASP.NETWebApplication

UML®

- Significa Unified Modeling Language™
  - OMG dice: “Una especificación que define un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema distribuido de Objetos”
  - Piense como una manera estandarizada de expresar modelos

MDA®/MDD®

MDA – Model-Driven Architecture

MDD – Model-Driven Development

- Ambos son marcas de la OMG
- Una aproximación para permitir que las aplicaciones sean definidas a través de modelos de manera independiente de la plataforma
  - Con un modelo especial que mapea a cada sistema sobre el cual será implementado

Qué es un DSL?

- DSL = Domain Specific Languaje, Lenguaje Especifico de Dominio
  - Es un lenguaje pequeño, altamente focalizado para resolver problemas claramente identificados
  - SQL es un DSL
  - De muchas maneras, DSLs son Lenguajes de propósito especifico, pero que usan un paradigma de programación

Qué es una Fabrica de Software (MS)?

- Un conjunto de DSLs para una variedad de dominios del ciclo de Vida del Desarrollo de Software (Arquitectura, Desarrollo, Distribución, etc.)
- Otras herramientas, wizards, generadores de código y asi sucesivamente en la medida en que sean requeridas.
- Un mecanismo unificado sobre todo lo anterior
- Un ejemplo es “Services Factory”
  - Genera código para todos contratos y artefactos a partir de los diagramas
  - DSL defines las posibilidades del contrato

CASE

- Computer-Aided Softwate Engineering
  - Lo más espectacular de los 80’s
  - No fue del todo exitoso
- Por qué CASE no funciono?
  - Demasiado ambicioso, apuntaba a reemplazar a los desarrolladores
  - Motores de generación de código inmensos
  - El nivel de abstracción llevaba a código complejo, difícil de entender, no depurable y leeeeeeeeeeeeeeento
  - Estándares que competían con poca interoperabilidad
- Termino siendo demasiado restringido para ser efectivo

Soy Desarrollador, me importa?

- Me quede sin trabajo?
- De todas maneras los modelos son para Gerentes de Proyecto/Arquitectos/<Tu “time waster” favorito>
- Yo solo escribo código
- El modelamiento no es Agile?
- Debo confiar en el código generado?
- Etc.

- El modelamiento puede no cambiar su vida, pero es seguro apostar que si.

Soy Arquitecto, me importa?

- Yo dibujo los diagramas del sistema, pero los desarrolladores los guardan y solo escriben código.
- Aconsejo en decisiones de plataforma, pero no tengo herramientas para asegurar la certeza de mi recomendación
- Escucho a los analista de negocio, tratando de hacer sentido de lo que dices
- Me esta “escuchando”, pero no tiene la mas remota idea del significado de lo que estoy diciendo
- Sigo usando modelos, a pesar de que me “queme” con CASE

Soy IT Pro, me importa?

- Estos %$%&(@ desarrolladores – me tiran sus sistemas debajo de la puerta y esperan que “trabajen”
- Como se como distribuirlos?
- Qué sucederá cuando una nueva instalación entre en línea?
- Pueden realmente los modelos ayudarme?

No existe un único modelo

- Una aplicación no puede ser definida en un único modelo
- Existe un conjunto interrelacionado:
  - Requerimientos
  - Procesos de Negocio
  - Arquitectura
  - Diseño y Desarrollo
  - Distribución
  - Operaciones
  - Monitoreo y Feedback
- Para ser convenientes estos modelos deben ser todos parte de la definición de la aplicación- creando estos modelos *es* crear la aplicación

Ciclo de Vida

- Si los modelos son pervasivos al desarrollo de aplicaciones, entonces necesitamos un conjunto de herramientas que puedan crear y manipular modelos para cada etapa de la vida de una aplicación
  - En software Factories, son “view points”
- El imperativo acá es asegurar que cualquier cambio a cualquier modelo u otro artefacto, sea apropiadamente reflejado en todos los modelos o artefactos afectados
- Los modelos no son solamente artefactos de tiempo de diseño
  - Un sistema verdaderamente dirigido por modelos usara modelos en tiempo de ejecución para determinar, por ejemplo, si un sistema esta ejecutándose dentro de los limites esperados