Patterns in Software Engineering

Lecturer: Raman Ramsin

Lecture 19

Analysis Patterns

Part 2

Supporting Patterns

- The supporting patterns describe how to take analysis patterns and apply them:
  - **Layered Architecture for Information Systems:** These patterns consider the architecture for a client/server information system and how such a system can be layered to improve its maintainability.
  - **Patterns for Type Model Design Templates:** These patterns focus on how conceptual models can be implemented, suggesting common patterns to turn analysis patterns into software.
  - **Association Patterns:** These patterns focus on examining modeling techniques themselves and how advanced modeling constructs can be viewed as patterns.

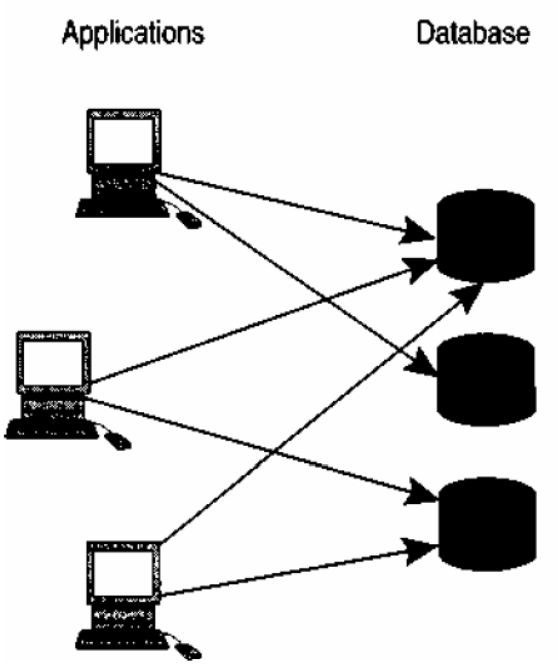
Supporting Patterns: Layered Architectures - *Two-Tier Architecture*

- **Problem:** Partitioning software on a client/server system.
- **Solution:** Put the user interface on the client and the database on the server.
  - The user interface classes access the database directly.

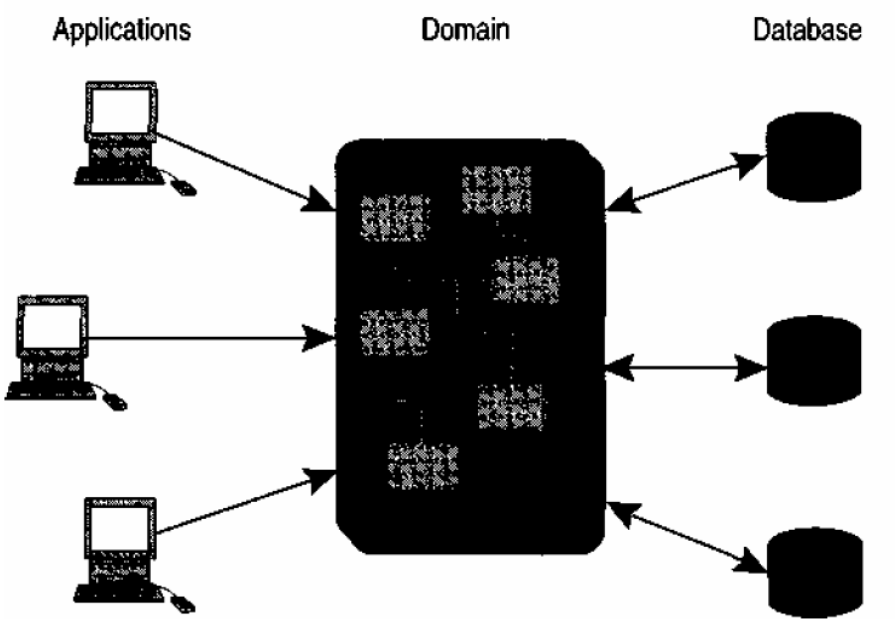
Supporting Patterns: Layered Architectures - *Three-Tier Architecture*

- **Problem:**
  - The two-tier architecture couples the user interface too tightly to the database design.
  - The database interface cannot support a rich model of the domain.
- **Solution:** Have three logical tiers: application, domain, and database.

Supporting Patterns: Layered Architectures - *Tiered Architectures*



Two Tier



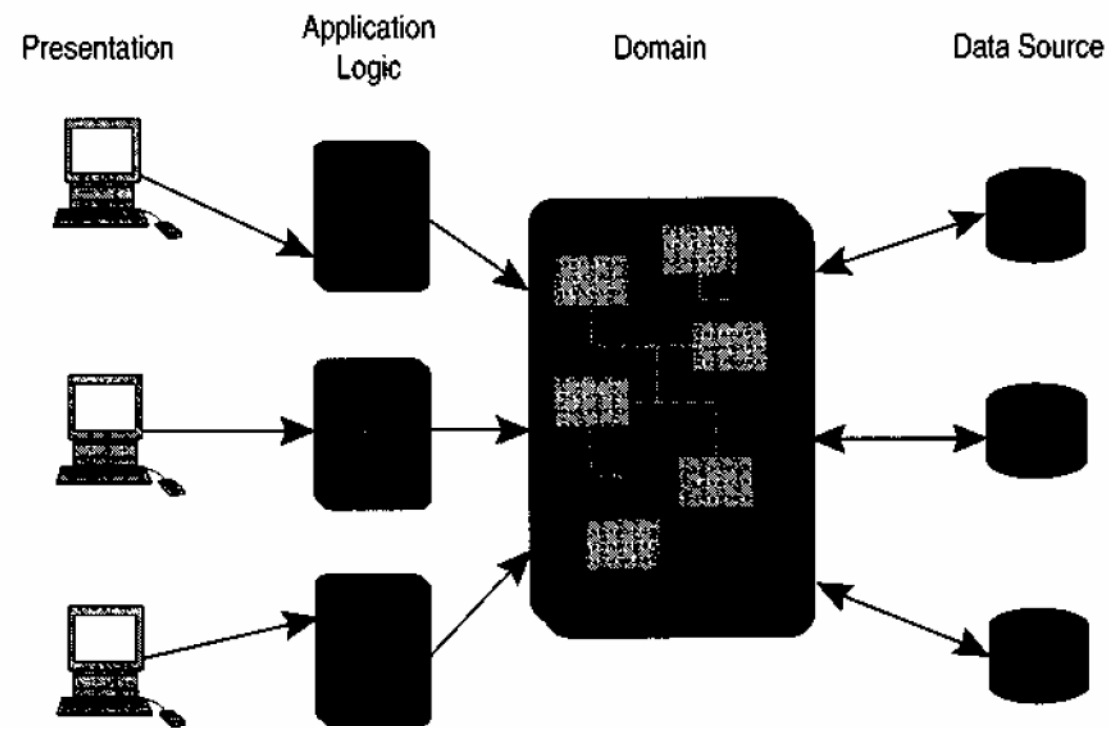
Three Tier

Supporting Patterns: Layered Architectures - *Presentation & Application Logic*

- **Problem:** Application software handles both interpretation of the domain model and driving the user interface.

- **Solution:**
  - Separate the application tier into presentation (user interface) and application logic (dealing with the domain model).
  - Structure the application logic as a set of facades for the presentation.

Supporting Patterns: Layered Architectures - *Presentation & Application Logic*



Supporting Patterns: Layered Architectures - *Database Interaction*

- **Problem:** Working with a database.

- **Solution:**
  - Let the domain classes be responsible for saving themselves in the database.
  - Create a separate layer to handle the interactions between database and domain objects.

Supporting Patterns: Type Model Design Templates – *Implementing Associations*

- **Problem:** Implementing a conceptual association.

- **Solution:**
  - Choose one direction to implement, and use an operation and a pointer.
  - Put operations and pointers in both directions.
  - Put operations in both directions but a pointer only in one. Use lookup for the other direction.
  - Put operations in both directions, and use a table and lookup for the pointers.

Supporting Patterns: Type Model Design Templates – *Implementing Generalizations*

- **Problem:** Implementing generalization, especially if multiple and dynamic classification is involved.

- **Solution:**
  - Use inheritance.
  - Use classes for each combination of subtypes with multiple inheritance.
  - Use an internal flag.
  - Delegate to a hidden class (state pattern).
  - Copy and replace.

Supporting Patterns: Type Model Design Templates – *Object Creation*

- **Problem:** Creating an object.

- **Solution:** Use a creation method with arguments for all mandatory and immutable mappings.

Supporting Patterns: Type Model Design Templates – *Object Destruction*

- **Problem:** Destroying an object.
- **Solution:** Have a specific destruction method.
  - Define how much the delete should cascade.

Supporting Patterns: Type Model Design Templates - *Entry Point*

- **Problem:** Starting to look for objects.

- **Solution:**
  - Let the class be responsible for storing and finding its instances.
  - Have a registrar find and store objects.

Supporting Patterns: Type Model Design Templates – *Implementing Constraints*

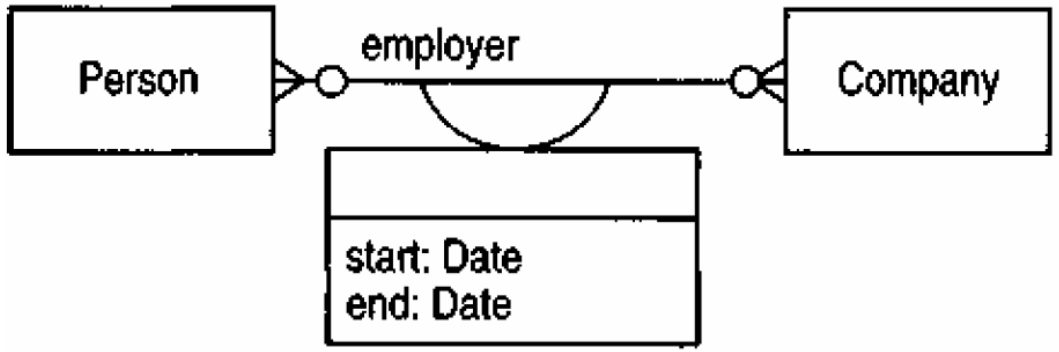
- **Problem:** Implementing constraints.

- **Solution:** Give each object an operation to check its constraint. Call it at the end of modifiers when debugging.

Supporting Patterns: Association Patterns - *Associative Type*

- **Problem:** Adding features to an association.
- **Solution:** Create a type for the association.
  - Use a special notation.

Supporting Patterns: Association Patterns - *Associative Type*

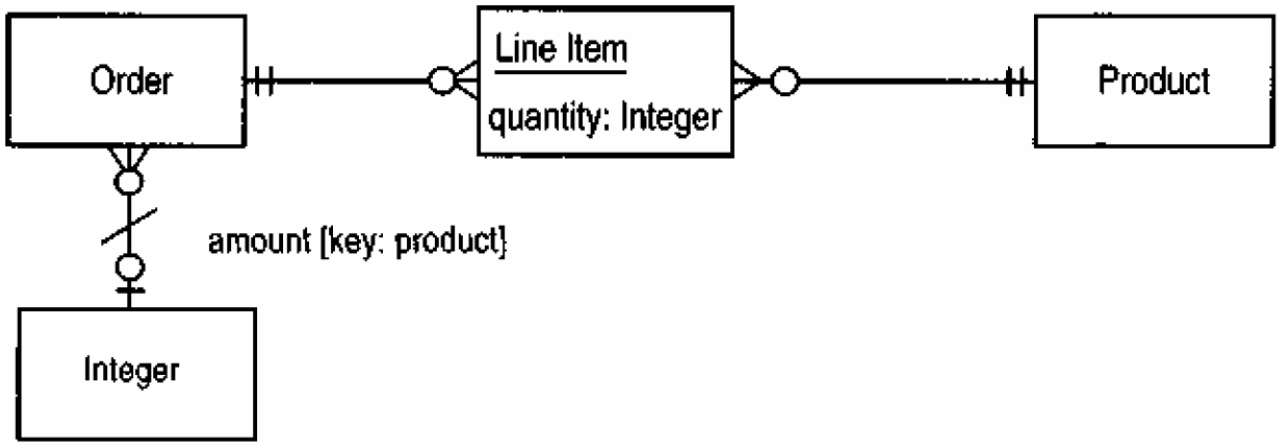


Supporting Patterns: Association Patterns - *Keyed Mapping*

- **Problem:** Representing values in a mapping that are keyed off another type.

- **Solution:** Use a keyed mapping.

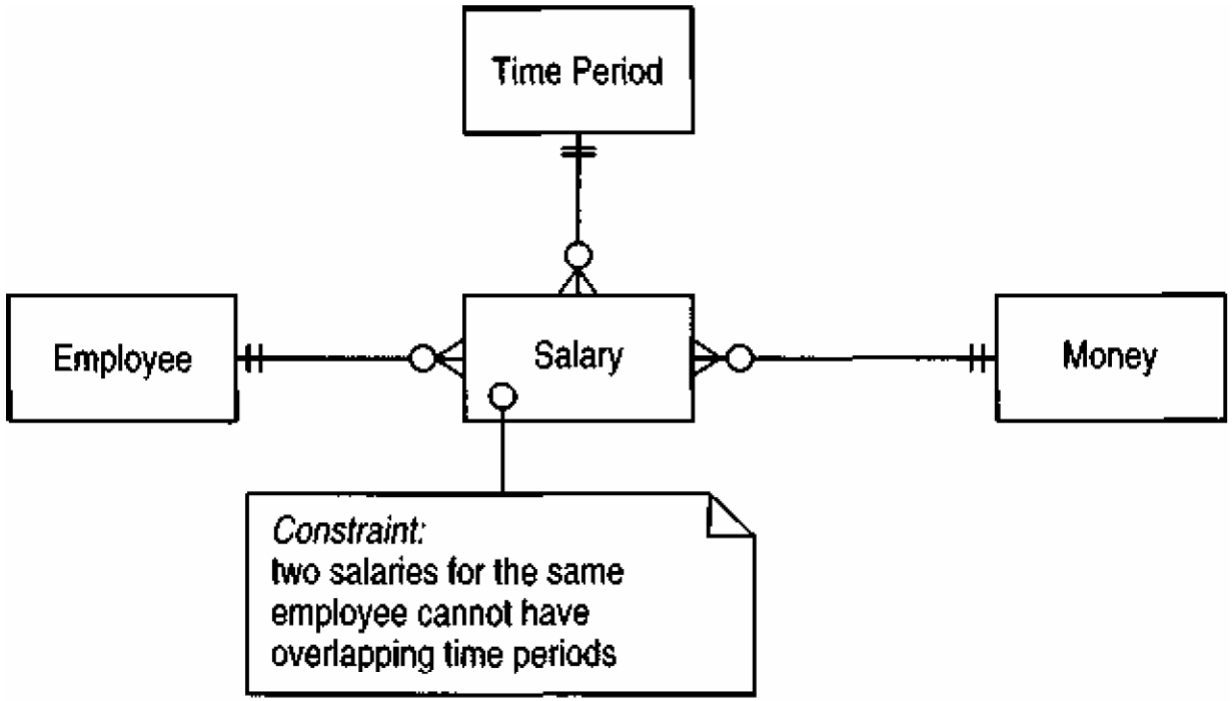
Supporting Patterns: Association Patterns - *Keyed Mapping*



Supporting Patterns: Association Patterns - *Historic Mapping*

- **Problem:** Recording previous values of a mapping.
- **Solution:** Use a historic mapping.

Supporting Patterns: Association Patterns - *Historic Mapping*



Reference

- Fowler, M., *Analysis Patterns: Reusable Object Models,* Addison-Wesley, 1997.