Chapter 06 (Cont.)
Specifying Relationships

Instructor: Aleenah Khan

Remaining Topics

- Multiplicity Indicators
- Reflexive Relationships
- Finding Relationships
- Package Relationships

Multiplicity Indicators

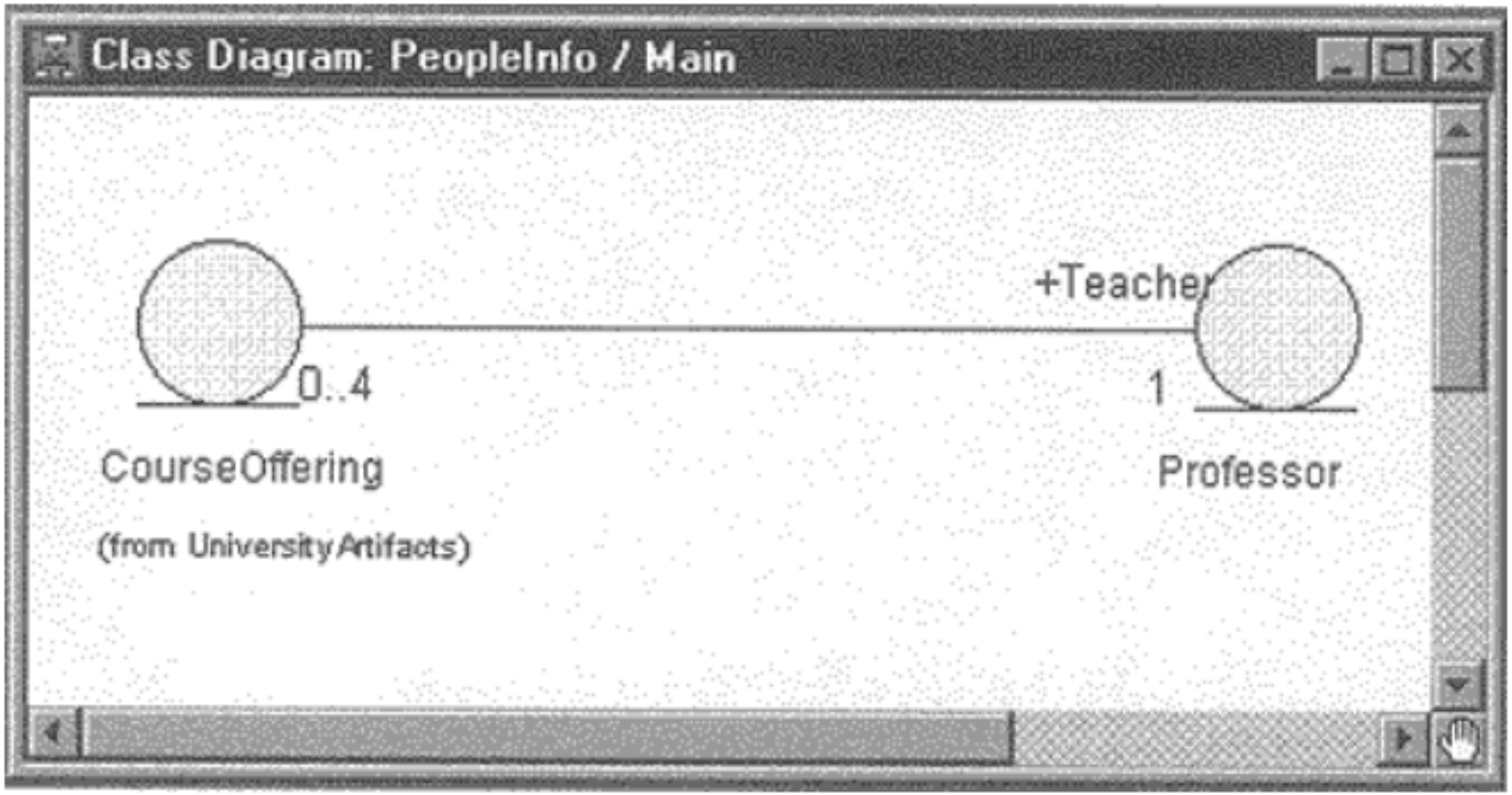
- Although multiplicity is specified for classes, it defines the number of objects that participate in a relationship.
- Multiplicity defines the number of objects that are linked to one another.
- There are two multiplicity indicators for each association or aggregation — one at each end of the line.

Multiplicity Indicators

- Some common multiplicity indicators are :
  - 1 Exactly one
  - 0.. *Zero or more
  - 1.. *One or more
  - 0..1 Zero or one
  - 5..8 Specific range (5, 6, 7, or 8)
  - 4..7,9 Combination (4, 5, 6, 7, or 9)

Multiplicity indicators are shown in Figure 6-7.

Figure 6-7. Multiplicity Indicators



Multiplicity Indicators

- The drawing in Figure 6-7 may be read in the following ways:
  - **One CourseOffering object is related to *exactly one* Professor object playing the role of the Teacher.**
    - For example, Math 101, Section 1 (a CourseOffering object) is related to Professor Smith (a Professor object).

Multiplicity Indicators

- **One Professor object playing the role of the Teacher is related to *zero to four* CourseOffering objects.**
  - For example, Professor Smith (a Professor object) is related to Math 101, Section 1; Algebra 200, Section 2; and Calculus 1, Section 3 (CourseOffering objects).

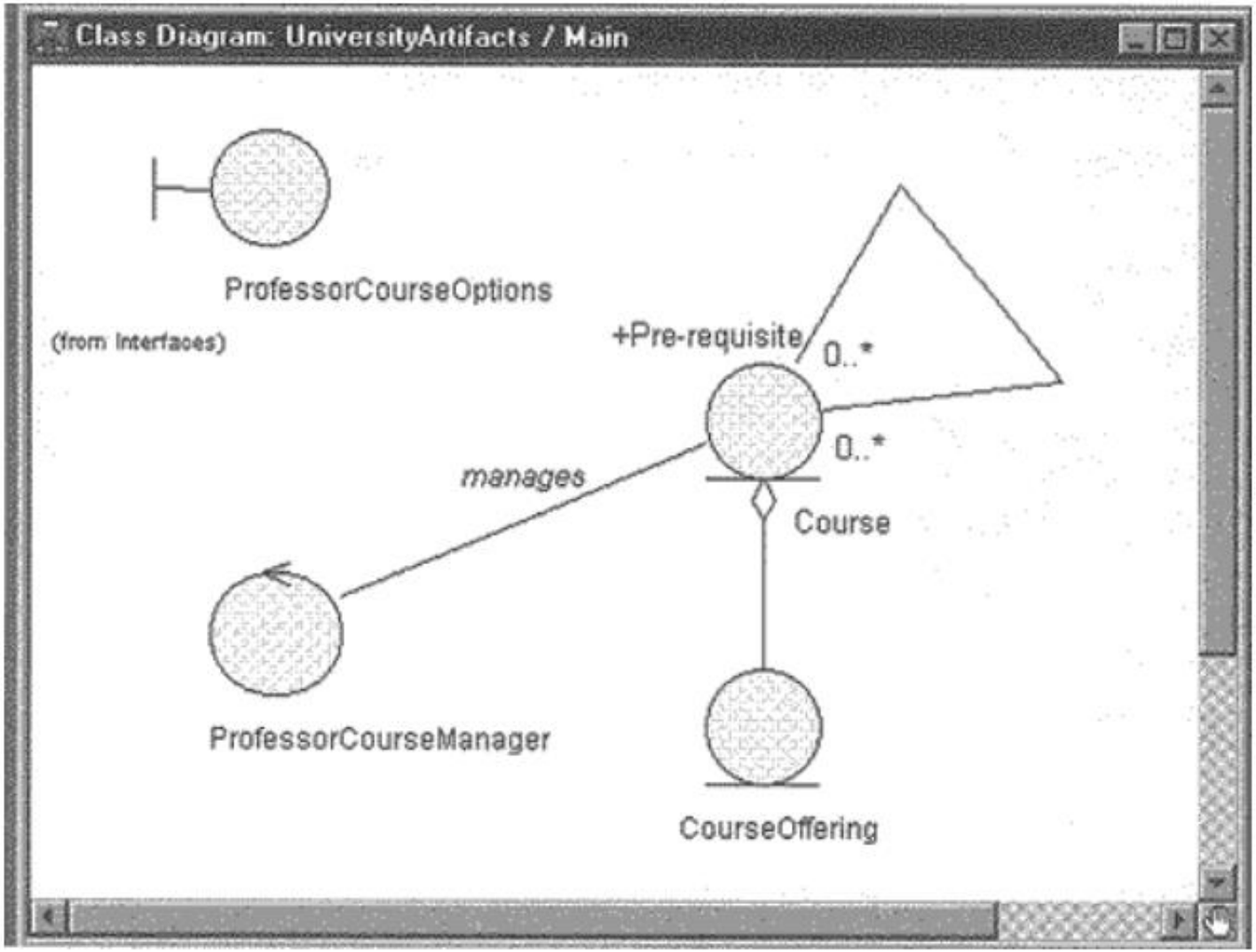
Since the multiplicity is a range of zero to four, as few as zero CourseOffering objects to a maximum of four CourseOffering objects may be linked to one Professor object.

Reflexive Relationships

- Multiple objects belonging to the same class may have to communicate with one another.
- This is shown on the class diagram as a ***reflexive association or aggregation.***
- Role names rather than association names typically are used for reflexive relationships.

A reflexive relationship is shown in Figure 6-8

Figure 6-8. Reflexive Relationship



Reflexive Relationships

- The reflexive relationship in Figure 6-8 may be read in the following ways:
  - **One Course object playing the role of Prerequisite is related to zero or more Course objects.**
  - **One Course object is related to zero or more Course objects playing the role of Prerequisite.**

Finding Relationships

- Scenarios are examined to determine if a relationship should exist between two classes.
- Messages between objects mean that the objects must communicate with each other.
- Associations and/or aggregations provide the pathway for communication.
- Relationships may also be discovered based on the signature of an operation. (To be discusssed later) This is discussed in Chapter 7.

Finding Relationships

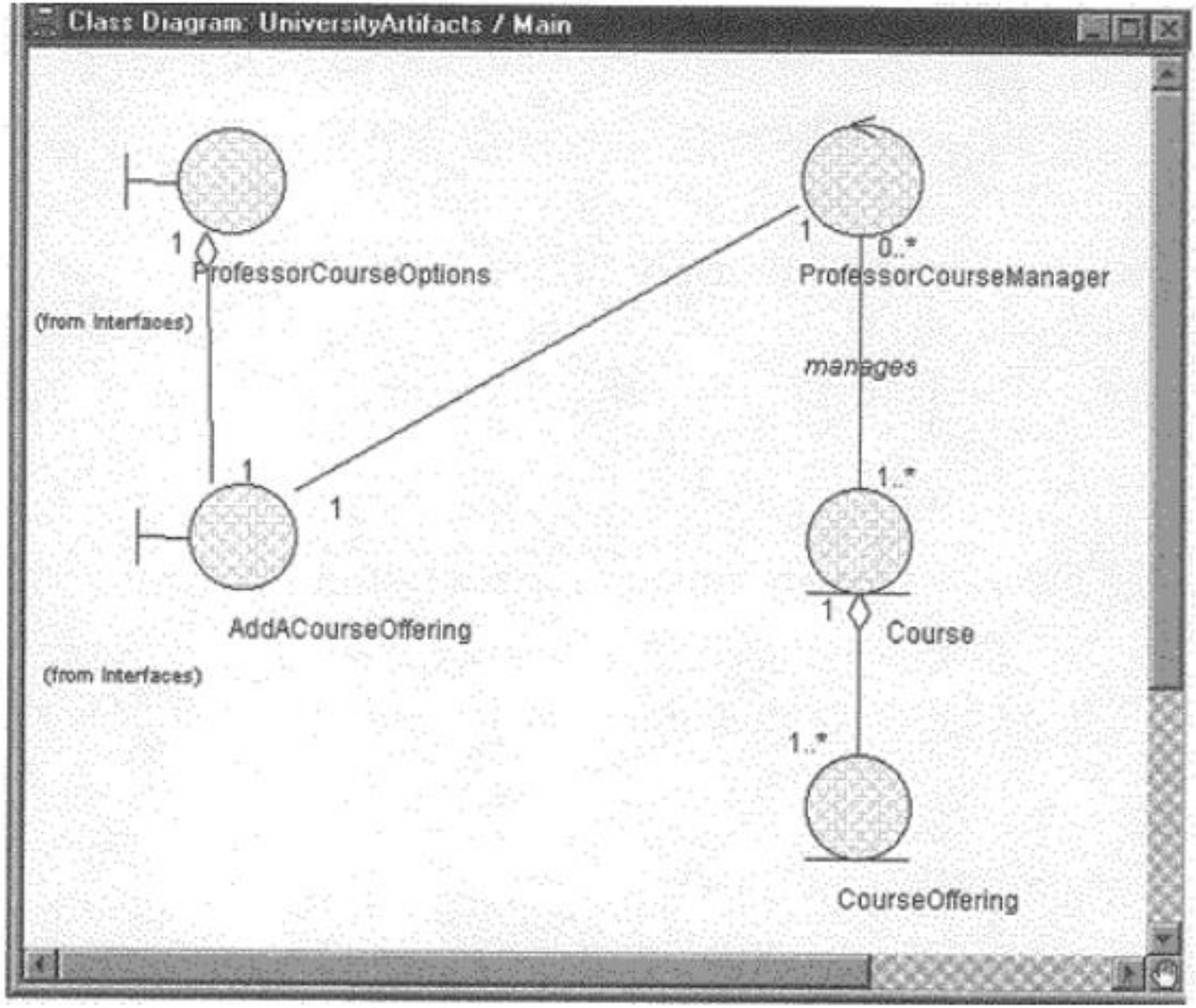
- In the *Add a Course Offering* scenario, the communicating objects along with the relationship-type decisions that have been made are shown in Table 6-1.

Table 6.1. Class Relationships

Sending Class	Receiving Class	Relationship Type
ProfessorCourseOptions	AddACourseOffering	Aggregation
AddACourseOffering	ProfessorCourseManager	Association
ProfessorCourseManager	Course	Association
Course	CourseOffering	Aggregation

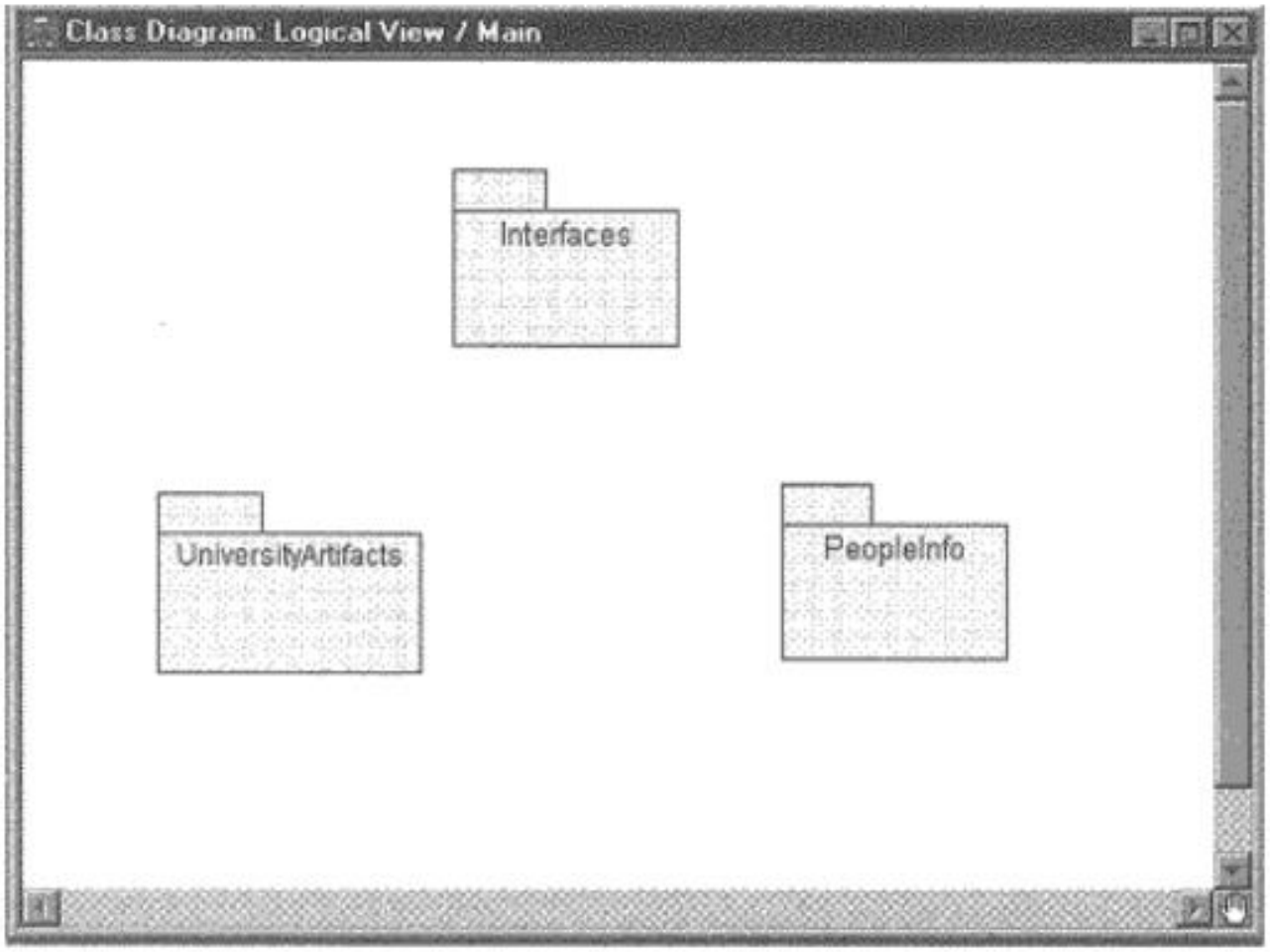
A class diagram with the added relationships is shown in Figure 6-9

Figure 6-9. Relationships in the Add a course Offering Scenario



Package

- A package in the logical view of the model is a collection of related packages and/or classes.
- By grouping classes into packages, we can look at the "higher" level view of the model (i.e., the packages) or we can dig deeper into the model by looking at what is contained by the package.


Class Diagram: Logical View / Main
Interfaces
UniversityArtifacts
PeopleInfo

Package Relationships

- Package relationships are also included to the model.
- The type of relationship is a dependency relationship, shown as a dashed arrow to the dependent package, as shown in Figure 6-10.

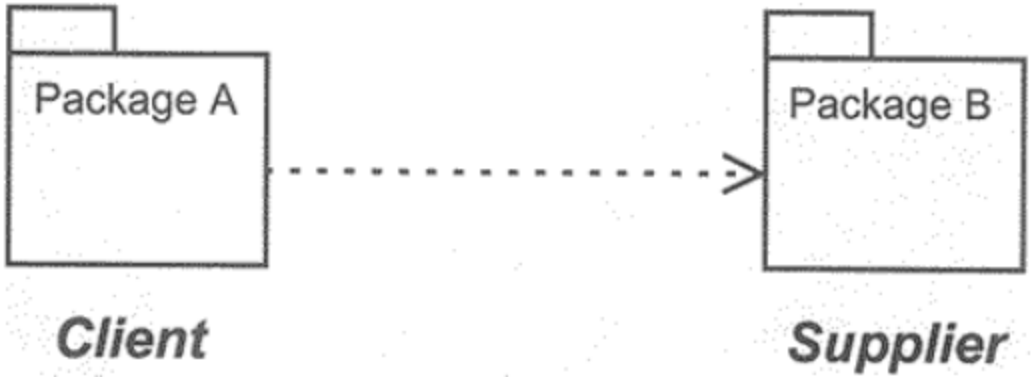


Figure 6-10. Package Relationships

Package Relationships

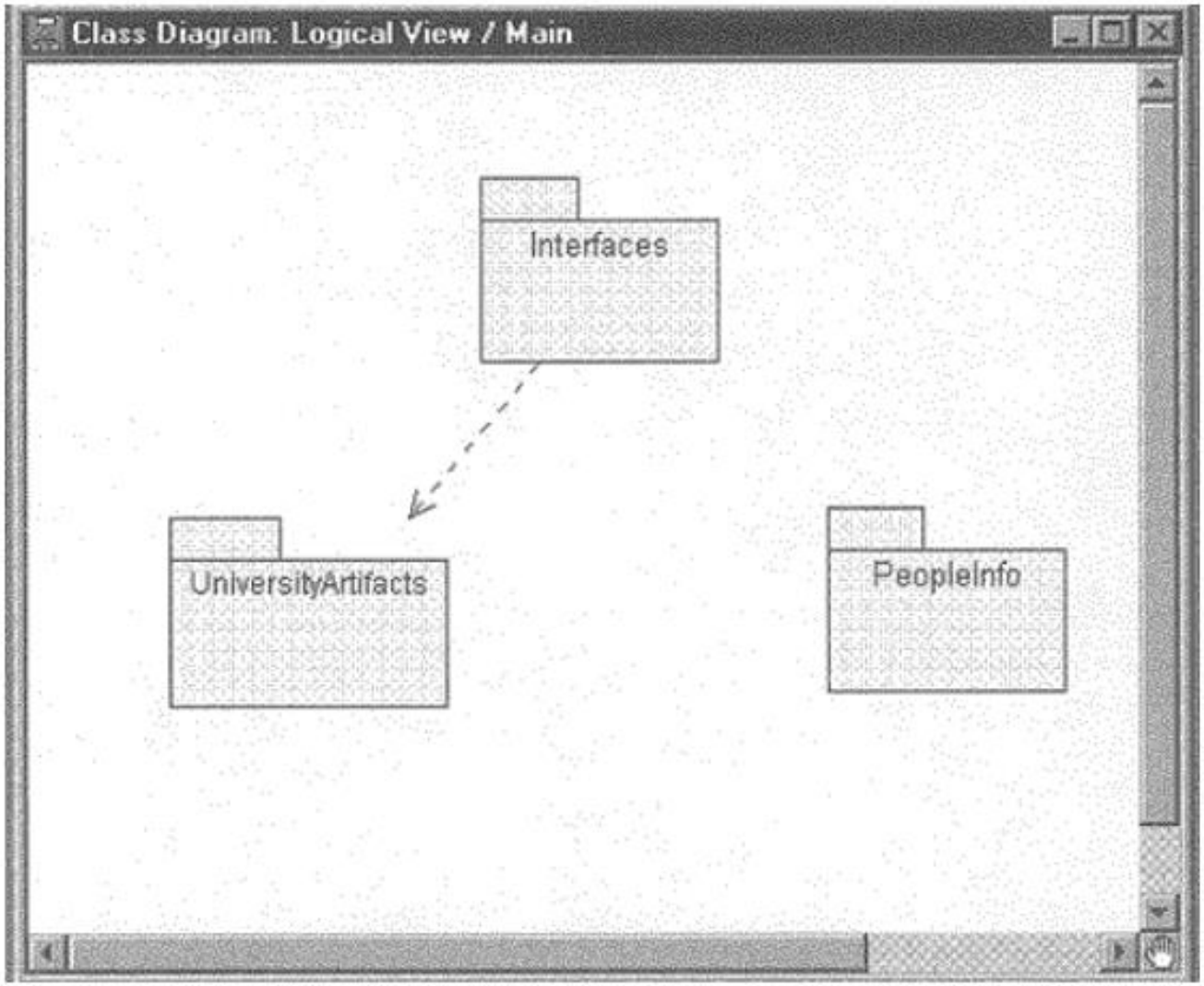
- If package A is dependent on package B this implies that one or more classes in package A initiates communication with one or more public classes in package B.
- Package A is referred to as the **Client** package and package B is referred to as the **Supplier** package.
- Package relationships are also discovered by examining the scenarios and class relationships for the system under development.
- This is an iterative process, and the relationships will change as analysis and design progresses.

Package Relationships

- In the *Add a Course Offering* scenario, the **AddACourseOffering** class sends a message to the **ProfessorCourseManager** class.
- This implies that there is a relationship between the **Interfaces** package and the **UniversityArtifacts** package.
- At this time, we have not discovered any relationships to the People package.

The package relationships for the Course Registration System are shown in Figure 6.11

Figure 6-11. Package Relationships in the ESU Course Registration System



Summary

- Relationships provide the conduit for object interaction.
- Two types of relationships, between classes that are discovered during analysis are associations and aggregations.
- An association is a bidirectional semantic connection between classes.
- An aggregation is a specialized form of association in which a whole is related to its part(s).
- An association may be named. Usually the name is an active verb or verb phrase that communicates the meaning of the relationship.

Summary

- Roles can be used instead of association names.
- A role name is a noun that denotes the purpose or capacity wherein one class associates with another.
- Multiplicity is the number of instances that participate in a relationship.
- There are two multiplicity indicators for each association or aggregation— one at each end of the relationship line.
- Multiple objects belonging to the same class may have to communicate with one another. This is shown on the class diagram as a reflexive association or aggregation.

Summary

- Scenarios are examined to determine if a relationship should exist between two classes.
- Packages are related via dependency relationships.
- If package A is dependent on package B, this implies that one or more classes in package A initiates communication with one or more public classes in package B.