CSSE 490 Model-Based Software Engineering: Introduction to MetaModels

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

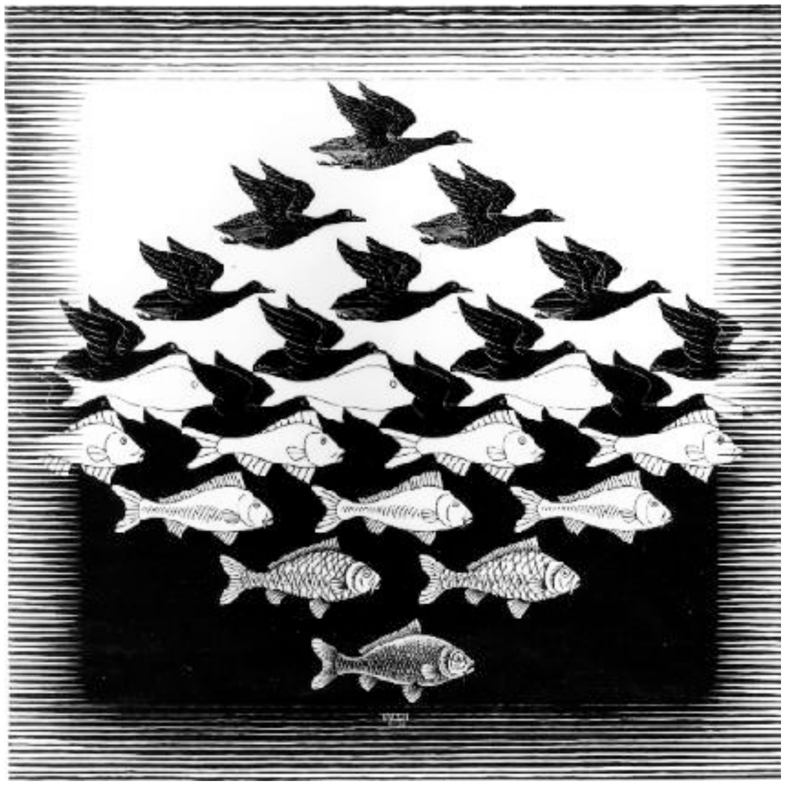
Email: bohner@rose-hulman.edu

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

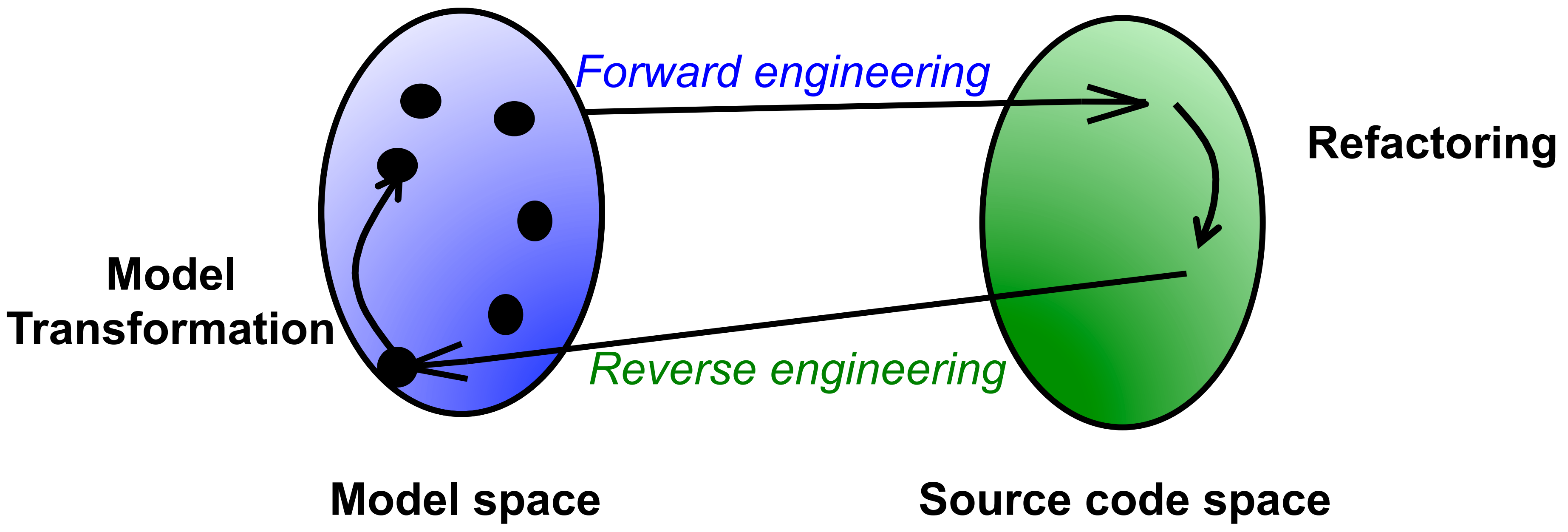
Learning Outcomes: Transformations

Define transformation rules for abstraction and refinement.

- Examine model transformations
- Explore Mappings with MDA Example
- Discuss paper (if time)

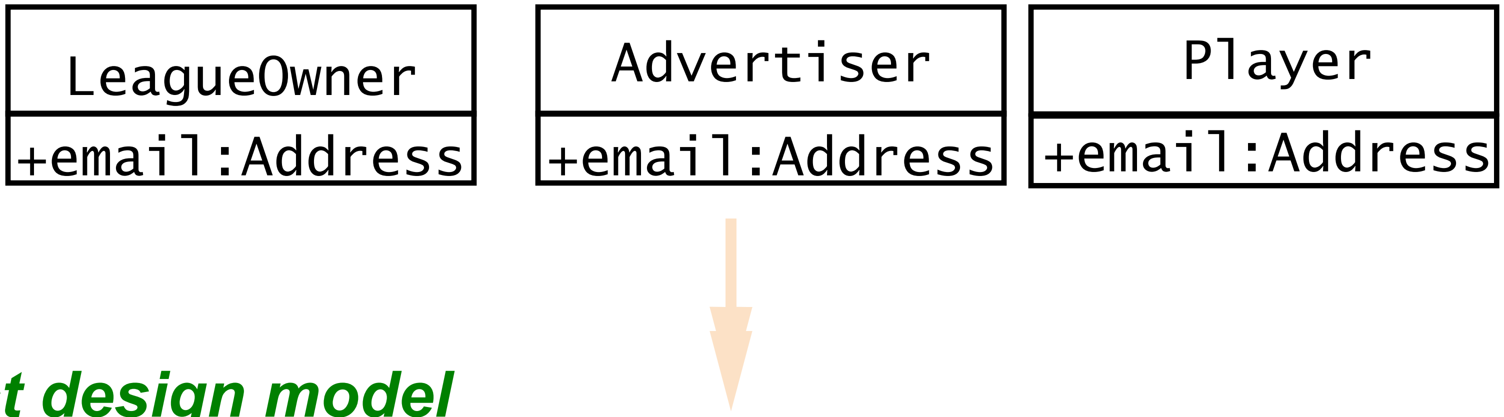


Model Transformations

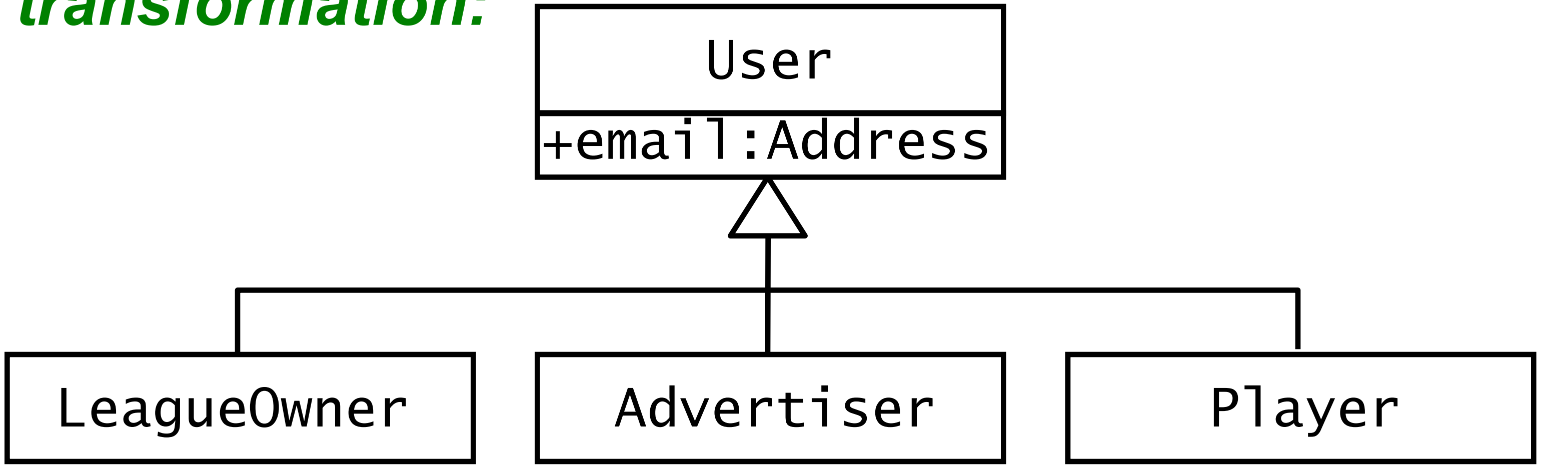


Model Transformation Example

Object design model before transformation



Object design model after transformation:



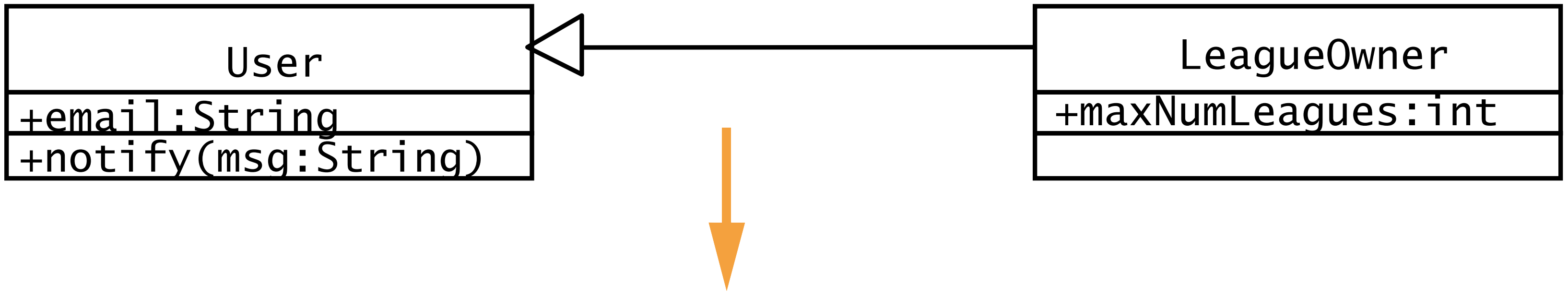
Refactoring Example: Pull Up Field

```
public class Player {
   private String email;
   //...
}
public class LeagueOwner {
   private String eMail;
   //...
}
public class Advertiser {
   private String email_address;
   //...
}
```

```
public class User {
   private String email;
}
public class Player extends
   User {
   //...
}
public class LeagueOwner
   extends User {
   //...
}
public class Advertiser extends
   User {
   //...
}
```

PIM to PSM Example

Object design model before transformation



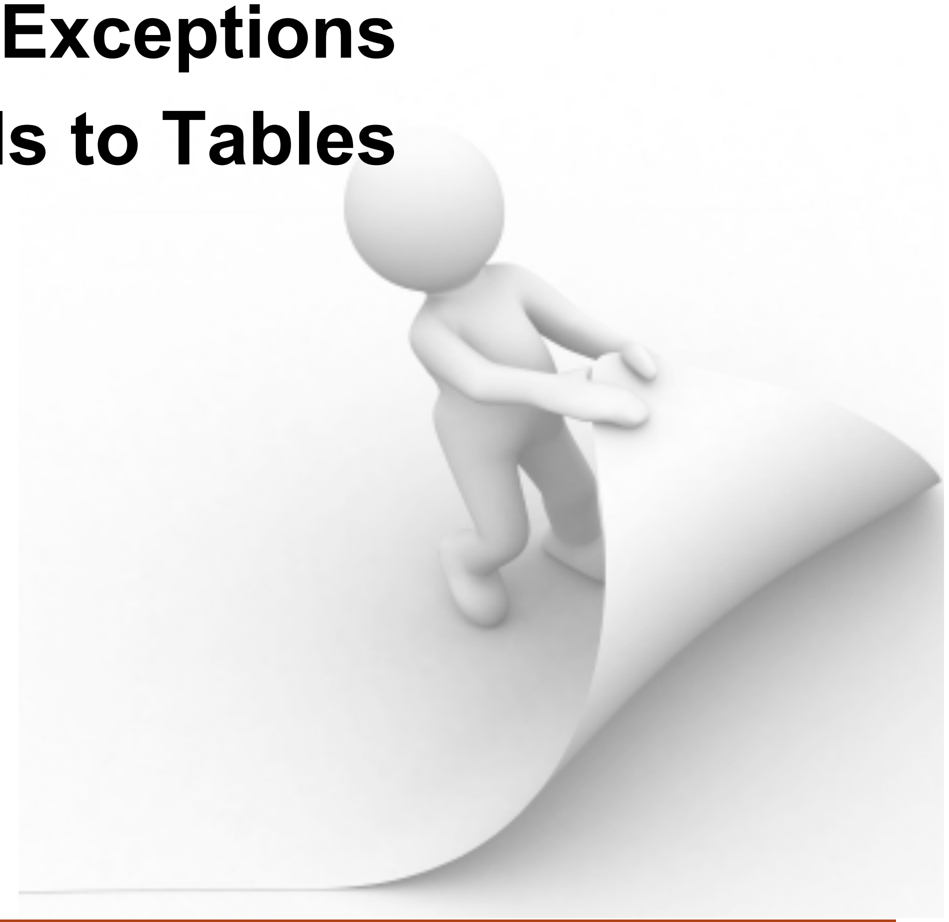
Source code after transformation

```
public class User {
    private String email;
    public String getEmail() {
          return email;
    }
    public void setEmail(String value){
          email = value;
    }
    public void notify(String msg) {
          // ....
    }
    /* Other methods omitted */
}
```

```
public class LeagueOwner extends User {
    private int maxNumLeagues;
    public int getMaxNumLeagues() {
          return maxNumLeagues;
    }
    public void setMaxNumLeagues
                    (int value) {
          maxNumLeagues = value;
    }
    /* Other methods omitted */
}
```

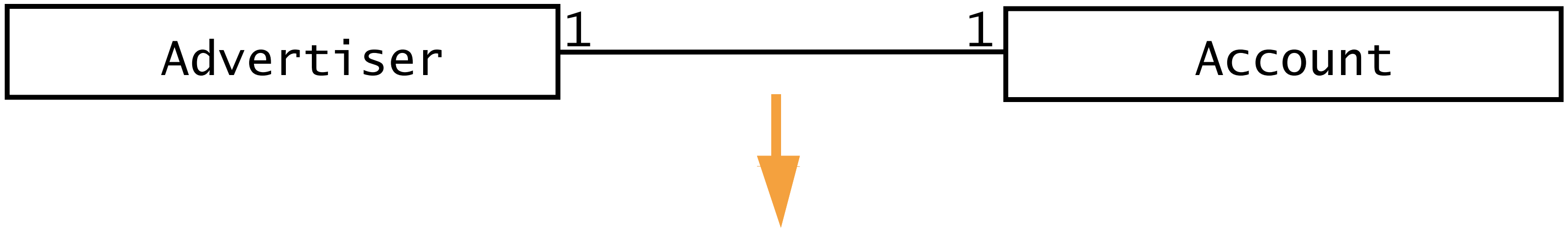
Some Other Mapping Activities

- Mapping Associations
- Mapping Contracts to Exceptions
- Mapping Object Models to Tables



Mapping: Unidirectional, 1-to-1 Association

Object design model before mapping

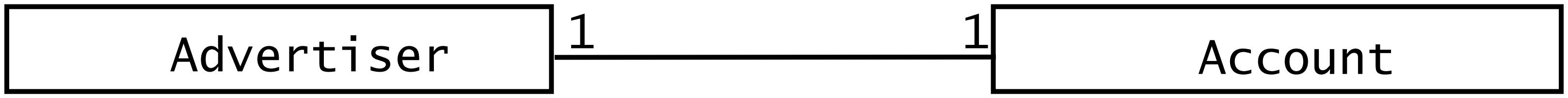


Source code after rendering

```
public class Advertiser {
   private Account account;
   public Advertiser() {
       account = new Account();
   }
   public Account getAccount() {
       return account;
   }
}
```

Example: Bidirectional 1-to-1 Association

Object design model before mapping



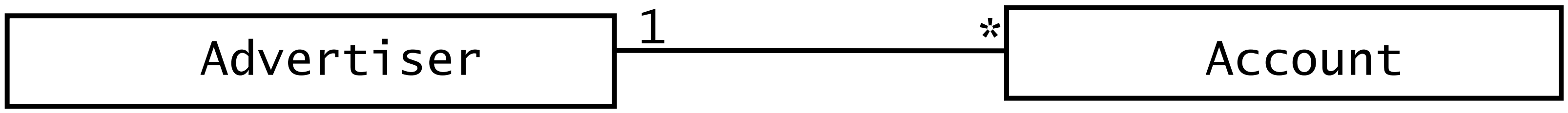
Source code after rendering

```
public class Advertiser {
   /* The account field initialized
    * in the constructor and never
    * modified. */
   private Account account;

   public Advertiser() {
         account = new Account(this);
   }
   public Account getAccount() {
         return account;
   }
}
```

```
public class Account {
   /* The owner field initialized
    * during the constructor and
    * never modified. */
   private Advertiser owner;

   public Account(owner:Advertiser)
   {
         this.owner = owner;
   }
   public Advertiser getOwner() {
         return owner;
   }
}
```

Bidirectional, 1-to-many Association

Object design model before mapping



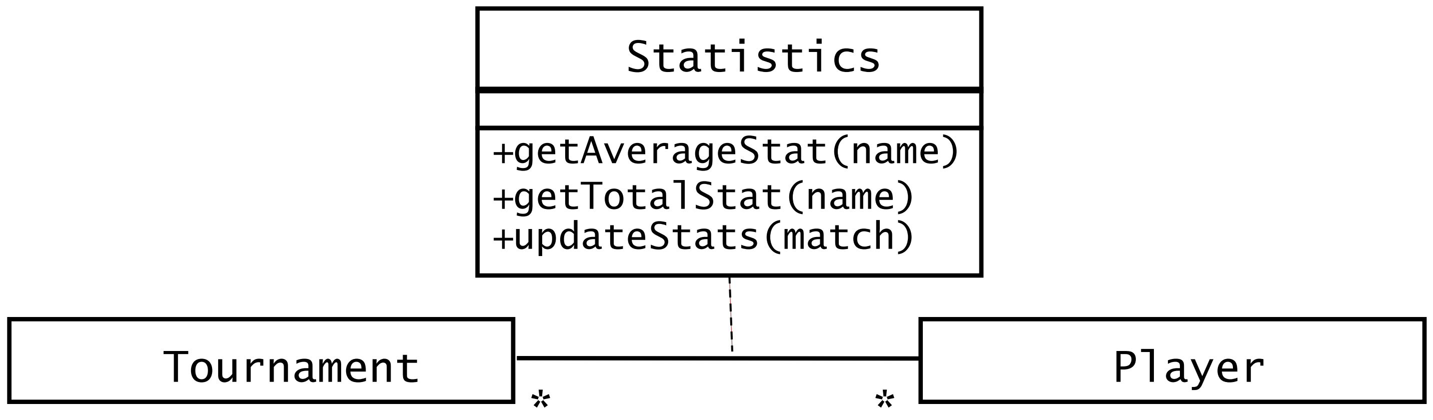
Source code after rendering

```
public class Advertiser {
   private Set accounts;
   public Advertiser() {
         accounts = new HashSet();
   }
   public void addAccount(Account
   a) {
         accounts.add(a);
         a.setOwner(this);
   }
   public void removeAccount
   (Account a) {
         accounts.remove(a);
         a.setOwner(null);
   }
}
```

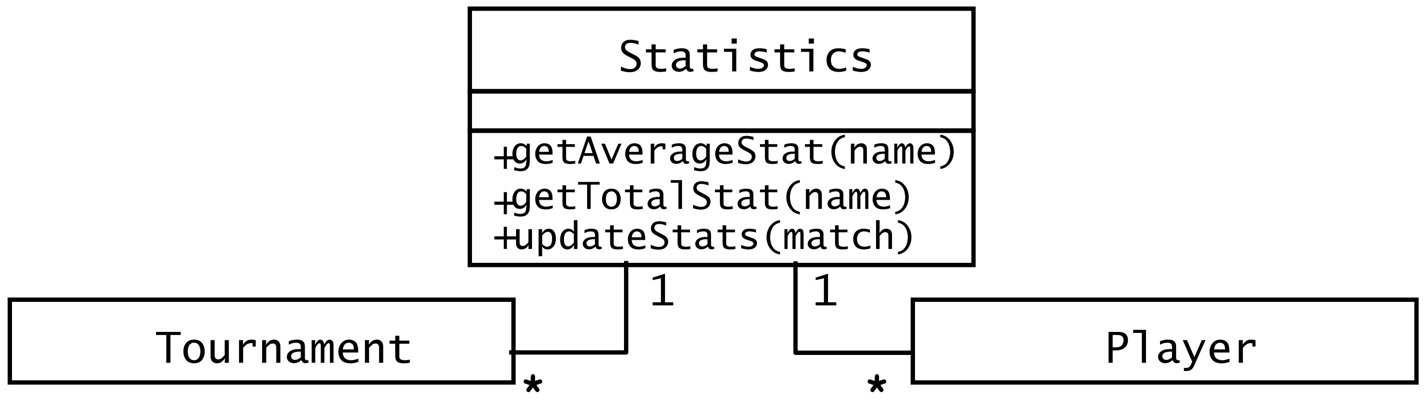
```
public class Account {
   private Advertiser owner;
   public void setOwner(Advertiser
   newOwner) {
      if (owner != newOwner) {
            Advertiser old = owner;
            owner = newOwner;
            if (newOwner != null)
              newOwner.addAccount(this);
            if (oldOwner != null)
              old.removeAccount(this);
      }
   }
}
```

Transforming an Association Class

Object design model before transformation



Object design model after transformation:
1 class and 2 binary associations



What are some ways we can handle constraints in modeling? In coding?

- Think for a minute…
- Turn to a neighbor and discuss it for a minute

Exceptions as Building Blocks for Contracts

- Most object-oriented languages do not support contracts directly
  - But, exception mechanisms can be building blocks for signaling and handling contract violations
  - Try-throw-catch mechanism used in Java
- Example:
  - Let's assume the acceptPlayer() operation of TournamentControl is invoked with a player who is already part of the Tournament
  - acceptPlayer() should throw KnownPlayer exception

Try-throw-catch Mechanism in Java

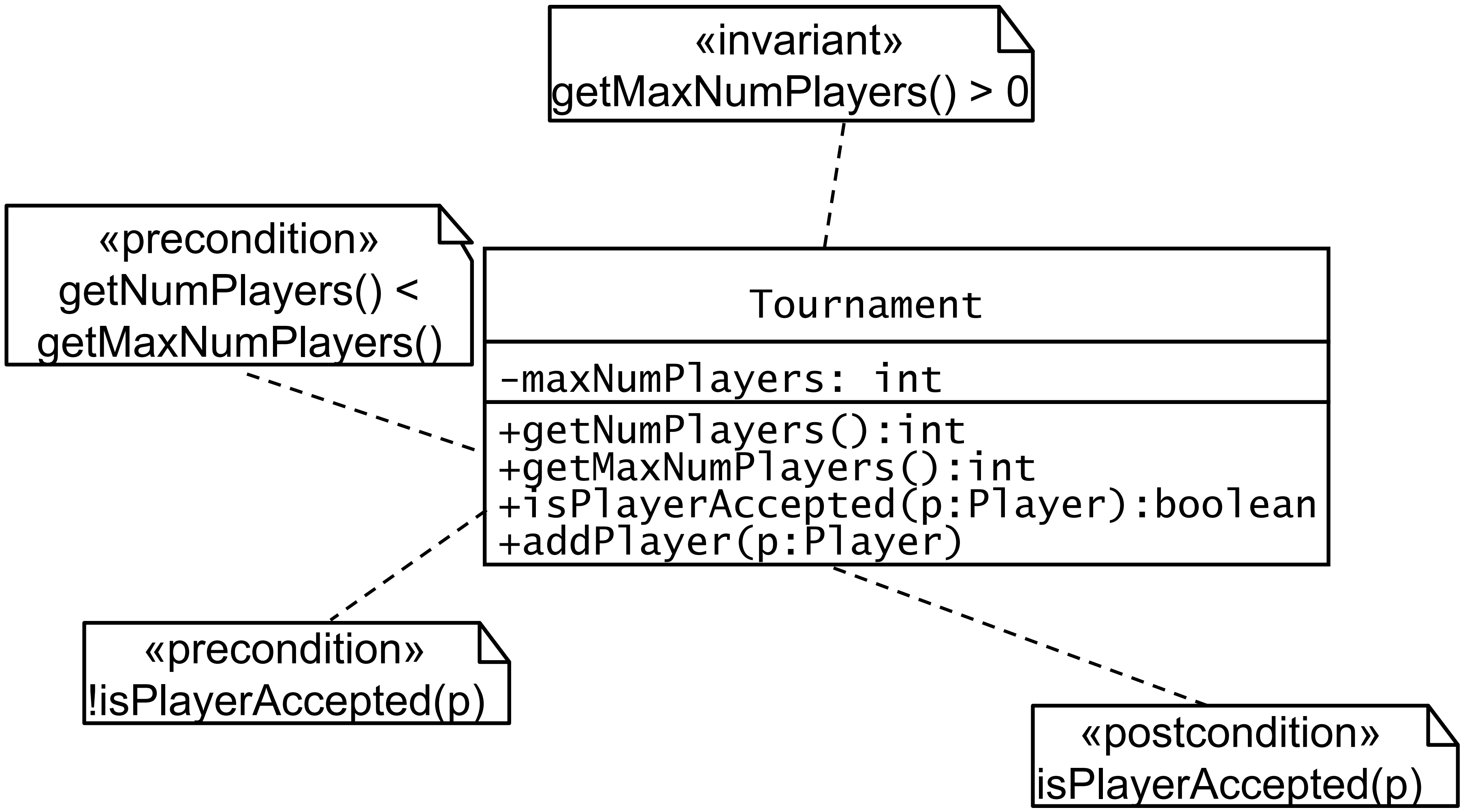
```
public class TournamentControl {
    private Tournament tournament;
    public void addPlayer(Player p) throws KnownPlayerException {
            if (tournament.isPlayerAccepted(p)) {
                    throw new KnownPlayerException(p);
            }
            //... Normal addPlayer behavior
    }
}
public class TournamentForm {
    private TournamentControl control;
    private ArrayList players;
    public void processPlayerApplications() {
            // Go through all the players
            for (Iteration i = players.iterator(); i.hasNext();) {
                    try {       // Delegate to the control object.
                            control.acceptPlayer((Player)i.next());
                    } catch (KnownPlayerException e) {
                            // If an exception was caught, log it to the
    console
                            ErrorConsole.log(e.getMessage());
                    }
            }
    }
}
```

Implementing a Contract

For each operation in the contract:

- **Check precondition**: Check the precondition before the beginning of the method with a test that raises an exception if the precondition is false
- **Check postcondition**: Check the postcondition at the end of the method and raise an exception if the contract is violated. If more than one postcondition is not satisfied, raise an exception only for the first violation
- **Check invariant**: Check invariants at the same time as postconditions
- **Deal with inheritance**: Encapsulate the checking code for preconditions and postconditions into separate methods that can be called from subclasses

Example Implementation of the Tournament.addPlayer() Contract



Heuristics for Mapping Contracts to Exceptions

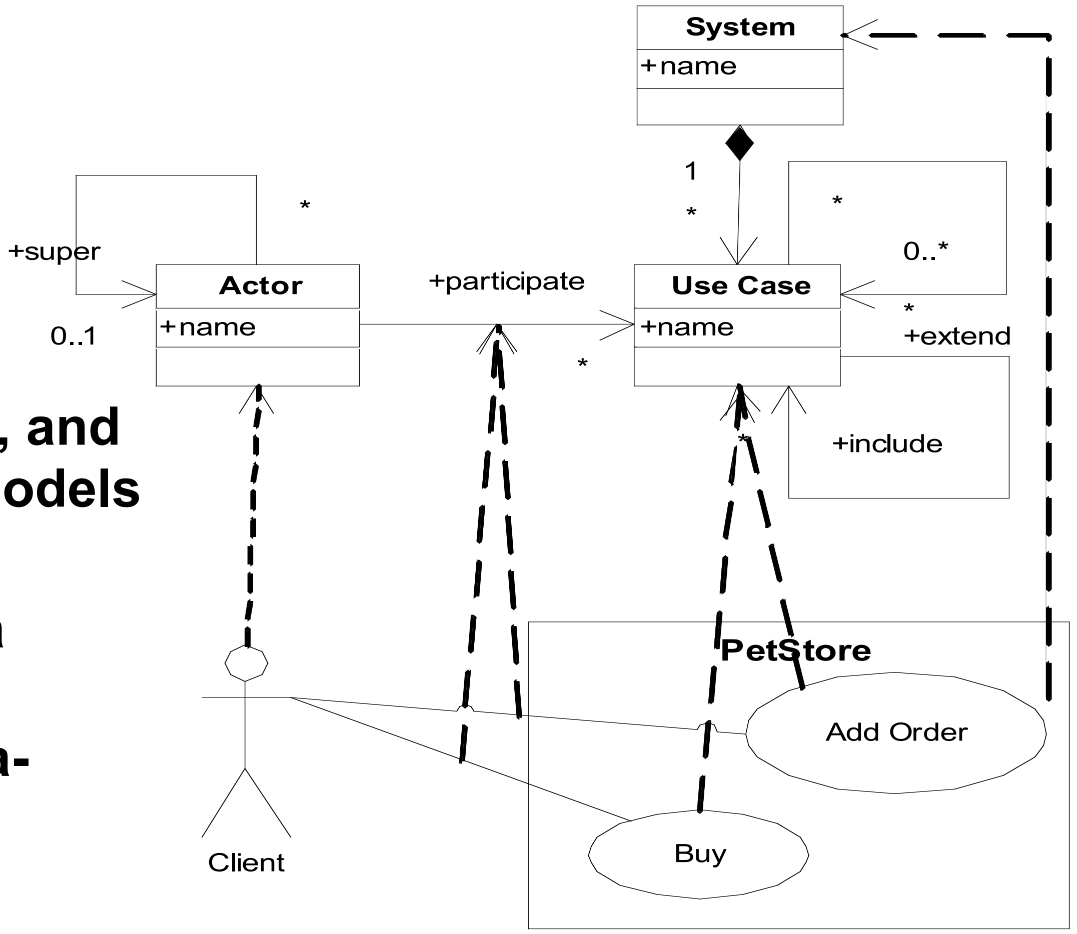
Be pragmatic

- Focus on components with the longest life
  - Focus on Entity objects, not on boundary objects associated with the user interface
- Reuse constraint checking code
  - Many operations have similar preconditions
  - Encapsulate constraint checking code into methods so that they can share the same exception classes

What is a Metamodel?

Pragmatic Definition:
A model that answers questions about (explains) a set of related models

- **Defines concepts, structures, and relationships for a class of models**
- **A "model" is an instance of a metamodel if it respects the structure defined by the meta-model**



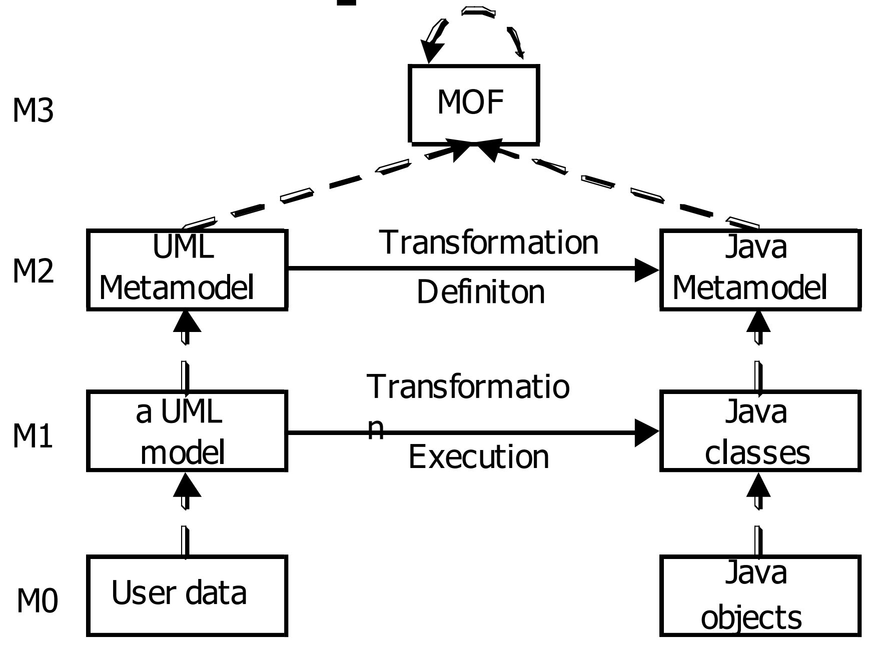
4 Layer Metamodel Architecture

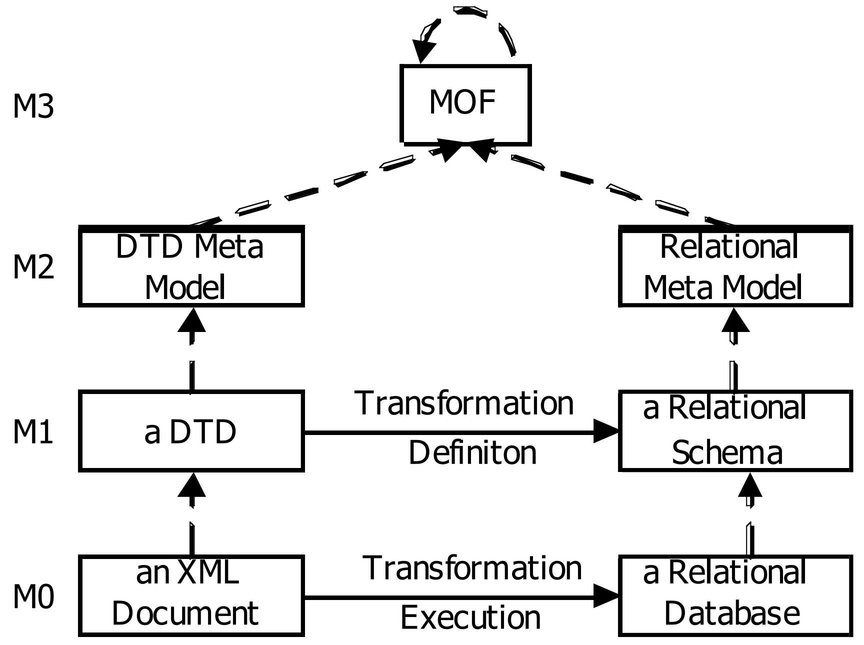
Layer	Description	Examples
M3: Metametamodel	**Foundation for a Metamodeling Architecture.** **Defining the language to describe metamodels.**	**MetaClass, MetaAttribute, MetaOperation**
M2: Metamodel	**An Instance of a metametamodel.** **Defining the language to describe models.**	**Class, Attribute, Operation, Component**
M1: Model	**An Instance of Metamodel. Defining a language to describe the information object domain.**	**Product, Unit Price, Customer, Sale, Detail**
M0: User Objects (User Data)	**An Instance of a Model.** **Defines specific information Domain**	**<Chair>, <Desk>, $100, $200**

Source: Colin Aktinson, Essence of Multilevel Metamodeling (UML2001)

Q6

Example Transformation Scenarios





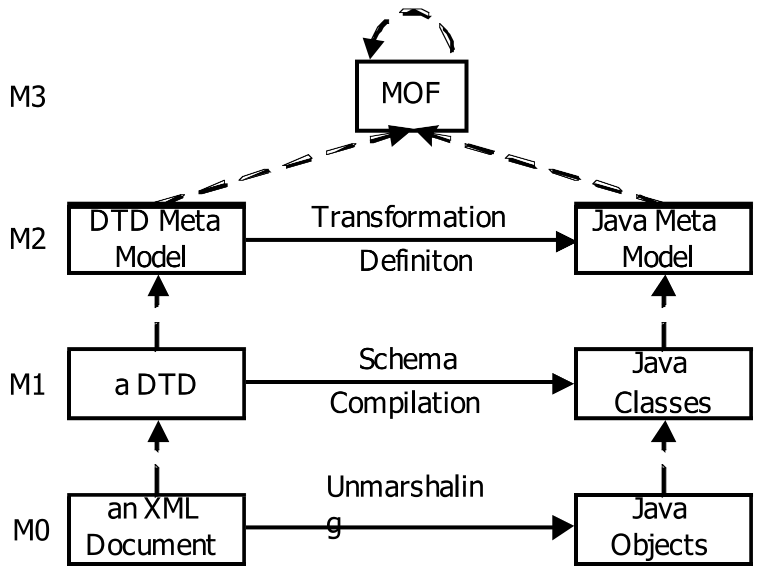
QVT Scenario

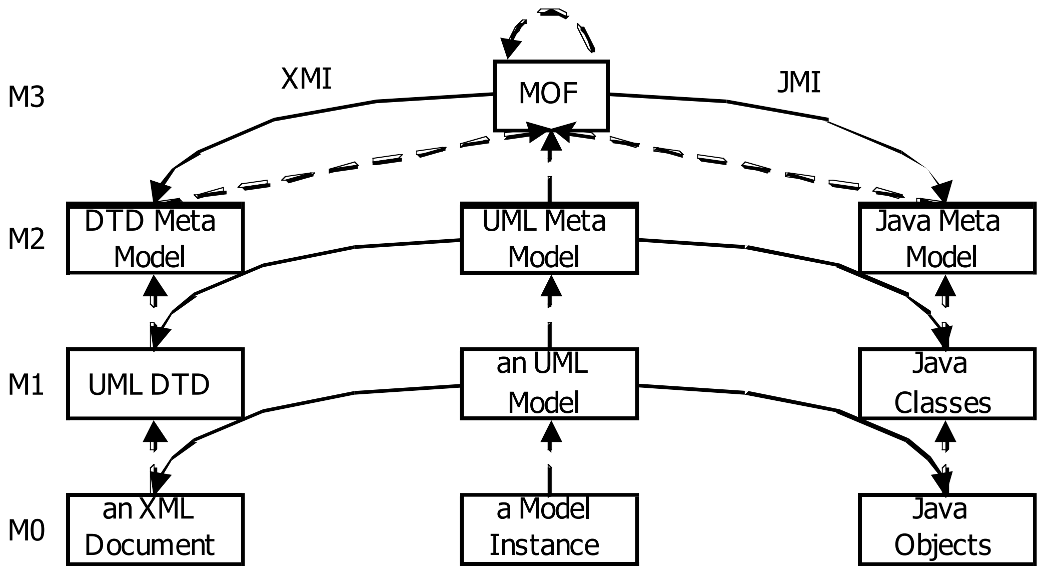
- Context of Query/Views/ Transformation
- Transformation specified between meta models

Data Transformation Scenario

- Transformation executed over concrete data instances at level M0
  - E.g. Common Warehousing Metamodel (CWM)

ExampleTransformation Scenerios (continued)





Data Binding in MOF Context

- Transformation specified at level M2 is executed twice in lower levels M1 & M0

Inter-level Transformations

- XML Metadata Interchange (XMI)
- Java Metadata Interchange (JMI)

Paper Discussion: Metamodel Paper

Model-Driven Development: A Metamodeling Foundation

- What are the main thrusts of the paper?
- What are the controversial points and your positions?
- What did you get out of reading about feature-based transformation approaches?

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Homework and Milestone Reminders

- Milestone 2: Establish a repository and structure for assembling components for your FacePamphlet application
  - Due by 11:55pm Friday, April 1st, 2011 (no foolin'!)