IT4490 - SOFTWARE DESIGN AND CONSTRUCTION

6. IDENTIFY DESIGN ELEMENTS

Some slides extracted from IBM coursewares

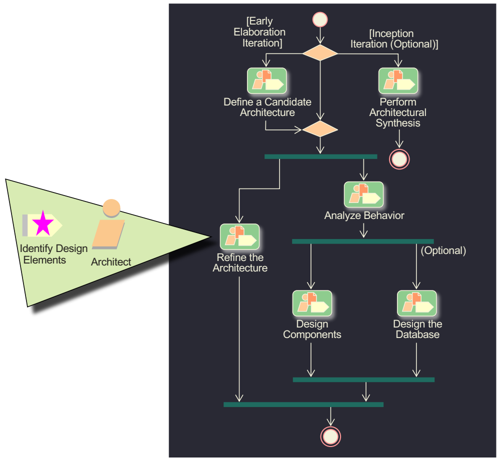
1

Objectives: Identify Design Elements

- Define the purpose of Identify Design Elements and demonstrate where in the lifecycle it is performed
- Analyze interactions of analysis classes and identify Design Model elements => Design classes

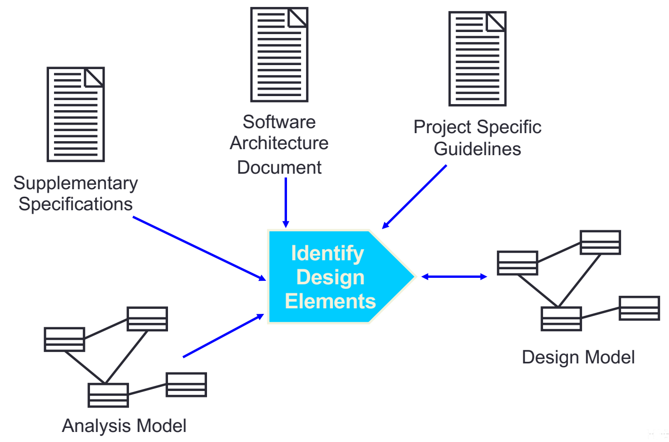
2

Identify Design Elements in Context


[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analyze Behavior
Identify Design Elements
Architect
Refine the Architecture
(Optional)
Design Components
Design the Database

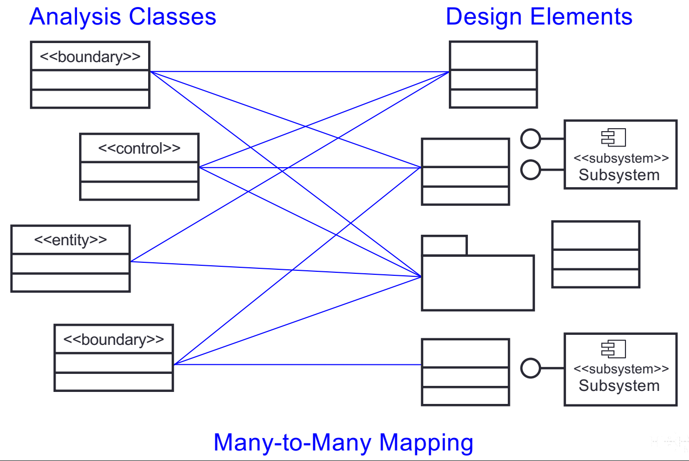
3

Identify Design Elements Overview


Supplementary Specifications
Software Architecture Document
Project Specific Guidelines
Identify Design Elements
Design Model
Analysis Model

4

From Analysis Classes to Design Elements



5

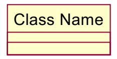
Identifying Design Classes

- An analysis class maps directly to a design class if:
  - It is a simple class
  - It represents a single logical abstraction
- More complex analysis classes may
  - Split into multiple classes
  - Become a package
  - Become a subsystem (discussed later)
  - Any combination …

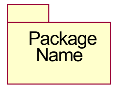
6

Review: Class and Package

- What is a class?
  - A description of a set of objects that share the same responsibilities, relationships, operations, attributes, and semantics



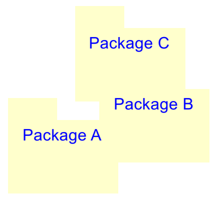
- What is a package?
  - A general purpose mechanism for organizing elements into groups
  - A model element which can contain other model elements



7

Group Design Classes in Packages

- You can base your packaging criteria on a number of different factors, including:
  - Configuration units
  - Allocation of resources among development teams
  - Reflect the user types
  - Represent the existing products and services the system uses



8

Packaging Tips: Boundary Classes

If it is **likely** the system interface will undergo considerable changes

Boundary classes placed in separate packages

If it is **unlikely** the system interface will undergo considerable changes

Boundary classes packaged with functionally related classes

9

Packaging Tips: Functionally Related Classes

- Criteria for determining if classes are functionally related:
  - Changes in one class' behavior and/or structure necessitate changes in another class
  - Removal of one class impacts the other class
  - Two objects interact with a large number of messages or have a complex intercommunication
  - A boundary class can be functionally related to a particular entity class if the function of the boundary class is to present the entity class
  - Two classes interact with, or are affected by changes in the same actor

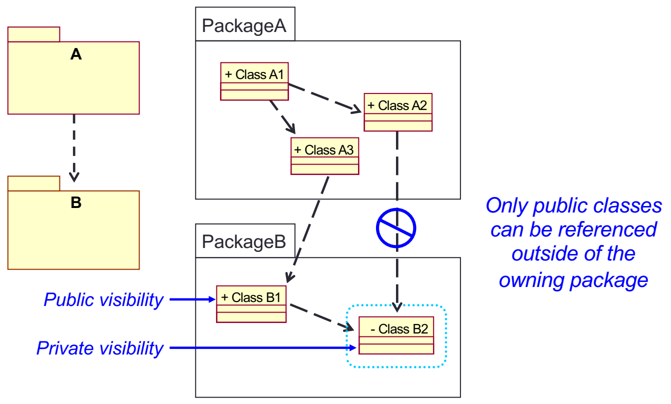
10

Packaging Tips: Functionally Related Classes (continued)

- Criteria for determining if classes are functionally related (continued):
  - Two classes have relationships between each other
  - One class creates instances of another class
- Criteria for determining when two classes should **NOT** be placed in the same package:
  - Two classes that are related to different actors should not be placed in the same package
  - An optional and a mandatory class should not be placed in the same package

11

Package Dependencies: Package Element Visibility

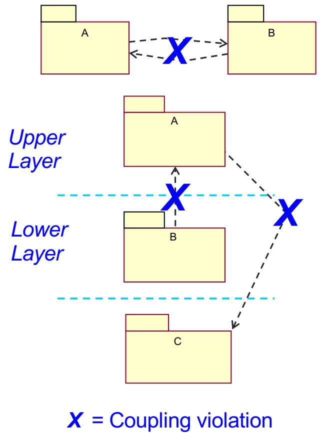


OO Principle: Encapsulation

12

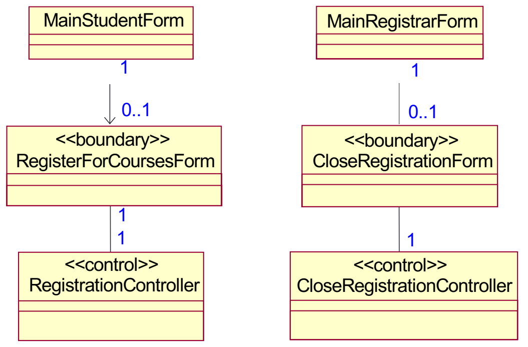
Package Coupling: Tips

- Packages should not be cross-coupled
- Packages in lower layers should not be dependent upon packages in upper layers
- In general, dependencies should not skip layers



13

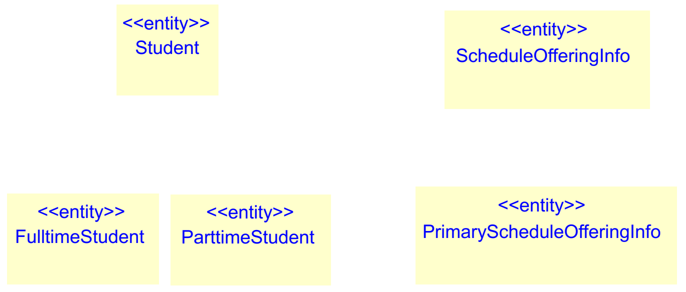
Example: Registration Package



MainStudentForm
1
0..1
<<boundary>> RegisterForCoursesForm
1
1
<<control>> RegistrationController
MainRegistrarForm
1
0..1
<<boundary>> CloseRegistrationForm
1
<<control>> CloseRegistrationController

14

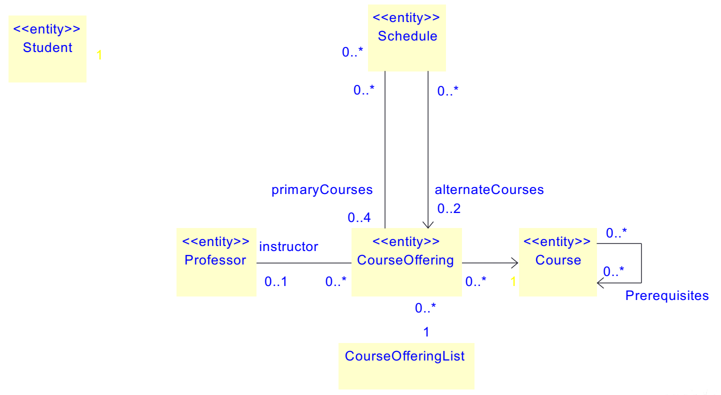
Example: University Artifacts Package: Generalization



<<entity>> Student
<<entity>> ScheduleOfferingInfo
<<entity>> FulltimeStudent
<<entity>> ParttimeStudent
<<entity>> PrimaryScheduleOfferingInfo

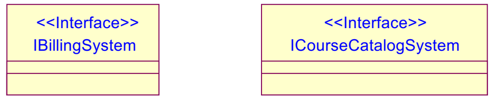
15

Example: University Artifacts Package: Associations



16

Example: External System Interfaces Package



17