Chapter 8

Conceptual Domain Modeling

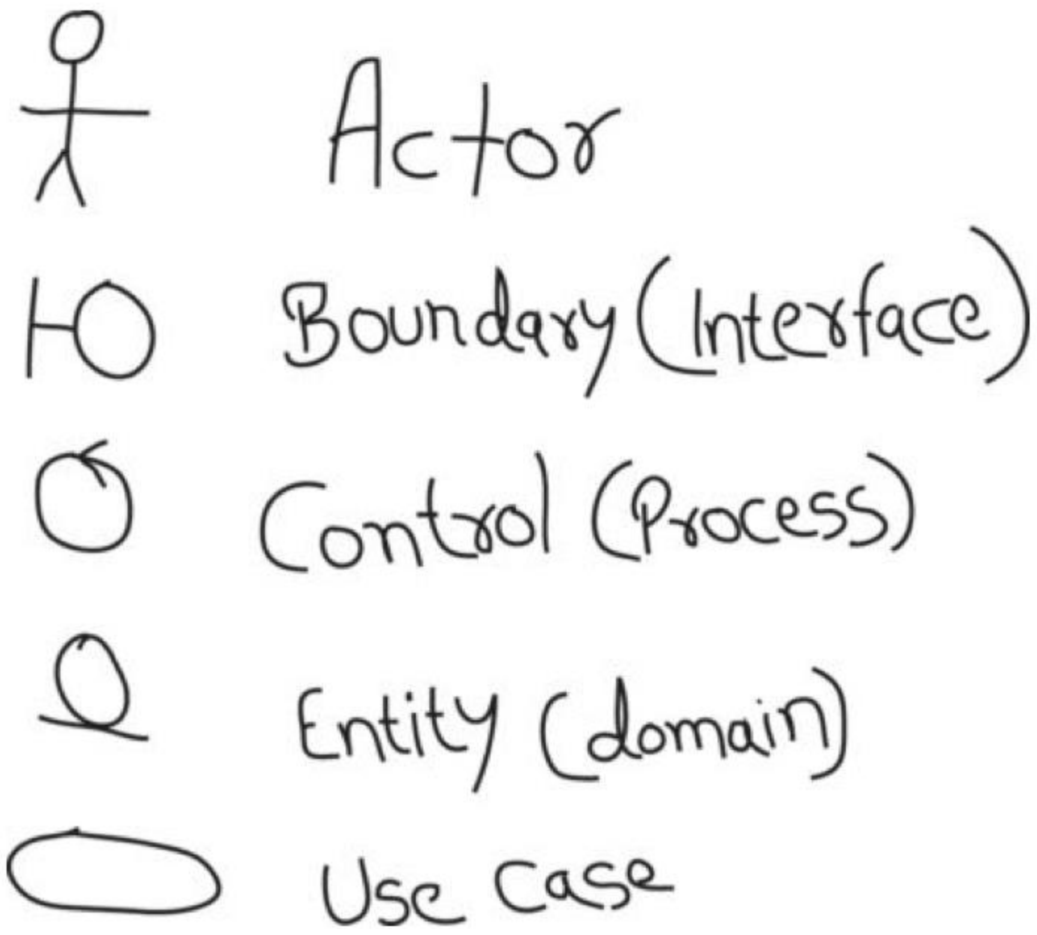
What Is It?

- The task of discovering the entity types that represent the things and concepts, and their relationships, pertinent to your problem space.
- Nouns and noun phrases within your requirements models are good candidates for concepts that should be included in your conceptual model.
- Many of the entities that appear in your conceptual model will also appear in your glossary.
- AKA domain modeling, conceptual modeling.

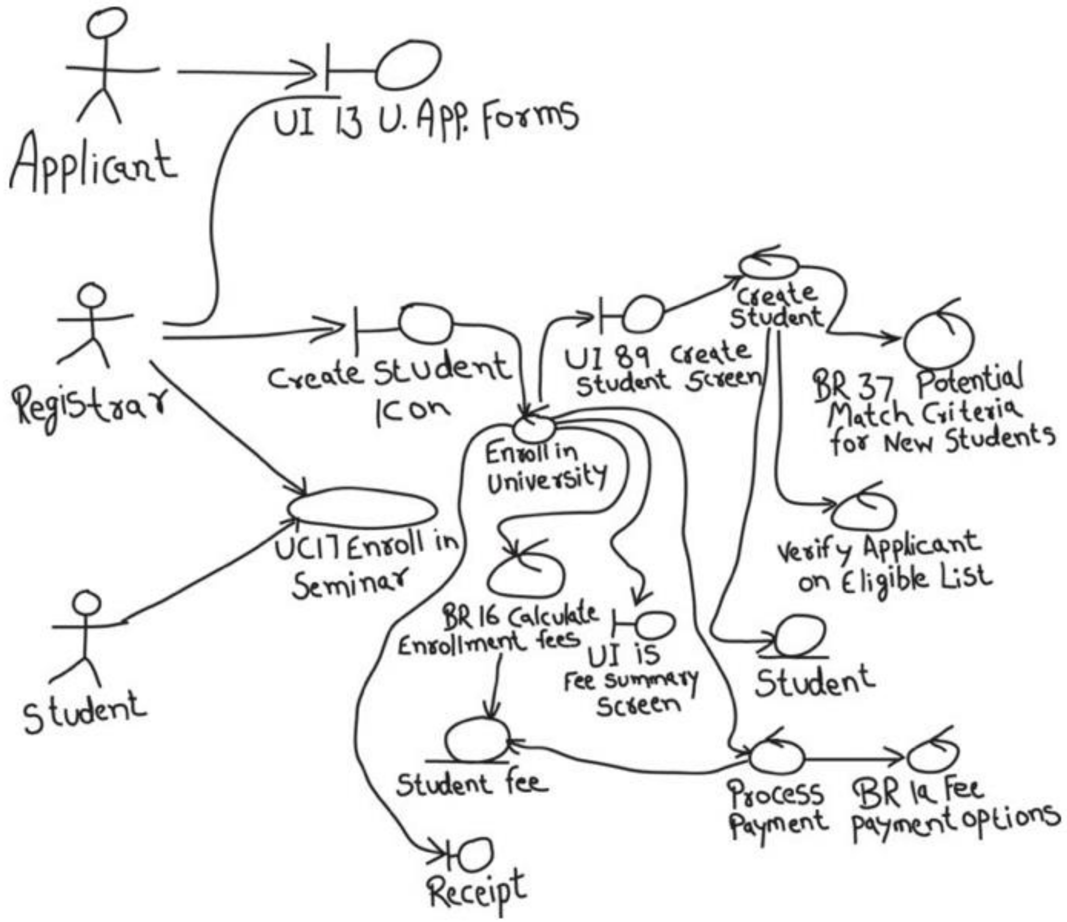
Robustness Diagrams

- Depict several types of concepts:
  - Actors. This is the same concept as actors on a UML use case diagram.
  - Boundary elements. These represent software elements such as screens, reports, HTML pages, or system interfaces that actors interact with. Also called interface elements.
  - Control elements. These serve as the glue between boundary elements and entity elements, implementing the logic required to manage the various elements and their interactions; they are also known as process elements or simply as controllers.
  - Entity elements. These are entity types typically found in your conceptual model. Examples include Student and Seminar.
  - Use cases (optional). Because use cases can invoke other use cases, you need to be able to depict this on your robustness diagrams.

Robustness Diagrams Visual Stereotypes



Robustness Diagram Example



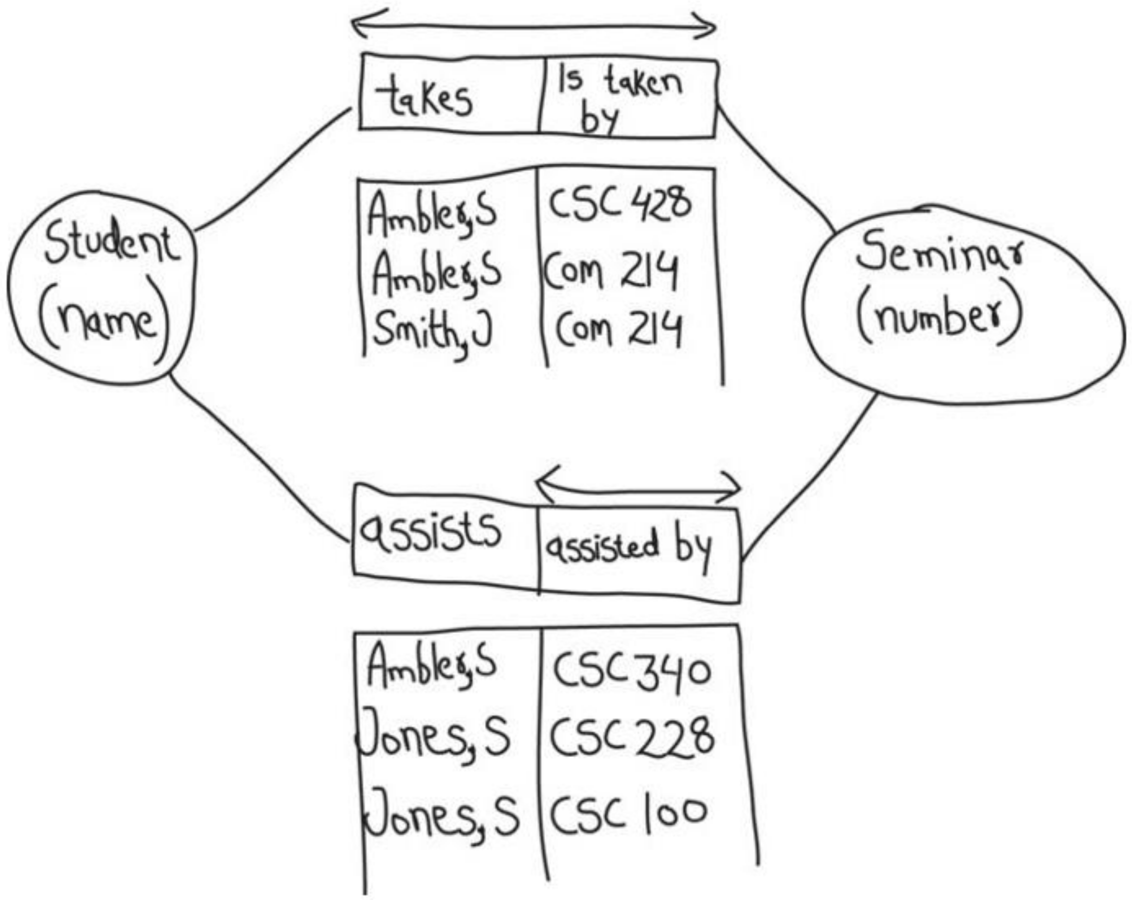
Robustness Diagram Rules Of Thumb

- Add a boundary element for each major user-interface element such as a screen or a report.
- Add a controller to manage the overall process of the scenario being modeled.
- Add a controller for each business rule. This helps to make the business rules explicit on your diagram.
- Add a controller for activities that involve several other elements.
- Add an entity for each business concept.
- Add a use case whenever one is included in the scenario.

Object Role Model Diagrams

- Depicts:
  - objects (entity types),
  - the relationships (fact types) between them,
  - the roles that the objects play in those relationships,
  - constraints within the problem domain, and, optionally,
  - examples (called fact-type tables).
- Ovals represent entity types
- Rectangles represent roles that the objects play in relationships.
- The double arrowhead above the roles indicates uniqueness constraints.
- Where there are no arrowheads at all above the roles that indicate an unrestricted many-to-many relationship

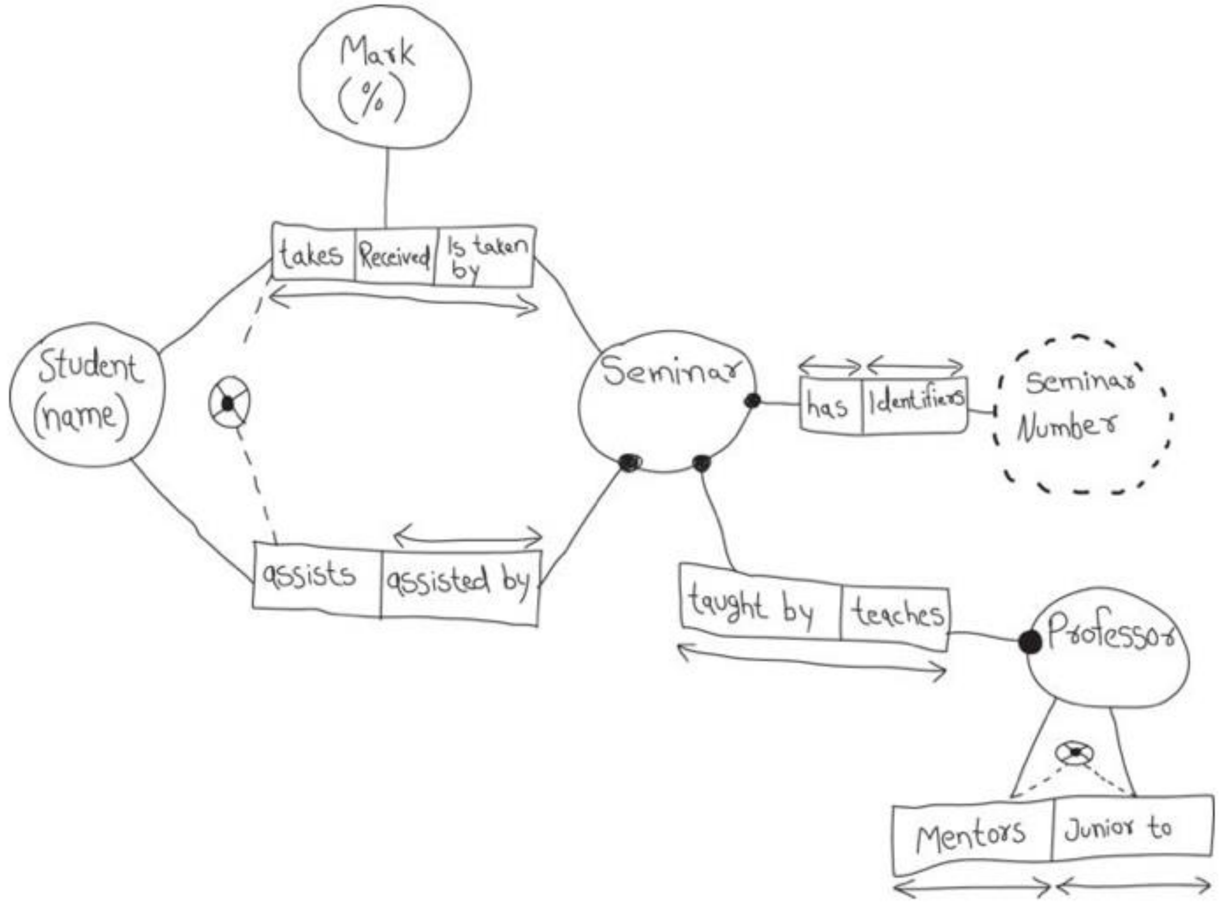
Simple ORM Diagram Example



Complex ORM Diagram

- It is possible to model n-ary relationships on ORM diagrams simply by adding more roles to the relationship.
- Dotted circle with an X inside indicates an exclusive (XOR) relationship.
- Dotted circle without an X inside indicates a OR relationship.
- Dark dot indicates mandatory relationship.
- Dashed oval represents attributes of an object.
- Can show recursive relationships.
- Try UML Object Diagram instead.

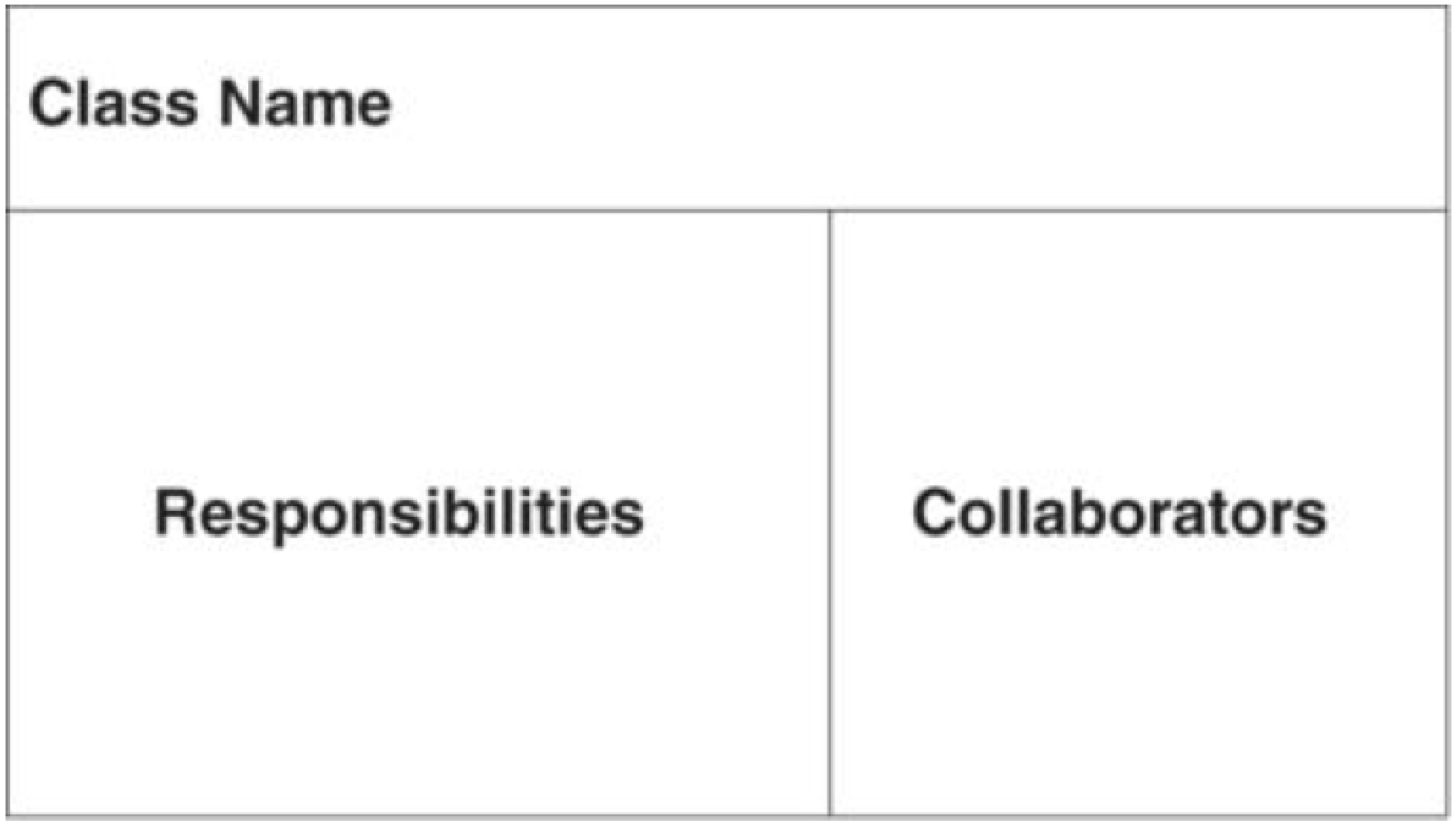
Complex ORM Diagram Example



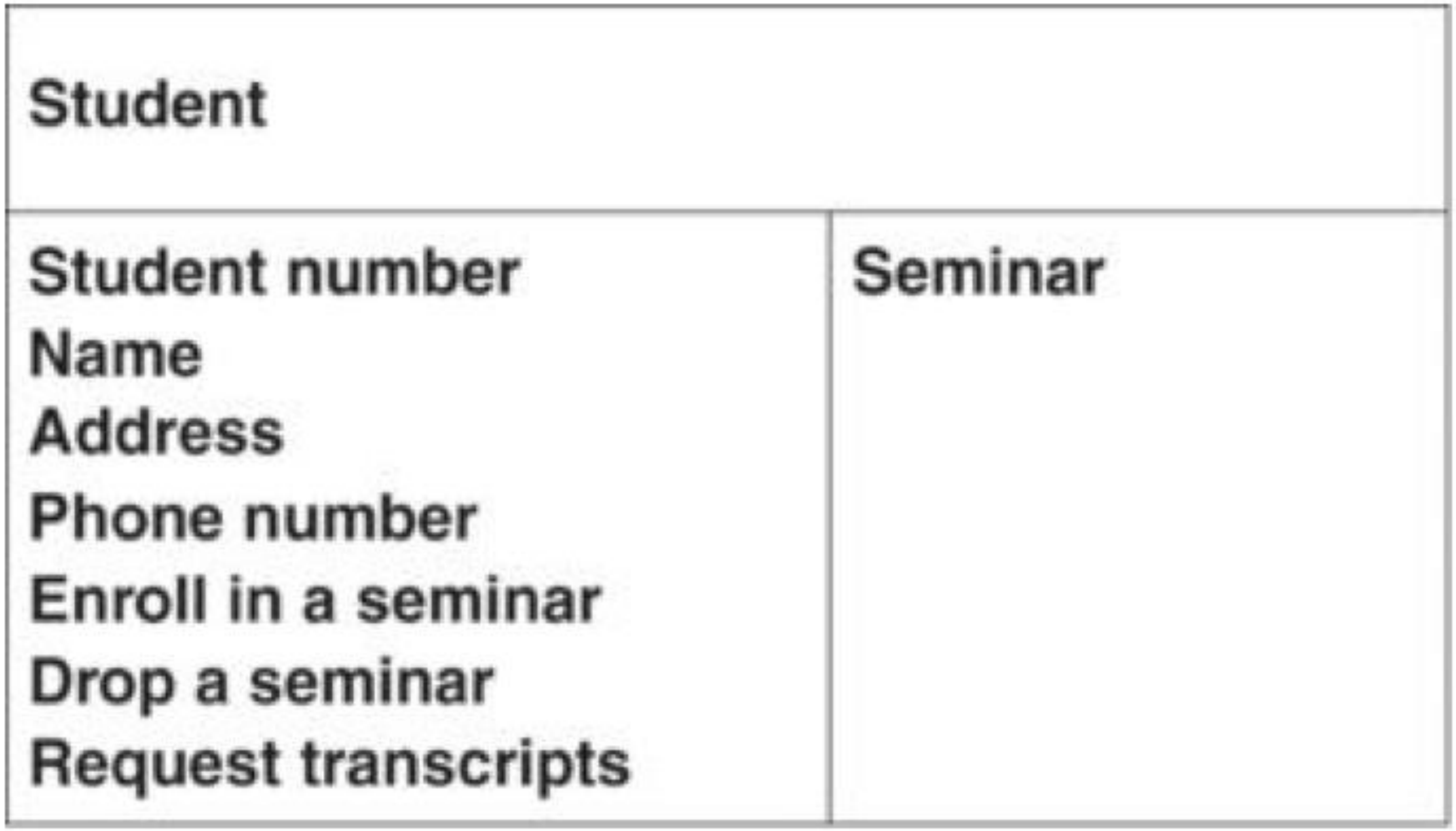
Class Responsibility Collaborator (CRC) Cards

- A collection of standard index cards that have been divided into three sections:
  - A class represents a collection of similar objects.
    - An object is a person, place, thing, event, or concept relevant to the system at hand.
  - A responsibility is something that a class knows (attribute) or does (method).
  - A collaborator is another class that a class interacts with to fulfill its responsibilities.
    - Collaboration takes one of two forms: A request for information or a request to do something.

CRC Card Layout

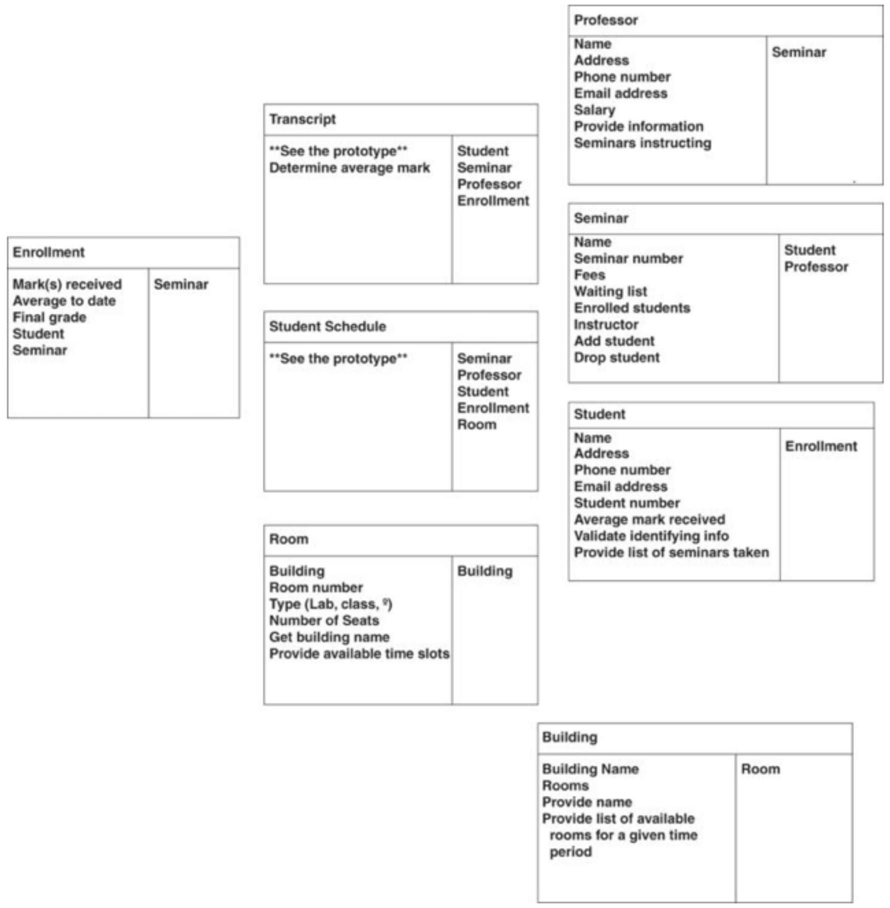


CRC Card Example



Student	
Student number Name Address Phone number Enroll in a seminar Drop a seminar Request transcripts	Seminar

A Stack Of CRC Cards



Enrollment	
Mark(s) received	Seminar
Average to date	
Final grade	
Student	
Seminar	

Transcript	
See the prototype	Student
Determine average mark	Seminar
	Professor
	Enrollment

Student Schedule	
See the prototype	Seminar
	Professor
	Student
	Enrollment
	Room

Room	
Building	Building
Room number	
Type (Lab, class, …)	
Number of Seats	
Get building name	
Provide available time slots	

Professor	
Name	Seminar
Address	
Phone number	
Email address	
Salary	
Provide information	
Seminars instructing	

Seminar	
Name	Student
Seminar number	Professor
Fees	
Waiting list	
Enrolled students	
Instructor	
Add student	
Drop student	

Student	
Name	Enrollment
Address	
Phone number	
Email address	
Student number	
Average mark received	
Validate identifying info	
Provide list of seminars taken	

Building	
Building Name	Room
Rooms	
Provide name	
Provide list of available rooms for a given time period	

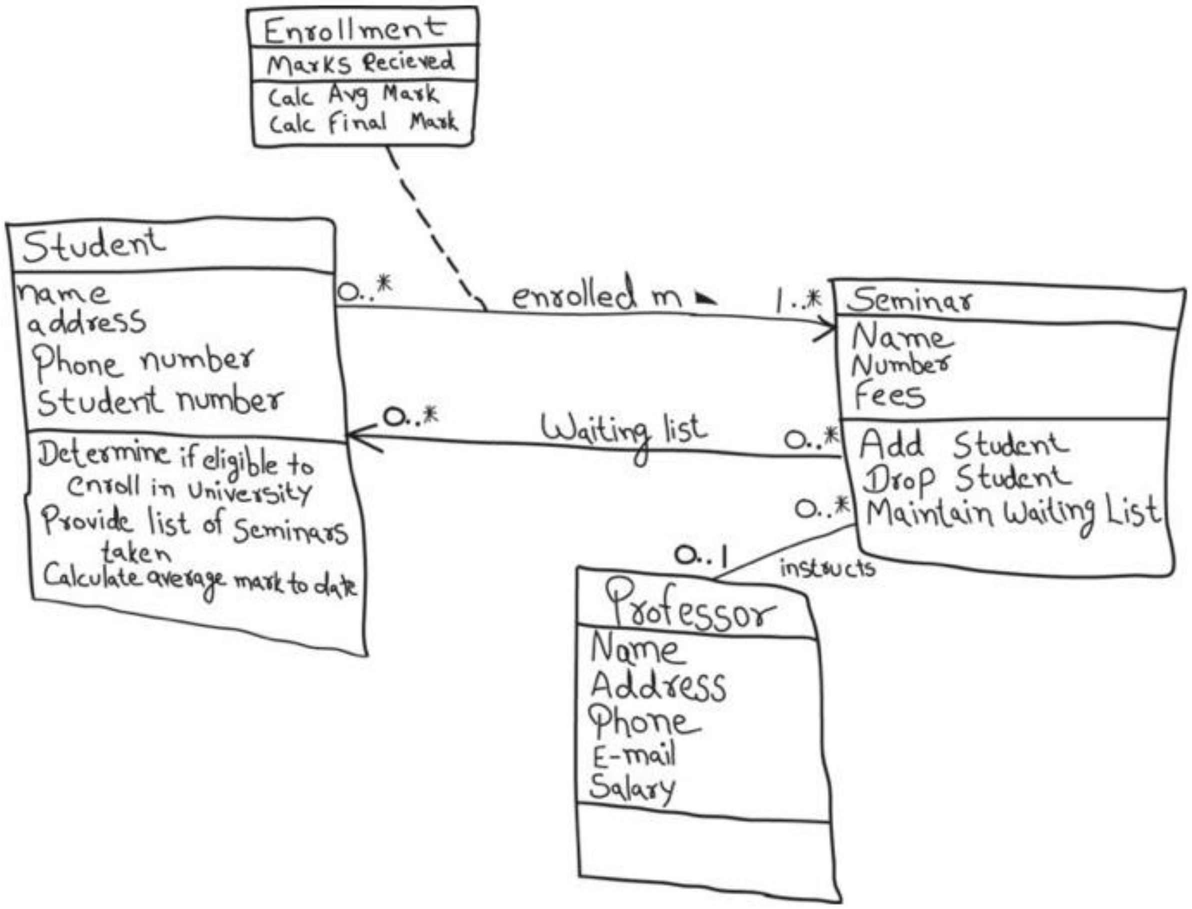
CRC Tips

- List a collaborator once on a card.
- List a collaborator only if there is a collaboration.
- Sometimes the collaborator does most of the work.
- Expect to move the cards around a lot in the beginning.
- Put “busy” cards towards the center of the table.
- Actually move the cards around.
- People will identify associations as they move cards around.
- The context determines whether something is a class or a responsibility.
- Choose a consistent style for data responsibilities. E.g., “Student Name,” not “Know Student Name”

Analysis Class Diagrams

- Class models show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes.
- Classes are depicted as boxes with three sections: the top one indicates the name of the class, the middle one lists the attributes of the class, and the third one lists the methods.

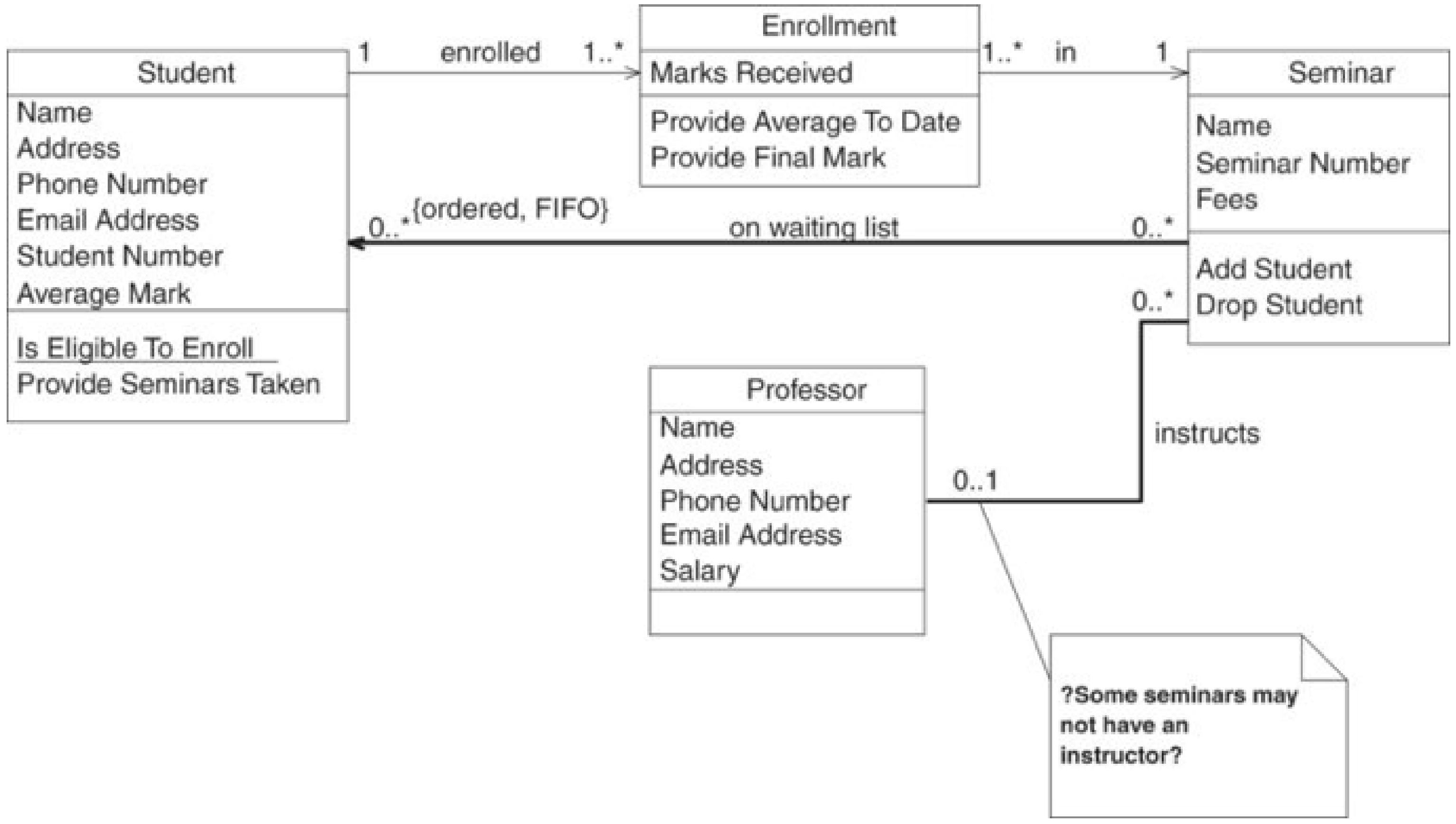
Analysis Class Diagram Example



Associative Class

- Used to model associations that have methods and attributes.
- Associative classes are typically modeled during analysis and then refactored.

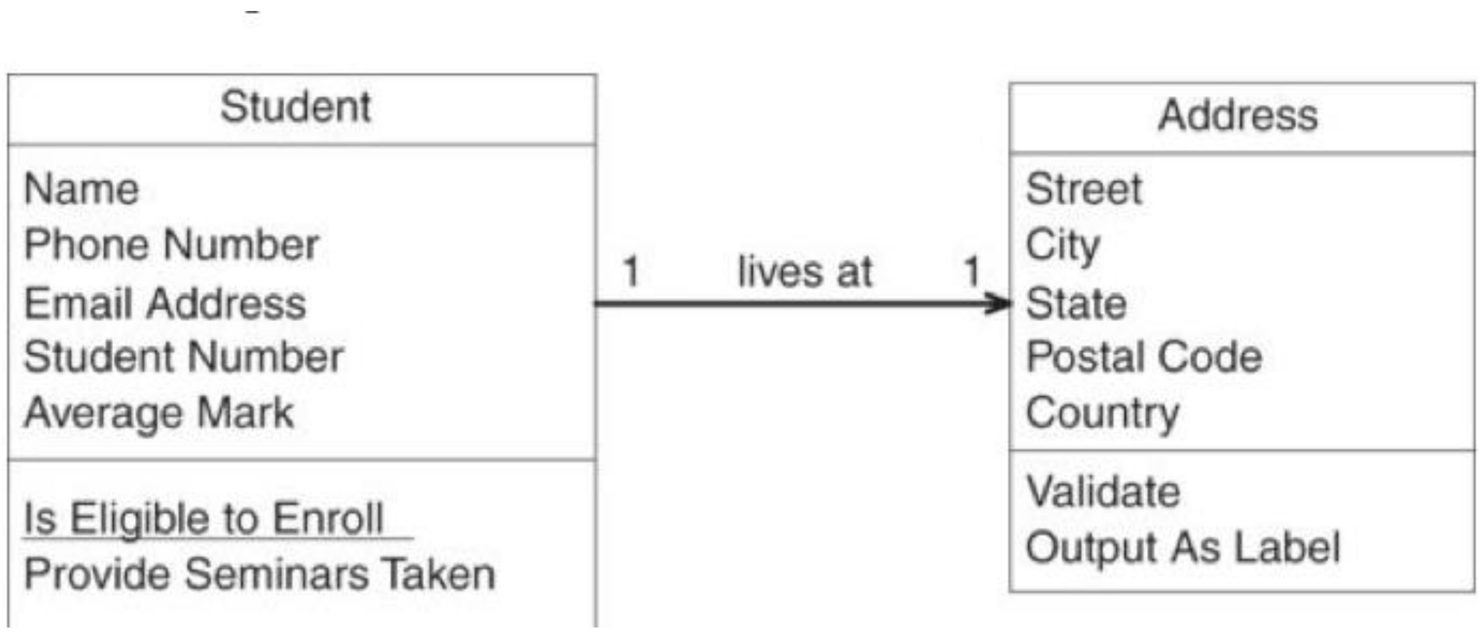
Refactored Class Diagram Example



Modeling Classes And Responsibilities

- An object is any person, place, thing, concept, event, screen, or report applicable to your system. Objects both know things (they have attributes) and they do things (they have methods).
- A class is a representation of an object and, in many ways, it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application.
  - Classes are typically modeled as rectangles with three sections: the top section for the name of the class, the middle section for the attributes of the class, and the bottom section for the methods of the class.
- Attributes are the information stored by an object (or at least information temporarily maintained about an object).
- Methods are the things an object or class do.

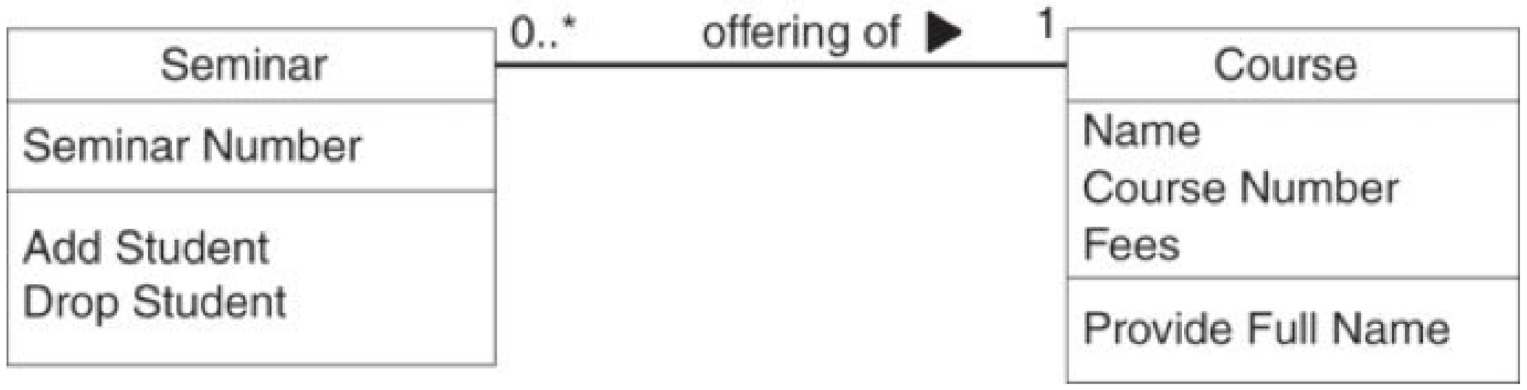
Class Diagram Example



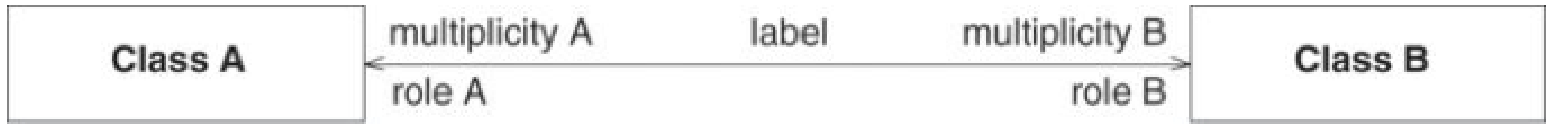
Class Normalization

- A process in which you refactor the behavior of classes to increase their cohesion and/ or to reduce the coupling between classes.

Normalized Class Example



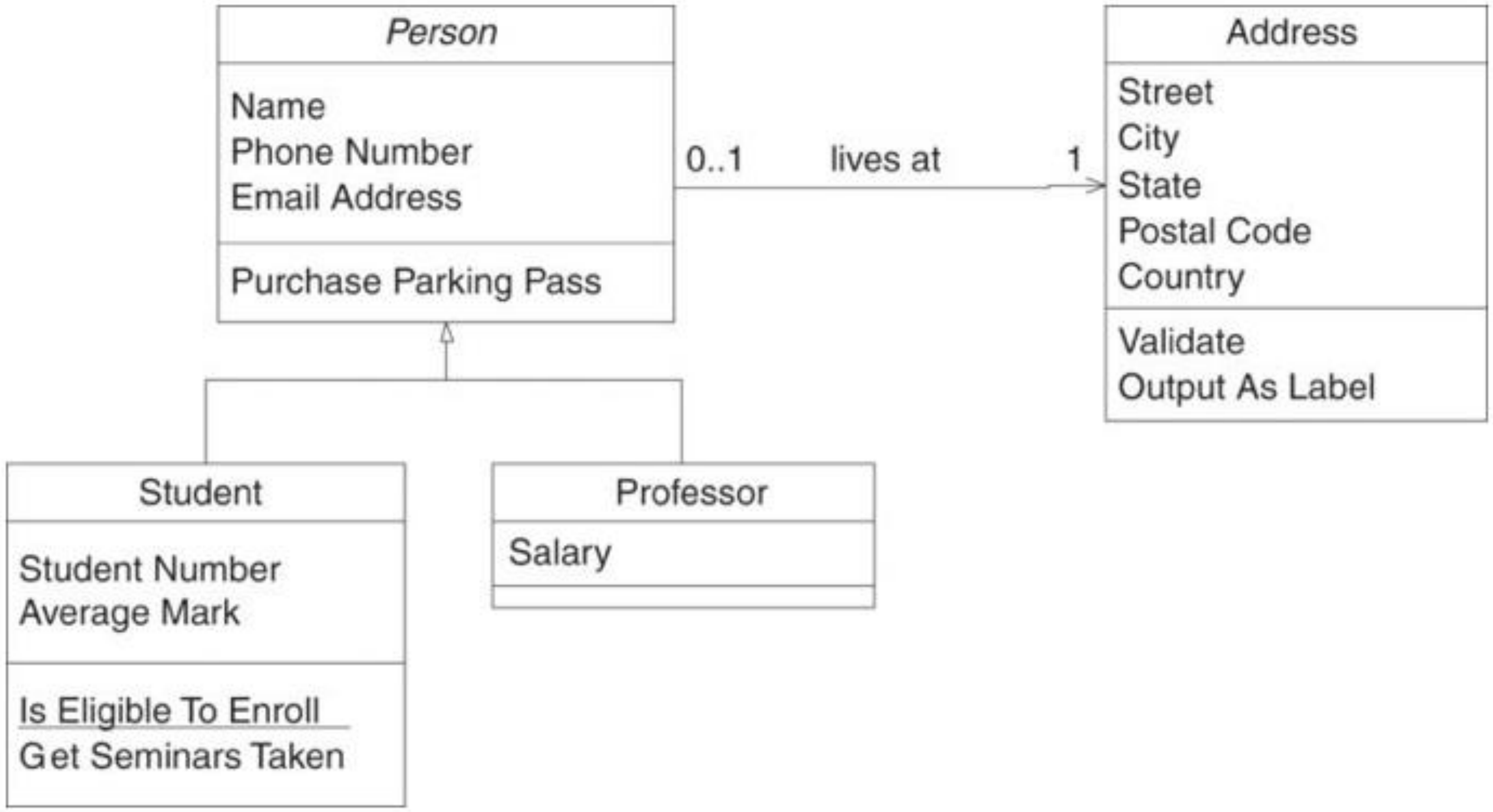
Associations Syntax



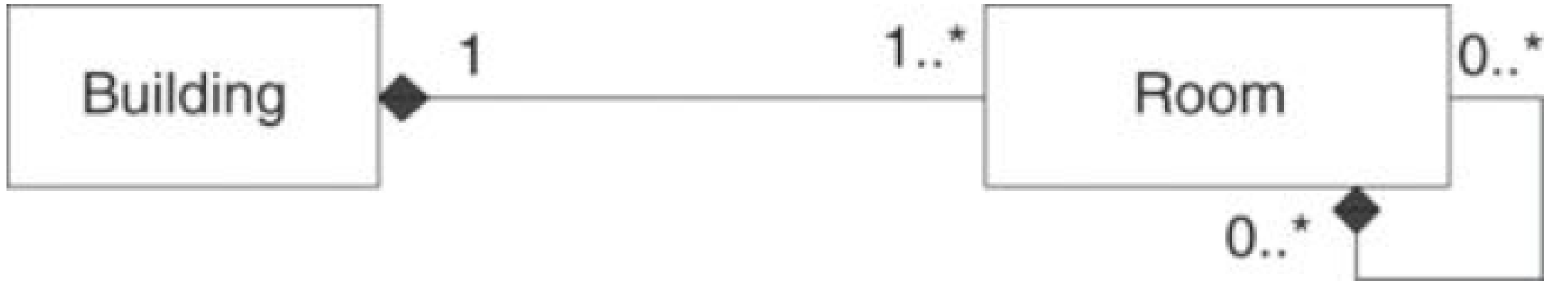
Multiplicity

0. .1	Zero or one
1	One only
0. .*	Zero or more
1. .*	One or more
n	Only *n* (where *n* > 1)
0. .*n*	Zero to *n* (where *n* > 1)
1. .*n*	One to *n* (where *n* > 1)

Modeling Inheritance



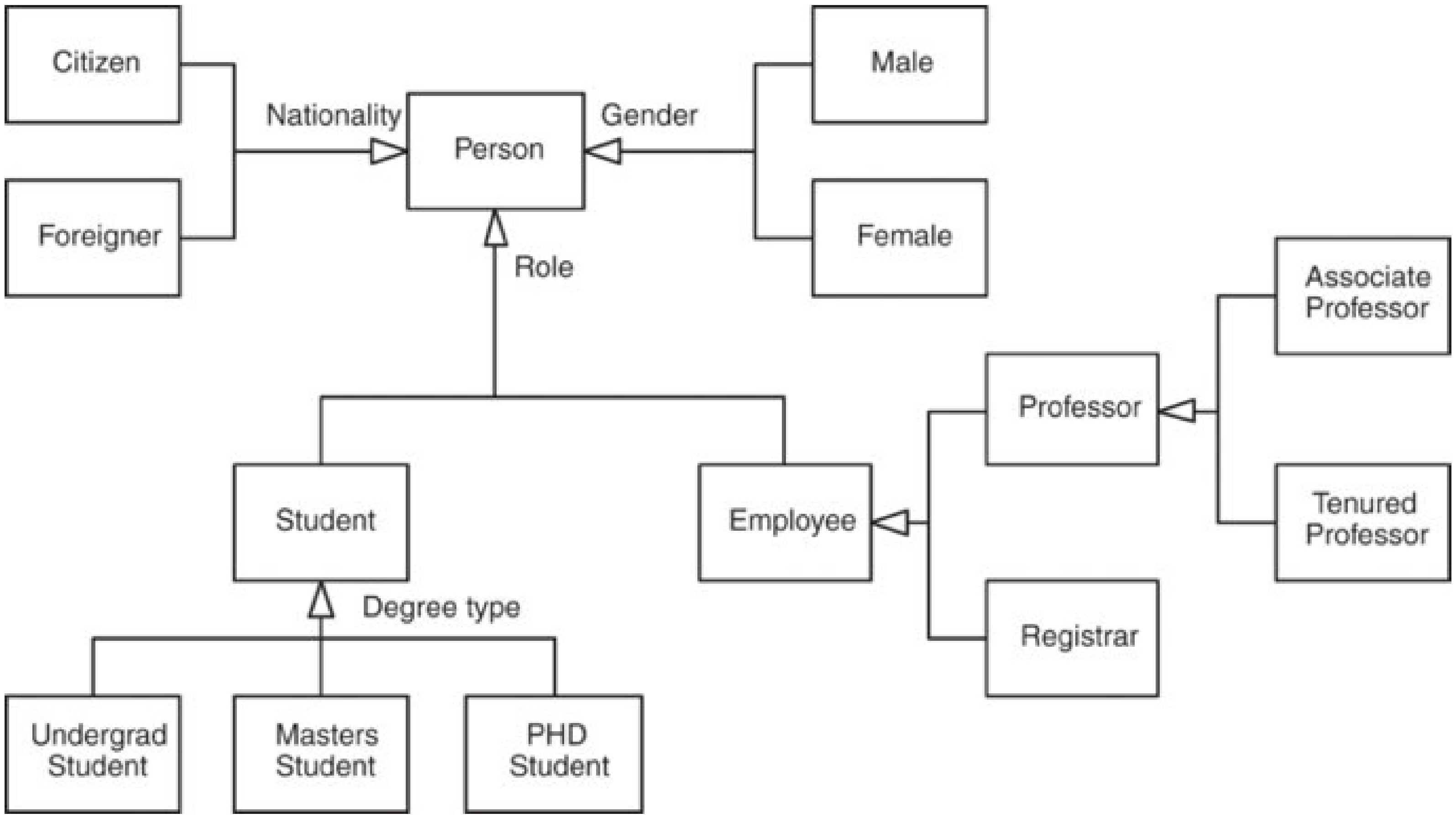
Modeling Composition



Modeling Vocabularies

- A vocabulary defines the semantics of entity types and their responsibilities, the taxonomical relationships between entity types, and the ontological relationships between entity types.
- Semantics is simply a fancy word for meaning— when we are defining the semantics of something we are defining its meaning.
- Taxonomies are classifications of entity types into hierarchies.
- Ontology goes beyond taxonomy. Where taxonomy addresses classification hierarchies ontology will represent and communicate knowledge about a topic as well as about a set of relationships and properties that hold for the entities included within that topic.

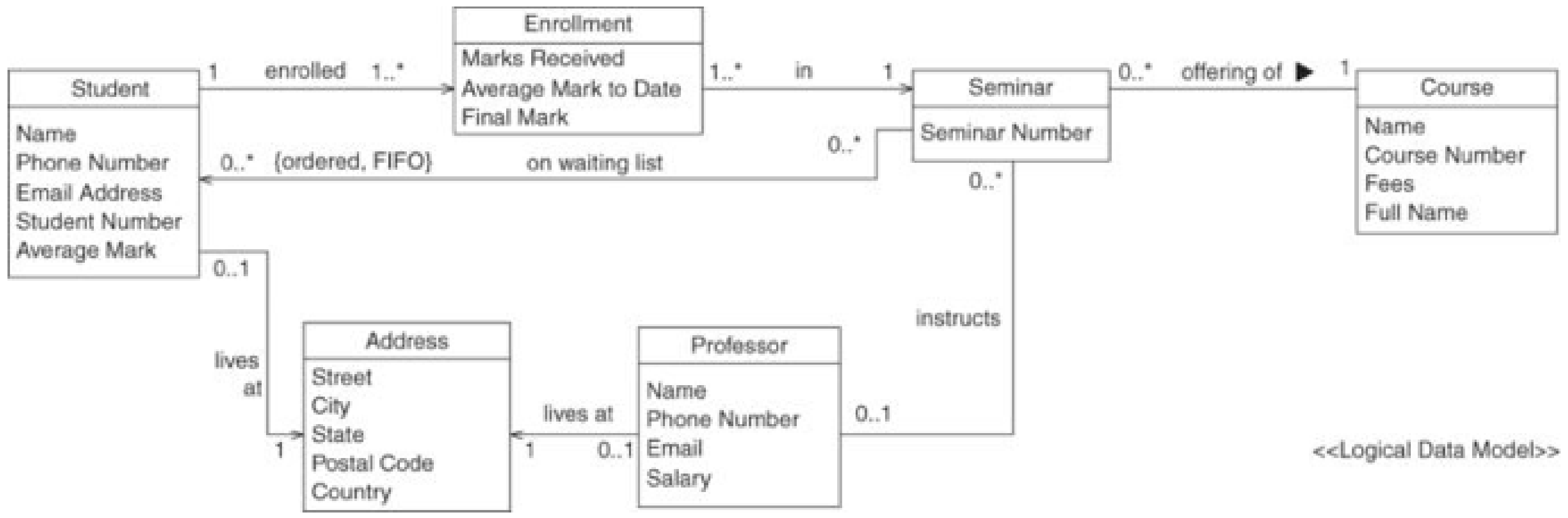
Vocabulary Example



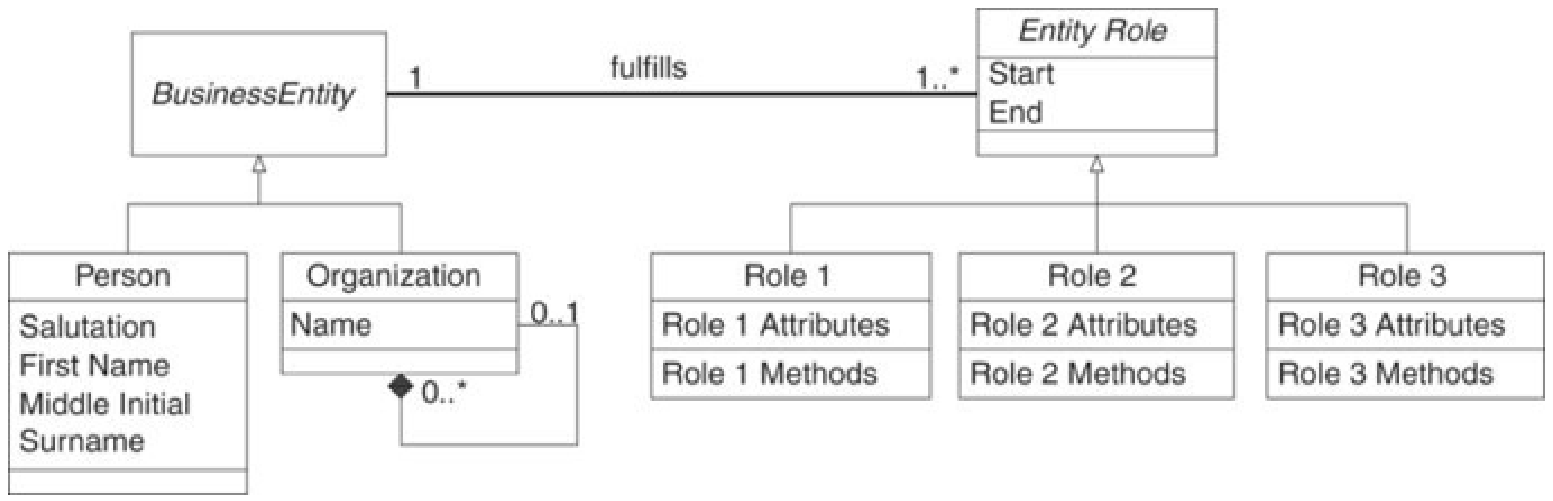
Logical Data Models (LDMs)

- Data modeling is the act of exploring data-oriented structures.
- Methods are not used in LDMs.

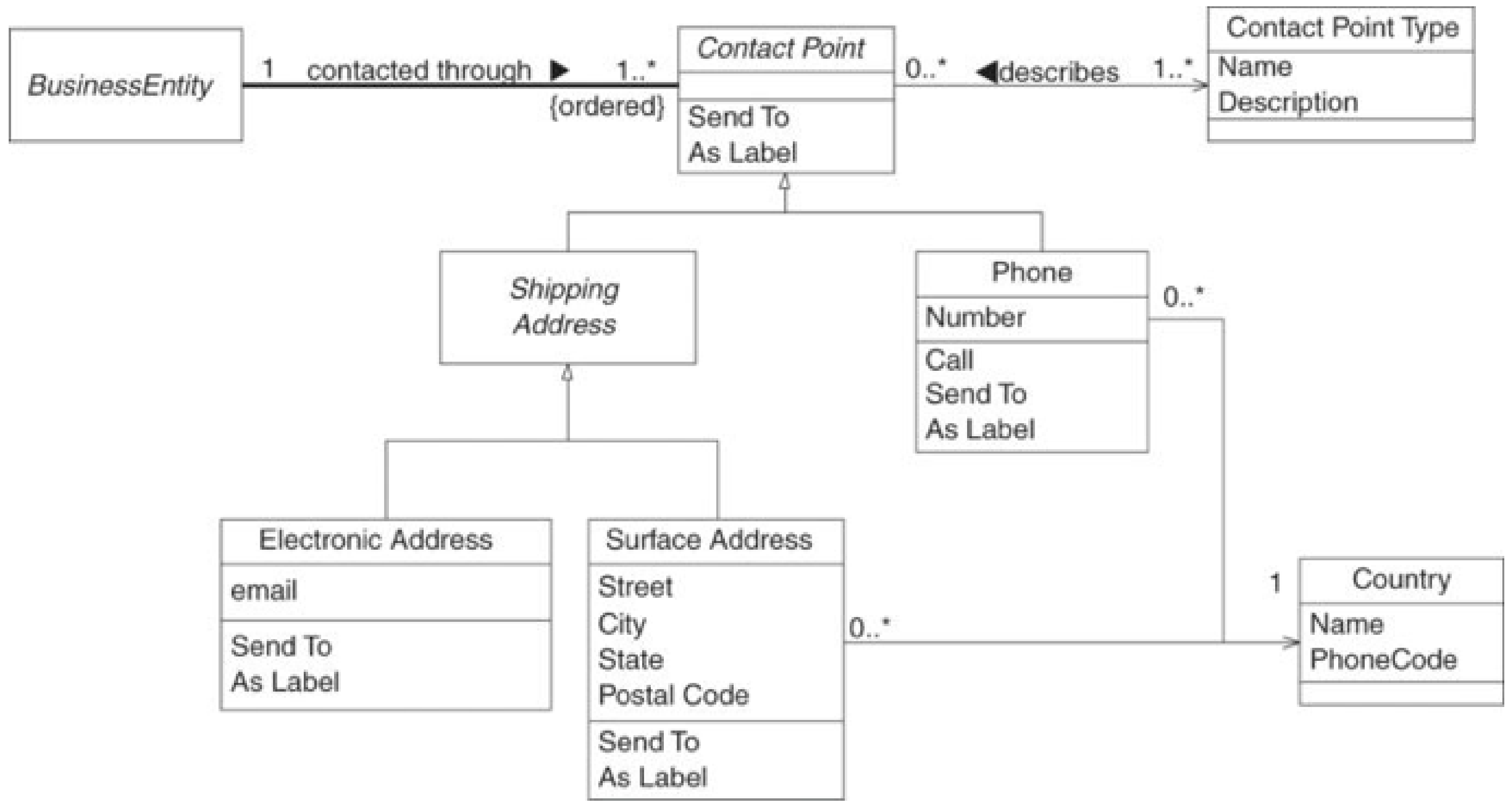
LDM Example


Student
Name
Phone Number
Email Address
Student Number
Average Mark
1
enrolled
1..*
Enrollment
Marks Received
Average Mark to Date
Final Mark
1..*
in
1
Seminar
Seminar Number
0..*
offering of ▶
1
Course
Name
Course Number
Fees
Full Name
0..*
{ordered, FIFO}
on waiting list
0..*
0..*
instructs
0..1
lives at
1
Address
Street
City
State
Postal Code
Country
lives at
1
0..1
Professor
Name
Phone Number
Email
Salary
0..1
<<Logical Data Model>>

Business Entity Analysis Pattern



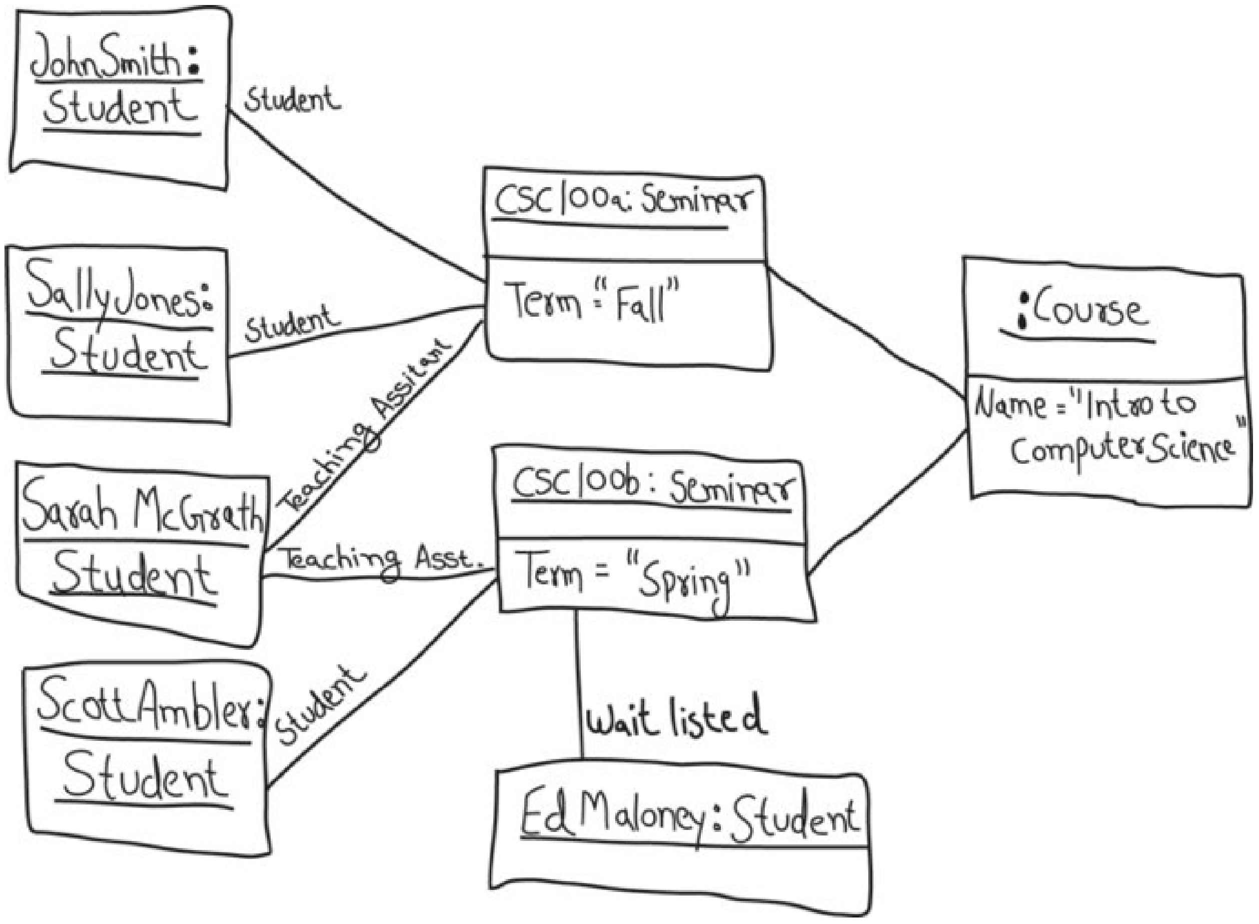
Contact Point Analysis Pattern



UML Object Diagram

- UML object diagrams, sometimes referred to as instance diagrams, are useful for exploring "real-world" examples of objects and the relationships between them.
- Although UML class diagrams are very good at describing this information some people find them too abstract— a UML object diagram can be a good option for explaining complex relationships between classes.

UML Object Diagram Example



Keeping Conceptual Domain Modeling Agile

- You do not need a complete model.
- Use the right model for the job.
- Focus on the problem space.
- Focus on fulfilling the requirements first.
- Use meaningful names.
- Conceptual model in parallel with other activities.
- Apply analysis patterns gently.