中国科学技术大学软件学院
SCHOOL OF SOFTWARE ENGINEERING OF USTC

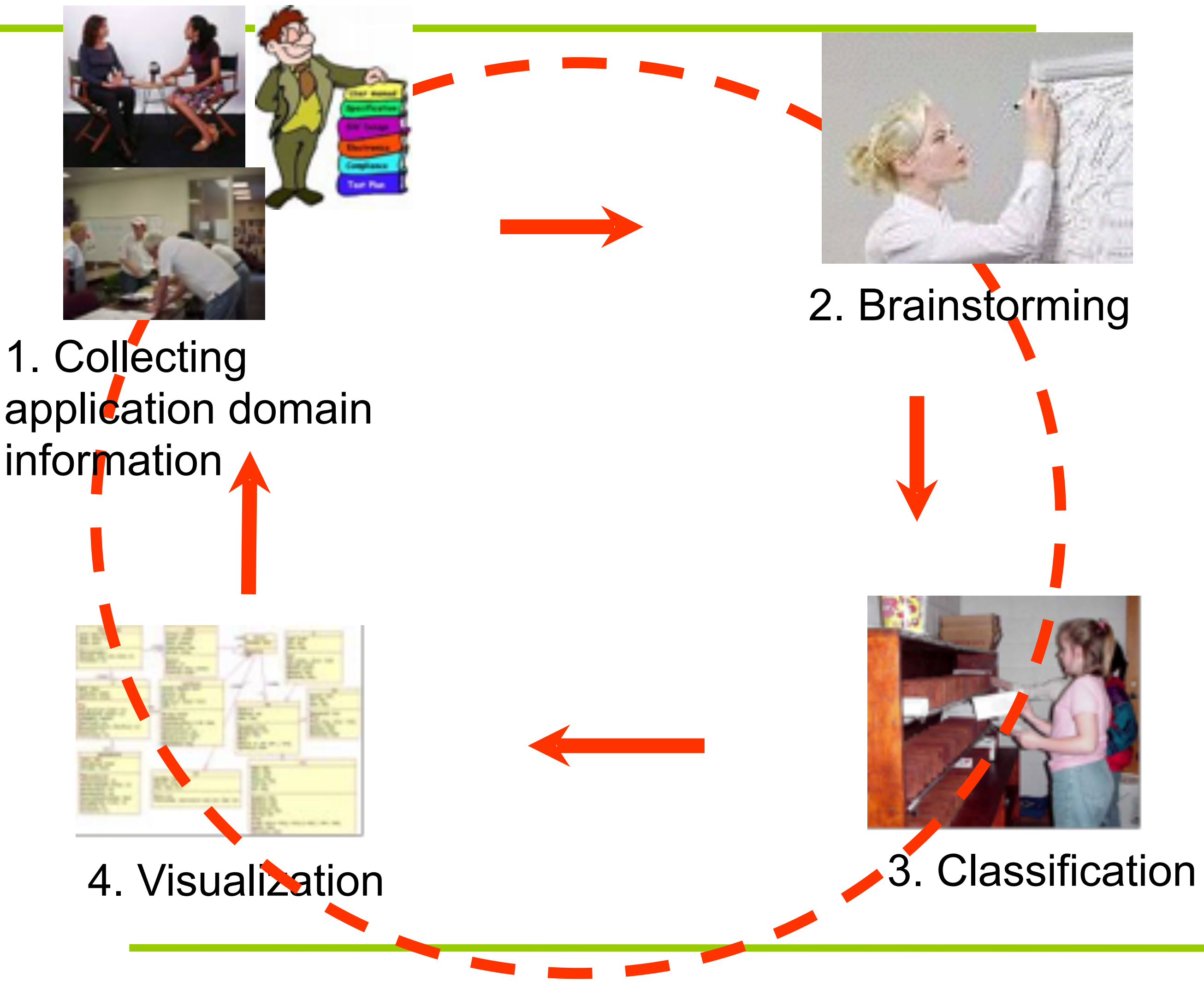
Domain Modeling

中国科学技术大学软件学院
孟宁

Domain Modeling

- What: A process performed by the development teams to acquire domain knowledge.
- Why:
  - Because software engineers need to work in different domains or different projects. They need domain knowledge to develop the system.
  - Software engineers come from different background. This may affect their perception of the application domain.
- How:
  - Collect domain information, perform team brainstorming and classification, and visualize the domain knowledge using UML class diagram
  - Detail is given in the next slide

Steps for Domain Modeling



Object and Attribute

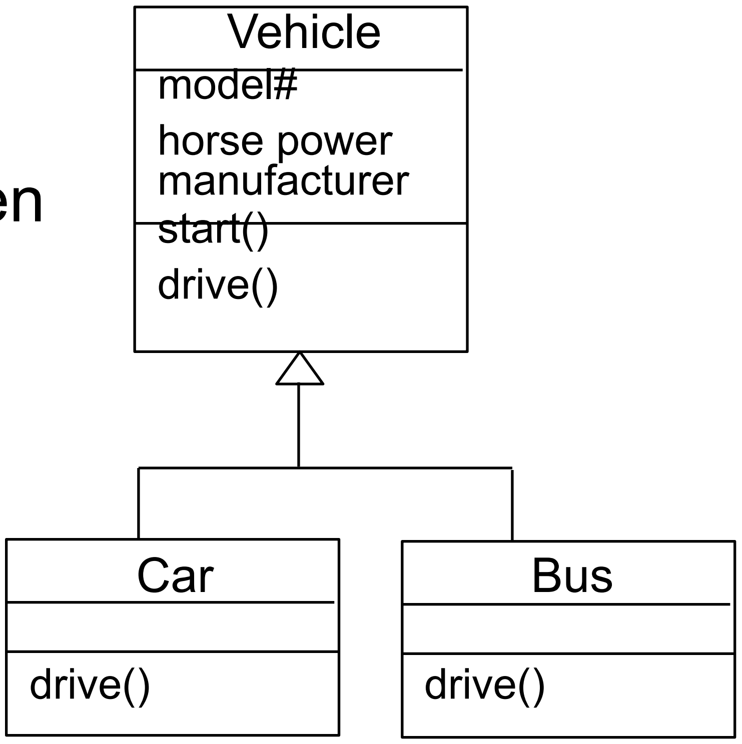
- A noun/noun phrase can be a class or an attribute, how do we distinguish?
- This is often a challenge for beginners.
- Rules to apply:
  - An object has an *"independent existence*" in the application/application domain, an attribute does not (have).
  - Example: "Number of seats", class or attribute?
  - Attribute, because "number of seats" cannot exist without referring to a car, airplane, or classroom as in "number of seats of a car"， "number of seats of a classroom"

Object and Attribute

- Rules to apply:
  - attributes describe objects or store state information of objects
  - objects must be created by invoking a constructor (explicitly or implicitly)

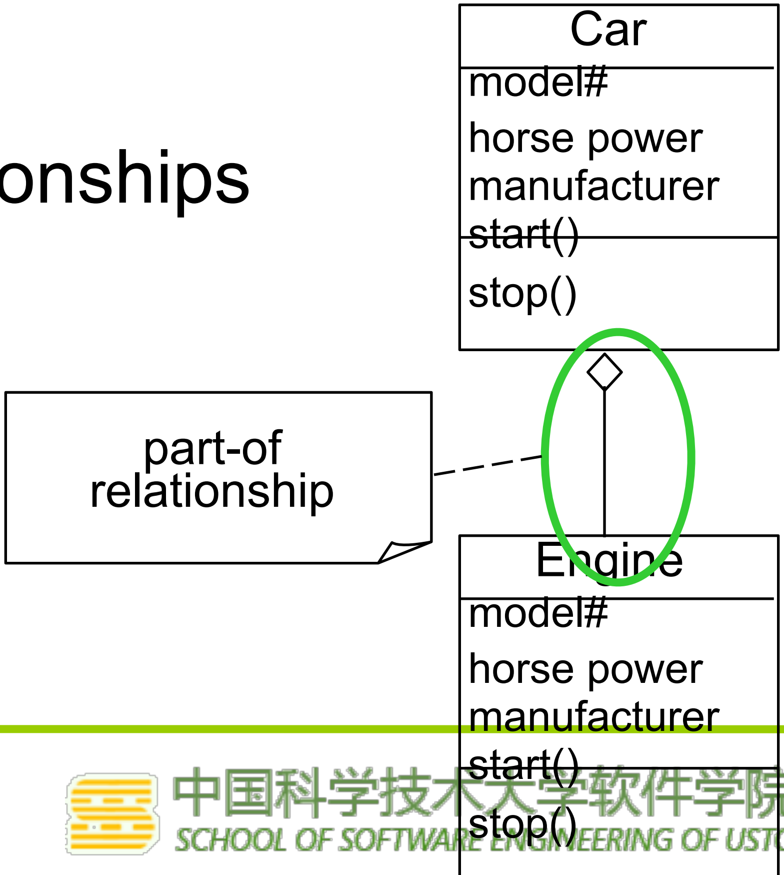
Inheritance Relationship

- Inheritance relationships
  - express the generalization/ specialization relations between concepts
  - one concept is more general/ specialized than the other
  - example: vehicle is a generalization of car, car is a specialization of vehicle
  - also called IS-A relation



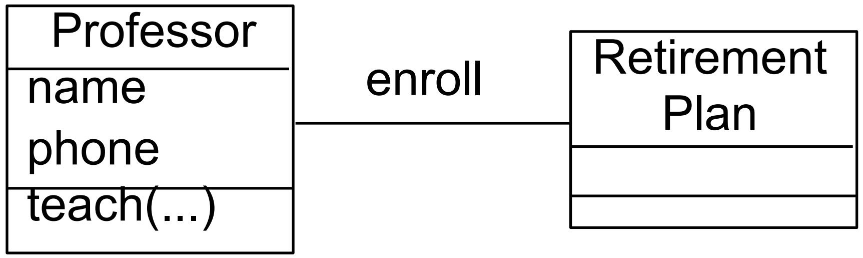
Aggregation Relationship

- Aggregation relationships
  - express the fact that one object is part of another object
  - engine is part of a car
  - also called part-of relationships



Association Relationship

- Association relationships
  - expressing general relationships other than inheritance and aggregation
  - these can be application specific relationships between two concepts
  - example: "instructor teach course", "user has account"



Neither inheritance nor aggregation can apply.

Steps for Domain Modeling

- 1) Collect application domain information
  - focus on the functional requirements
  - also consider other requirements and documents
- 2) Brainstorming
  - listing important application domain concepts
  - listing their properties/attributes
  - listing their relationships to each other
- 3) Classifying the domain concepts into:
  - classes
  - attributes / attribute values
  - relationships
    - association, inheritance, aggregation
- 4) Document result using UML class diagram

Brainstorming: Rules to Apply

- The team members get together to identify & list
  - nouns / noun phrases
  - "X of Y" expressions (e.g., color of car)
  - transitive verbs
  - adjectives
  - numeric
  - possession expressions (has/have, possess, etc.)
  - "constituents / part of" expressions
  - containment / containing expressions
  - "X is a Y" expressions

Example

Functional requirement:
[PFR1] The web based application must provide a search capability for overseas exchange study programs using a variety of search criteria.

adjective, but not domain specific

nouns but not domain specific

domain specific transitive verb

domain specific noun/noun phrase

Brainstorming result:

- nouns/noun phrases

programs
search criteria

- transitive verbs

search for

Classifying Brainstorming Result

♦ nouns/noun phrases	⇒	class or attributes
♦ "X of Y" expressions	⇒	X is an attribute of Y
	⇒	X is a role in an association
♦ transitive verbs	⇒	association relationships
♦ adjectives		te values
♦ numeric		attribute / multiplicity values
♦ possession expressions (has/have, possess, etc.)	⇒	aggregation relationships or attributes
♦ "consist of/part of" expression	⇒	aggregation relationships
♦ containment / containing expressions	⇒	association or aggregation relationships
♦ "X is a Y" expressions	⇒	inheritance

Objects have independent existence, attributes do not.

Example

Rule: noun/noun phrase ➔ class or attribute

Domain specific
nouns/noun phrases

transitive verbs

(c) program
(c) search criteria
(c) user

$$\begin{tikzcd}\text{user} \arrow[r, "\text{search for}"] & \text{program}\end{tikzcd}$$

search for (programs)

Rule: transitive verbs ⇒
association relationships

Because they can exist independently.

Example

A car has a make, model, horse power, number of seats ...

(c) car
(a) make
(a) model
(a) horse power
(a) number of seats

Car has independent existence.
Make, model, horse power, and
number of seats do not.

A customer can rent one or more cars ...

(c) customer
(c) car
rent
1+

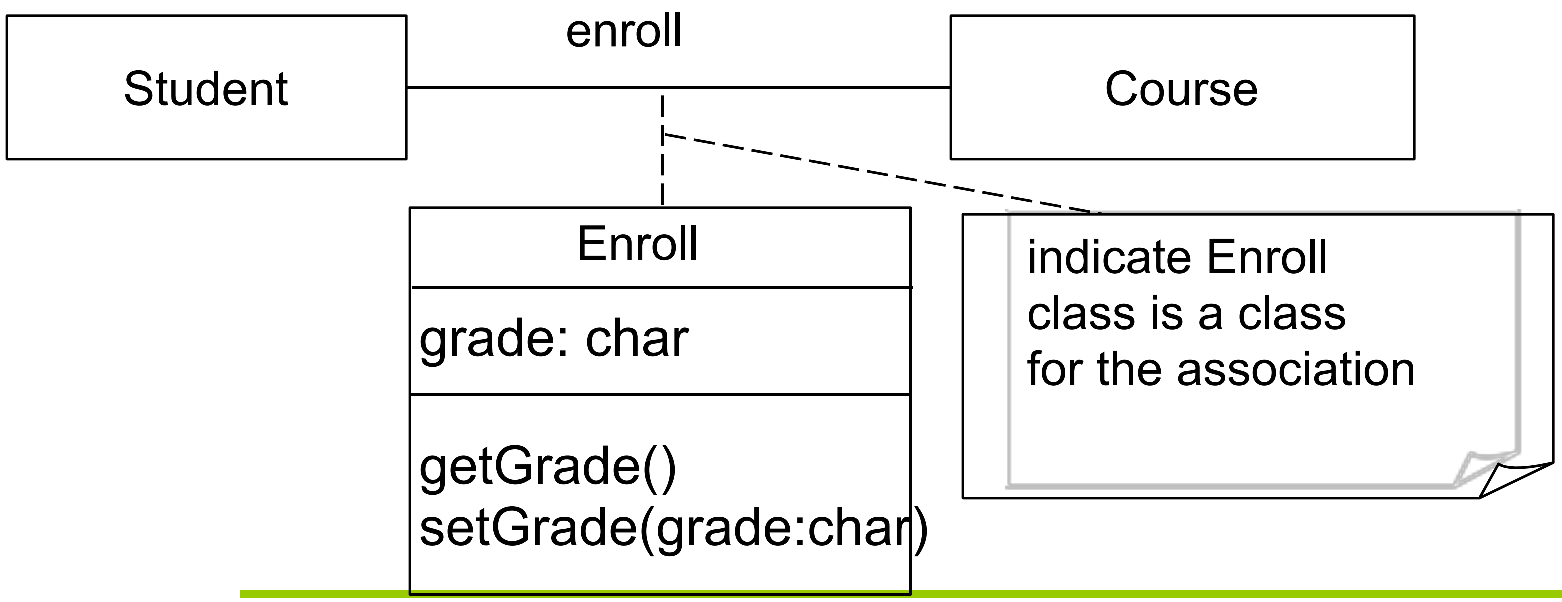
Class Exercise

- Do the following for your team project
- Identify the concepts that exist in the application domain.
- Classify the concepts in terms of
  - classes
  - attributes of classes
  - relationships between the classes
    - inheritance
    - aggregation and
    - association

Association Class

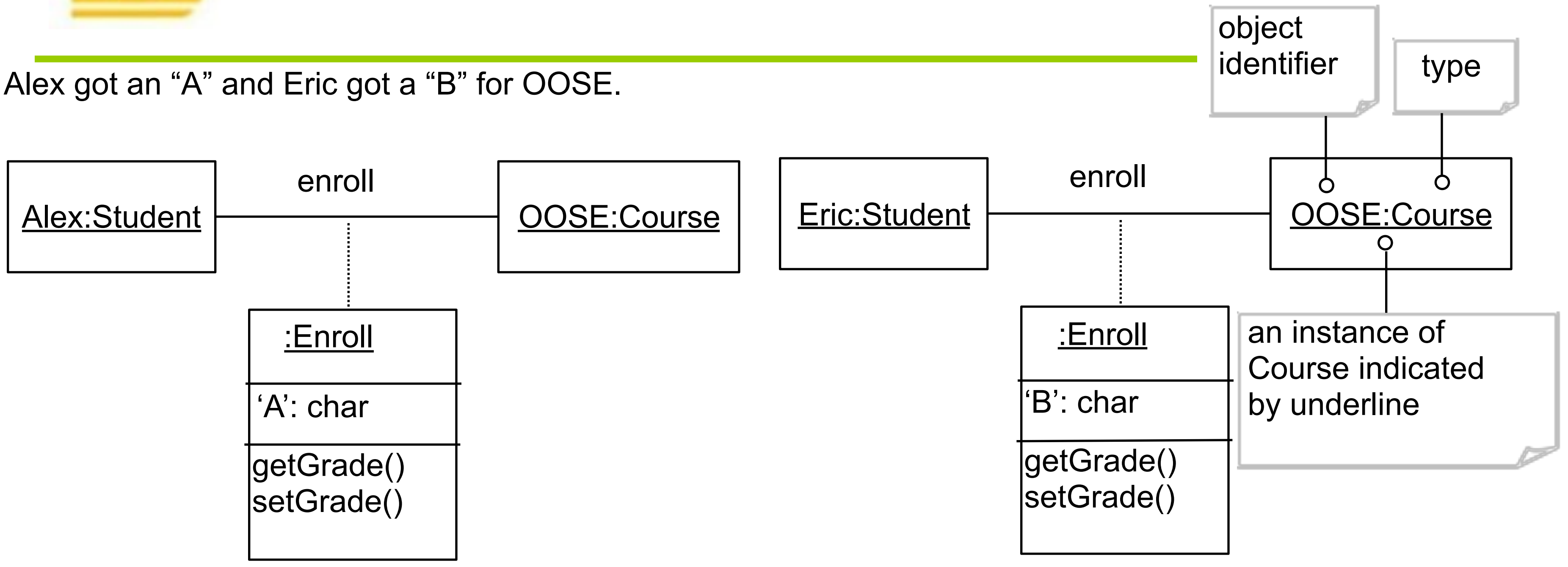
An association class defines properties and operations for an association between two classes.

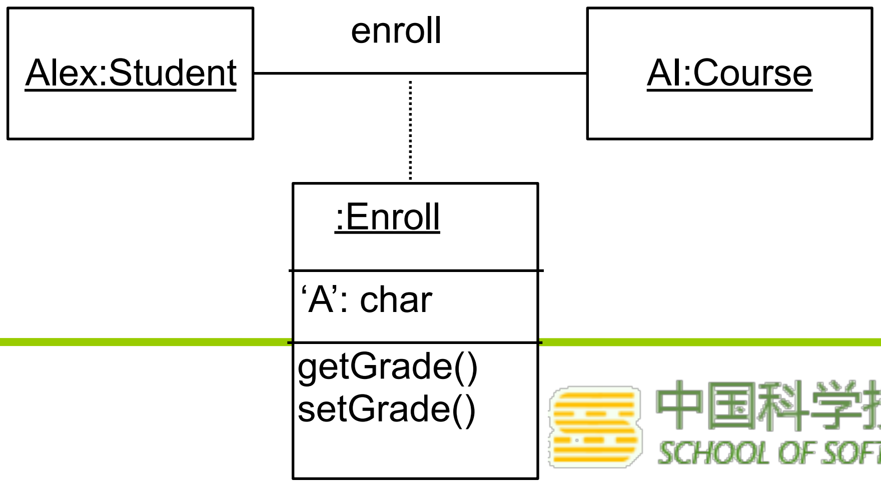
Students enroll in courses and receive grades.



Understand Association Class

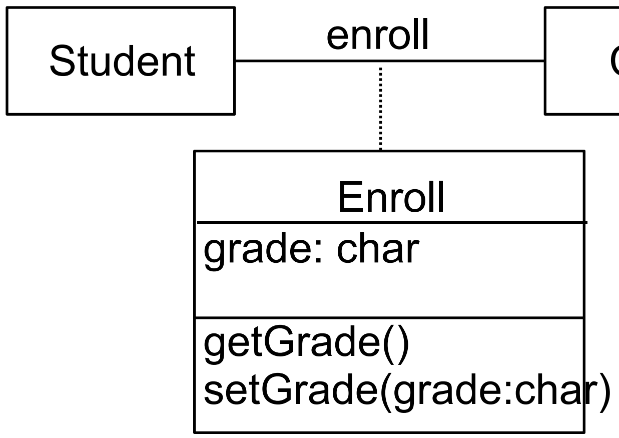
Alex got an “A” and Eric got a “B” for OOSE.





Alex also got an “A” for AI.

Understand Association Class

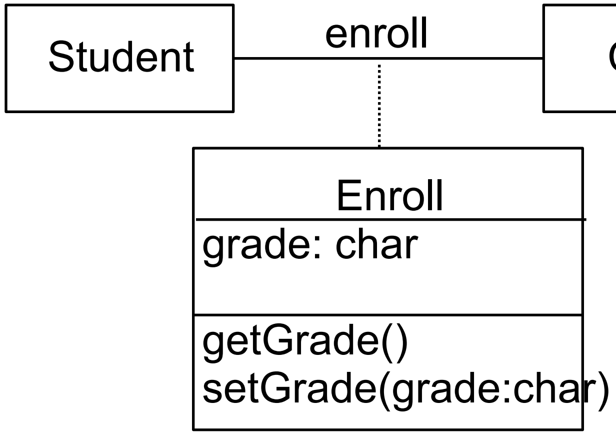


```
Student *alex=new Student( ... );
Course *oose=new Course ( ... );
...
Enroll *e=new Enroll(alex, oose);
e->setGrade('A');
```

Design Implementation

```
class Student { ... }
class Course {...}
class Enroll {
private:
   char grade;
   Student* student;
   Course* course;
public:
   Enroll (Student* s, Course* c);
   char getGrade();
   void setGrade(char grade);
}
Enroll::Enroll(Student* s, Course* c) {
   student=s; course=c;
}
```

Understand Association Class



OO Realm

Student

sid	name	phone	…	…
001	Alex	…	…	…
002	Eric	…	…	…

Course

cn	title	desc	…	…
c1	oose	…	…	…
c2	AI	…	…	…

Enroll

sid	cn	grade	…	…
001	c1	A	…	…
001	c2	A	…	…
002	c1	B	…	…

RDB Realm

Tip for Domain Modeling

Do not do brainstorming and drawing at the same time. The result could be very poor.

1) List the concepts, and then classify them on a whiteboard.

2) Take a picture(s) of the whiteboard using a digital camera.

3) Email the digital images to team members.

4) Have a member or two to convert the result to UML class diagram.

5) Email the UML class diagram to all members to review.

6) Modify the diagram to reflect corrections and comments.

谢谢大家！

References
Dr. David Kung University of Texas Arlington May 2010