UML and ER

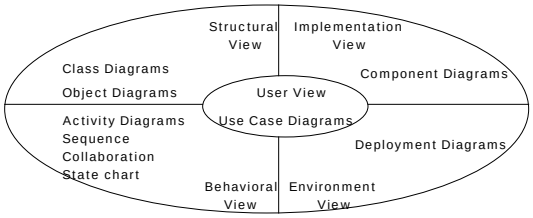
Roadmap

- UML Class Structure diagrams vs EER diagrams
  - Equivalences
  - Differences
  - Classification and Generalisation
- UML's role in database development
  - Database classes and Application classes
  - Interfaces to databases
  - Diagrams other than Class Diagram
  - Object Constraint Language

UML Notation (1)

- UML defines nine types of diagrams
- Concepts are depicted as symbols and relationships among concepts are depicted as paths (lines)
- 7 types of Diagram:
  - Use Case
  - Activity
  - Class <--- Relate to EER Static Structural Model
  - Object
  - State chart
  - Sequence
  - Collaboration

Architectural views organise knowledge



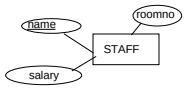
Perspectives on Class Diagrams

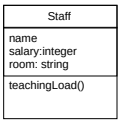
- Conceptual
  - the diagram represents the concepts in the domain
  - the concepts relate to classes that implement them but there may not be a direct mapping
  - no regard for software
- Specification
  - the interfaces of the software.
  - abstract data types rather than classes
- Implementation
  - the implementation of the interfaces

Concepts in the domain will be reflected in the database and the application

Entity Types ≡ Classes

- Class diagrams describe the static structure of a system, or how it is declared rather than how it behaves.
- Classes are represented by boxes, usually divided into three:
  - name of class
  - attribute and its information
  - operations of the class

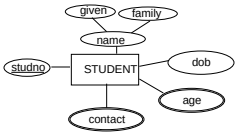


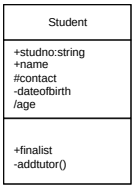


{name unique}

Attributes

- Every entity type must have a *key* attribute or set of attributes
- *Composite or Atomic* *Derived*
- *Single-valued or Multi-valued* *Null valued*



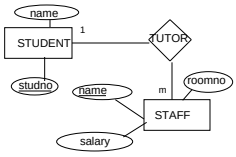


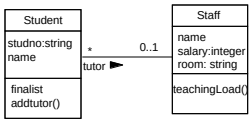
Relationships between objects

- Generalisation: one object is a more generalised type of another object
- Attribution: one object is a property of another
- Association: one object is associated with another object
- Classification: grouping objects together
- Aggregation: one object is part of another
- Collections: one object is made up of a collection of other objects

Relationship Types ≡ Associations

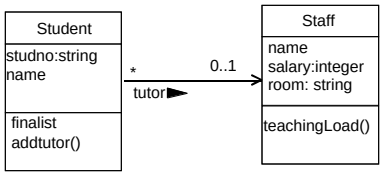
- Paths between boxes are ***relationships***
- ***Relationships*** may be of various kinds, be labelled, and have numbers (***multiplicity***) describing the number of occurrences.





Navigability on Relationships

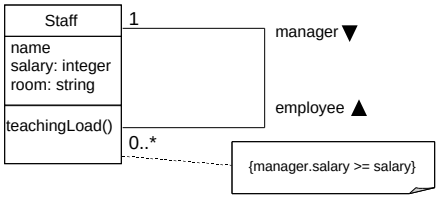
- uni-directional associations
- more useful in specification and implementation models



Constraints

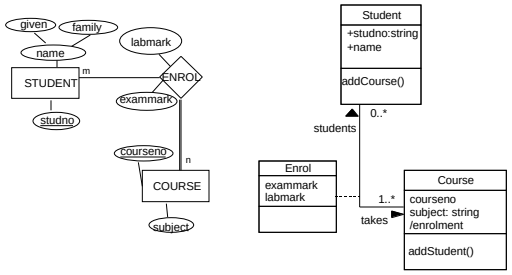
Constraints on Entity Types and Relationship Types

- may be added to these diagrams
- arrows (for direction)
- notes (for additional explanation)

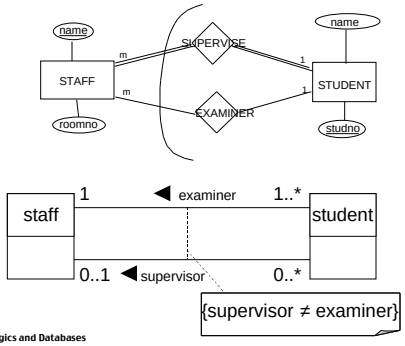


Association Class

- Adding attributes, operations and other features to relationships



Association Constraints



Aggregation

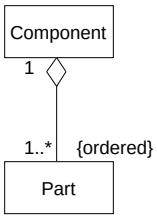
- Not catered for in EER modelling.
- Aggregation is an abstraction concept for building composite entities from their components
- 1. aggregate attribute values to form a whole entity
- 2. combining entities that are related by an association relationship into higher-level aggregate entity

- IS-A-PART-OF
- IS-A-COMPONENT-OF

- is a team an aggregation of its members?

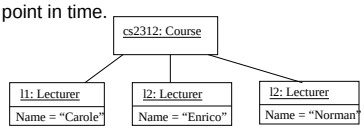
Aggregation

- Component contains an ordered collection of Parts
- The Part may be changed as the Component is updated



Object diagrams

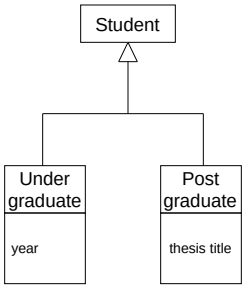
- An object diagram is the same as a class diagram but addresses the view of the system from the perspective of real or prototypical cases.
- They are concrete forms of the abstractions, showing a set of objects and the relationships of these objects at a point in time.



- Like Scenarios in Sequence Diagrams...

Generalisation

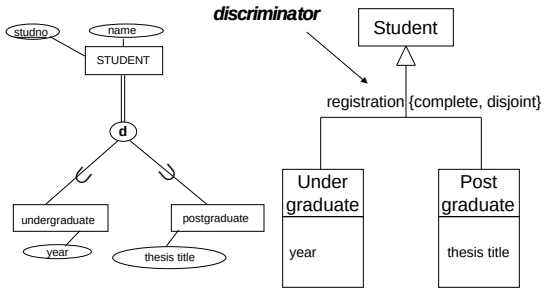
- A relationship between classes/types
- A subclass is a specialised type of superclass
- Attribute inheritance



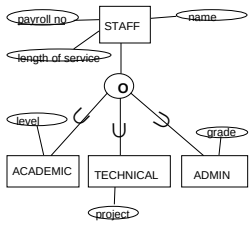
Classification

- The relationship between an object and its type
- A subclass entity type represents a subset or subgrouping of superclass entity type's instances
- User defined or Predicate/Condition defined
- Disjointness
  - *Overlap* the same object may be a member of *more than one* subclass of the specialisation
  - *Disjoint* the same object may be a member of *only one* subclass of the specialisation
- Completeness
  - *Total* every object in the superclass *must be* a member of some subclass in the specialisation
  - *Partial* an object in the superclass need not be a member of any subclass in the specialisation

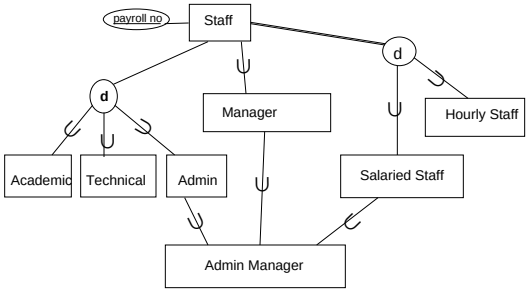
Classification



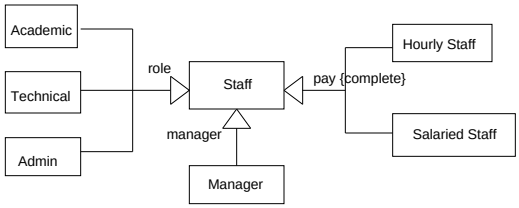
Single Classification



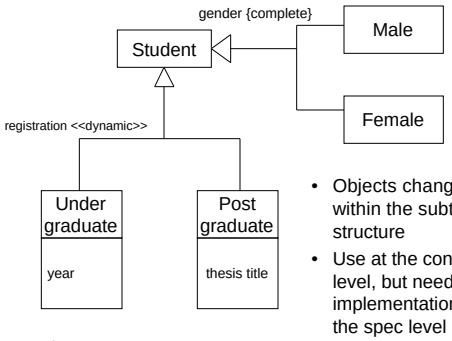
Multiple Classification



Multiple Classification



Dynamic Classification



- Objects change type within the subtyping structure
- Use at the conceptual level, but need implementation tricks at the spec level