2002年4月OMG横浜会議のMDA Information Dayでの講演内容

MDAとシステム設計

2002年4月23日

(株)日立製作所
大谷　真

序：OMG の動き

- 1989年設立
- 分散オブジェクトミドルウェアの標準化
  - 1995年： CORBA2; 2001年： CORBA2.5
- ドメイン（業種別、業種共通）標準化
  - 1995年～： 各分野標準
- モデリング（＝システム設計）の標準化
  - 1997年： UML（Unified Modeling Language）
  - 1997年： MOF； 1999年： XMI； 2000年：CWM
  - 2001年： 分野固有UMLプロファイル（EDOC, EAI）
- アーキテクチャ（参照モデル）
  - 1990年： OMA (Object Management Architecture)
  - 2001年： **MDA (Model Driven Architecture)**
- 2001年後半～：MDAに基づく各種標準化
- 2002年：UMLバージョン2（予定）・・・MDA基本機能を整備

アジェンダ

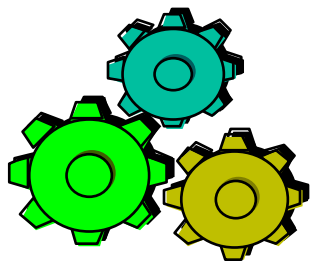
- MDAの背景とビジョン
- MDAのアプローチ
- MDAの実現に向けて
- まとめ

MDAの背景とビジョン

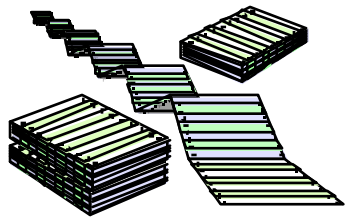
業務のインテグレーション

営業

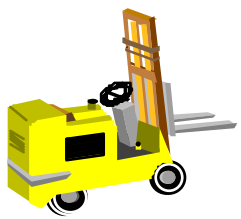
設計



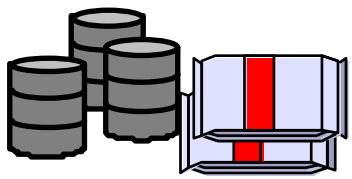
製造



財務・会計



出荷・入荷



入庫・出庫

入金・出金
・売掛

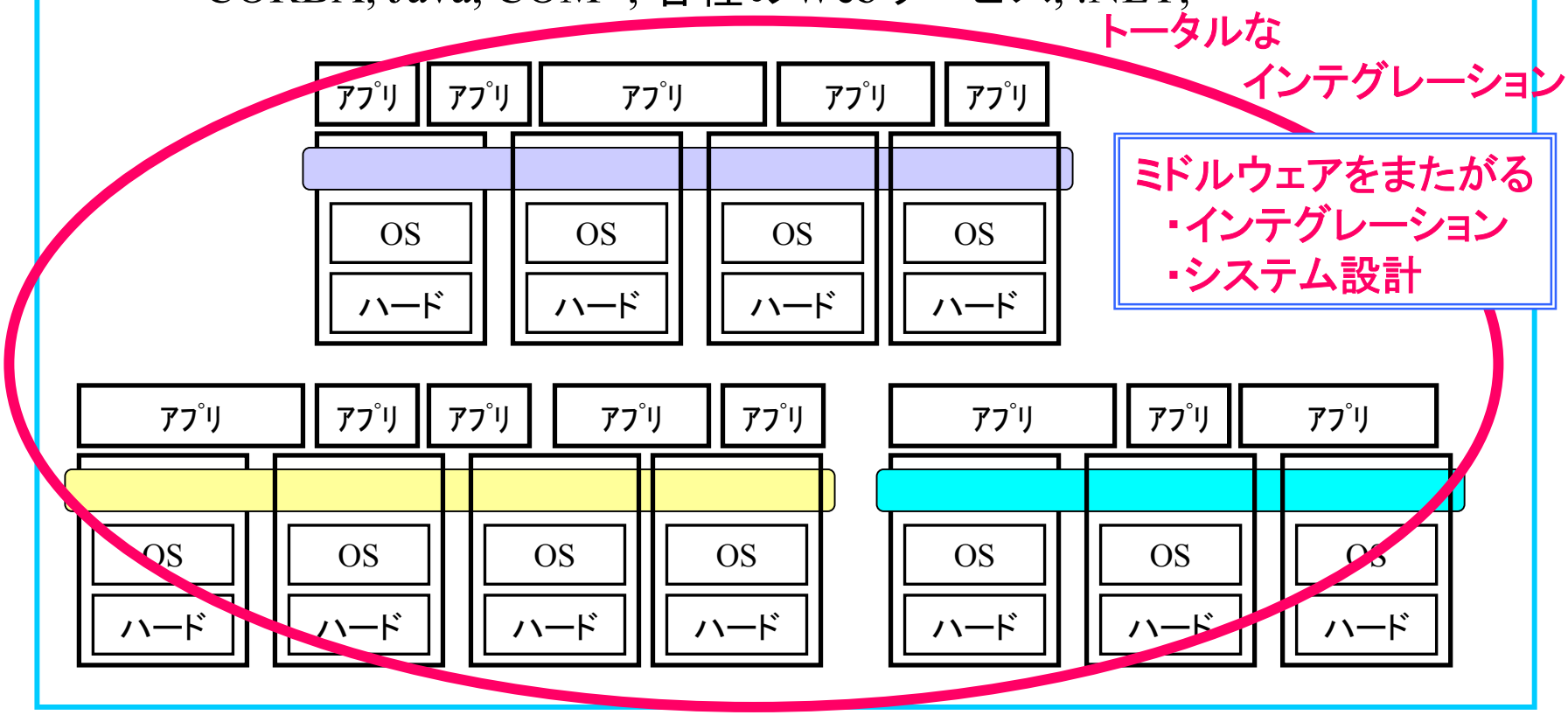
- 20年近くにわたって着実に進歩してきた
- しかし、現在も大きな課題を抱えている

問題の根源＝多種多様なプラットフォーム

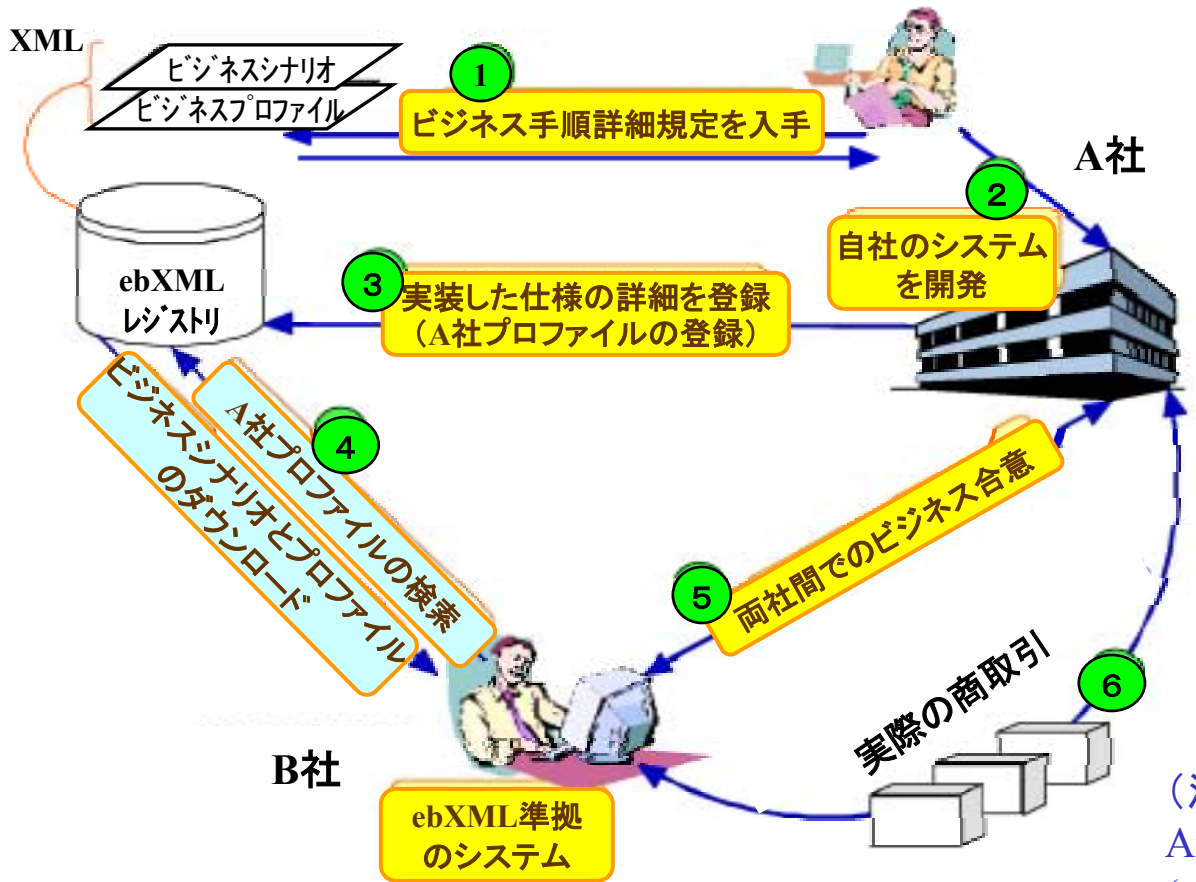
- 複数のハードウェアアーキテクチャ
  - Pentium, PowerPC, PA-RISC, Sparc, 370, ・・・
- 複数のネットワーク
  - イーサネット、ATM、IP、SS7、Appletalk、USB、Firewire、・・・
- 複数のプログラミング言語
  - C/C++, Java, Visual Basic, C#, Perl, JavaScript, VBScript, COBOL, PL/I, Fortran, ・・・
- 複数のOS
  - Unix、Windows、NT/XP、メインフレームOS、MacOS、Windows CE、携帯電話、セットトップボックス、ゲーム機、・・・
- そして、複数の分散ミドルウェア
  - JAVA/CORBA, COM+/.NET, Webサービス（SOAP, ebXML,...）

分散ミドルウェアの成功、進化、そして増殖

- ミドルウェアの標準化と普及により、異なるハード／OS上でのインテグレーションは解決の方向にある。
- 新たな問題：<u>複数のミドルウェア</u>
  CORBA, Java, COM+, 各種のWebサービス, .NET, ・・・



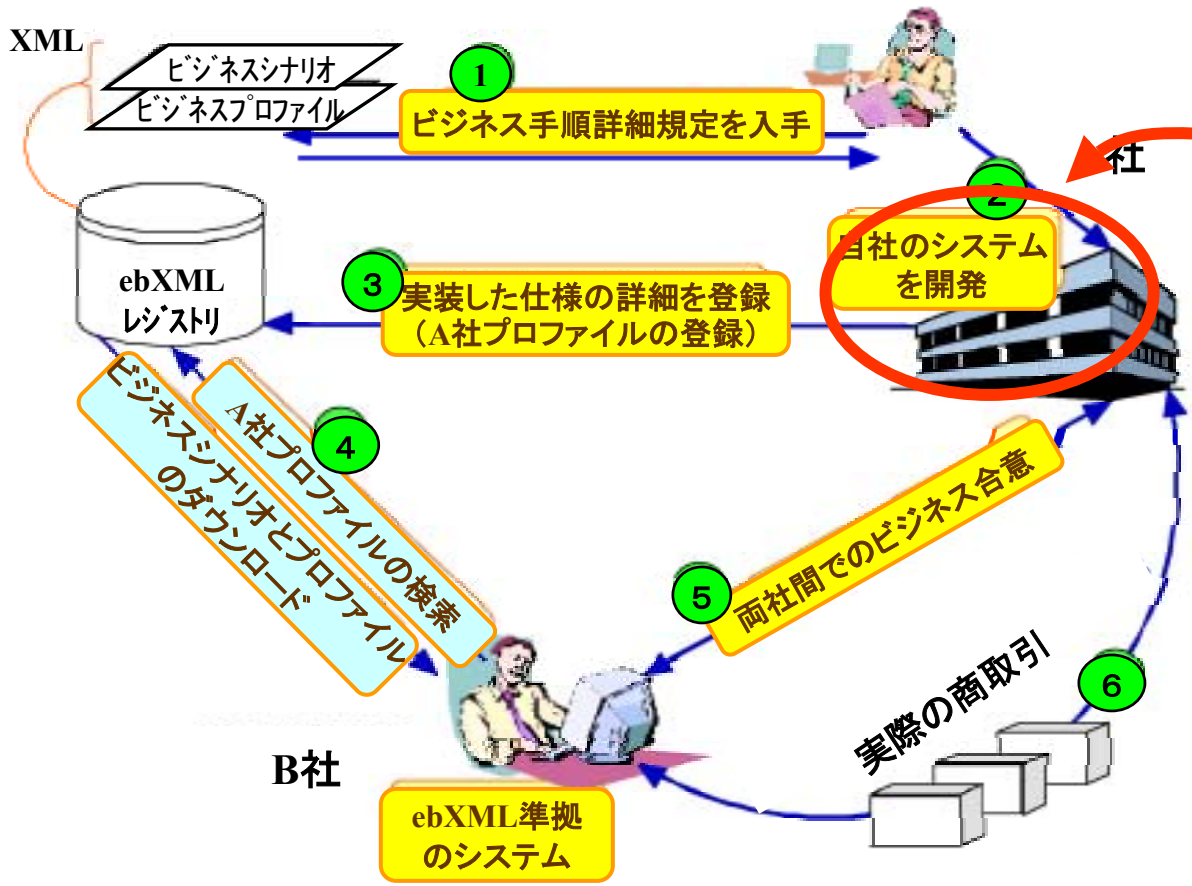
第1の課題:
ミドルウェアをまたがるインテグレーション



ミドルウェアの
一種である
Webサービス
（ebXML）
を例として
考えてみよう

（注）ebXMLの“Technical Architecture Specification” (v1.0.4)から引用

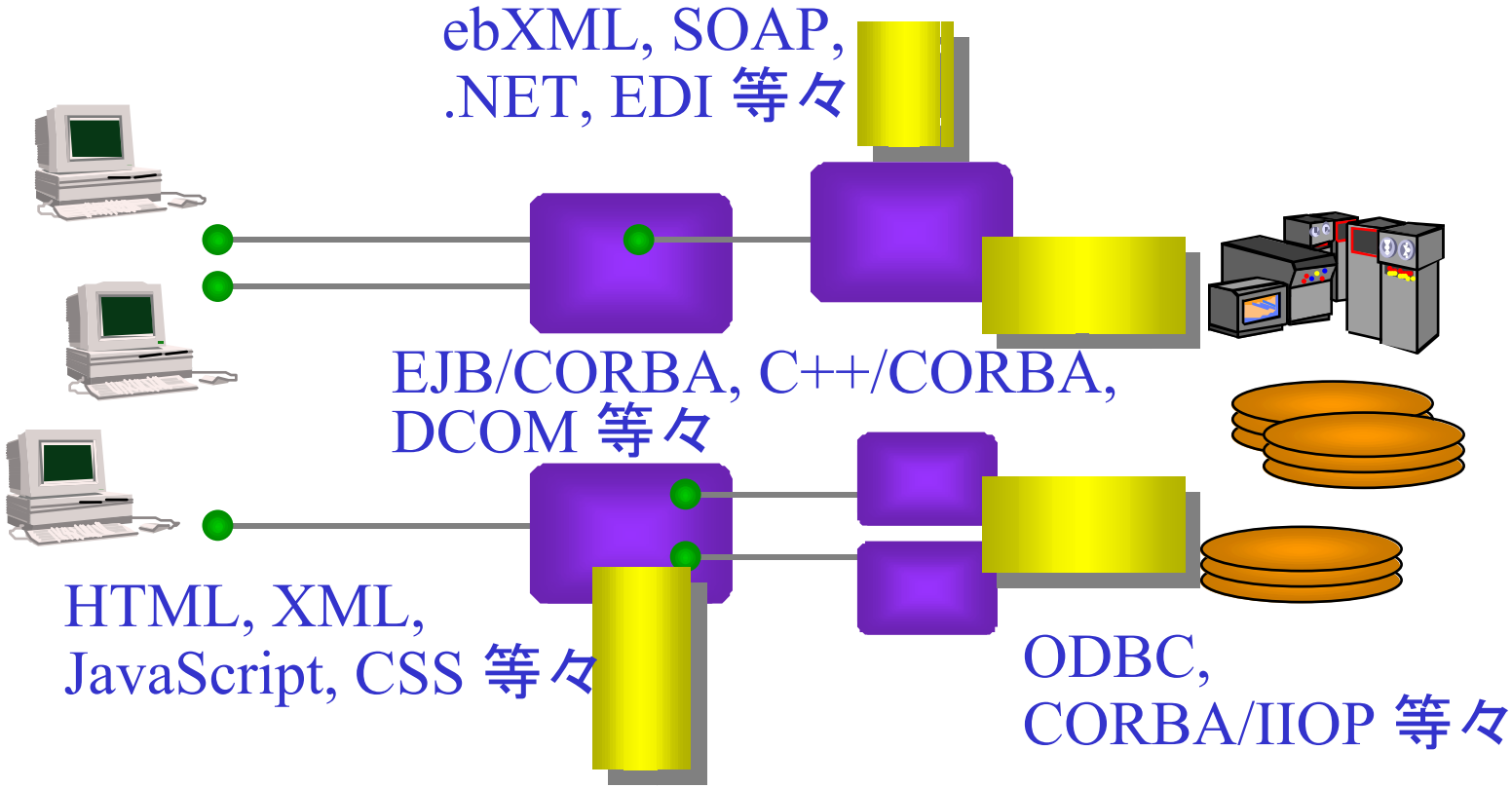
ミドルウェアをまたがるインテグレーション



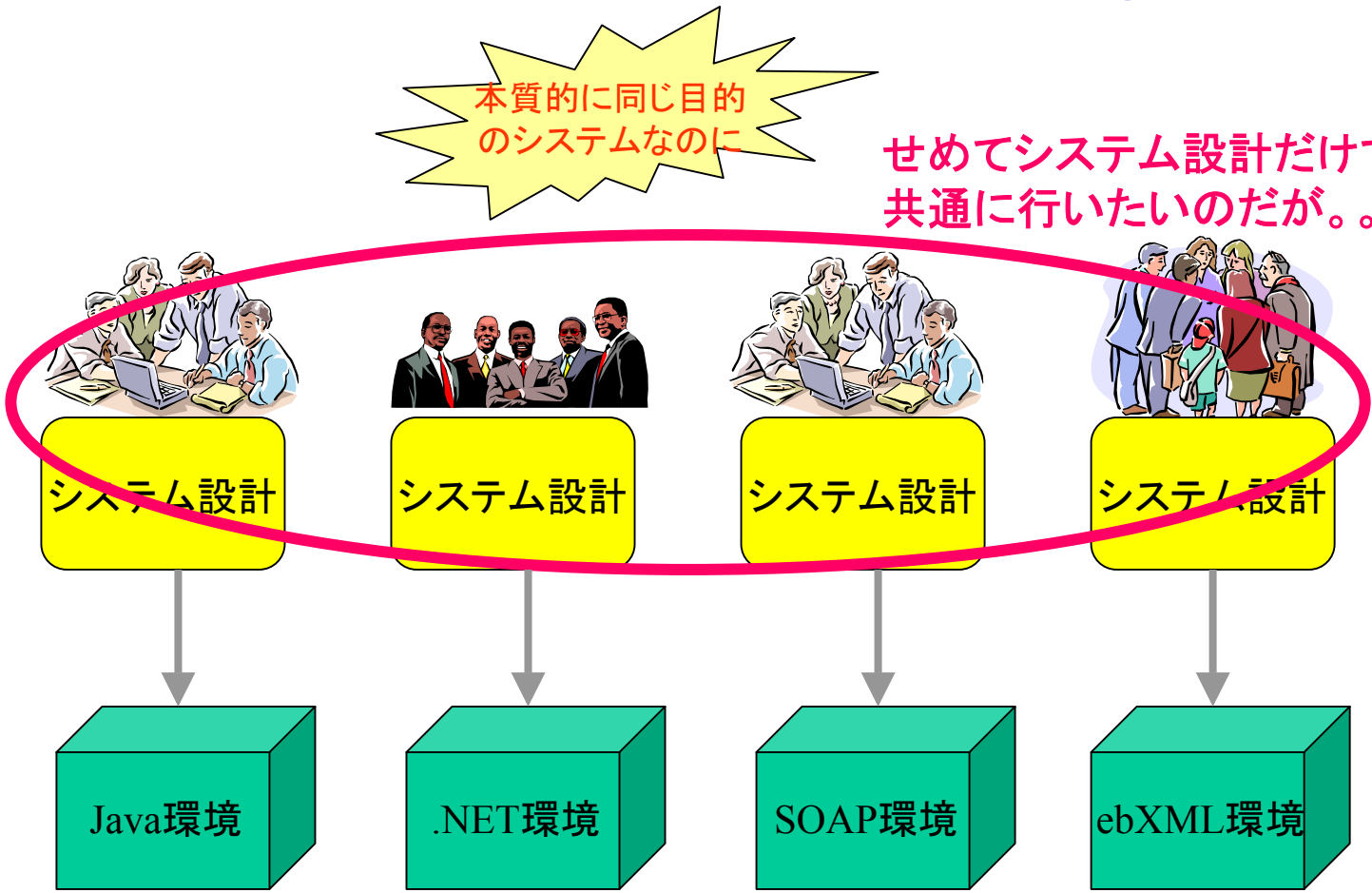
ここがWebサービスとその他のIT世界との接点

- 新規開発？
- 否。Webサービスを既存の企業システムにつなぎ込む。インテグレートする

それは、社内のミドルウェア環境に つなぎ込むことを意味する



第2の課題:
ミドルウェアをまたがるシステム設計



MDAのビジョン

- 複数の(ミドルウェア)プラットフォームの共存は不可避
- MDA(Model Driven Architecture)
  ＝モデル中心アーキテクチャ

補足:ここでいう「モデル」とは？

- プラットフォームに依存しない設計
  - UML (Unified Modeling Language)で記述
  - **PIM (Platform Independent Model)**と呼ばれる
- PIMから、実際のプラットフォームに依存した設計を導出
  - **PSM (Platform Specific Model)**と呼ばれる
- PSMから、現物のインターフェース情報を導出

補足:「モデル」とは？

- ここでいう「モデル」とは？
  - 概念を整理したもの
  - 実物の縮小、実物の特徴の一部を示すもの
  - 新たに作るとき参考にする既存物
  - 設計書 ---------- より正確には、「システムの具体的な設計書」
- UML（Unified Modeling Language）
  - システムの設計書を記述するための標準記法
    - 論理的なモジュール構成→クラス図
    - 状態遷移図→アクティビティ図、コラボレーション図
    - など

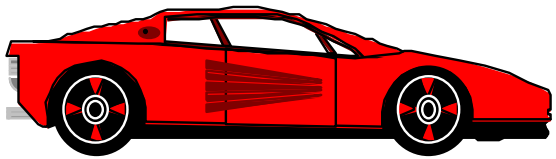
MDAのビジョン

- 複数の(ミドルウェア)プラットフォームの共存は不可避
- MDA(Model Driven Architecture)
  ＝モデル中心アーキテクチャ

モデル＝設計書

- プラットフォームに依存しない設計
  - UML (Unified Modeling Language)で記述
  - **PIM (Platform Independent Model)**と呼ばれる
- PIMから、実際のプラットフォームに依存した設計を導出
  - **PSM (Platform Specific Model)**と呼ばれる
- PSMから、現物のインターフェース情報を導出

簡単な例



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

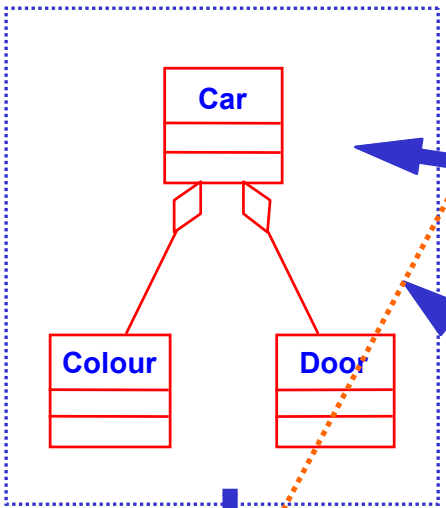
```
public class Car {
  public colour colour;
  public int door#; }
```

```
<auto doors=”2”  colour=”red”/>
```

XMIの例

OMGのXMI標準で
詳細なマッピング規則が定められている

PIM（UMLで記述）




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

Model in XMI

モデルの交換

```
<element name="Car"/>

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

XMI Schema & DTD

```
Class Car
  { Colour colour
    Door   door
  }
```

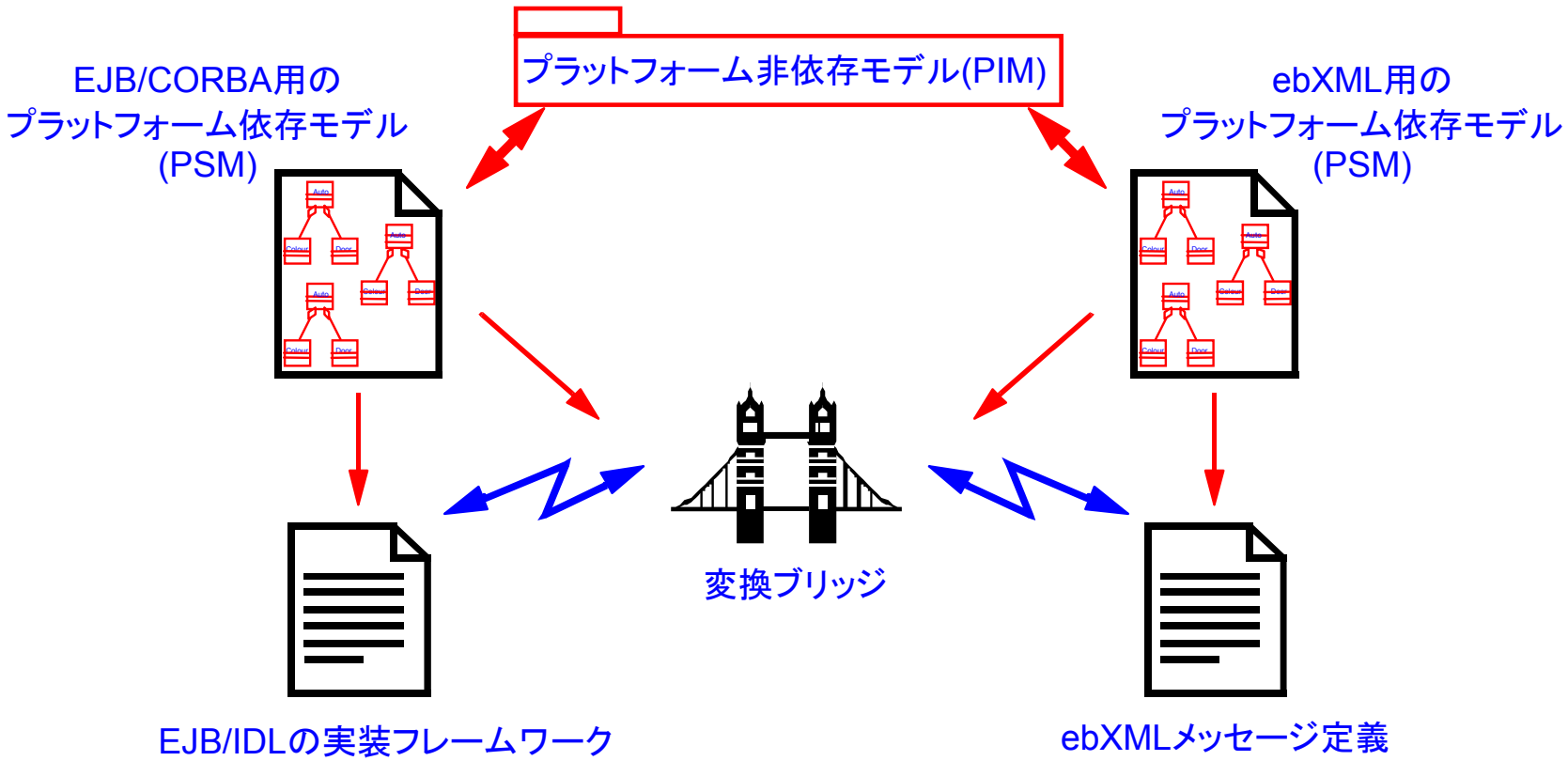
Java, IDL

```
<Car>
 <Colour>Red</Colour>
 <Door>2</Door>
</Car>
```

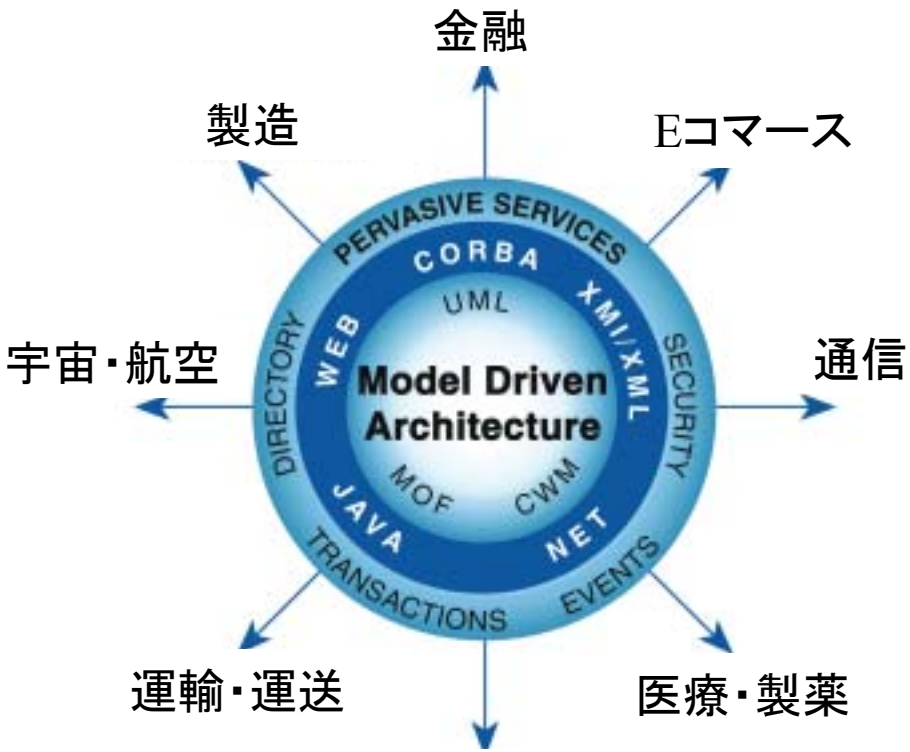
XMI doc.

XMLデータの交換

モデル中心のアーキテクチャ

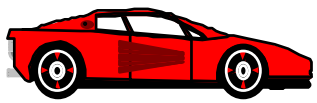


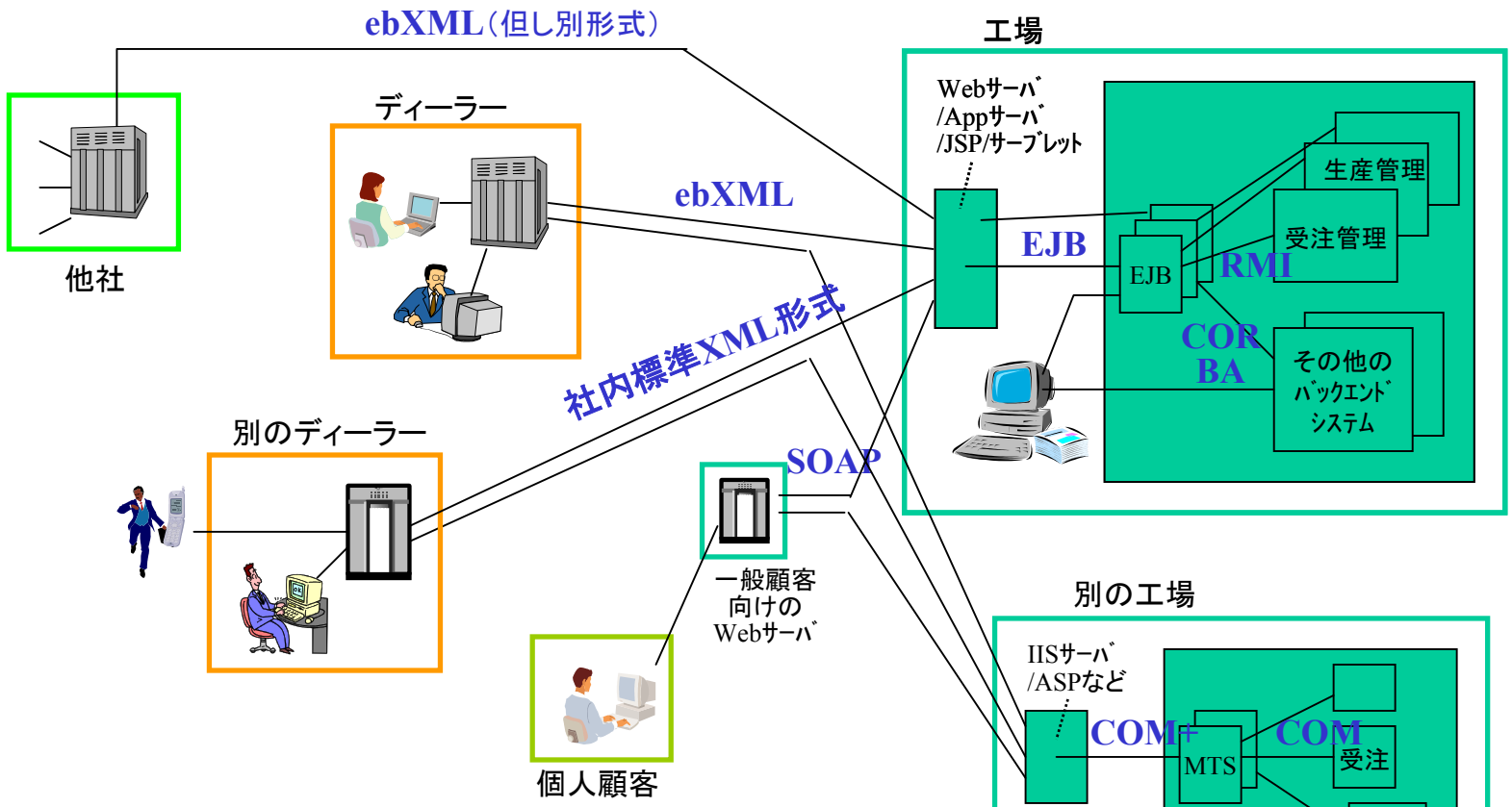
MDA (Model Driven Architecture) : モデル駆動アーキテクチャ



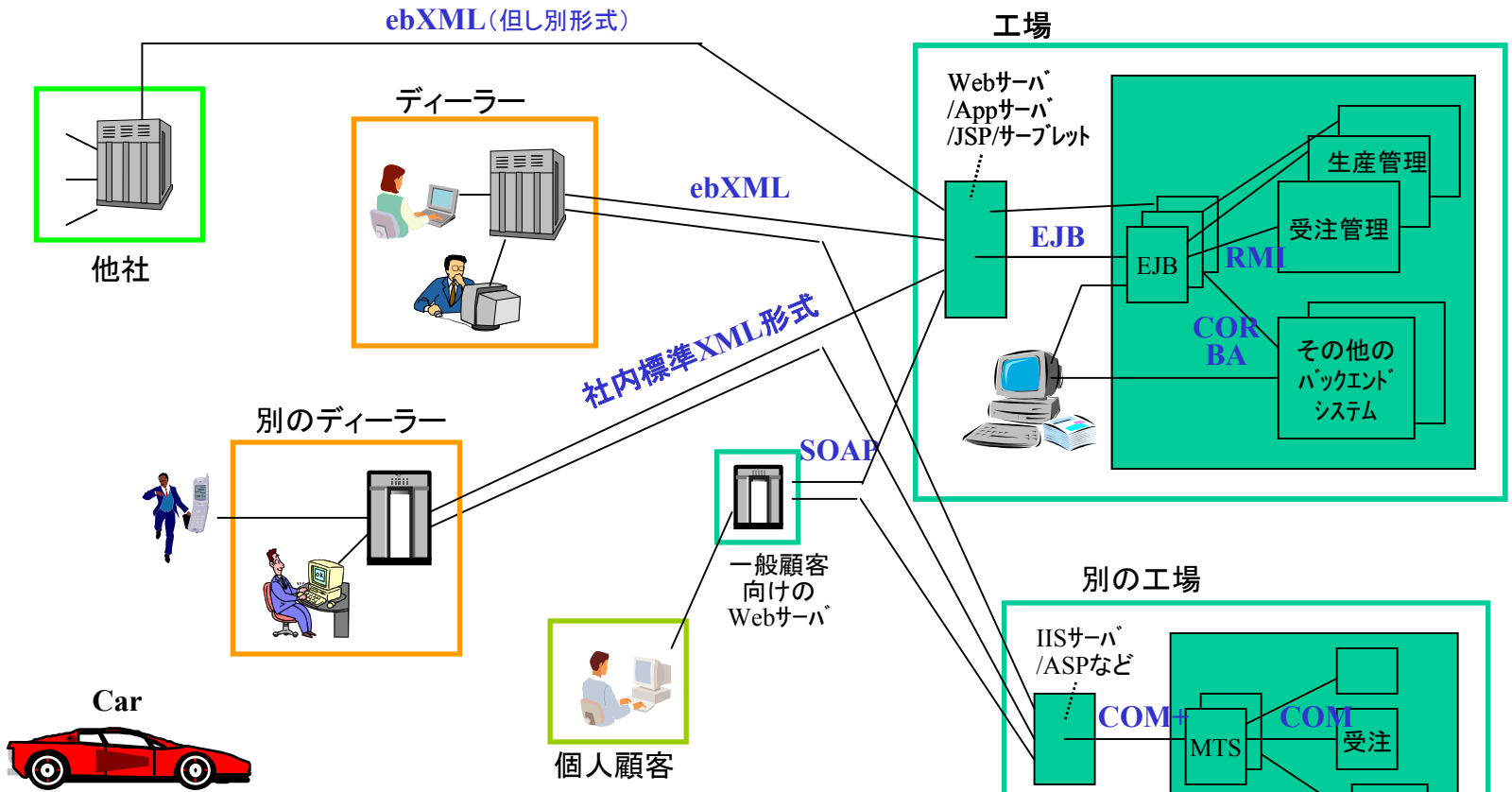
MDAのアプローチ

MDAのアプローチを示す例

- Car()の販売システム
  - 商品のオプション：Carの色とドア種別だけ・・・おもちゃの車！(^_^)
- ポイント
  - Carを表す“同じ“データがあちこちに現れる
  - “同じ”処理もあちこちに現れる
- MDAのアプローチ
  - PIM（Platform Independent Model）
  - PSM (Platform Specific Model）

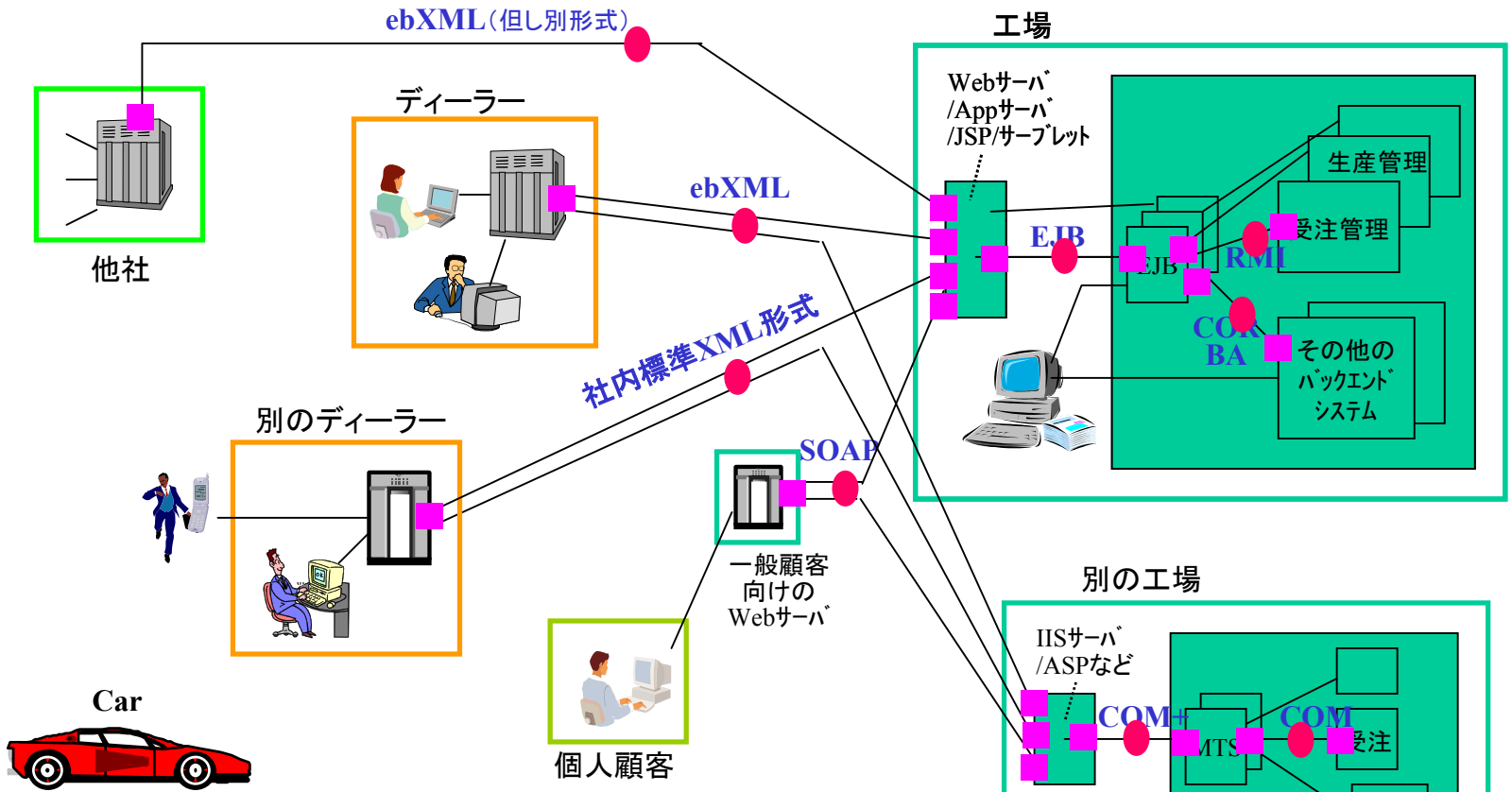


システム全体



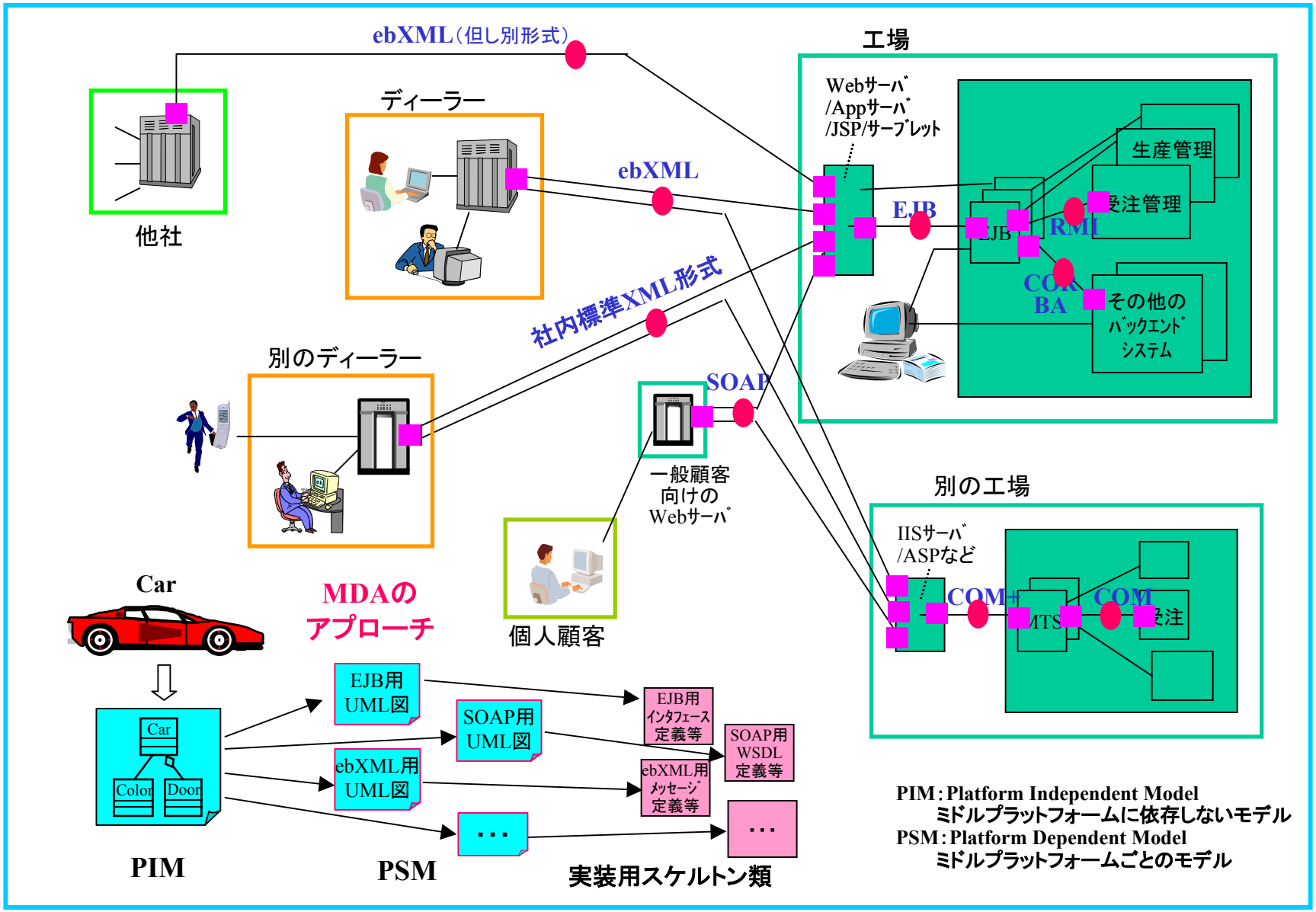
- Carを表すデータはどこに現れる？

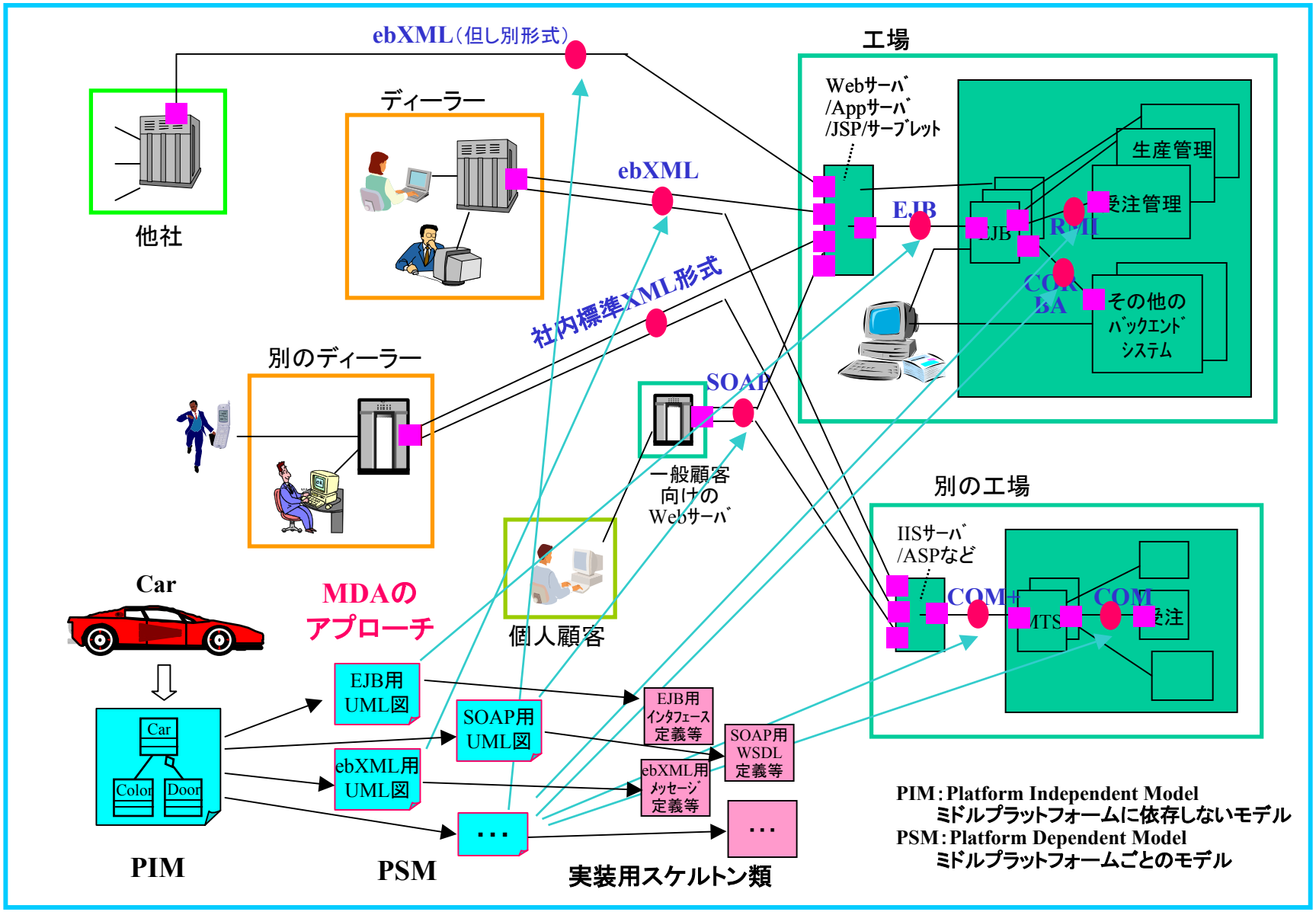
■ インテグレート点（処理の端点）はどこか？

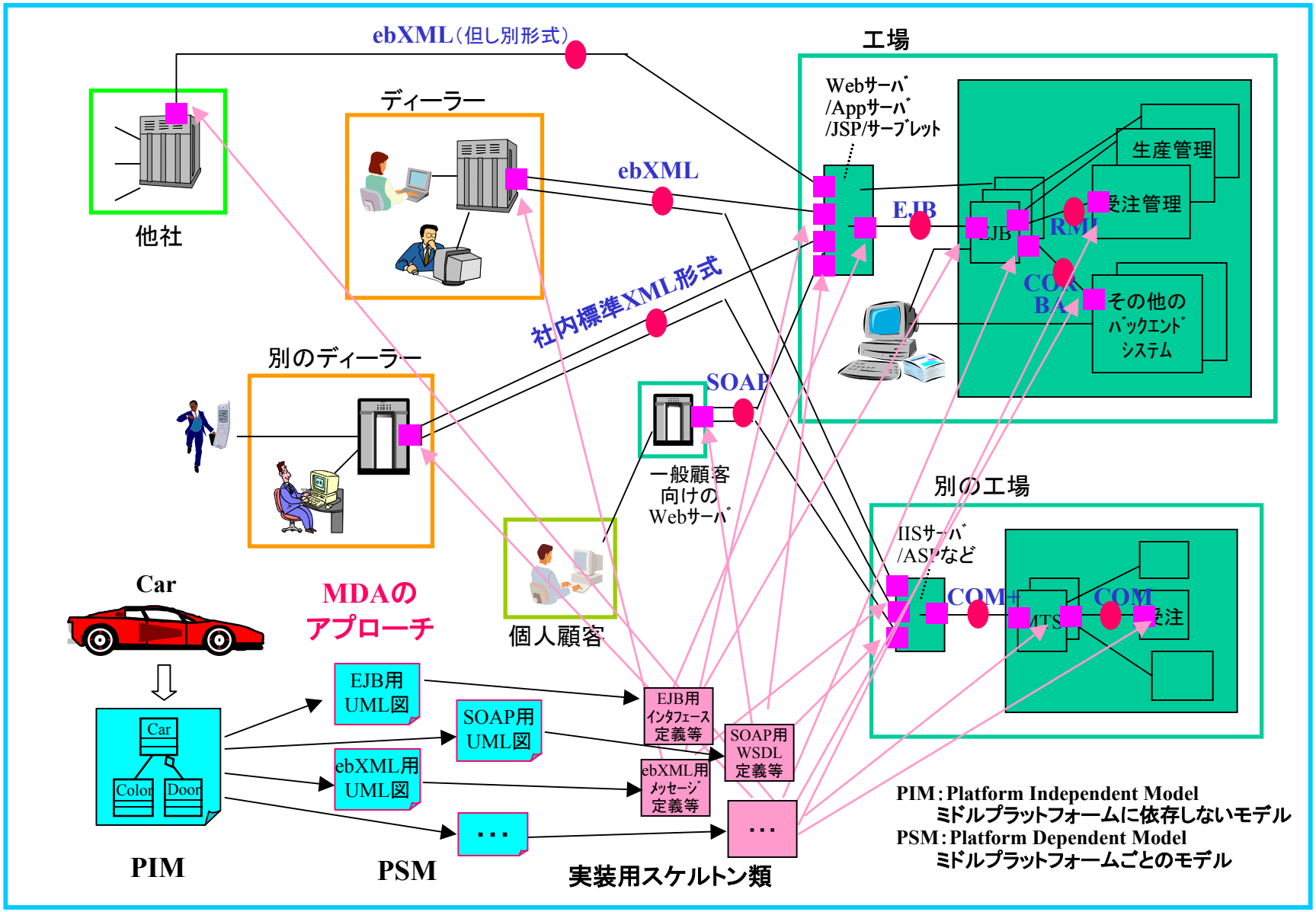


- Carを表すデータはどこに現れる？

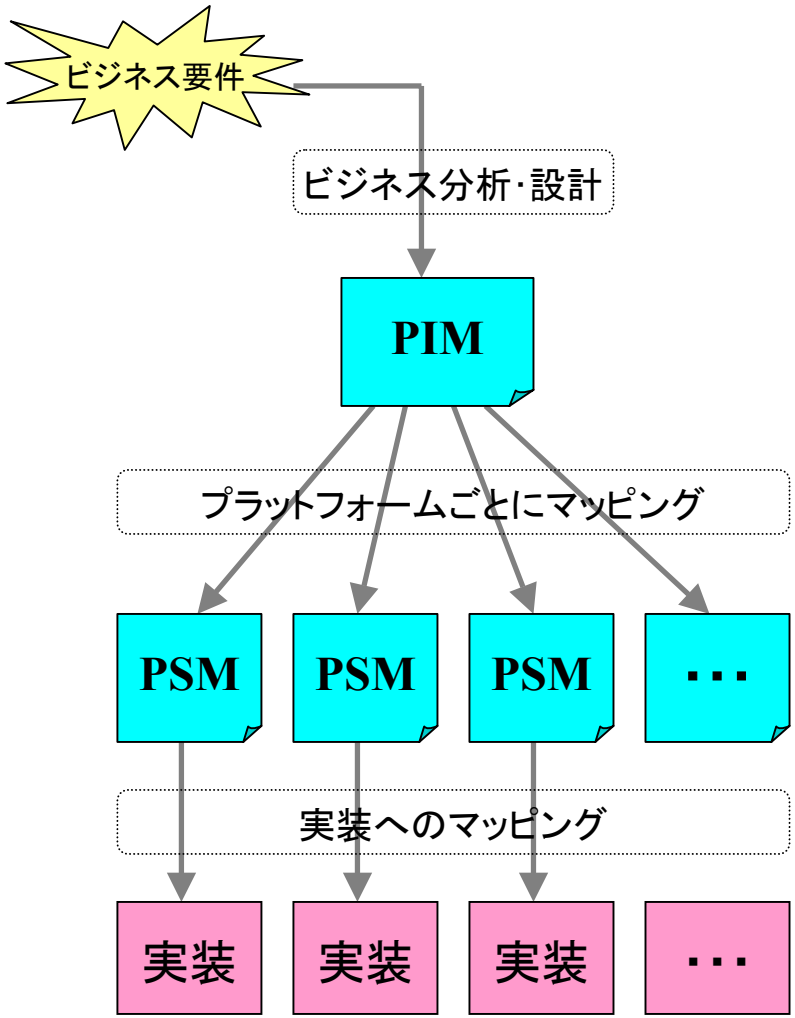
■ インテグレート点（処理の端点）はどこか？







MDAのアプローチ



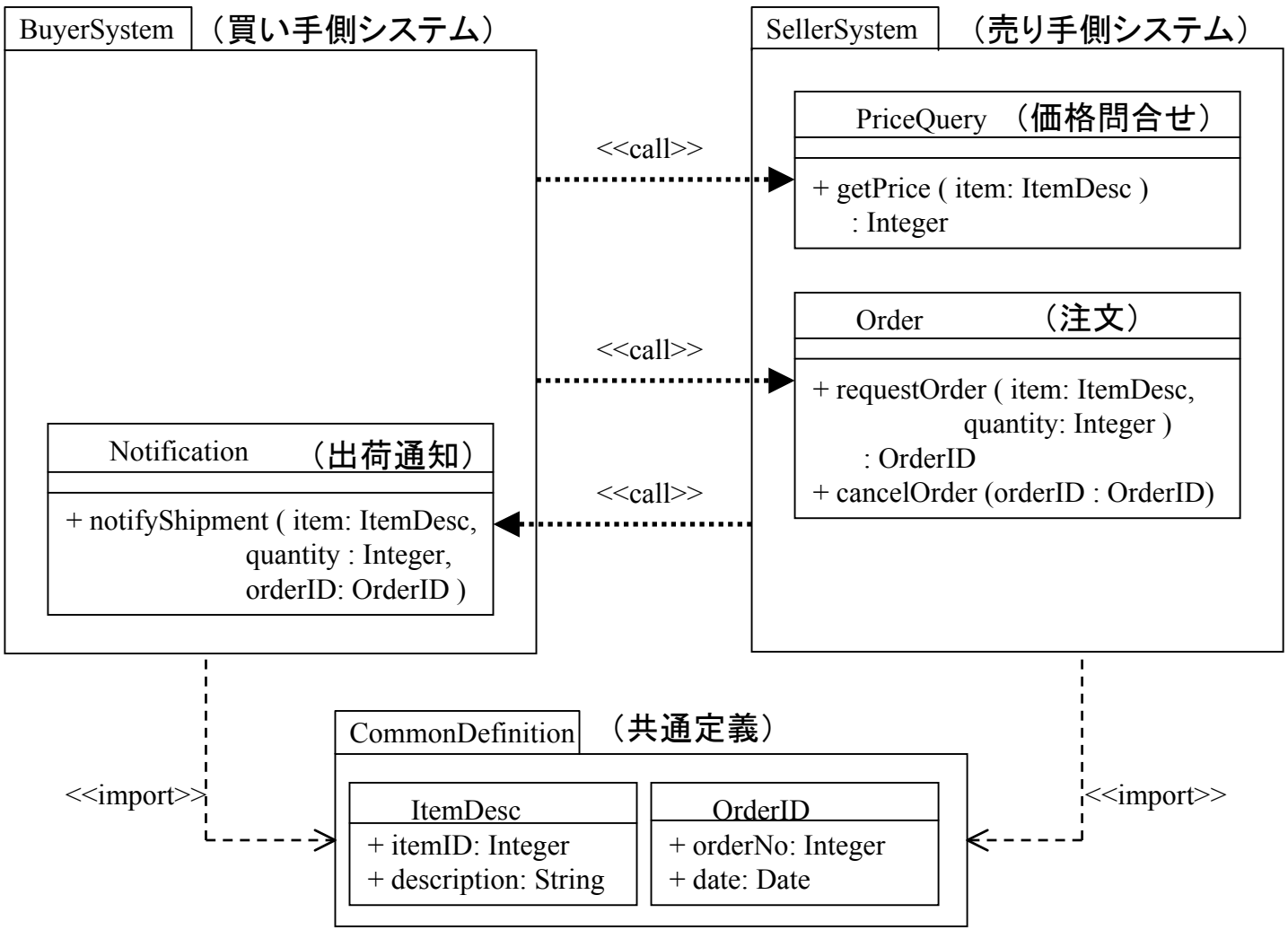
- 設計モデル中心
- PIMとPSM
- PIMはミドルウェアプラットフォーム非依存の設計図
- PSMはミドルウェアプラットフォーム固有機能を使った実装に即した設計図
- マッピング：
  - PIM→PSM
  - PSM→実装
- 開発プロセス／開発ライフサイクルへの柔軟な対応
  - PSM→PIM
  - PIM→PIM、PSM→PSM
  - （実装→PSM）

PIMとPSMの例

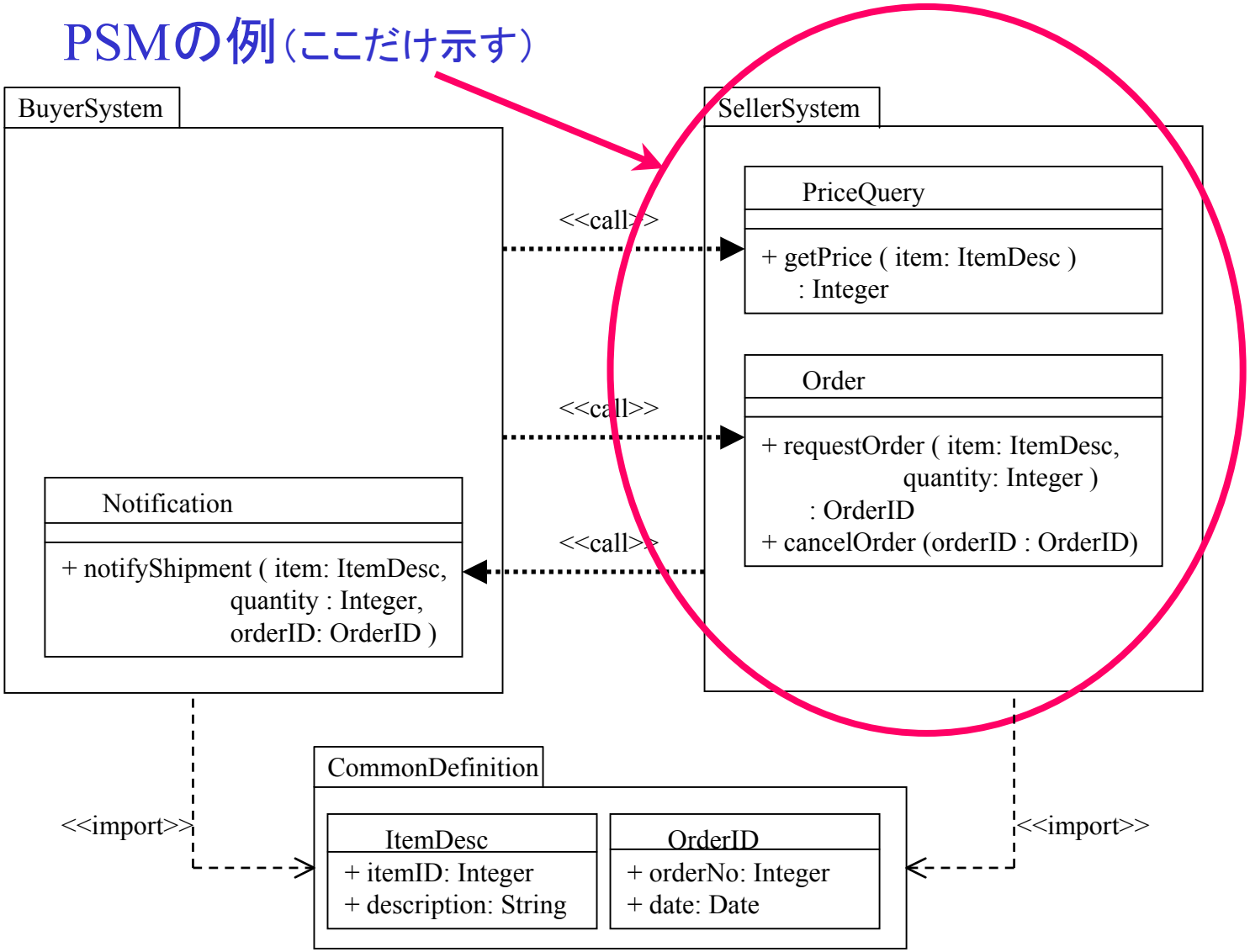
- 簡単な注文システム
  - 価格問合せ（PriceQuery）
  - 注文（Order）
  - 出荷通知（Notification）
- PSM
  - EJBマッピング例　　・・・企業内取引を想定
  - SOAPマッピング例　・・・Webサービス；企業間取引を想定

（注）PIM：プラットフォーム非依存モデル；Platform Independent Model
PSM：プラットフォーム依存モデル；Platform Specific Model

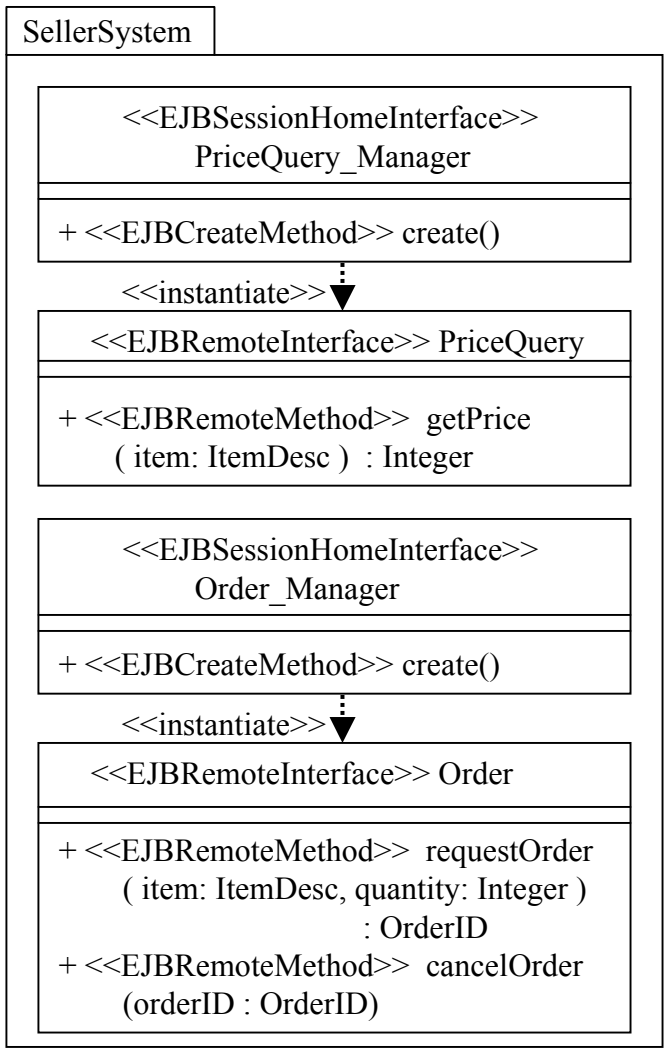
PIM



PSMの例（ここだけ示す）

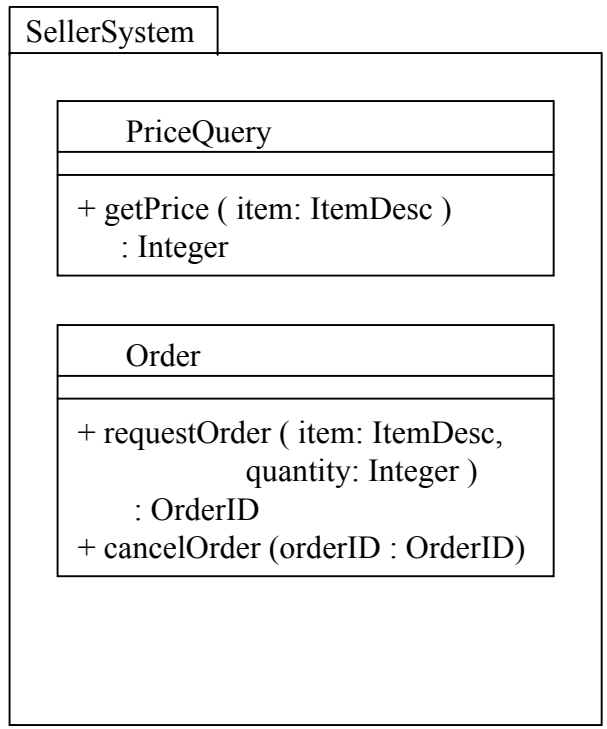


PSM(EJB用)

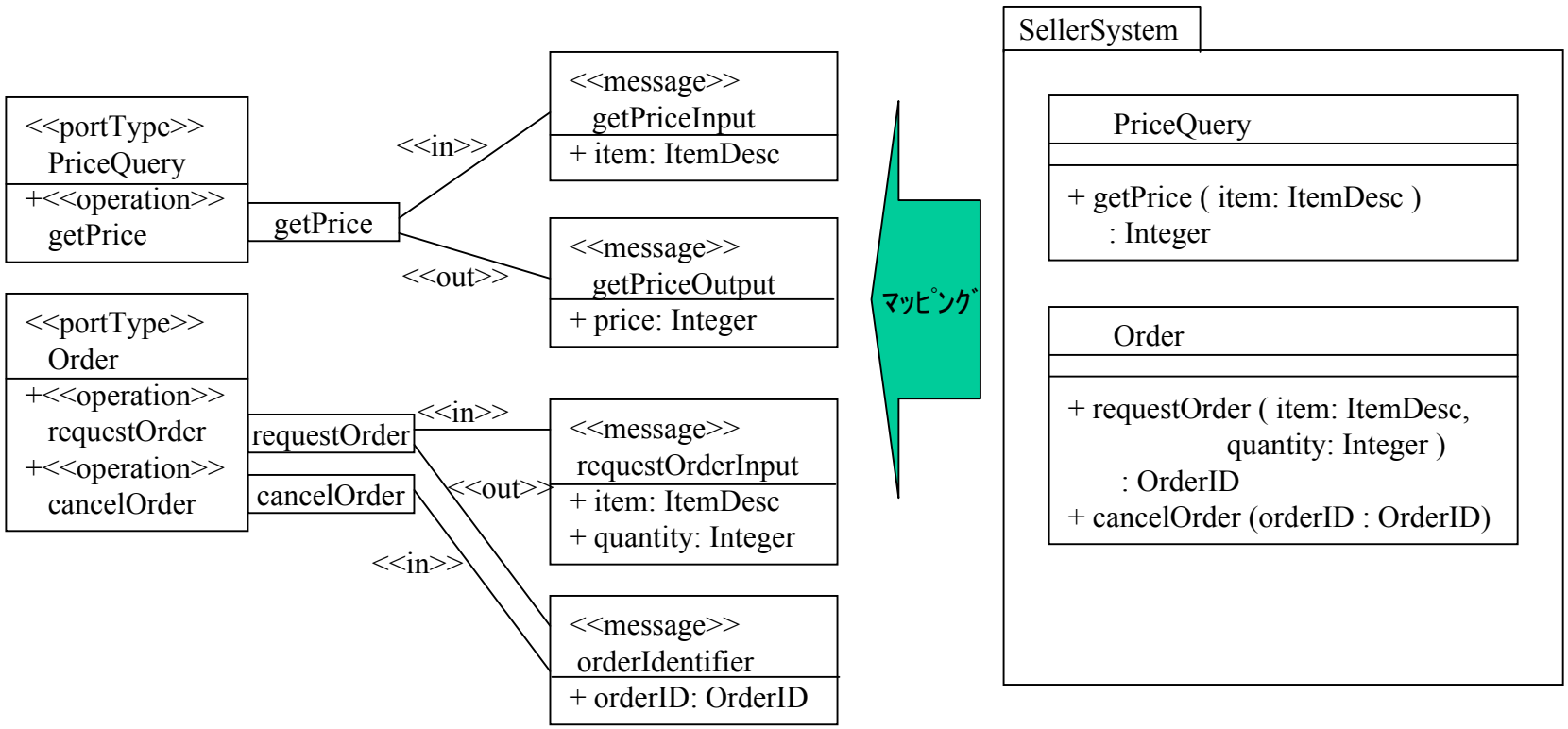


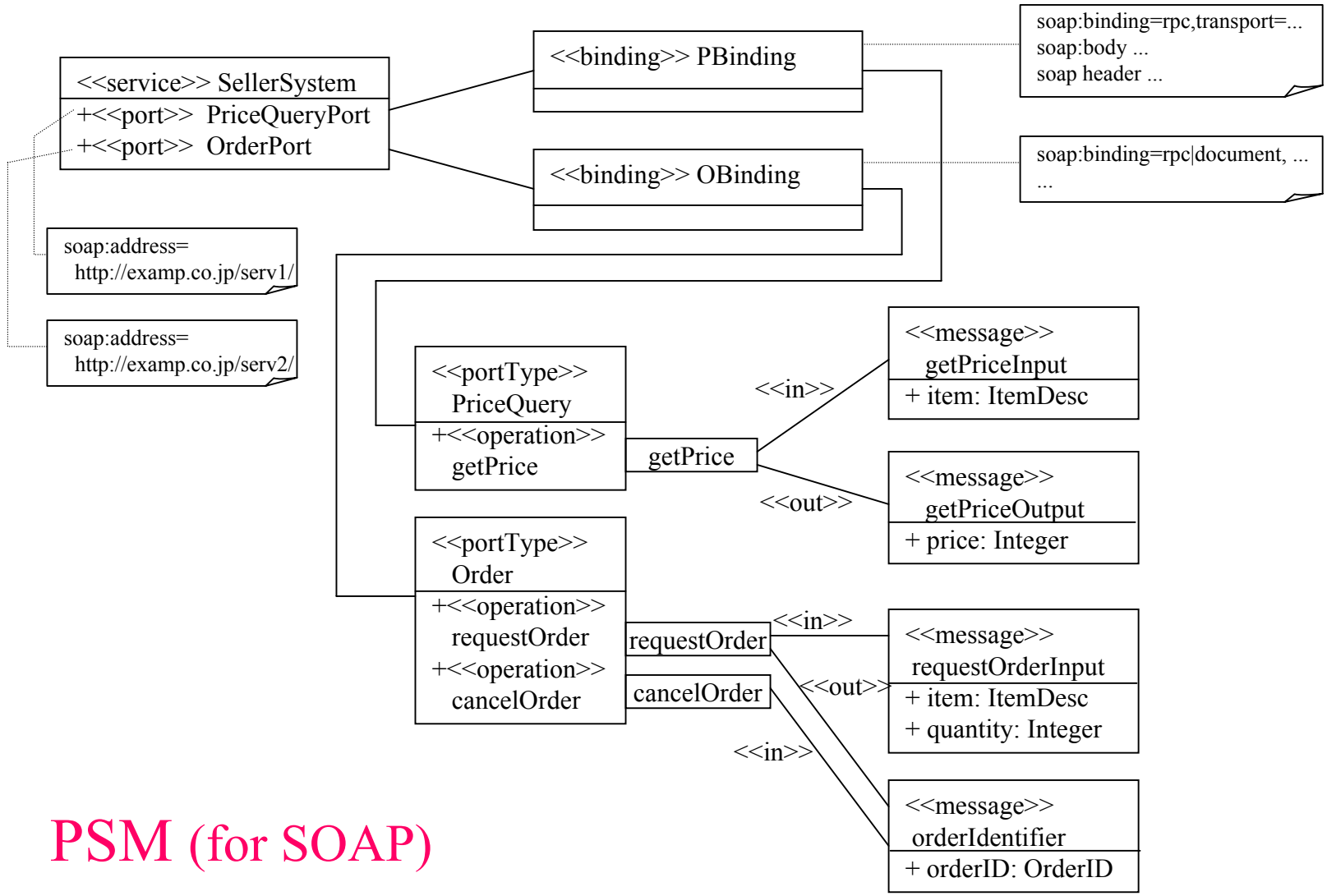
マッピング

PIM



PSM（SOAP用）　PIM




<<service>> SellerSystem
+<<port>> PriceQueryPort
+<<port>> OrderPort
<<binding>> PBinding
<<binding>> OBinding
soap:binding=rpc,transport=...
soap:body ...
soap header ...
soap:binding=rpc|document, ...
...
soap:address=
http://examp.co.jp/serv1/
soap:address=
http://examp.co.jp/serv2/
<<portType>>
PriceQuery
+<<operation>>
getPrice
getPrice
<<in>>
<<out>>
<<message>>
getPriceInput
+ item: ItemDesc
<<message>>
getPriceOutput
+ price: Integer
<<portType>>
Order
+<<operation>>
requestOrder
+<<operation>>
cancelOrder
requestOrder
cancelOrder
<<in>>
<<out>>
<<in>>
<<message>>
requestOrderInput
+ item: ItemDesc
+ quantity: Integer
<<message>>
orderIdentifier
+ orderID: OrderID

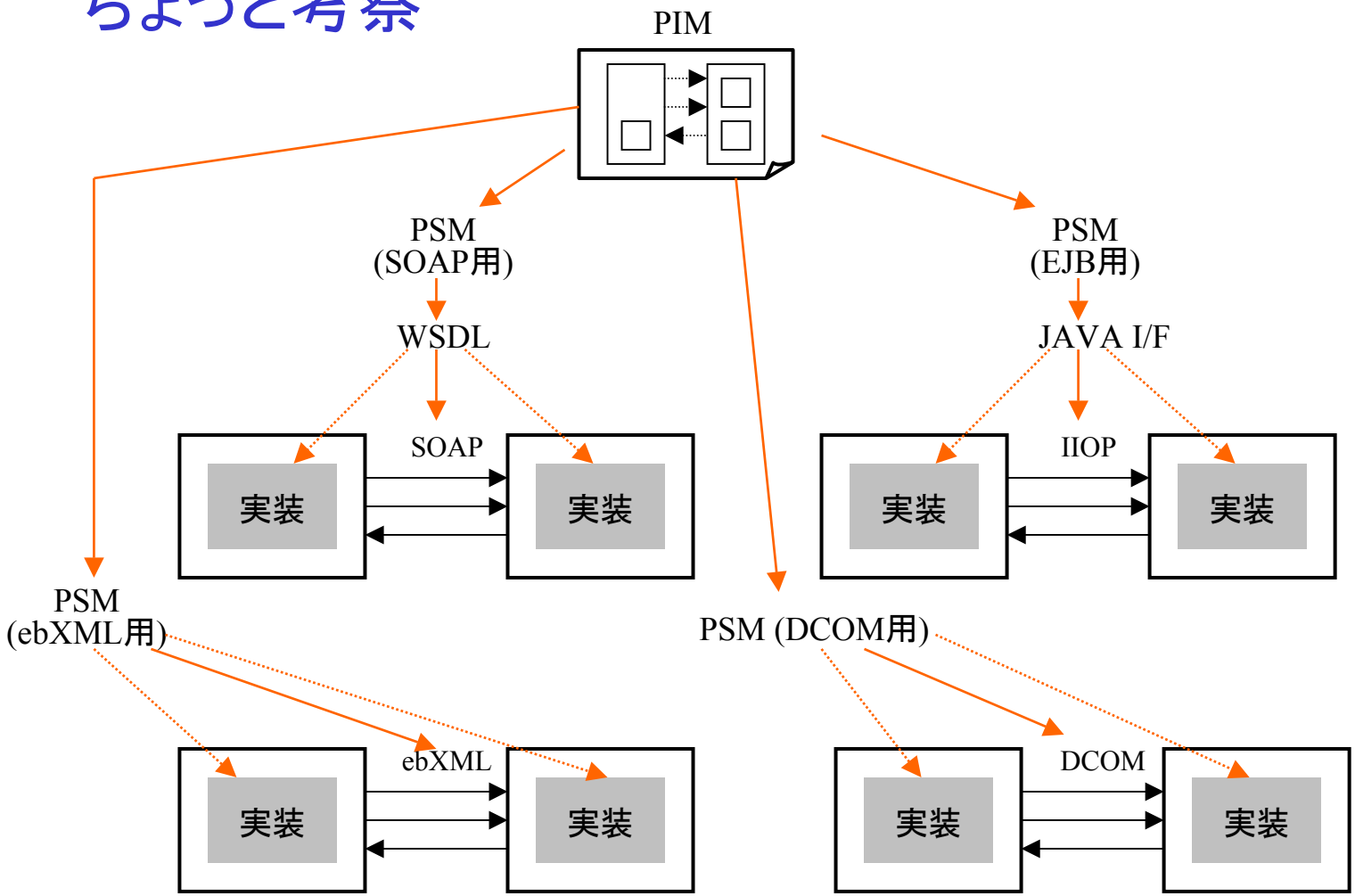
PSM (for SOAP)

WSDL

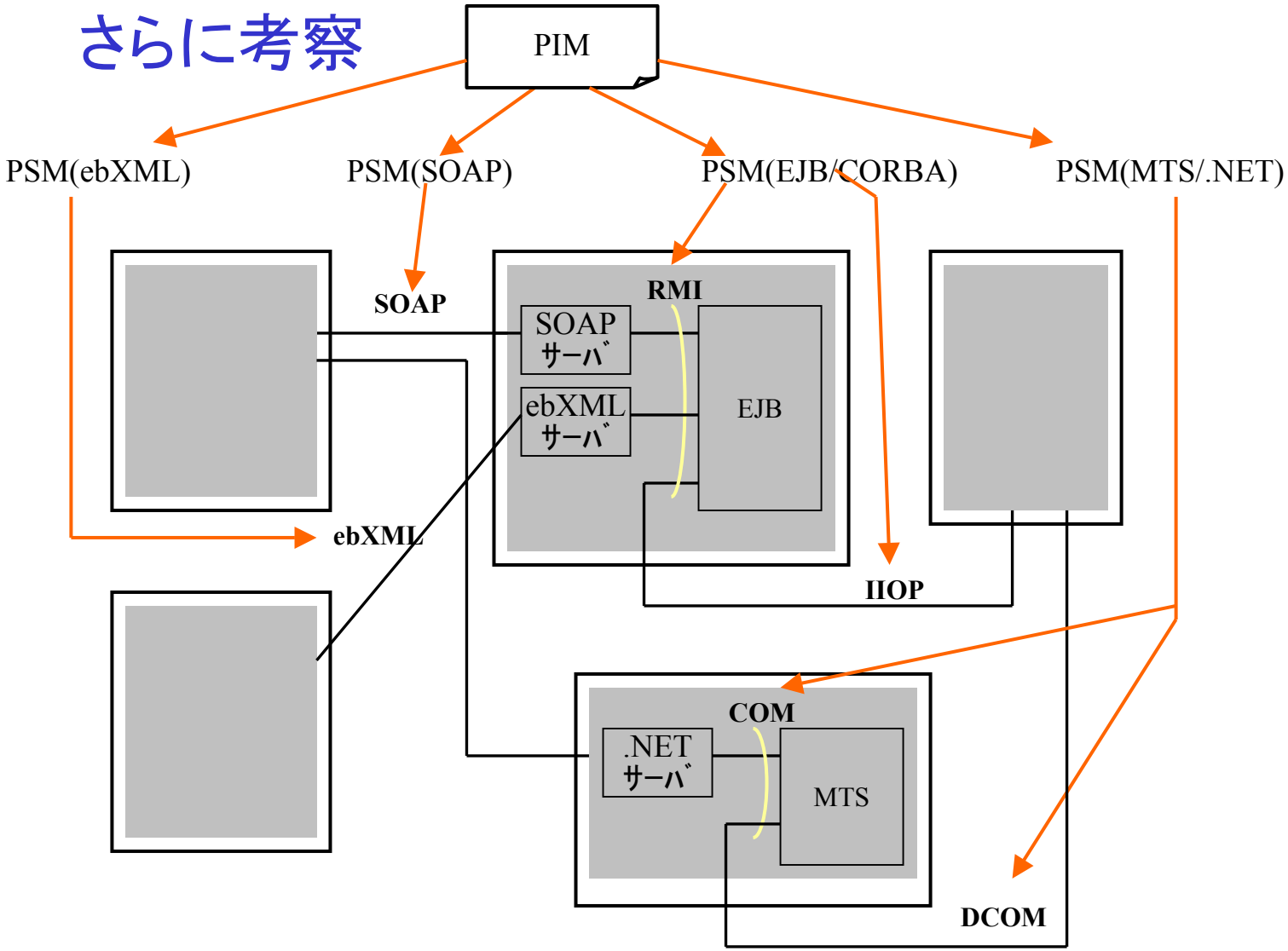
```
<definitions name="uri-BuySellSystem" ...
  xmlns:cd="uri-CommonDefinition" ...>
  <import namespace="uri-CommonDefinition"/>

  <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>
```

```
  <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>

  <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>
```

ちょっと考察

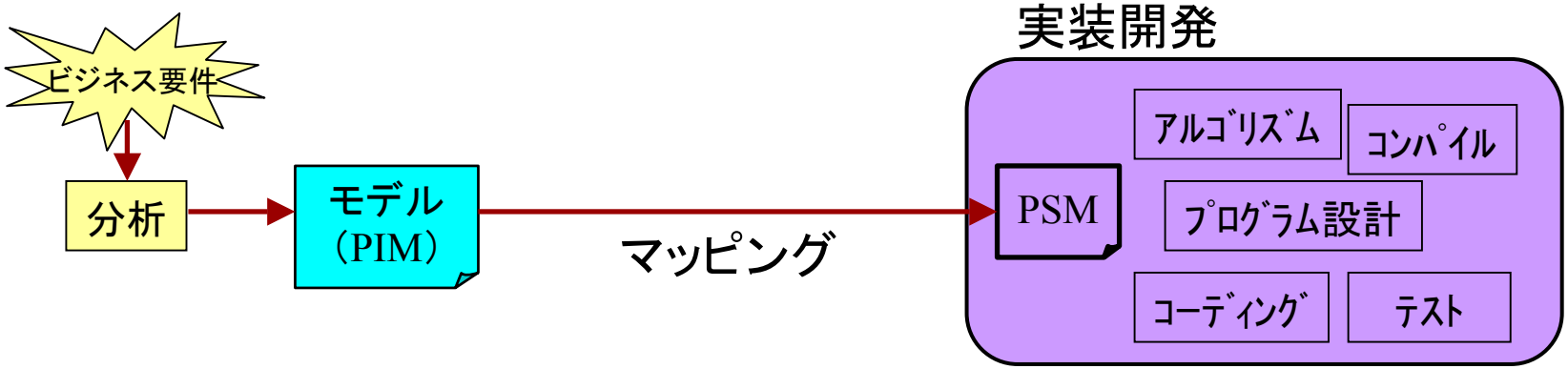


さらに考察



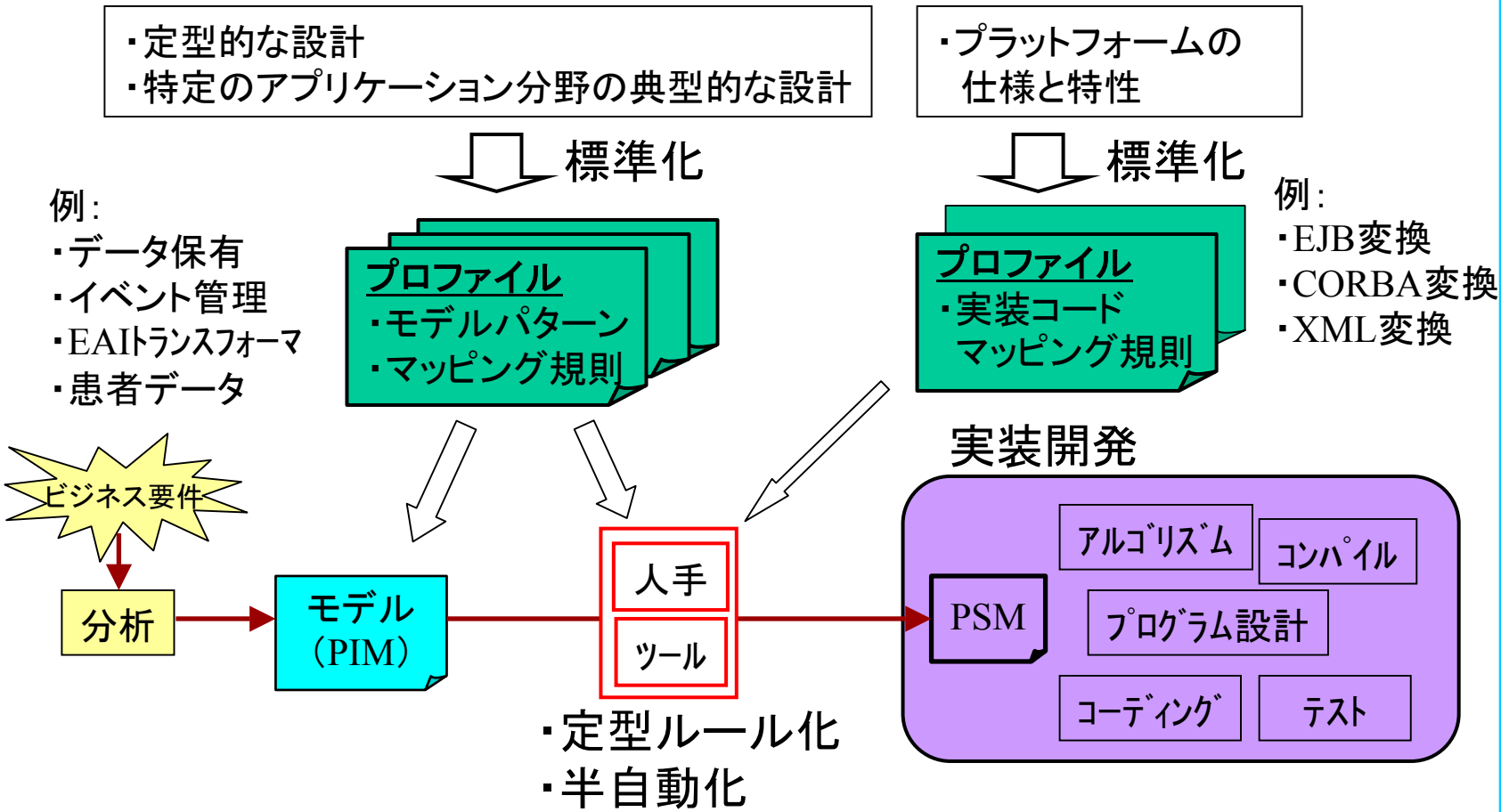
MDAの実現に向けて

MDAを実現するには

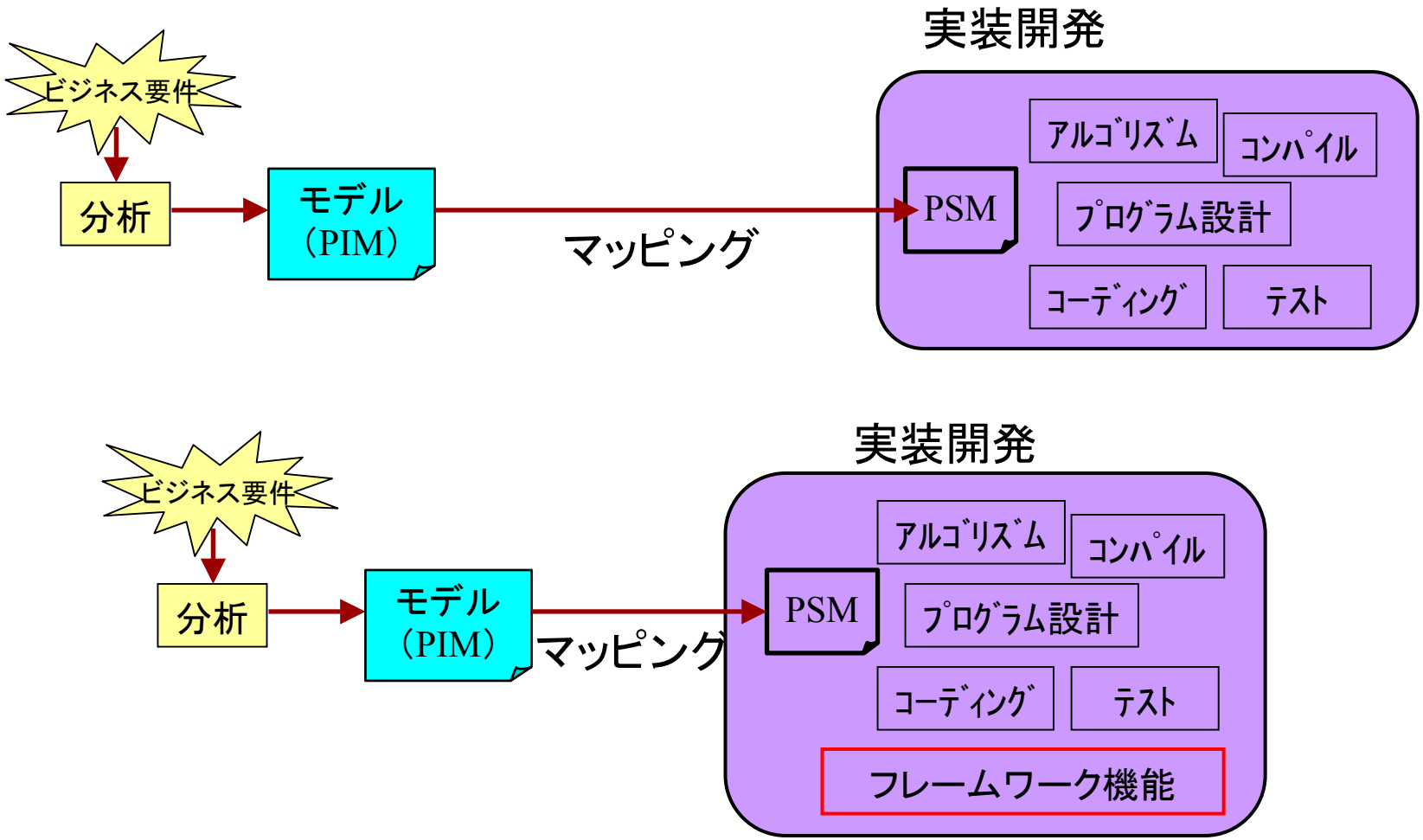


- PIMからPSMへのマッピングをどうするか
  - いままでのように、『アーキテクトの経験と実力』や『実装システム設計者・開発者の努力と涙』に頼るのか？
  - MDAの“キモ”:
    - 定型的／典型的なマッピングのルールを決める／標準化する
    - ツールによる半自動化をめざす
- PIMとPSMの“距離”を縮める
    - フレームワーク機能の充実

PIMからPSMへのマッピング

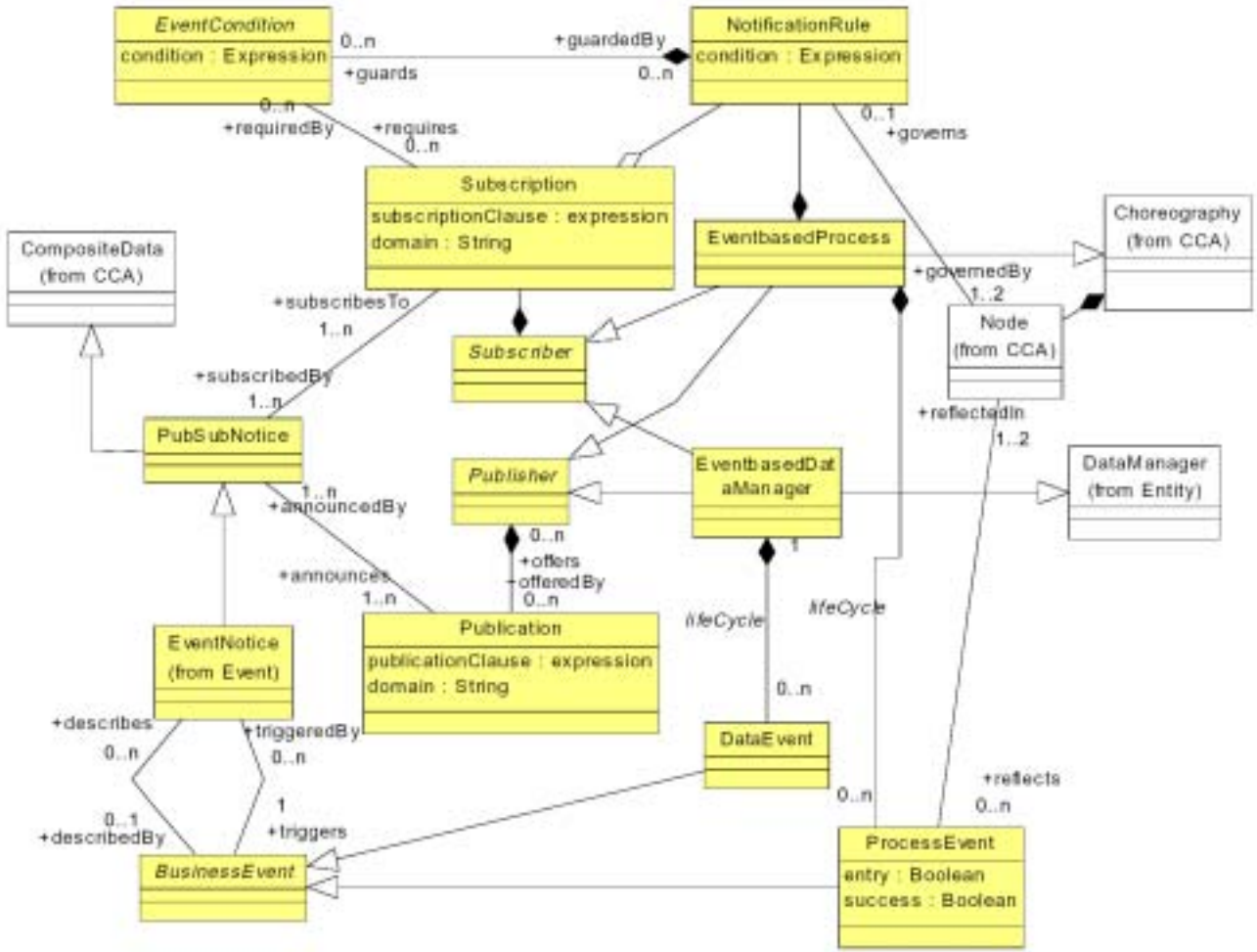


PIMとPSMの"距離"を縮める



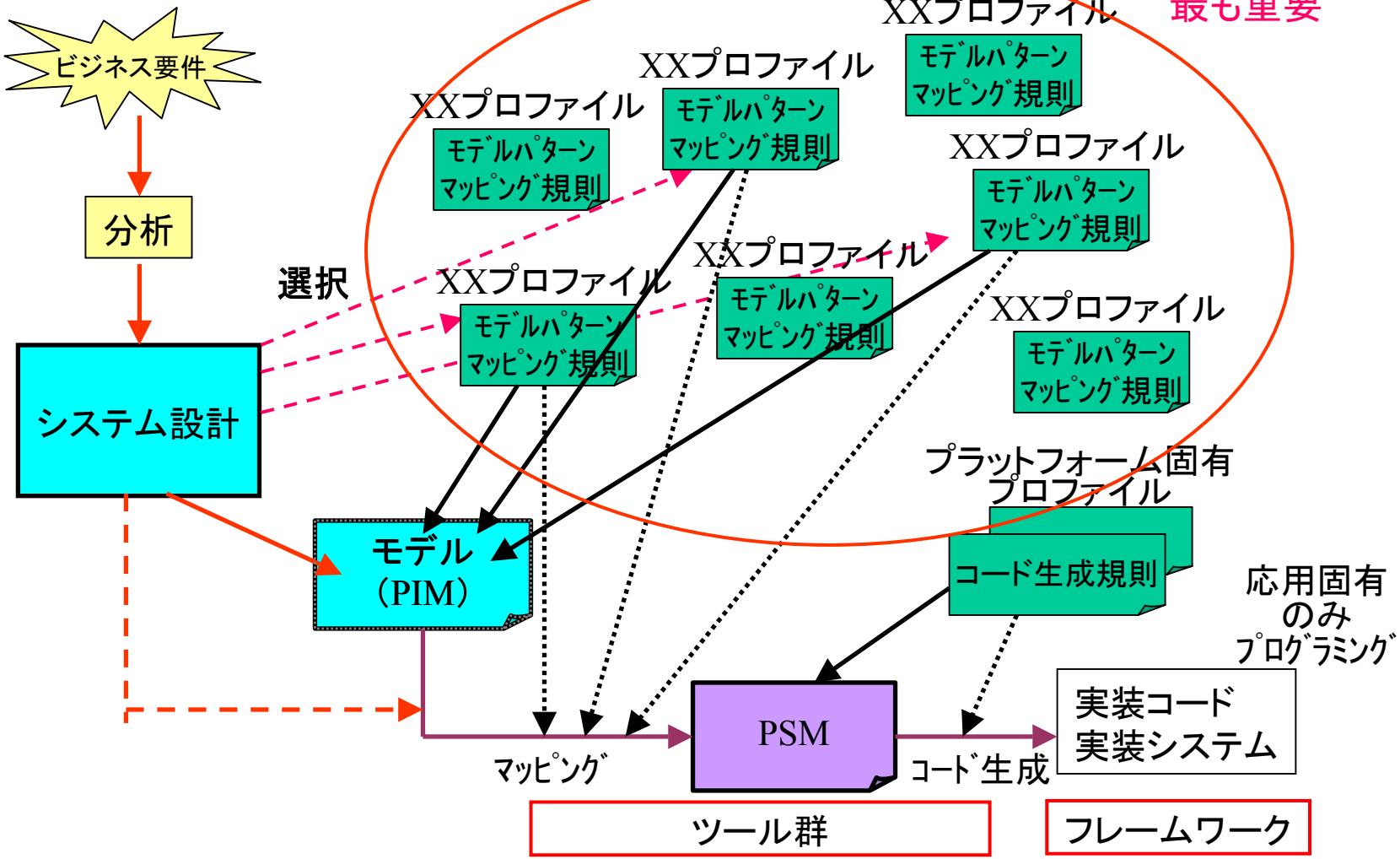
プロファイルの例

（UML Profile for EDOCのEvents Profileの規定）



究極の形

標準(国際/国内、業種内、企業内など)のプロファイルの蓄積が最も重要



補足：標準化完、標準化作業中、議論されている各種のプロファイル

OMG（標準化完）

- UML Profile for EAI (Enterprise Application Integration)
- UML Profile for EDOC (Enterprise Distributed Object Computing)
  - CCA (Component Collaboration Architecture)
  - Entities Profile
  - Events Profile
  - Business Process Profile
  - Relationship Profile
- UML Profile for Schedulability, Performance and Time
- UML Profile for CORBA

OMG（標準化作業中）

- UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms
- UML for Systems Engineering

JCP（標準化完）

- UML Profile for EJB (JCP)

その他（議論、話題、うわさ）

- UML Profile for WSDL
- UML Profile for XML Schema
- UML Profile for Persistence Model
- UML Profile for Reverse Engineering
- UML Profile for Framework Architectures
- UML Profile for DCL
- UML Profile for Business Modeling
- UML profile for Business Analysis
- UML Profile for .NET
- UML profile for Interaction design
- UML Profile for Database Design
- UML profile for hypermedia
- UML for Ontology Development
- UML profile for DAML
- UML Profile for Web applications

まとめ

MDAサマリ

- PIMとPSM
- ２種類のマッピング：PIM→PSM、PSM→実装
- PIM作成者のために、アプリケーションに対応した各種標準プロファイルを準備。PSMへの標準マッピングも同時に規定
- PIM→PSMは半自動化、PSM→実装は自動化
- 現実の実装開発に直結し、かつ、プラットフォーム非依存なシステム設計を可能にする
- 多種多様な標準プロファイル開発がMDAの最重要事項だ
  国際標準、国内標準、業界標準、団体標準、企業/組織内標準、部門標準、インテグレーター標準、ベンダー標準、個人標準
  - CBOPのMDAプロジェクト構想

まとめ

- ミドルウェアプラットフォームは今後も複数存在し、かつ、最新技術を吸収しつつ進化し続けるだろう。
  - CORBA/Javaや.NETは進化し続けるだろう。
  - Webサービスも進化し続けるだろう。
  - 新たなミドルウェアも登場するだろう。
  - ユーザは、先端の技術を駆使し、最新のITシステムを維持したい。
- ビジネス要求も次々変化する。最新のビジネス要求を迅速に企業ITシステムに実装していくことが必要だ。
- ビジネス進化と技術進化のそれぞれに応じてPIMとPSMは別々に設計・開発したい。ただし、両者が乖離しては困る。また、生産性を上げてくれるやり方でないと困る。
- MDAがこの理想を実現するカギだ。

END