Modeling Server-side Components with UML

Junichi Suzuki, Ph.D.

jxs@computer.org

http://www.jks.la/jxs/

School of Information and Computer Science
University of California, Irvine

Who am I?

- Research fellow, UC Irvine (2000-)
  - biologically-inspired software designs for scalable and adaptable distributed computing
- Ph.D. from Keio U (2001)
- ex- Technical director, Object Management Group Japan
- ex.ex- Technical director, Soken Planning Co., Ltd.

Where is UC Irvine?

- UCI (U of California, Irvine)
  - One of eight UC system universities
- Irvine
  - in between LA and San Diego
  - reported by FBI, as the safest city in the US
  - 1 hour to LA downtown
  - 10 minutes to Newport Beach
  - 20 minutes to Huntington Beach
  - 20 minutes to Anaheim Disneyland
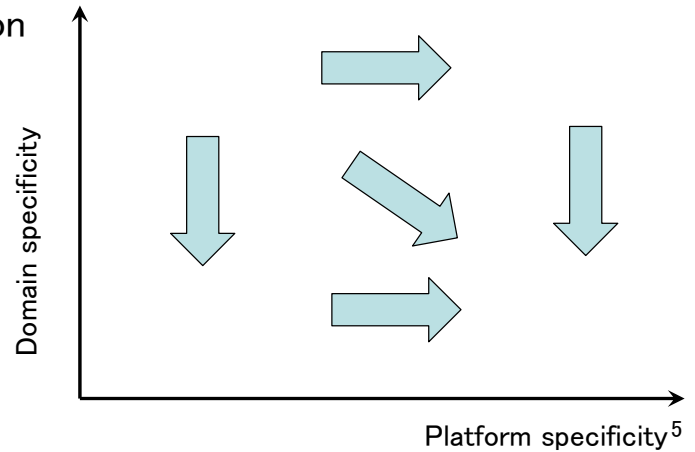 - 5 hours to Las Vegas

Overview

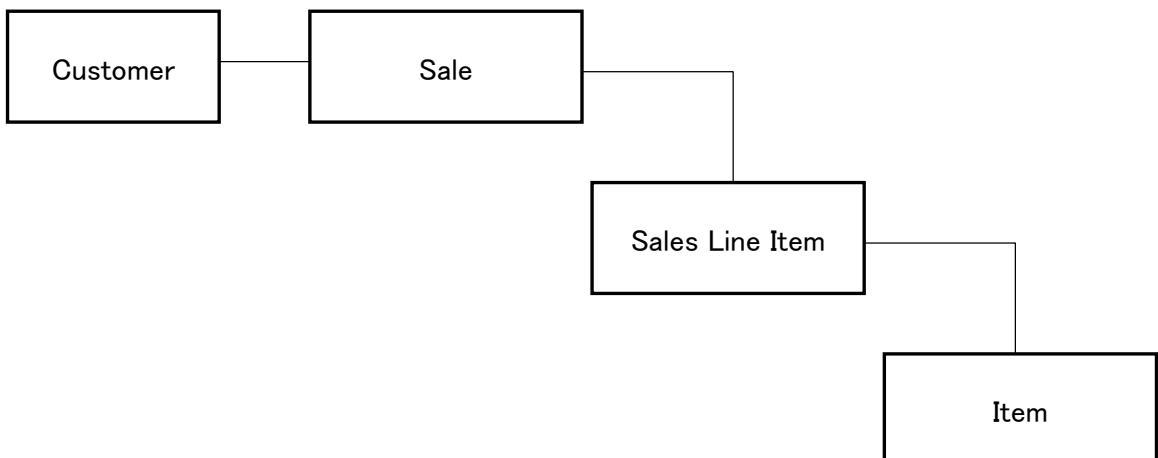- UML Profiles
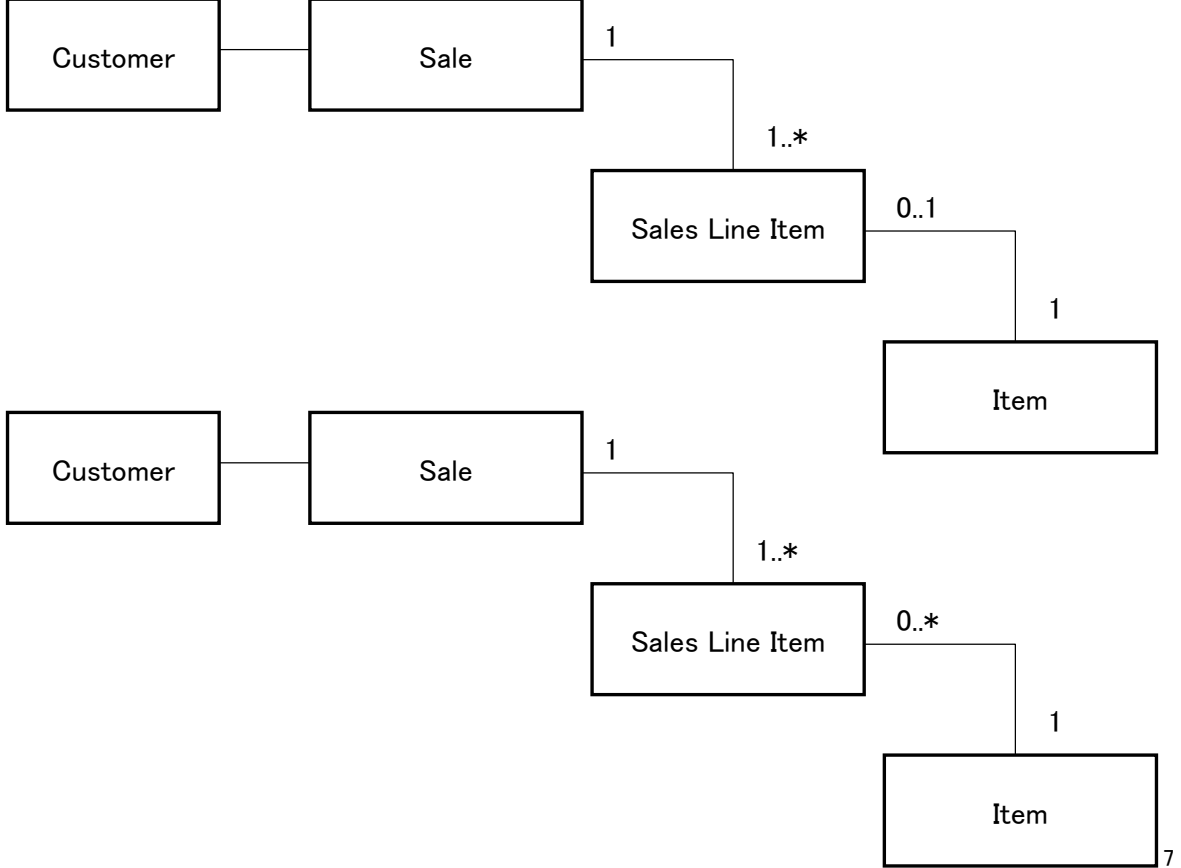- UML Profile for EJB
- UML Profiles and MDA

Model Transformation

- 2 dimensions of model transformation
  - Domain specialization
  - Platform specialization
- Several forms of model transformation
  - Manual transformation
  - Automatic transformation

Platform specificity[5]

An Example of Manual Domain Specialization

Diagram (top half)

```
┌──────────┐   ┌──────────┐
│ Customer │───│   Sale   │  1
└──────────┘   └──────────┘ ┐
                            │
                      1..*  │
              ┌──────────────┐  0..1
              │Sales Line Item│──────────┐
              └──────────────┘           │
                                      1  │
                                  ┌──────────┐
                                  │   Item   │
                                  └──────────┘
```

```
┌──────────┐   ┌──────────┐
│ Customer │───│   Sale   │  1
└──────────┘   └──────────┘ ┐
                            │
                      1..*  │
              ┌──────────────┐  0..*
              │Sales Line Item│──────────┐
              └──────────────┘           │
                                      1  │
                                  ┌──────────┐
                                  │   Item   │
                                  └──────────┘ 7
```

Technologies for Model Transformations

- UML profiles
  - for EJB
  - for CORBA
  - for Realtime scheduling
- Action semantics
  - allows modelers to embed actions (behaviors) into model elements.
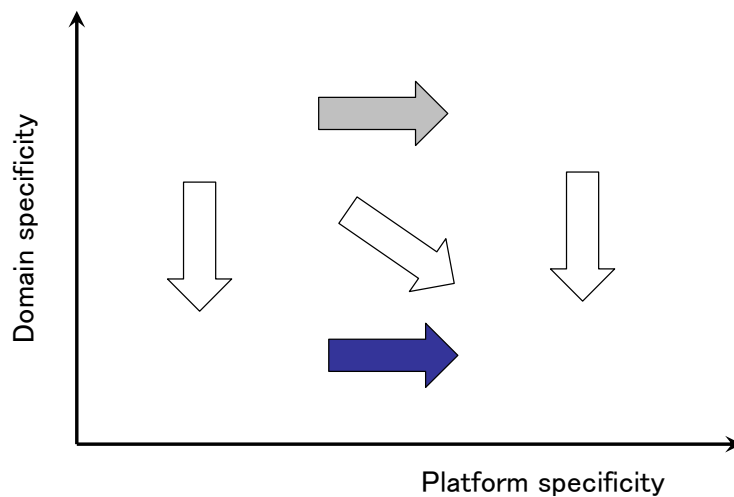
UML Profiles

- A UML profile
  - provides a means to specialize UML models to a specific domain or implementation technology.
  - is defined with the UML extension mechanisms
    - i.e. stereotypes, tag definition/tagged values, and constraints
  - may extend the UML standard meta model.
    - virtual meta model

UML Profile for EJB

- used for specializing platform independent models to EJB specific models

Domain specificity

Platform specificity

- What *UML Profile for EJB* defines include:
  - Design model
    - Java design model
    - EJB design model
      - External model
      - Internal model
  - Implementation model
    - Java implementation model
    - EJB implementation model

Java Design Model

- Defines UML representations of Java language constructs
  - Java class, interface, etc.
- Java package
  - mapped to a UML package
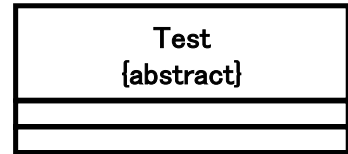 - e.g. *package edu.uci.ics;*

```
┌──────────────┐
│ edu::uci::ics │
├──────────────┴──┐
│                 │
│                 │
│                 │
└─────────────────┘
```
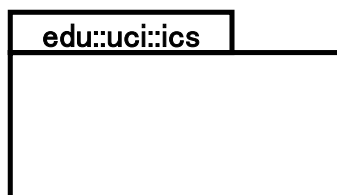
- Java Class
  - mapped to a UML class
  - e.g. *public abstract class Test {}*

Test {abstract}

- Java Interface
  - mapped to a UML interface or UML class stereotyped as <<JavaInterface>>.
  - e.g. *public interface Test {}*

<<JavaInterface>> Test

- Java method
  - mapped to a UML operation
  - e.g. *public void test() throw Foo{}*
    - + test(): void {JavaThrows=Foo}
- Others
  - Single type import
  - On demand type import
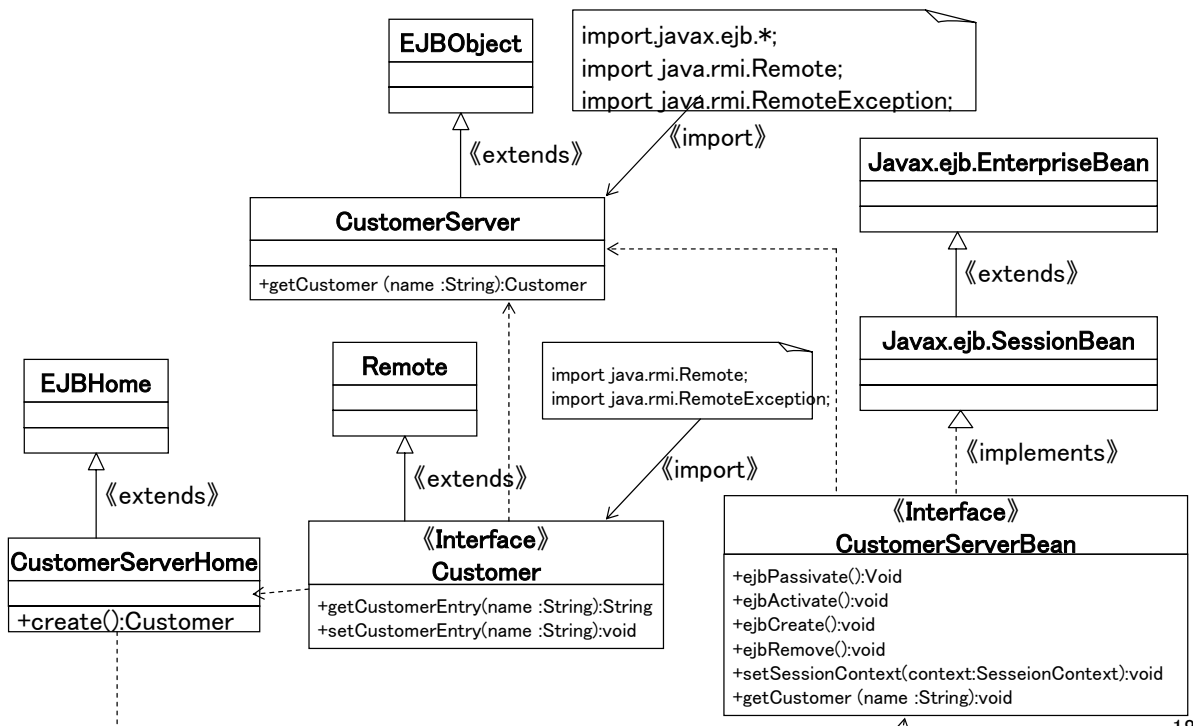
EJB Design Model

- Defines UML representations of EJB specific constructs
  - e.g. EJB remote interface, home interface, etc.
  - External view
    - Defines logical constructs visible to the clients of an EJB Enterprise Bean
  - Internal view
    - Defines logical constructs visible to the developers of an EJB Enterprise Bean

EJB Design Model: External View

- EJB remote interface
  - Mapped to a UML class stereotyped as ≪EJBRemoteInterface≫.
- EJB home interface
  - Mapped to a UML class stereotyped as ≪EJBHomeInterface≫.
- EJB session home
  - Mapped to a UML class stereotyped as ≪EJBSessionHomeInterface≫.
- EJB entity home
  - Mapped to a UML class stereotyped as ≪EJBEntityHomeInterface≫.

Slide 17

```
┌─────────────────────────────┐
│         EJBHome             │
├─────────────────────────────┤
│                             │
└─────────────────────────────┘
              △
              │  <<extends>>
┌─────────────────────────────┐
│       CustomerHome          │
├─────────────────────────────┤
│                             │
├─────────────────────────────┤
│     create(): Customer      │
└─────────────────────────────┘
```

```
┌─────────────────────────────┐
│   <<EJBHomeInterface>>      │
│       CustomerHome          │
├─────────────────────────────┤
│                             │
├─────────────────────────────┤
│    +create():Customer       │
└─────────────────────────────┘
```

Slide 18

EJBObject

import.javax.ejb.*;
import java.rmi.Remote;
import java.rmi.RemoteException;

《extends》 《import》

Javax.ejb.EnterpriseBean

CustomerServer

+getCustomer (name :String):Customer

《extends》

Javax.ejb.SessionBean

EJBHome

Remote

import java.rmi.Remote;
import java.rmi.RemoteException;

《import》

《implements》

《extends》

《extends》

CustomerServerHome

+create():Customer

《Interface》
Customer

+getCustomerEntry(name :String):String
+setCustomerEntry(name :String):void

《Interface》
CustomerServerBean

+ejbPassivate():Void
+ejbActivate():void
+ejbCreate():void
+ejbRemove():void
+setSessionContext(context:SesseionContext):void
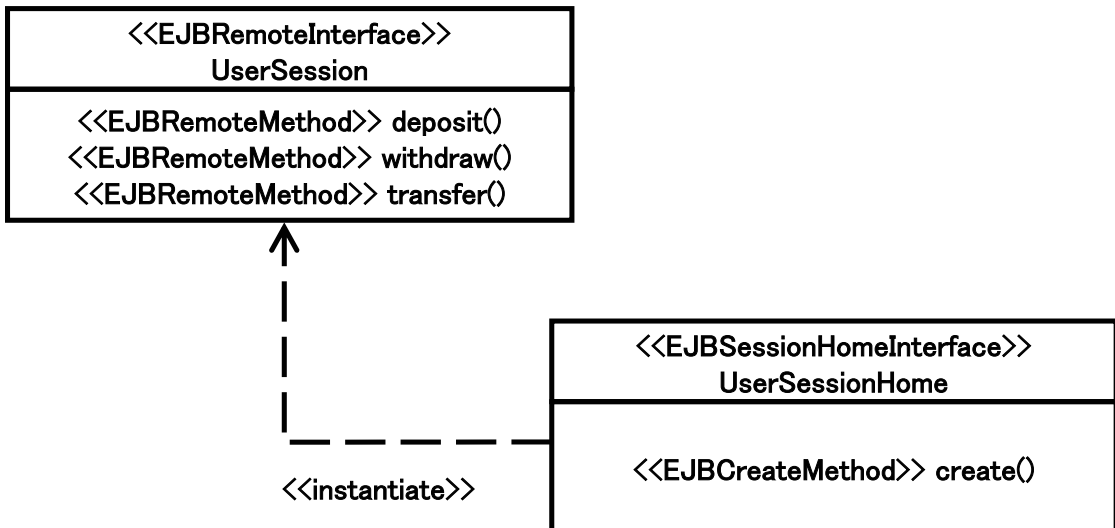+getCustomer (name :String):void

- EJB Method
  - Means methods declared in EJB Remote and Home interfaces
  - Mapped to a UML operation
  - <<EJBCreateMethod>>
    - Represents a create method in a home interface
  - <<EJBFinderMethod>>
    - Represents a finder interface in a home interface
  - <<EJBRemoteMethod>>
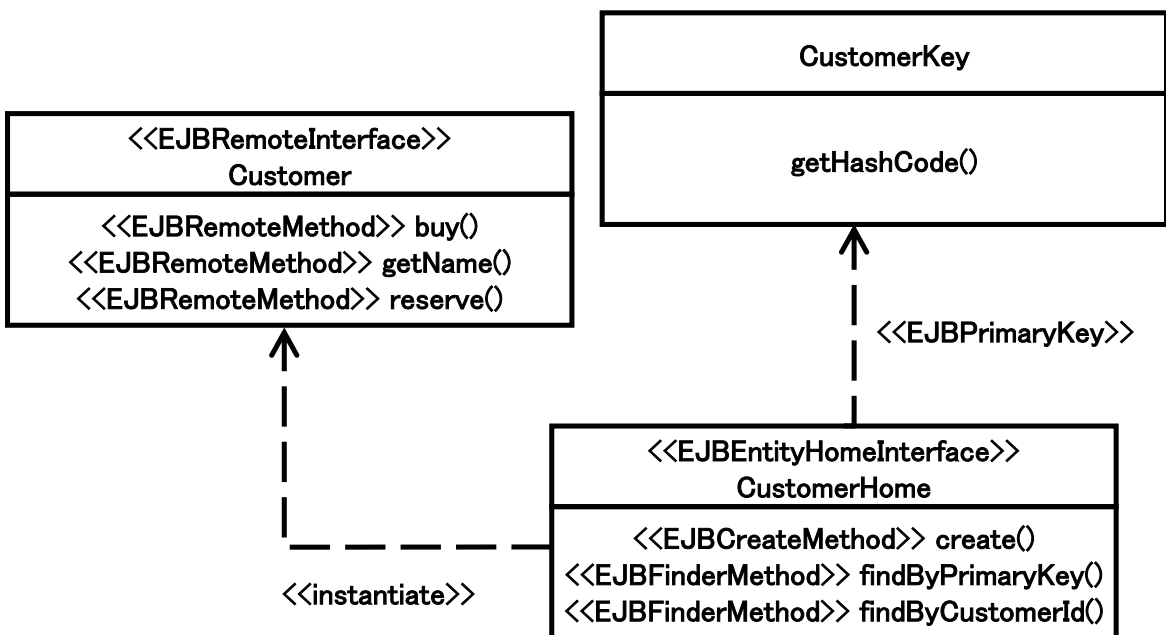   - Represents a method in a remote interface.

<<EJBRemoteInterface>> UserSession
<<EJBRemoteMethod>> deposit() <<EJBRemoteMethod>> withdraw() <<EJBRemoteMethod>> transfer()

<<EJBSessionHomeInterface>> UserSessionHome
<<EJBCreateMethod>> create()

<<instantiate>>

- EJB primary key
  - Mapped to a UML usage association stereotyped as
    <<EJBPrimaryKey>>.
    - between EJB primary key class and EJB entity home

CustomerKey
getHashCode()

<<EJBRemoteInterface>> Customer
<<EJBRemoteMethod>> buy() <<EJBRemoteMethod>> getName() <<EJBRemoteMethod>> reserve()

<<EJBPrimaryKey>>

<<EJBEntityHomeInterface>> CustomerHome
<<EJBCreateMethod>> create() <<EJBFinderMethod>> findByPrimaryKey() <<EJBFinderMethod>> findByCustomerId()

<<instantiate>>

EJB Design Model: Internal View

- EJB enterprise bean
  - Mapped to a UML subsystem stereotyped as 《EJBEnterpriseBean》.
- EJB session bean
  - Mapped to a UML subsystem stereotyped as 《EJBSessionBean》.
- EJB entity bean
  - Mapped to a UML subsystem stereotyped as 《EJBEntityBean》
  - 《EJBCmpField》 represents a container-managed field (attribute).

《EJBEntityBean》

Customer

| Specification elements | Realization elements |
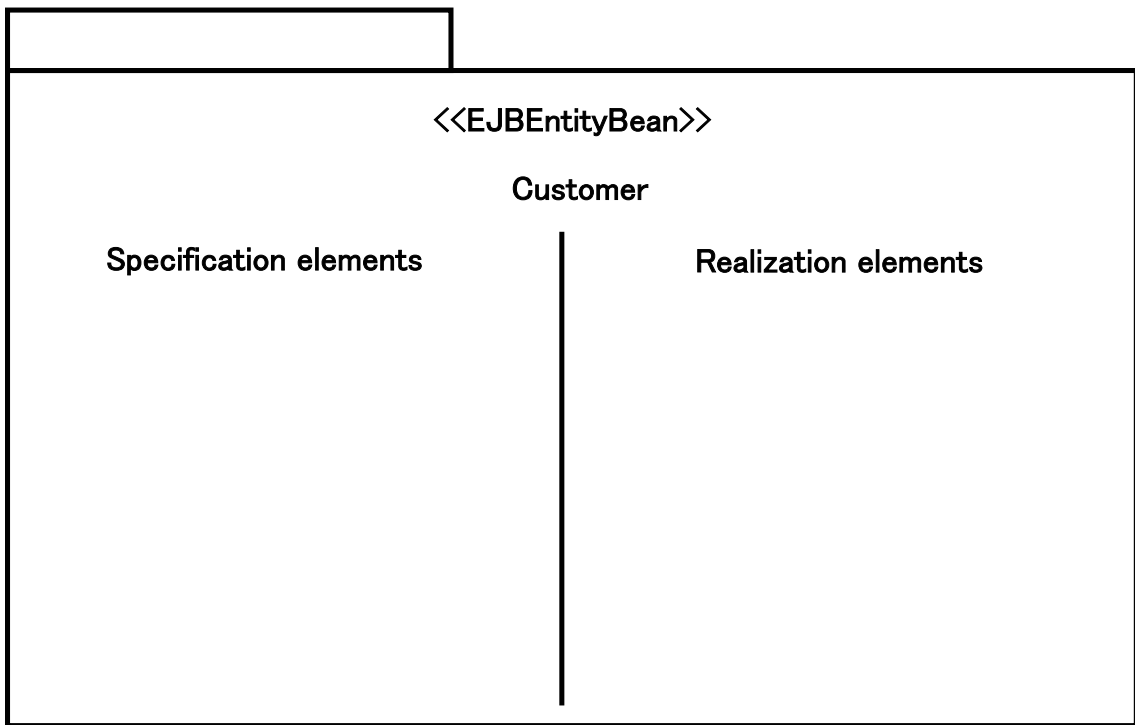
- EJB enterprise bean is declared by
  - an EJB home interface,
  - an EJB remote interface,
  - an EJB implementation class
  - Supplemental Java classes and interfaces, and
  - EJB deployment descriptor.

- EJB implementation class
  - Mapped to a UML class stereotyped as ≪EJBImplementation≫.
- EJB remote interface
  - Mapped to a UML abstraction association stereotyped as ≪EJBRealizeRemote≫.
    - between EJB remote interface and EJB implementation class.
- EJB home interface
  - Mapped to a UML abstraction association stereotyped as ≪EJBRealizeHome≫.
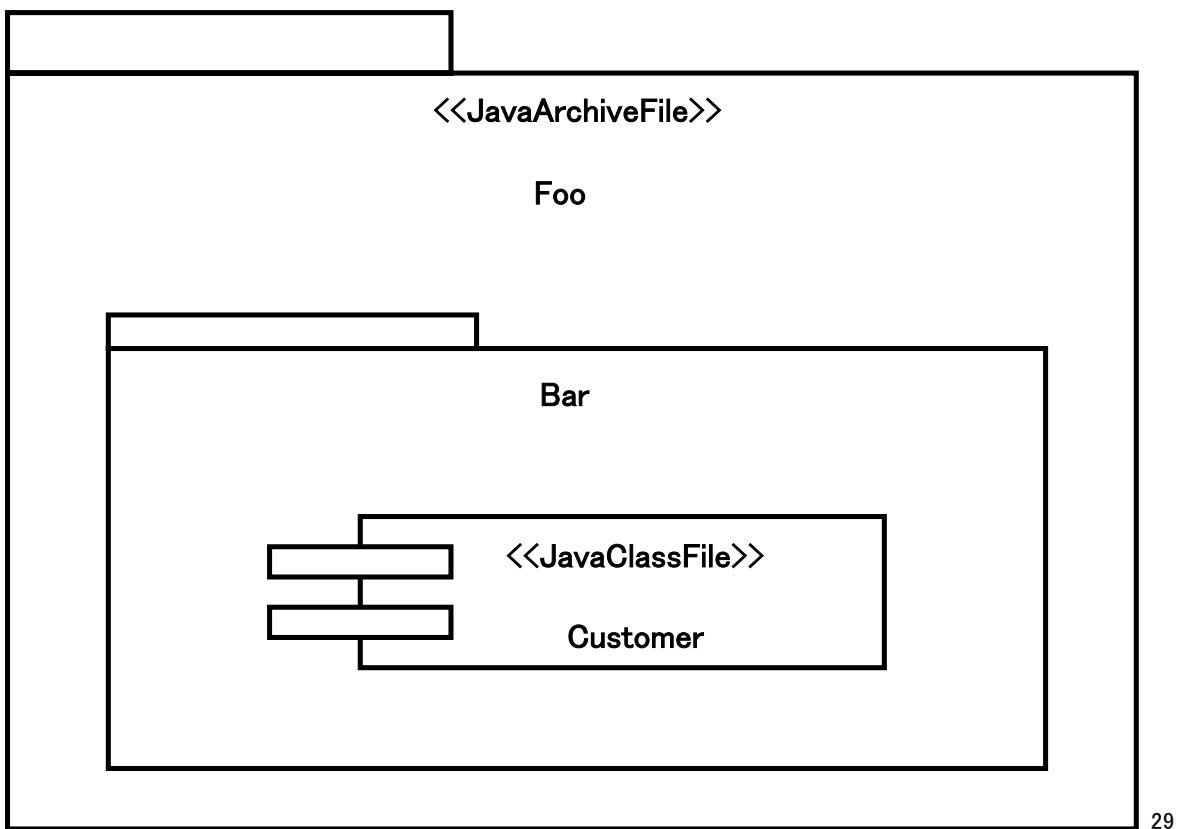   - between EJB home interface and EJB implementation class.
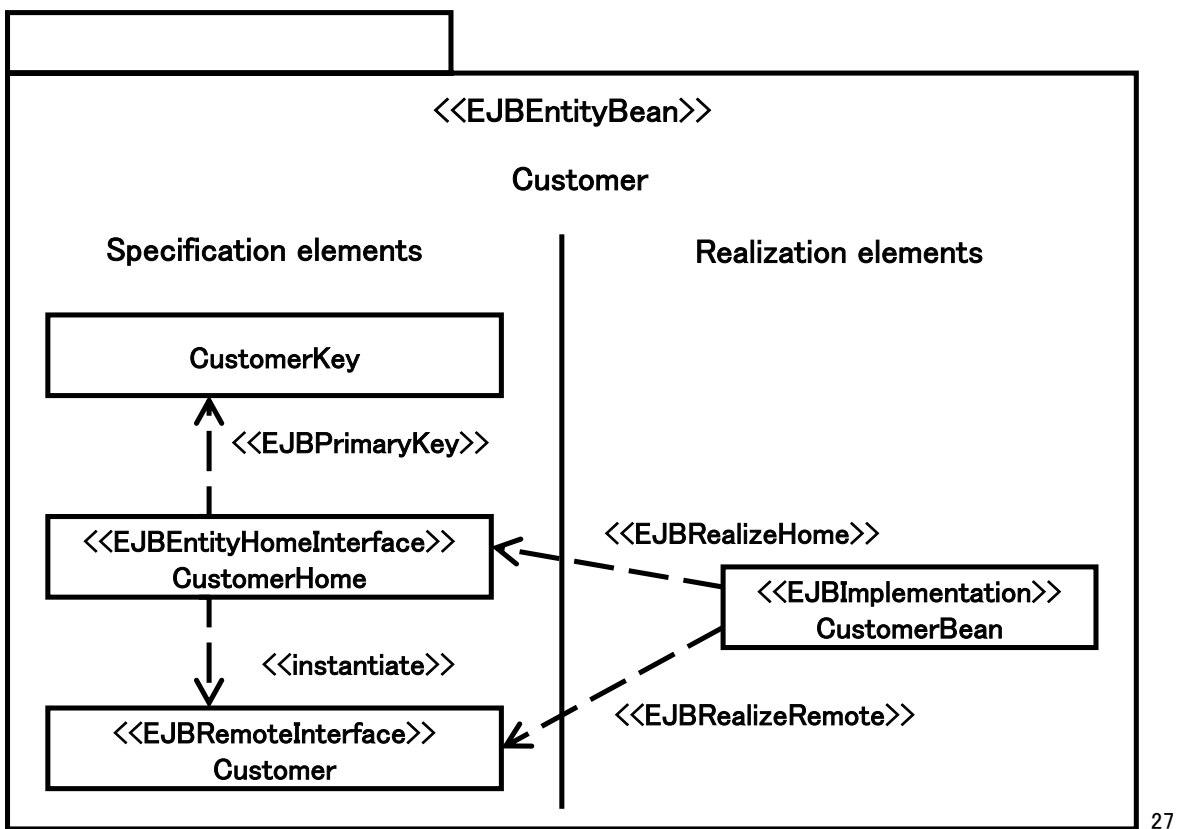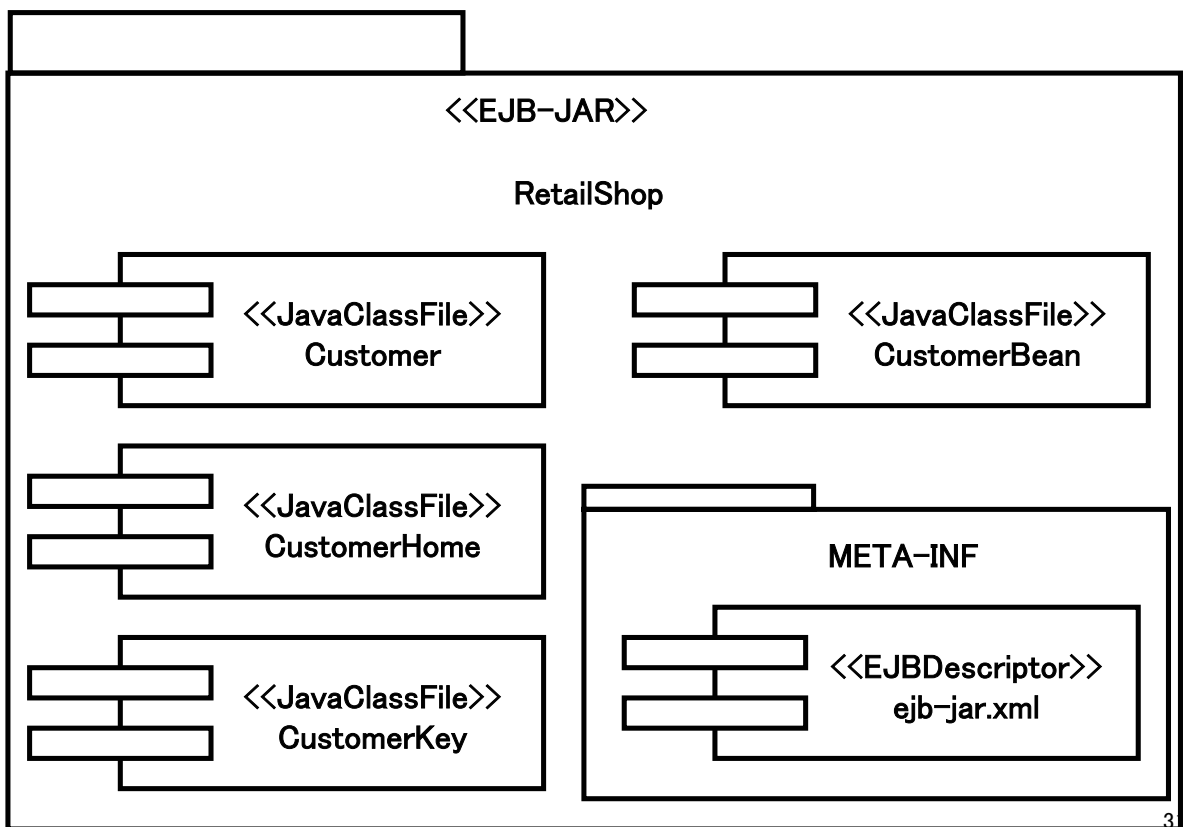
Java Implementation Model

- Java class file
  - Mapped to a UML component stereotyped as ⟪JavaClassFile⟫.

- Java archive (JAR) file
  - Mapped to a UML package stereotyped as ⟪JavaArchiveFile⟫.

Slide 29

```
┌─────────────────────────────────────────────┐
│  <<JavaArchiveFile>>                          │
│                                               │
│         Foo                                   │
│   ┌──────────────────────────────────────┐   │
│   │         Bar                           │   │
│   │                                       │   │
│   │   ┌──┬─────────────────────────┐      │   │
│   │   ├──┤  <<JavaClassFile>>       │      │   │
│   │   └──┤                          │      │   │
│   │      │  Customer                │      │   │
│   │      └──────────────────────────┘      │   │
│   └──────────────────────────────────────┘   │
└───────────────────────────────────────────────┘
```

Slide 30

EJB Implementation Model
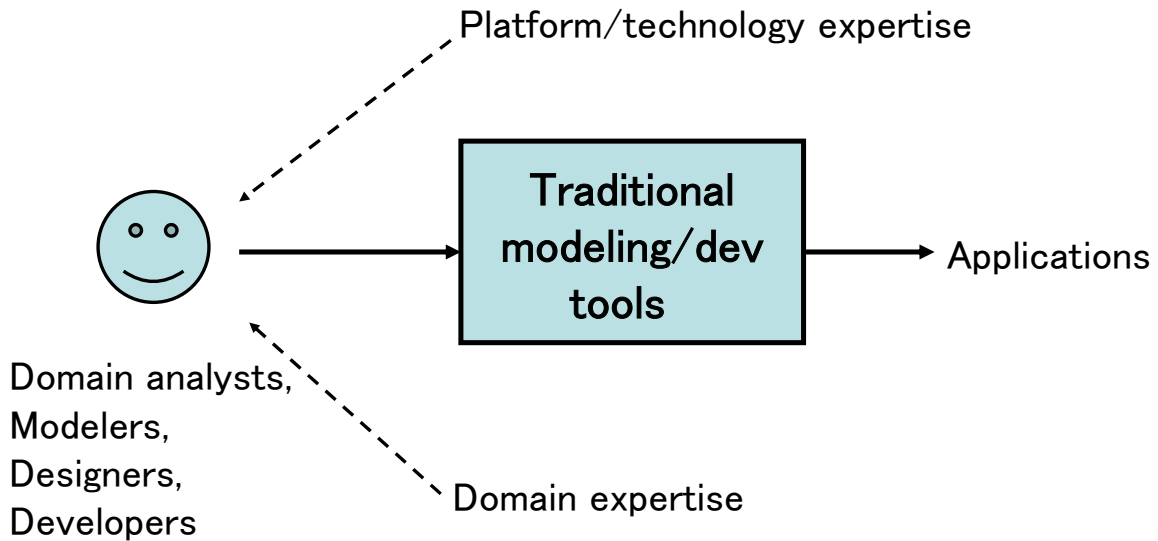
- EJB-JAR
  - Mapped to a UML package stereotyped as <<EJB-JAR>>
- EJB deployment descriptor
  - Mapped to a UML component stereotyped as <<EJBDescriptor>>

Slide (UML Component Diagram)

<<EJB-JAR>>

RetailShop

<<JavaClassFile>>
Customer

<<JavaClassFile>>
CustomerBean

<<JavaClassFile>>
CustomerHome

META-INF

<<JavaClassFile>>
CustomerKey

<<EJBDescriptor>>
ejb-jar.xml

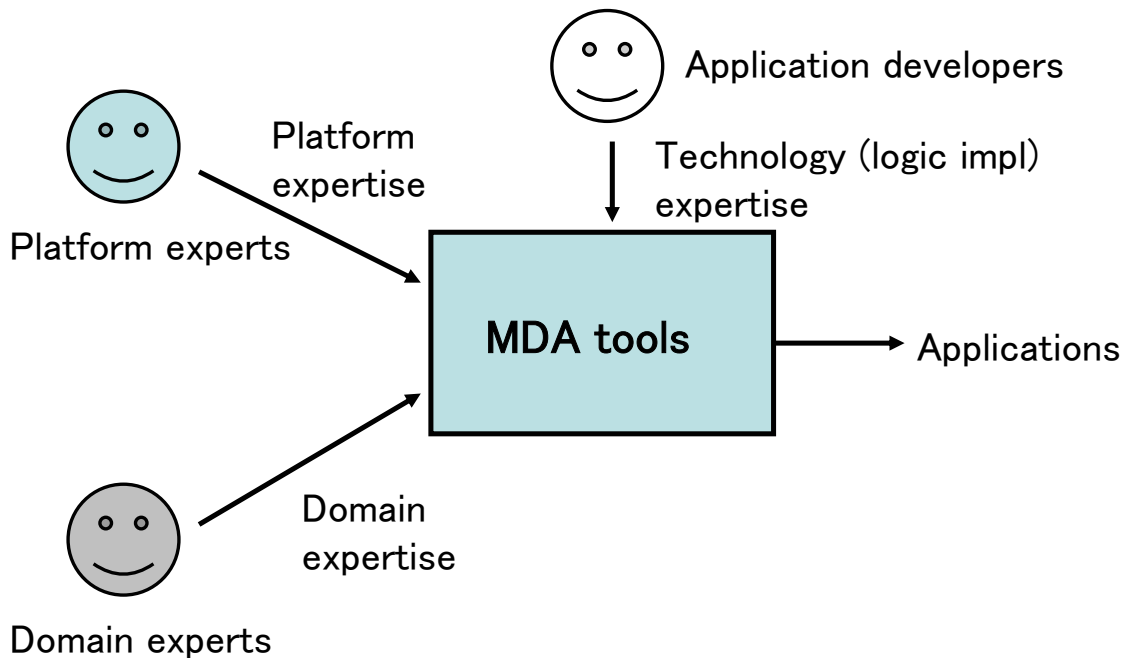
UML Profiles and MDA

- UML profiles
  - are key components to achieve MDA vision.
  - defines a set of mapping rules for model transformations.

Traditional Modeling and Development

Platform/technology expertise

Traditional modeling/dev tools

Applications

Domain analysts,
Modelers,
Designers,
Developers

Domain expertise

MDA-based Modeling and Development

Application developers

Platform expertise

Technology (logic impl) expertise

Platform experts

MDA tools

Applications

Domain expertise

Domain experts

Goals in MDA

- Model continuation
  - Maximizing model continuation during software development process.
- Separation of concerns
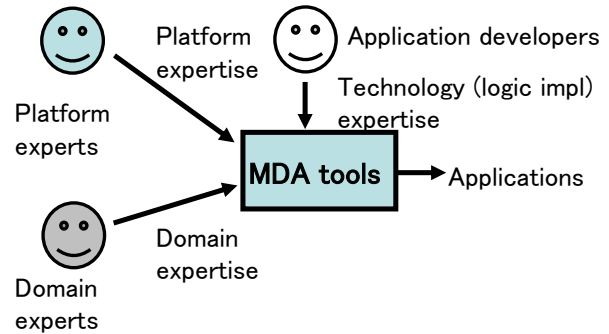 - Maximizing separation of concerns

Benefits from MDA

- Reduced software development cost
- Reduced software development time
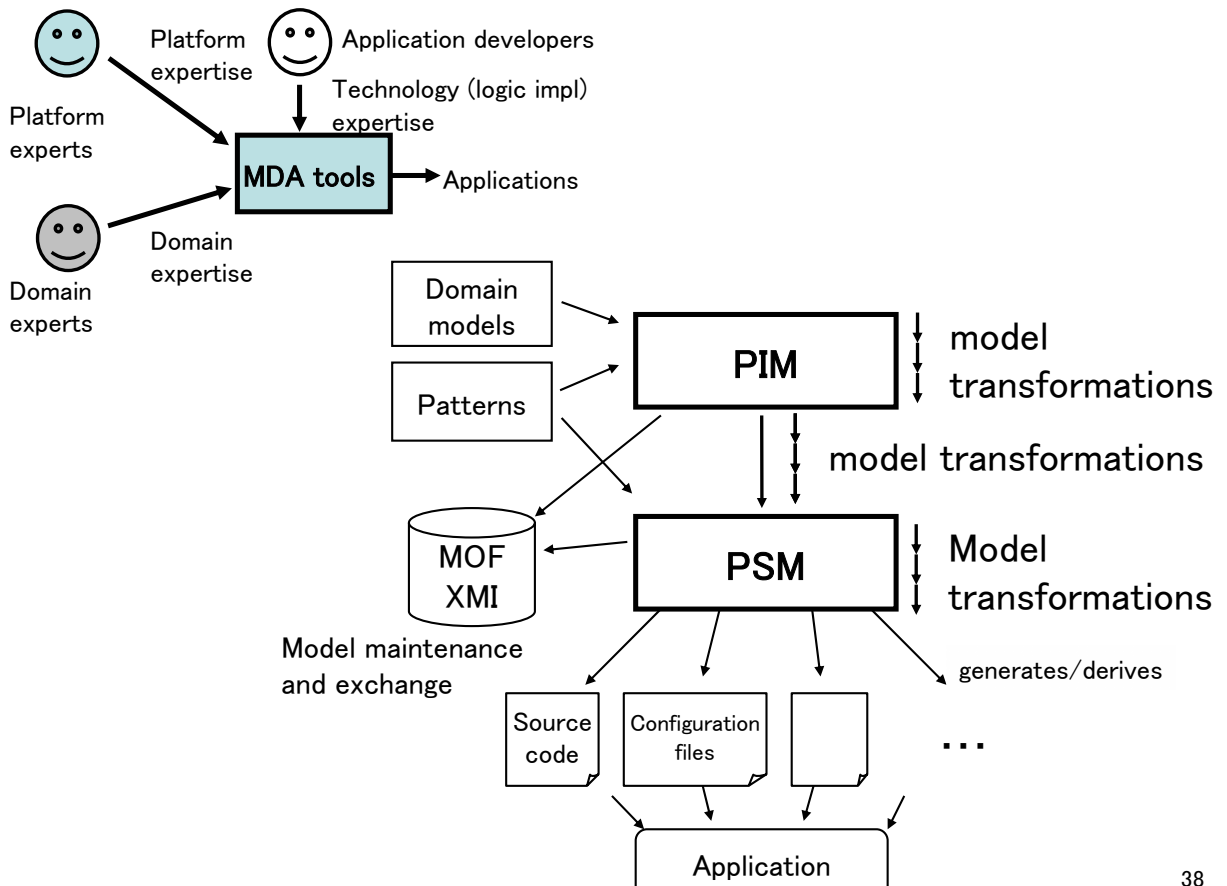- Rapid and smooth integration of legacy and emerging technologies

Model Transformation
and Integration

- Model transformation
  - Domain specialization
  - Platform specialization
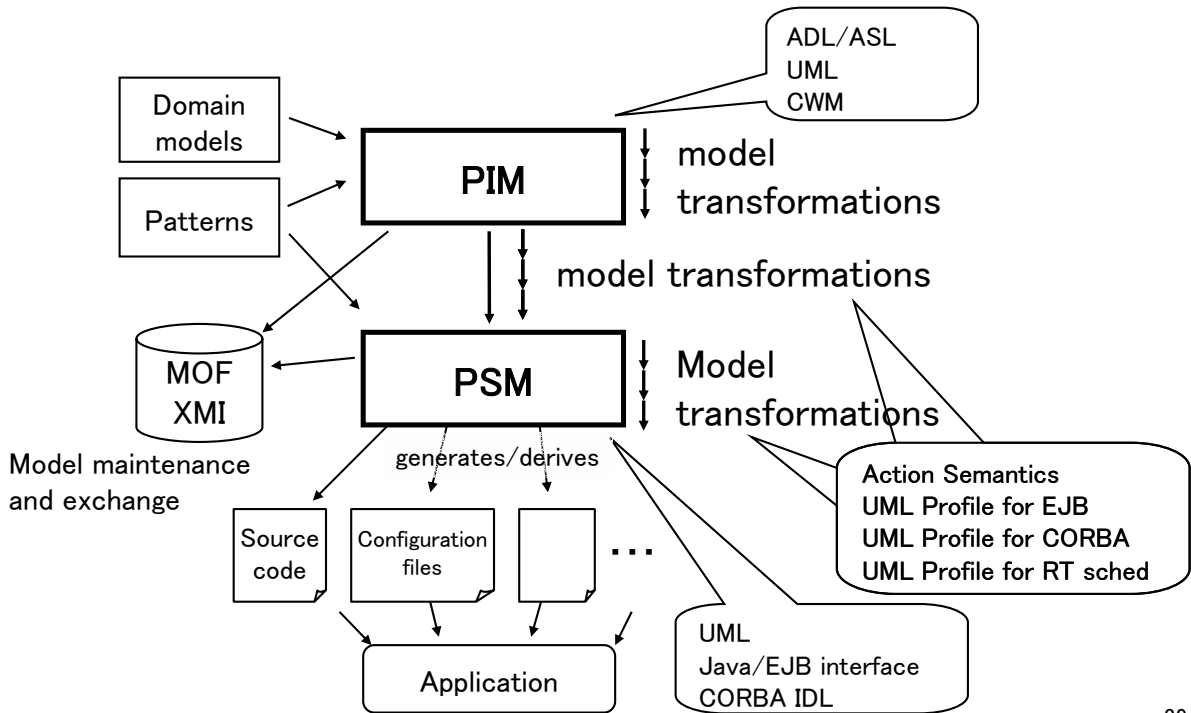- Model integration
  - Model weaving

Platform expertise

Application developers

Technology (logic impl) expertise

Platform experts

MDA tools → Applications

Domain experts

Domain expertise

Platform expertise

Application developers

Technology (logic impl) expertise

Platform experts

MDA tools → Applications

Domain experts

Domain expertise

Domain models

Patterns

PIM → model transformations

model transformations

MOF XMI

PSM → Model transformations

Model maintenance and exchange

generates/derives

Source code | Configuration files | | ...

Application

Model Transformation

Domain models

Patterns

PIM

ADL/ASL
UML
CWM

model transformations

model transformations

MOF
XMI

PSM

Model transformations

Model maintenance and exchange

generates/derives

Source code

Configuration files

...

Application

Action Semantics
UML Profile for EJB
UML Profile for CORBA
UML Profile for RT sched

UML
Java/EJB interface
CORBA IDL
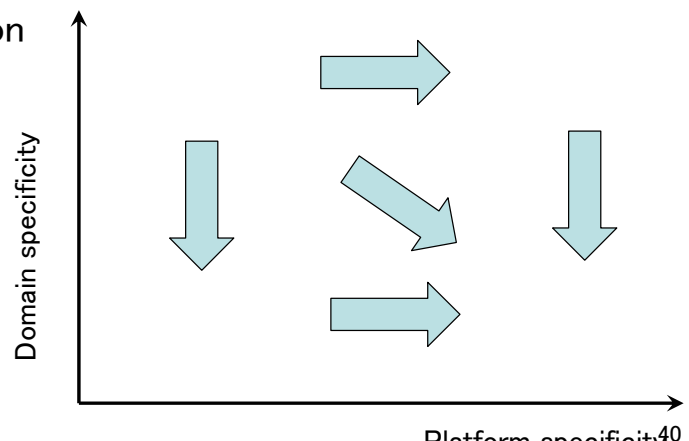

Model Transformation

- 2 dimensions of model transformation
  - Domain specialization
  - Platform specialization
- Several forms of model transformation
  - Manual transformation
  - Automatic transformation

Domain specificity

Platform specificity

Scope of UML Profile for EJB