



# Building Compilable Class Models

David S. Frankel  
Chief Consulting Architect  
IONA Technologies  
david.frankel@iona.com

# The Challenge



- **Requirement:** Preserve investment
  - As new platforms emerge
  - As platforms themselves change
    - EJB™ 1.1 → EJB 2.0
    - XML → XML Schema
    - MTS → COM+
    - CORBA™ 2.X → CORBA 3.0
- **Solution:** Isolate information and processing logic from technology specifics
  - Build platform-independent models
  - Map these models to specific platforms

# Unified Modeling Language™



- UML™ is independent of
  - CORBA
  - COM
  - EJB
  - XML
  - Etc.



# MOF™ Background



- Standard Passed by OMG, 1997
- Standard Constructs for Describing metamodels
- Premise: There will be more than one metamodel
- Supplemented by XML Metadata Interchange (XMI™) Specification, 1998
- Sun JSR-40 defining Java Metadata Interface (JMI)

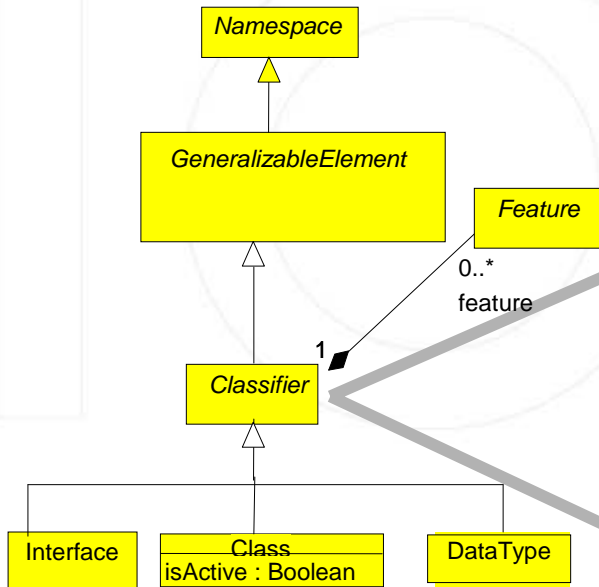
# Platform-Independent Metamodel



UML  
Metamodel

XMI's UML-XML  
Mapping Rules  
Produce

XML DTD

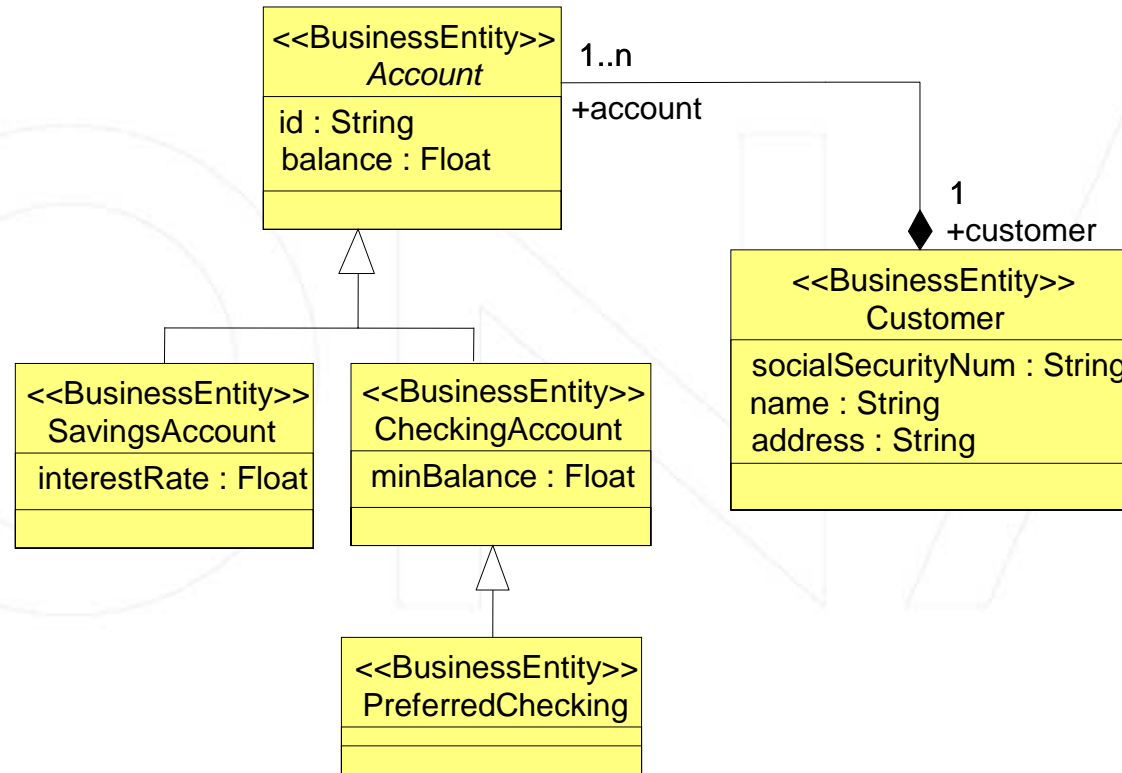


...

```
<!ELEMENT Foundation.Core.Classifier
    (Foundation.Core.ModelElement.name?,
    Foundation.Core.ModelElement.visibility?,
    Foundation.Core.ModelElement.isSpecification?,
    Foundation.Core.GeneralizableElement.isRoot?,
    Foundation.Core.GeneralizableElement.isLeaf?,
    Foundation.Core.GeneralizableElement.isAbstract?,
    ...
    Foundation.Core.Classifier.feature*)?
>
```

...

# Platform Independent Business Information Model



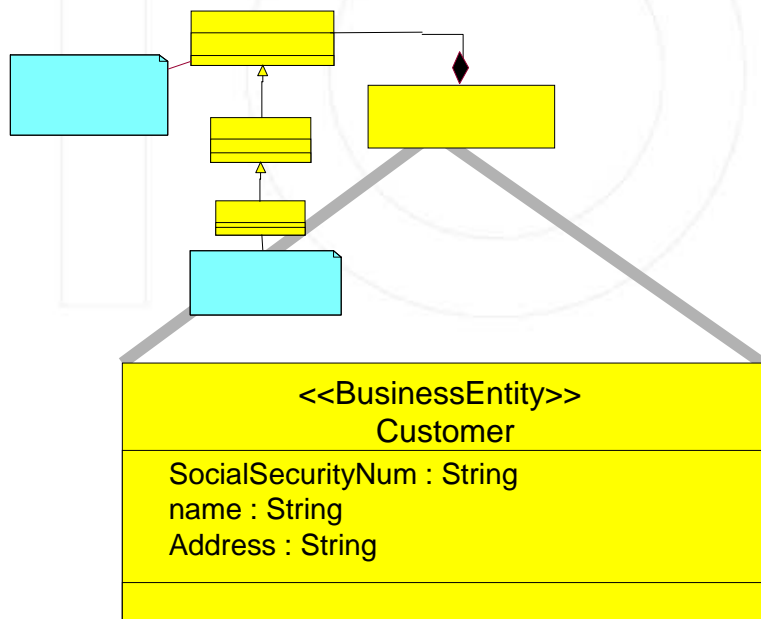
# Mapping the Business Information Model to XML



Platform-Independent Model

XMI's UML-XML Mapping Rules  
Produce

XML DTD (or Schema)



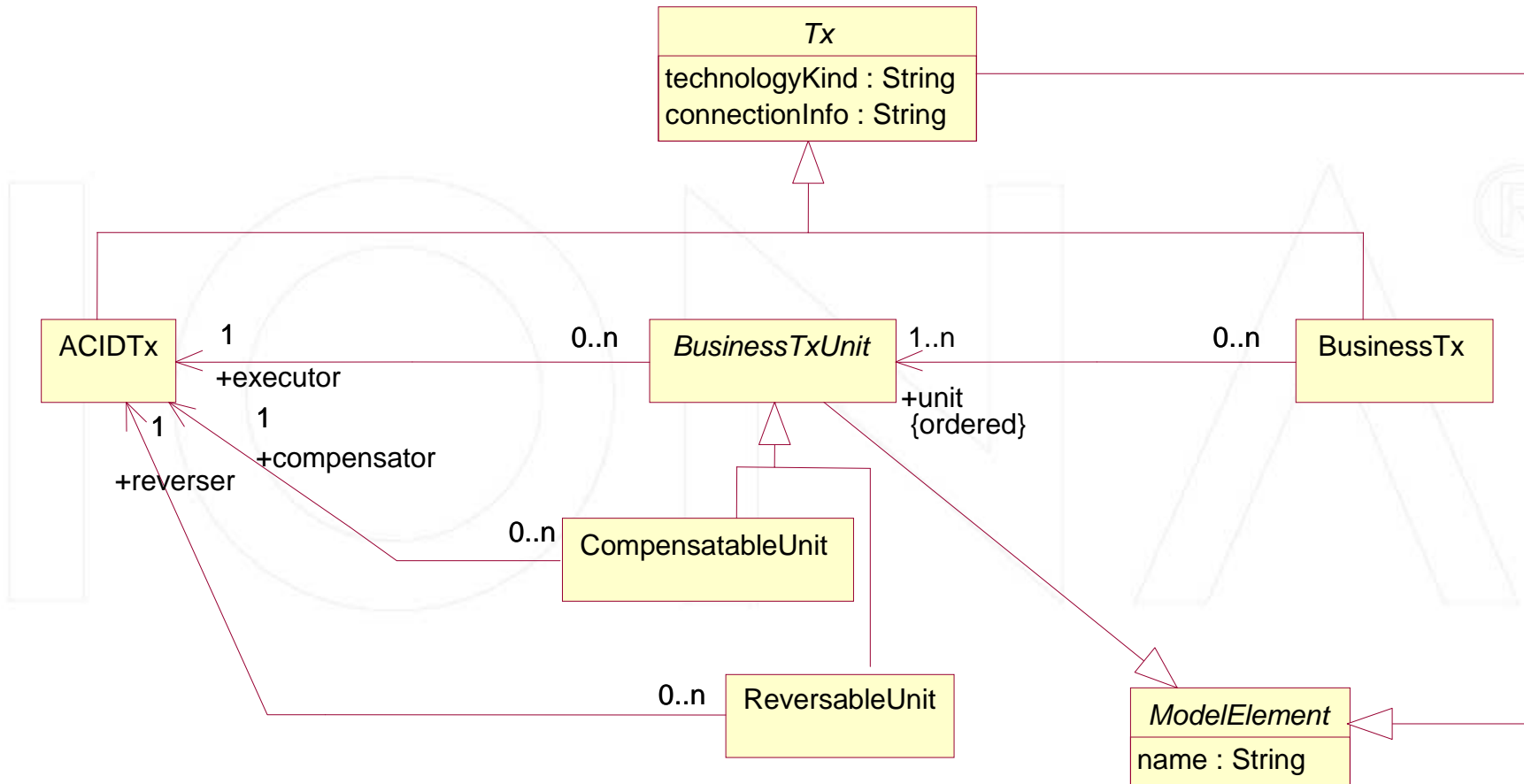
```
...  
<!ELEMENT Bank.Customer.SocialSecurityNum (#PCDATA | XMI.reference)*>  
<!ELEMENT Bank.Customer.name (#PCDATA | XMI.reference)* >  
<!ELEMENT Bank.Customer.Address (#PCDATA | XMI.reference)* >  
...
```

# Don't...



- Specify basic accessor and mutator operations
  - Simply declare attributes and navigable associations
  - Let language mappings produce accessors and mutators

# Example



# Mapping Attributes



```
ModelElement  
name : String
```

General Mapping Rule



Contract of ModelElement supports getting and setting the value of the name, usually by generating accessor and mutator operations.

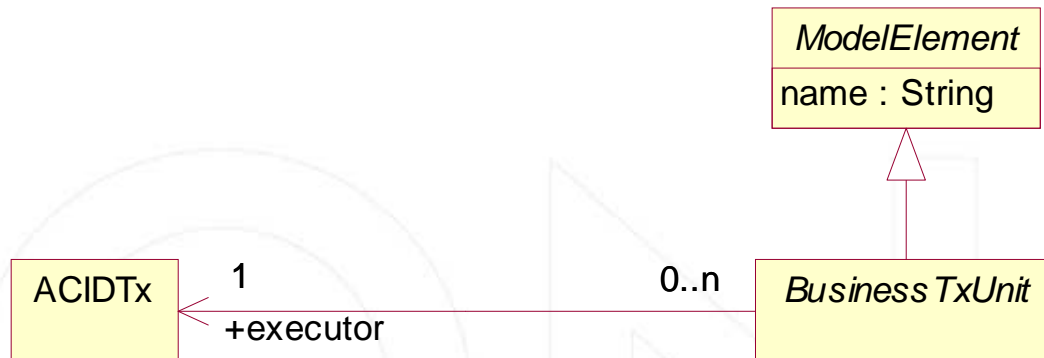
JMI Mapping Rule



```
public interface ModelElement  
{  
    public java.lang.String getName();  
    public setName (java.lang.String newValue);  
}
```

# Mapping

## Navigable Association Ends



General Mapping Rule



Contract of BusinessTxUnit supports getting and setting the value of the reference to ACIDTx.

JMI Mapping Rule

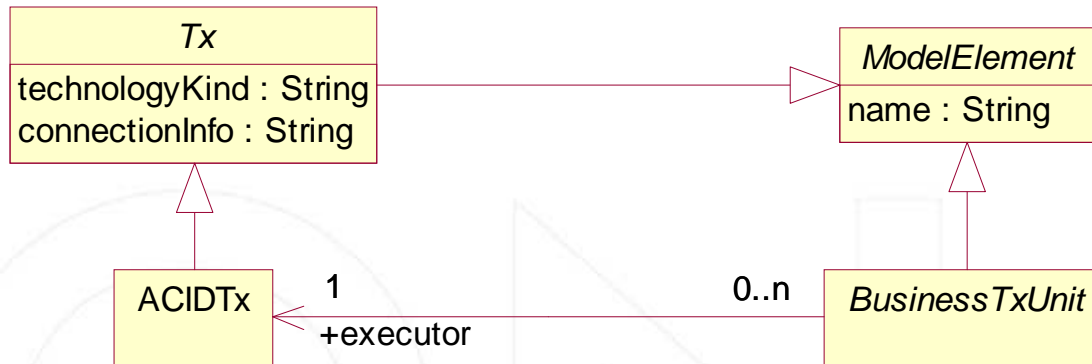


```
public interface BusinessTxUnit extends ModelElement
{
    public ACIDTx getExecutor();
    public setExecutor (ACIDTx newValue);
}
```

# Mapping



## Non-Navigable Association Ends



General Mapping Rule



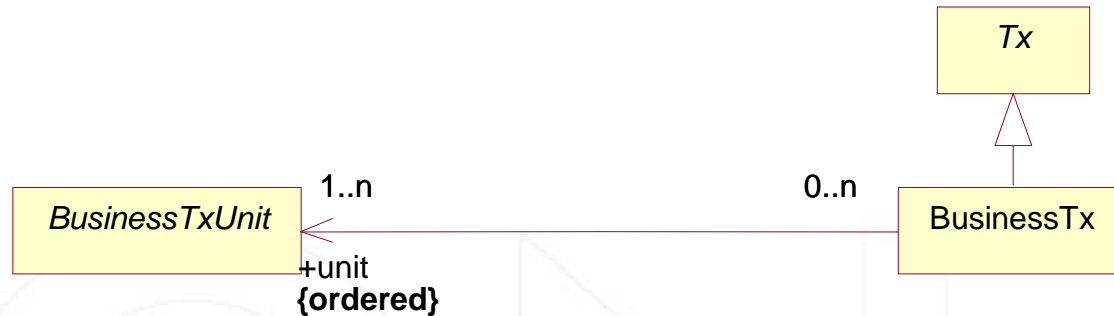
Contract of ACIDTx does *not* support getting and setting the value of the BusinessTxUnit

JMI Mapping Rule



```
public interface ACIDTx extends Tx
{
}
```

# Mapping Navigable, Multi-Valued Association Ends



General Mapping Rule



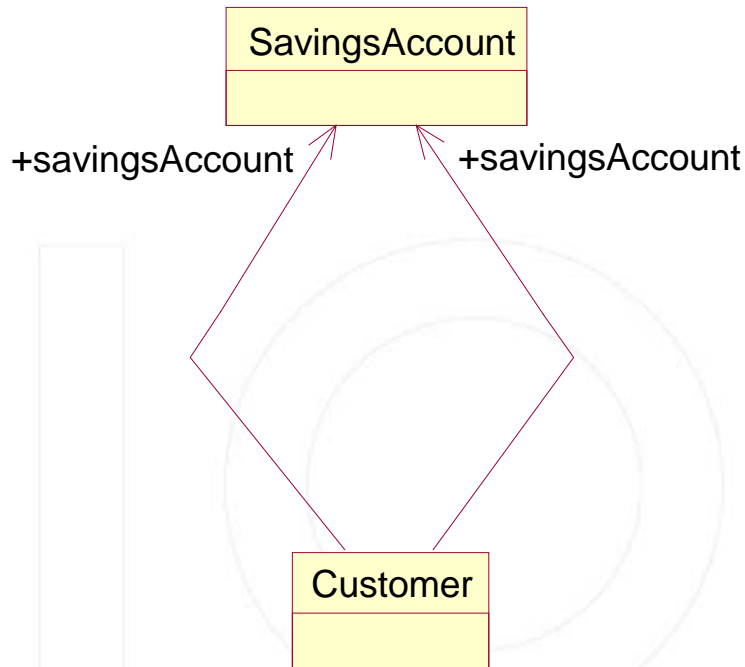
Contract of BusinessTx supports getting and setting the value of the reference to BusinessTxUnit. Mapping differs if not ordered.

JMI Mapping Rule

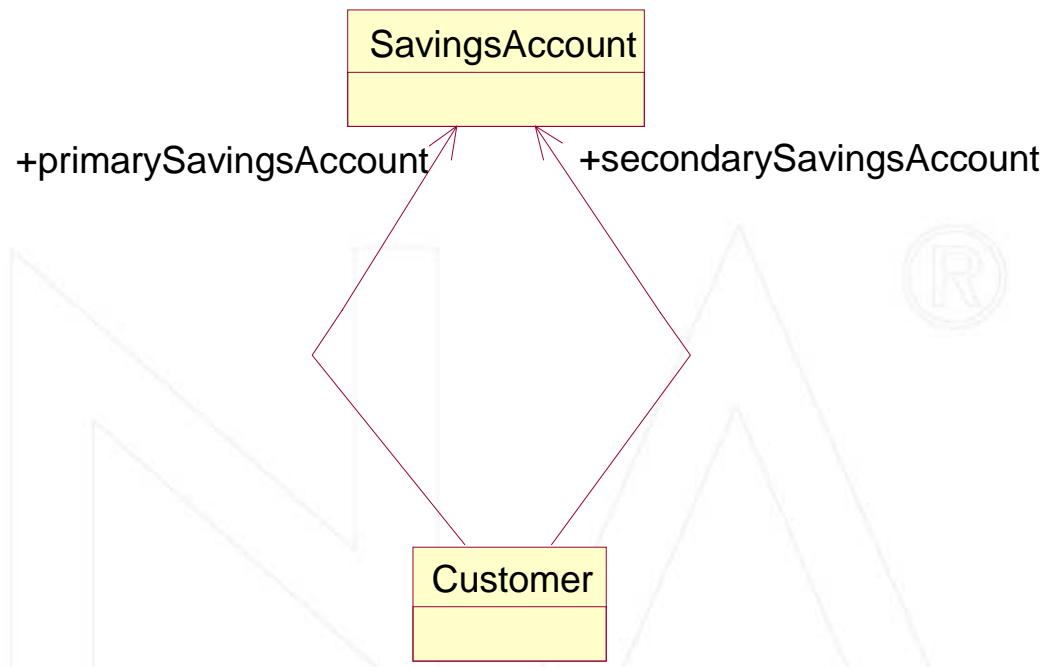


```
public interface BusinessTx extends Tx {
    public java.util.List getUnit();
    public void setUnit(java.util.List newValue);
    public void addUnit(BusinessTxUnit newElement);
    public void modifyUnit(BusinessTxUnit oldElement,
        BusinessTxUnit newElement);
    public void removeUnit(BusinessTxUnit oldElement);
    public void addUnitBefore(BusinessTxUnit newElement,
        BusinessTxUnit beforeElement);}
```

# Avoid Name Clashes



**Not Ok.** Using same association end name means two properties of Customer have the same name.



**Ok.** Different association end names distinguish the two properties of Customer.

# Read Only



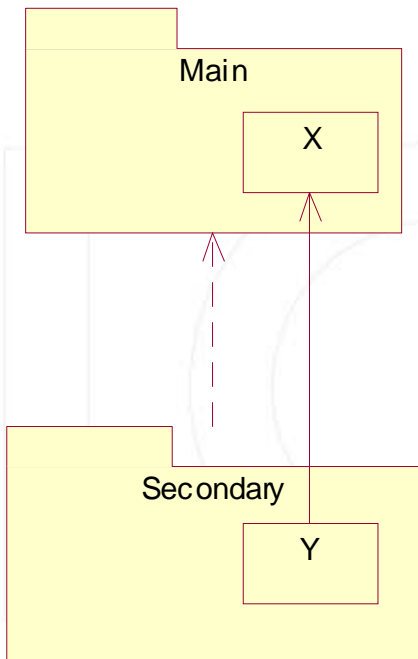
- MOF (not UML 1.x) allows the modeler to specify whether a client can change the value of an attribute or association end
  - isChangeable tagged value
  - If true, no mutator is generated

# Do...

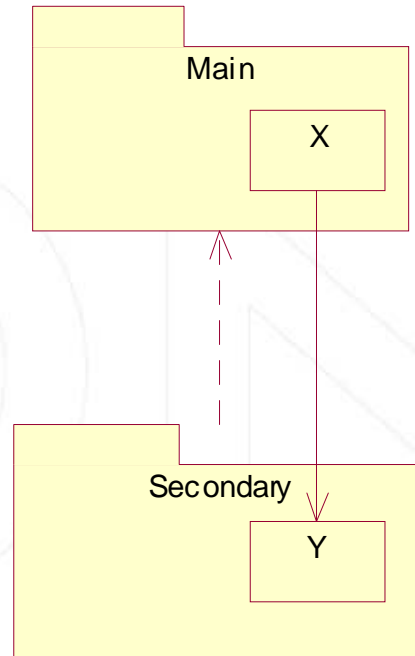


- Consider navigability of association ends carefully
- Use aggregations...properly
- Specify multiplicities on both sides of all associations
- Name all associations and association ends
  - Not all names need to show on diagrams
    - Convention: Suppress display of association names
    - Convention: Suppress display of non-navigable end names
- Use abstract classes where appropriate

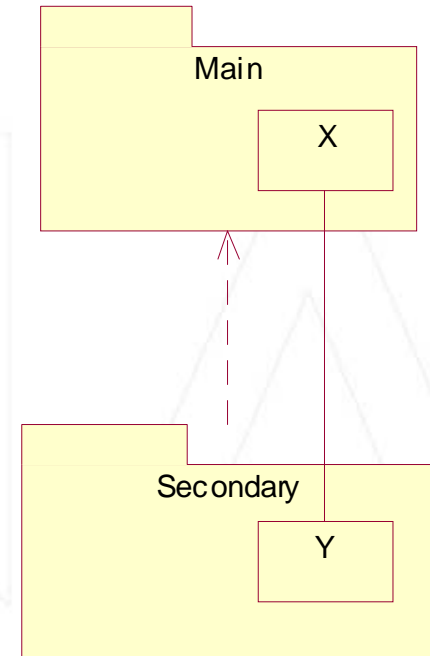
# Navigable Association Ends Imply Dependencies!



**Ok.** *Main* remains independent of *Secondary*.  
Contract of Y has getX and setX.

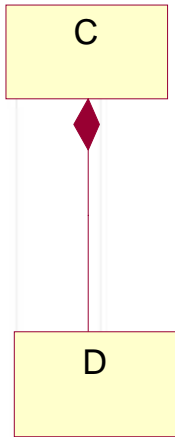


**Not Ok.** *Main* is actually dependent on *Secondary*.  
Contract of X has getY and setY.

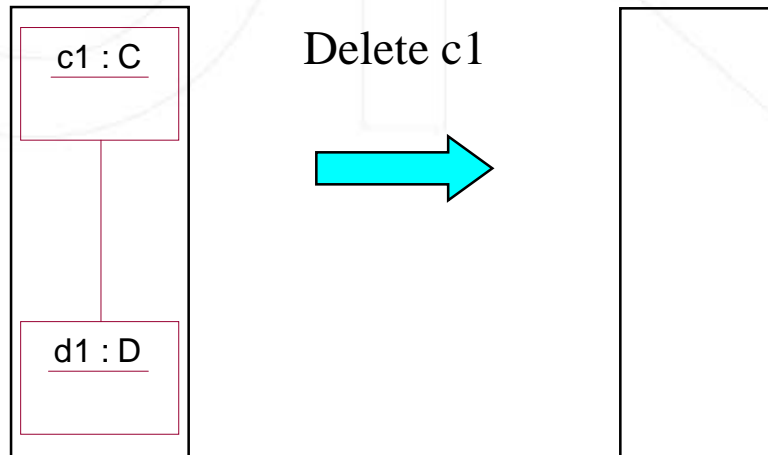


Neither end is navigable: **Ok**  
Both ends navigable: **Not Ok**

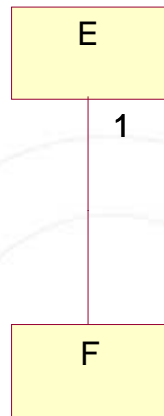
# Composite (Strong) Aggregation Semantics



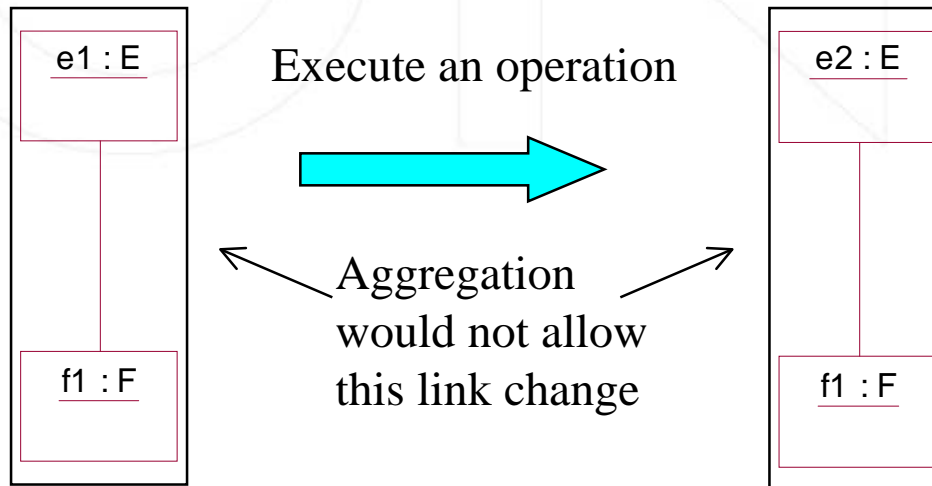
- An instance of D can be owned by only 1 instance of C
- Multiplicity of C can be 0..1 or 1..1, so it must be specified !!
  - 0..1 means that a D can exist without being linked to a C
    - But if a D *is* linked to a C, then it is owned by that C, i.e. its lifetime cannot extend beyond the lifetime of that C
- Multiplicity of D unconstrained
- A.k.a “black diamond”



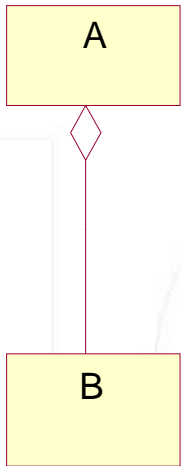
# No Aggregation Semantics



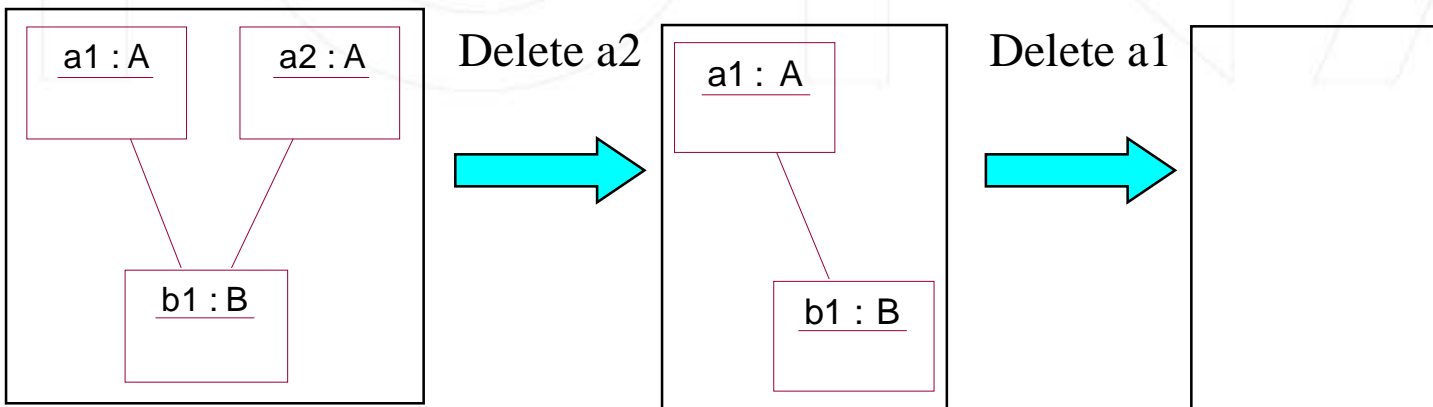
How does this differ from composite (strong) aggregation?



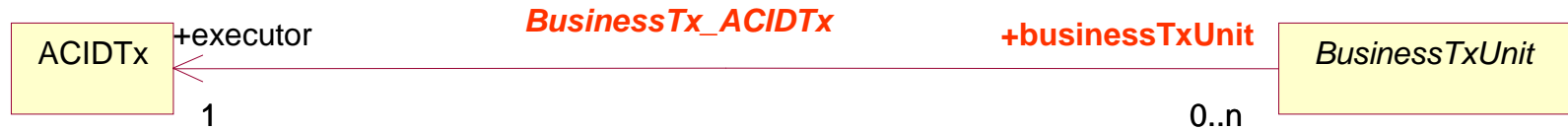
# Beware of Shared (Weak) Aggregation



- One possible interpretation
  - An instance of B can be owned by 1 or more instances of A
- Multiplicity of A can be 0..\* or 1..\*
- Multiplicity of B unconstrained
- Less common than composite (strong) aggregation
- A.k.a. “white diamond”



# Name all associations and ends



```
public interface BusinessTx_ACIDTx extends javax.jmi.reflect.RefAssociation {
    public boolean exists(ACIDTx executor, BusinessTxUnit businessTxUnit);
    public ACIDTx executor(BusinessTxUnit businessTxUnit);
    public void add(ACIDTx executor, BusinessTxUnit businessTxUnit);
    public void addBeforeExecutor(ACIDTx executor,
        BusinessTxUnit businessTxUnit,
        ACIDTx before);
    public void addBeforeBusinessTxUnit(ACIDTx executor,
        BusinessTxUnit businessTxUnit,
        BusinessTxUnit before);
    public void modifyExecutor(ACIDTx executor,
        BusinessTxUnit businessTxUnit,
        ACIDTx newExecutor);
    public void modifyBusinessTxUnit(ACIDTx executor,
        BusinessTxUnit businessTxUnit,
        BusinessTxUnit newBusinessTxUnit);
    public void remove(ACIDTx executor, BusinessTxUnit businessTxUnit);
}
```

# Mapping Abstract Classes

<i>Tx</i>
technologyKind : String connectionInfo : String

General Mapping Rule



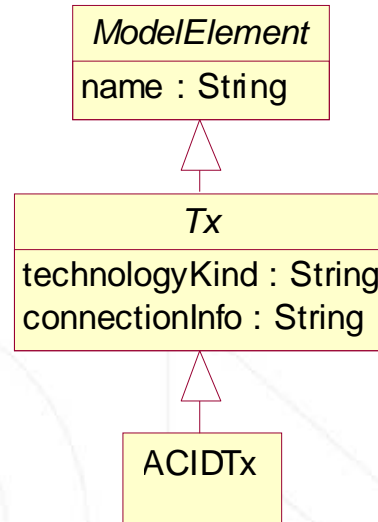
Separate instance management interface is always generated. But if the class is abstract (denoted by italicized name), it has no create operations

JMI Mapping Rule



```
public interface TxClass extends javax.jmi.reflect.RefClass
{
}
```

# Mapping Non-Abstract Classes



General Mapping Rule



Separate instance management interface is always generated. But if the class is abstract (denoted by italicized name), it has no create operations

JMI Mapping Rule



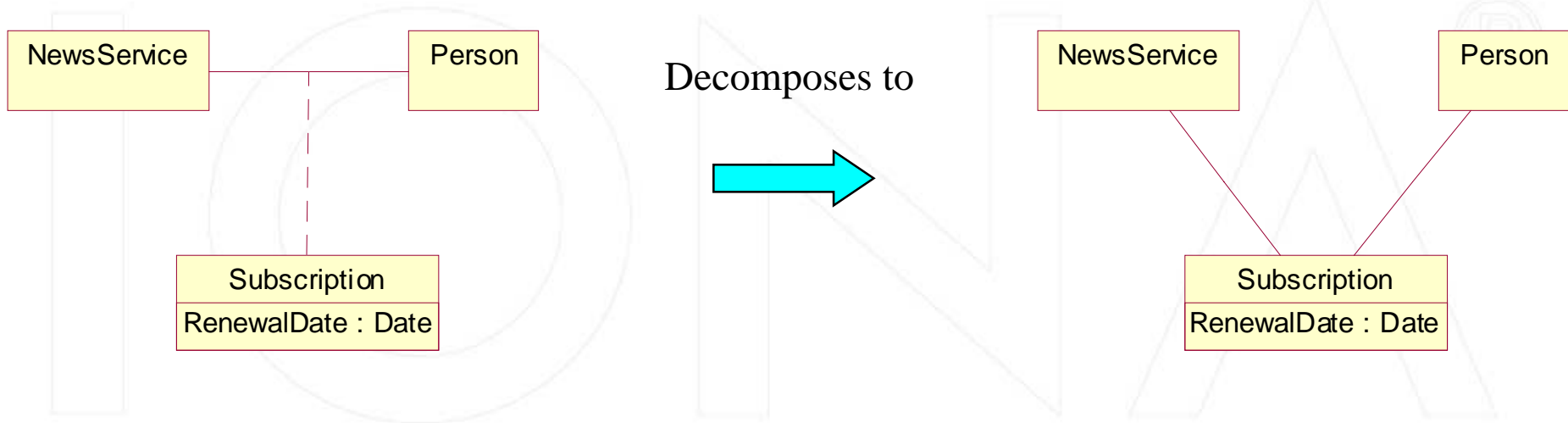
```
public interface ACIDTxClass extends javax.jmi.reflect.RefClass
{
    public ACIDTx createACIDTx();
    public ACIDTx createACIDTx(
        java.lang.String name,
        java.lang.String technologyKind,
        java.lang.String connectionInfo);
}
```

# Using UML to Create MOF Metamodels: Don't Use...

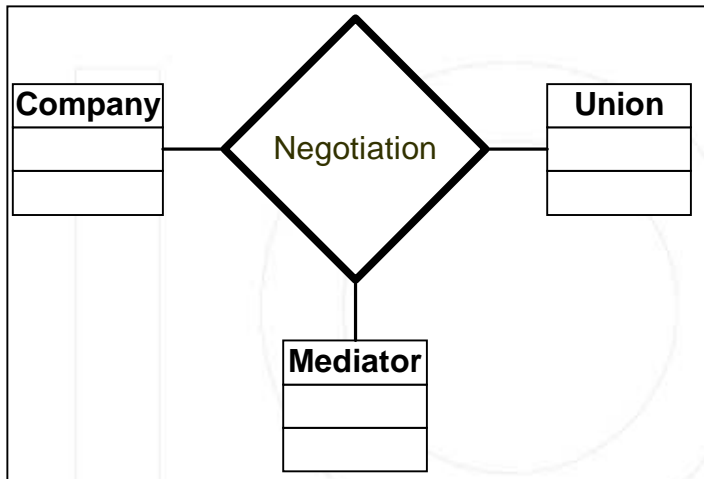


- Association classes
- N-ary associations (associations among more than two classes)
- Qualifiers

# Decomposing Association Classes

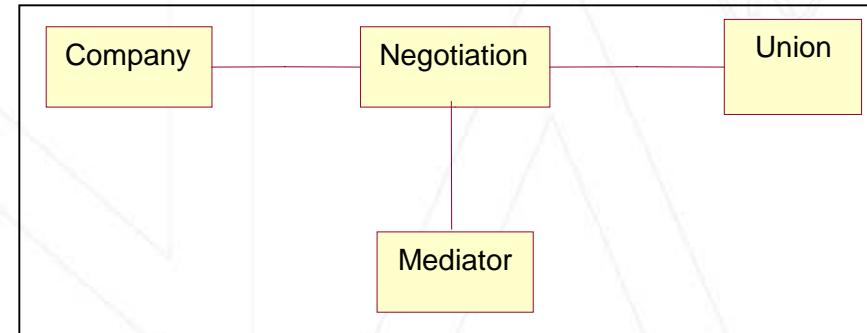
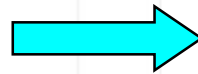


# Decomposing an N-Ary Association



N-Ary Association *Negotiation*,  
Associating Three Classes

Decomposes to



Binary Associations Only

# Doing Without Qualifiers



- No set formula
- Work around on a case-by-case basis



# UML Profile for MOF



- Rules for using UML to define MOF models
- Submitted to OMG as Part of UML Profile for EDOC
- OMG document ad/2001-08-19
  - Part I, Chapter 6.