Presentation at "MDA Information Day"
during the OMG Technical Meeting in April 2002

MDA and System Design

Makoto Oya
Hitachi, Ltd.
April 23, 2002
Activities in OMG


Standardization of Distributed Object Middleware
- 1995: CORBA2; 2001: CORBA2.5

Domain (industry specific & cross-industry) Standardization
- 1995-: Standards in various domains

Modeling Standardization
- 1997: UML (Unified Modeling Language)
- 2001: Application-specific UML Profiles (EDOC, EAI)

Architecture (Reference Model)
- 1990: OMA (Object Management Architecture)
- 2001: MDA (Model Driven Architecture)

2001-: starting Standardization based on MDA

2002(planned): UML V2 --- expected to include MDA base functionality
Agenda

- Background and Vision
- MDA's Approach
- Toward Realization of MDA
- Summary
Background and Vision
Integration of Business Processes

- Sales
- Engineering
- Manufacturing
- Inventory
- Accounting
- Payables/Receivables

- Improved steadily for more than 20 years.
- However, still existing big challenges.
Root of Problems = Varieties of Platforms

Variety of Hardware Architectures
- Pentium, PowerPC, PA-RISC, Sparc, 370, …

Variety of Networks
- Ethernet, ATM, IP, SS7, Appletalk, USB, Firewire, …

Variety of Programming Languages
- C/C++, Java, Visual Basic, C#, Perl, JavaScript, VBScript, COBOL, PL/I, Fortran, …

Variety of Operating Systems
- Unix, Windows, NT/XP, Mainframe OS, MacOS, Windows CE, Mobile phone, Set-top box, Game machine, …

Then, Variety of Middlewares
- JAVA/CORBA, COM+/.NET, Web Services(SOAP, ebXML, …)

(based on Andrew Watson's presentation)
Standardization and popularization of middleware have solved integration problems across different hardware architectures and operating systems.

Newly arising challenges: Mixture of Middlewares CORBA, Java, COM+, various Web Services, .NET, …

Success, Evolution and Proliferation of Middleware

Total Integration

Cross-Middleware - Integration - System Design
Challenge:
Integration across Middlewares

Let us consider about ebXML Web Services as an example of middleware.

(Note) from ebXML “Technical Architecture Specification” (v1.0.4)
Integration across Middlewares

New development?
No. Integration to connect Web Service and existing enterprise system.

Integration Point between Web Services and other IT world

- New development?
- No. Integration to connect Web Service and existing enterprise system.

Makoto Oya
(based on Andrew Watson's presentation)
This means integration to connect Web service and in-house middleware environment is needed.

- ebXML, SOAP, .NET, EDI etc.
- EJB/CORBA, C++/CORBA, DCOM etc.
- HTML, XML, JavaScript, CSS etc.
- ODBC, CORBA/IIOP etc.

(based on Andrew Watson's presentation)
The Other Challenge:
System Design across Middlewares

Essentially the same system, but …

System design, at least, should be done commonly, but, …
MDA Vision

- Cannot avoid co-existence of plural (middleware) platforms

- MDA: Model-Driven Architecture

- Platform independent system design
  - described using UML (Unified Modeling Language) in general
  - called as PIM (Platform Independent Model)

- From PIM, system design for each platform is driven
  - called as PSM (Platform Specific Model)

- From PSM, actual skeleton of codes is driven

Note: What is a "Model"?
Notes: What is a "Model"?

"Model" mentioned here means?
- Something showing concepts
- Scale-downed description or presentation
- Existing thing presenting some characteristics to design a new thing
- System Design --- precisely speaking: Design documents/information to create actual systems

UML (Unified Modeling Language)
- Standardized notation to describe system design
  - Logical module structure => Class Diagram
  - Status transition => Activity Diagram, Collaboration Diagram
  - etc.
MDA Vision

- Cannot avoid co-existence of plural (middleware) platforms

MDA

= Model Driven Architecture

- Platform independent system design
  - described using UML (Unified Modeling Language) in general
  - called as PIM (Platform Independent Model)

- From PIM, system design for each platform is driven
  - called as PSM (Platform Specific Model)

- From PSM, actual skeleton of codes is driven
Simple Example

```xml
<Car>
  <doors>2</doors>
  <colour>red</colour>
</Car>
```

```java
public class Car {
    public String colour;
    public int door#;
}
```

```xml
<auto doors="2" colour="red"/>
```
Example in XMI

Precise mapping rule is defined in OMG XMI Standard.

PIM (denoted by UML)

Model in XMI

XMI Schema & DTD

Exchange of Model

Java, IDL

XMI doc.

Exchange of XML Data

Class Car
{ Colour colour
  Door door
}

<Class>
  <Name> Car</Name>
  </Class>

<!ELEMENT Car(Colour*, Door*)>

{ Colour colour
  Door door
}

<element name="Car"/>

<?ELEMENT Car(Colour*, Door*)>
Driven by Model

Platform Independent Model (PIM)

Platform Specific Model (PSM) for EJB/CORBA

Bridge

Platform Specific Model (PSM) for ebXML

Implementation framework of EJB/IDL

ebXML Message Definition

Makoto Oya
(based on Andrew Watson's presentation)
MDA (Model Driven Architecture):
MDA's Approach
An Example to show the MDA's Approach

Order/Sales system of "Car" (🚗)  
- Order option: Color and kind of door --- Toy car! (^_^)

Points to see:
- Logically same data representing Car appears at various places.
- "Integrate point" performing logically same process also appears at various points.

MDA's approach
- PIM (Platform Independent Model)
- PSM (Platform Specific Model)
Total system
Where does data representing "Car" appear?

Where are integration points (= end-points of processes)?
Where does data representing Car appear?

Where are integration points (= end-points of processes)?
MDA’s Approach

PIM

PSM

Skeletons for implementation

ebXML (different form)

MDA Information Day 2002/04/23

Copyright (c) 2002, Hitachi, Ltd.
MDA's Approach

- Model driven
- PIM and PSM
- PIM represents system design independently of platforms
- PSM represents implementation level design based on a particular platform specific characteristics
- Mappings:
  - PIM => PSM
  - PSM => Implementation
- Flexible development process and life cycle:
  - PSM => PIM
  - PIM => PIM, PSM => PSM
  - (Implementation => PSM)
Example of PIM and PSM

- A simple order/response system
  - Query of price (PriceQuery)
  - Ordering (Order)
  - Shipment notification (Notification)

- PSM
  - EJB mapping example
    - assuming intra-trade in an enterprise.
  - SOAP mapping example
    - Web Service; assuming inter-enterprise trade.

Note: PIM: Platform Independent Model, PSM: Platform Specific Model
Example of PIM

BuyerSystem

Notification
+ notifyShipment (item: ItemDesc, quantity: Integer, orderID: OrderID)

<<call>>

SellerSystem

PriceQuery
+ getPrice (item: ItemDesc) : Integer

Order
+ requestOrder (item: ItemDesc, quantity: Integer) : OrderID
+ cancelOrder (orderID: OrderID)

CommonDefinition

ItemDesc
+ itemID: Integer
+ description: String

OrderID
+ orderNo: Integer
+ date: Date

<<import>>
<<call>>
<<call>>
<<import>>
Example of PSM
(showing this portion)

---

**BuyerSystem**

- **Notification**
  - + notifyShipment (item: ItemDesc, quantity: Integer, orderID: OrderID)

---

**SellerSystem**

- **PriceQuery**
  - + getPrice (item: ItemDesc) : Integer
- **Order**
  - + requestOrder (item: ItemDesc, quantity: Integer) : OrderID
  - + cancelOrder (orderID: OrderID)

---

**CommonDefinition**

- **ItemDesc**
  - + itemID: Integer
  - + description: String
- **OrderID**
  - + orderNo: Integer
  - + date: Date

---

Copyright © 2002, Hitachi, Ltd.
PSM (for EJB)

SellerSystem

<<EJBSessionHomeInterface>>
PriceQuery_Manager

+ <<EJBCreateMethod>> create()

<<instantiate>>

<<EJBRemoteInterface>> PriceQuery

+ <<EJBRemoteMethod>> getPrice
  ( item: ItemDesc ) : Integer

<<EJBSessionHomeInterface>>
Order_Manager

+ <<EJBCreateMethod>> create()

<<instantiate>>

<<EJBRemoteInterface>> Order

+ <<EJBRemoteMethod>> requestOrder
  ( item: ItemDesc, quantity: Integer )
  : OrderID
+ <<EJBRemoteMethod>> cancelOrder
  (orderID : OrderID)

PIM

SellerSystem

PriceQuery

+ getPrice ( item: ItemDesc ) : Integer

Order

+ requestOrder ( item: ItemDesc,
  quantity: Integer )
  : OrderID
+ cancelOrder (orderID : OrderID)
PSM (for SOAP)

**SellerSystem**

- **PriceQuery**
  - + getPrice (item: ItemDesc) : Integer

- **Order**
  - + requestOrder (item: ItemDesc, quantity: Integer) : OrderID
  - + cancelOrder (orderID: OrderID) : OrderID

**Mapping**

- **<<portType>> PriceQuery**
  - <<operation>> getPrice

- **<<portType>> Order**
  - <<operation>> requestOrder
  - <<operation>> cancelOrder

**<<message>> getPriceInput**
- + item: ItemDesc

**<<message>> getPriceOutput**
- + price: Integer

**<<message>> requestOrderInput**
- + item: ItemDesc
- + quantity: Integer

**<<message>> orderIdentifier**
- + orderID: OrderID

Makoto Oya

Copyright (c) 2002, Hitachi, Ltd.
PSM (for SOAP)
<definitions name="uri-BuySellSystem" ...>
  xmlns:cd="uri-CommonDefinition" ...
  <import namespace="uri-CommonDefinition"/>

  <message name="getPriceInput">
    <part name="item" element="cd:ItemDesc"/>
  </message>

  <message name="getPriceOutput">
    <part name="price" element="int"/>
  </message>

  <message name="requestOrderInput">
    <part name="item" element="cd:ItemDesc"/>
    <part name="quantity" element="int"/>
  </message>

  <message name="orderIdentifier">
    <part name="orderID" element="cd:OrderID"/>
  </message>

  <portType name="PriceQuery">
    <operation name="getPrice">
      <input message="getPriceInput"/>
      <output message="getPriceOutput"/>
    </operation>
  </portType>

  <portType name="Order">
    <operation name="requestOrder">
      <input message="requestOrderInput"/>
      <output message="orderIdentifier"/>
    </operation>
    <operation name="cancelOrder">
      <input message="orderIdentifier"/>
    </operation>
  </portType>

  <service name="SellerSystem">
    <port name="PriceQueryPort" binding="PBinding">
      <soap:address location="http://examp.co.jp/serv1/"/>
    </port>
    <port name="OrderPort" binding="OBinding">
      <soap:address location="http://examp.co.jp/serv2/"/>
    </port>
  </service>
</definitions>

<binding name="PBinding" type="PriceQuery">
  <soap:binding style="rpc"
    transport="schemas.xmlsoap.org/soap/http"/>
  <operation name="getPrice">
    <input>
      <soap:body use="encoded" namespace=.../>
      <soap:header .../>
    </input>
    <output>
      ...
    </output>
  </operation>
</binding>

<binding name="OBinding" type="Order">
  <soap:binding style="rpc|document" transport=.../>
  <operation name="requestOrder">
    ...
  </operation>
  <operation name="cancelOrder">
    ...
  </operation>
</binding>
Additional Consideration
More Consideration

PSM(ebXML) → PIM
PSM(SOAP) → PIM
PSM(EJB/CORBA) → PIM
PSM(MTS/.NET) → PIM

PSM(SOAP) → SOAP
PSM(SOAP) → ebXML

PSM(ebXML) → ebXML

PSM(EJB/CORBA) → EJB
PSM(EJB/CORBA) → IIOP
PSM(EJB/CORBA) → COM
PSM(EJB/CORBA) → MTS

PSM(MTS/.NET) → DCOM
Toward Realization of MDA
To Realize MDA

How to map PIM to PSM?

- Do we depend on "Experience and talent of architects" or "Effort and tear of system developers" as we do now?

- "Secret" of MDA:
  - Define/standardize common/typical mapping rules
  - Aim semi-automatic PSM generation.

Need to shorten the "Distance" between PIM and PSM

- Rich frameworks for application
Mapping from PIM to PSM

- Typical design
- Common design in a particular domain/application

- Specification and characteristics of platform

Example:
- Data sharing
- Event management
- EAI transformer
- Patient data

Requirement → Analyze
Model (PIM) → Profile
- Model patterns
- Mapping rules

Profile
- Implementation code mappings

Standardize

Standardize

Human Tool

PSM

Implementation Development

- Algorithm
- Compile
- Program Design
- Coding
- Test

- Standard rules
- Semi-automation

Example:
- EJB trans.
- CORBA trans
- XML trans.
Shorten the "Distance" between PIM and PSM

Implementation Development
- Algorithm
- Compile
- Program design
- Coding
- Test

Rich Frameworks
Example of Profile
(from Event Profile in "UML Profile for EDOC")
Ultimate Goal

The most important is to develop and heap standardized profiles.

- Analyze
- Select profiles
- System Design
- Model (PIM)
- PSM
- Code generation
- Mapping rules
- Model patterns
- Platform dependent profiles
- Impl. codes
- Impl. system
- Frameworks

Copyright (c) 2002, Hitachi, Ltd.
### Note: Various profiles - already standardized, in process, under discussion

<table>
<thead>
<tr>
<th>OMG(standardized)</th>
<th>- CCA (Component Collaboration Architecture)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Entities Profile</td>
</tr>
<tr>
<td></td>
<td>- Events Profile</td>
</tr>
<tr>
<td></td>
<td>- Business Process Profile</td>
</tr>
<tr>
<td></td>
<td>- Relationship Profile</td>
</tr>
<tr>
<td>OMG(in process)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms</td>
</tr>
<tr>
<td></td>
<td>- UML for Systems Engineering</td>
</tr>
<tr>
<td>JCP(standardized)</td>
<td>- CCA (Component Collaboration Architecture)</td>
</tr>
<tr>
<td></td>
<td>- Entities Profile</td>
</tr>
<tr>
<td></td>
<td>- Events Profile</td>
</tr>
<tr>
<td></td>
<td>- Business Process Profile</td>
</tr>
<tr>
<td></td>
<td>- Relationship Profile</td>
</tr>
<tr>
<td>Others (discussing, topics, rumor)</td>
<td></td>
</tr>
<tr>
<td>- UML Profile for WSDL</td>
<td>- UML Profile for .NET</td>
</tr>
<tr>
<td>- UML Profile for XML Schema</td>
<td>- UML Profile for Interaction design</td>
</tr>
<tr>
<td>- UML Profile for Persistence Model</td>
<td>- UML Profile for Database Design</td>
</tr>
<tr>
<td>- UML Profile for Reverse Engineering</td>
<td>- UML Profile for hypermedia</td>
</tr>
<tr>
<td>- UML Profile for Framework Architectures</td>
<td>- UML for Ontology Development</td>
</tr>
<tr>
<td>- UML Profile for DCL</td>
<td>- UML profile for DAML</td>
</tr>
<tr>
<td>- UML Profile for Business Modeling</td>
<td>- UML Profile for Web applications</td>
</tr>
<tr>
<td>- UML profile for Business Analysis</td>
<td></td>
</tr>
</tbody>
</table>
Summary
MDA Summary

- **PIM and PSM**
- Two kinds of mappings:
  - PIM=>PSM and PSM=>Implementation
- For PIM creators, standardized application specific profiles are provided. Standard mappings to PSM are also defined.
- Aiming semi-automatic PIM=>PSM transformation and automatic PSM=>Implementation transformation.
- Directly connecting to actual implementation development, and/or system design is platform independent.
- The most important thing for realization of MDA is development of wide range of standardized profiles.
Conclusion

Middleware will continue to evolve and proliferate with emerging technologies.

- CORBA, Java and .NET will also evolve.
- Web Services will evolve.
- New middleware may appear.
- Users want their IT systems up to date using such state-of-art technologies.

Business requirements will also evolve and need to be quickly implemented in enterprise IT systems.

PIM and PSM should to be independently designed and developed corresponding to business evolution and technology evolution, without breaking consistency, and in the way improving development productivity.

MDA is the Key to this vision.
END

Note: All names in this presentation, including company names and product names, are used identification purpose only, may be trademark or registered trademark of their respective holders.