面向对象软件方法学

张波

Textbook and Reference

- Mark Priestley, 2000, Practical object-oriented design with UML (面向对象设计的UML实践), Tsinghua University Press, Reprint. Originally published: McGraw-Hill, 2000. ISBN 7302040982.
- Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, 2002, Design Patterns: Elements of Reusable Object-Oriented Software, 机械工业出版社.
- 张波，Qt中的C++技术，2012年, 机械工业出版社

Outline of this lecture

Chapter 1 Introduction

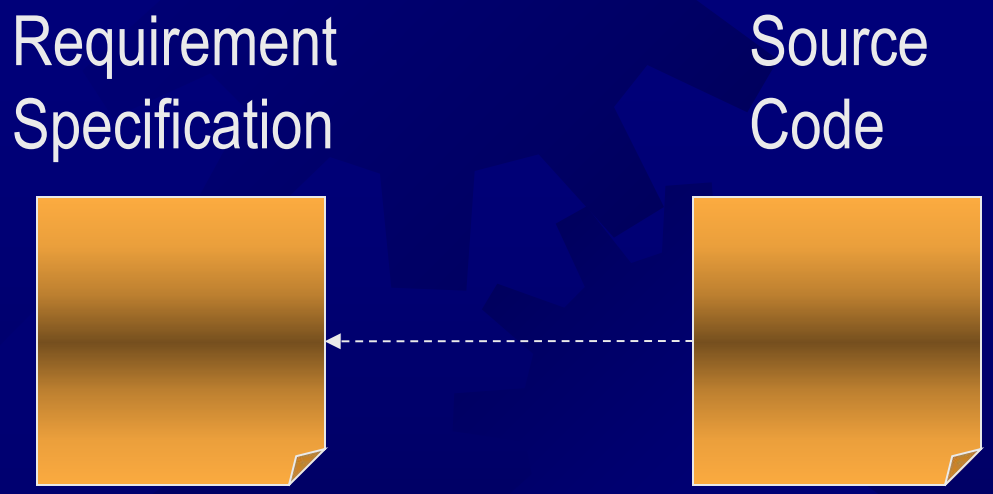
- Why do we need modeling?
- What kinds of modeling methods available?
- Why do we need UML?
- Fundamental elements of UML
  - Views
  - Diagrams

Chapter 2 Modeling with Objects

- Properties, Navigibility, Message Passing,
- Strength of OOP

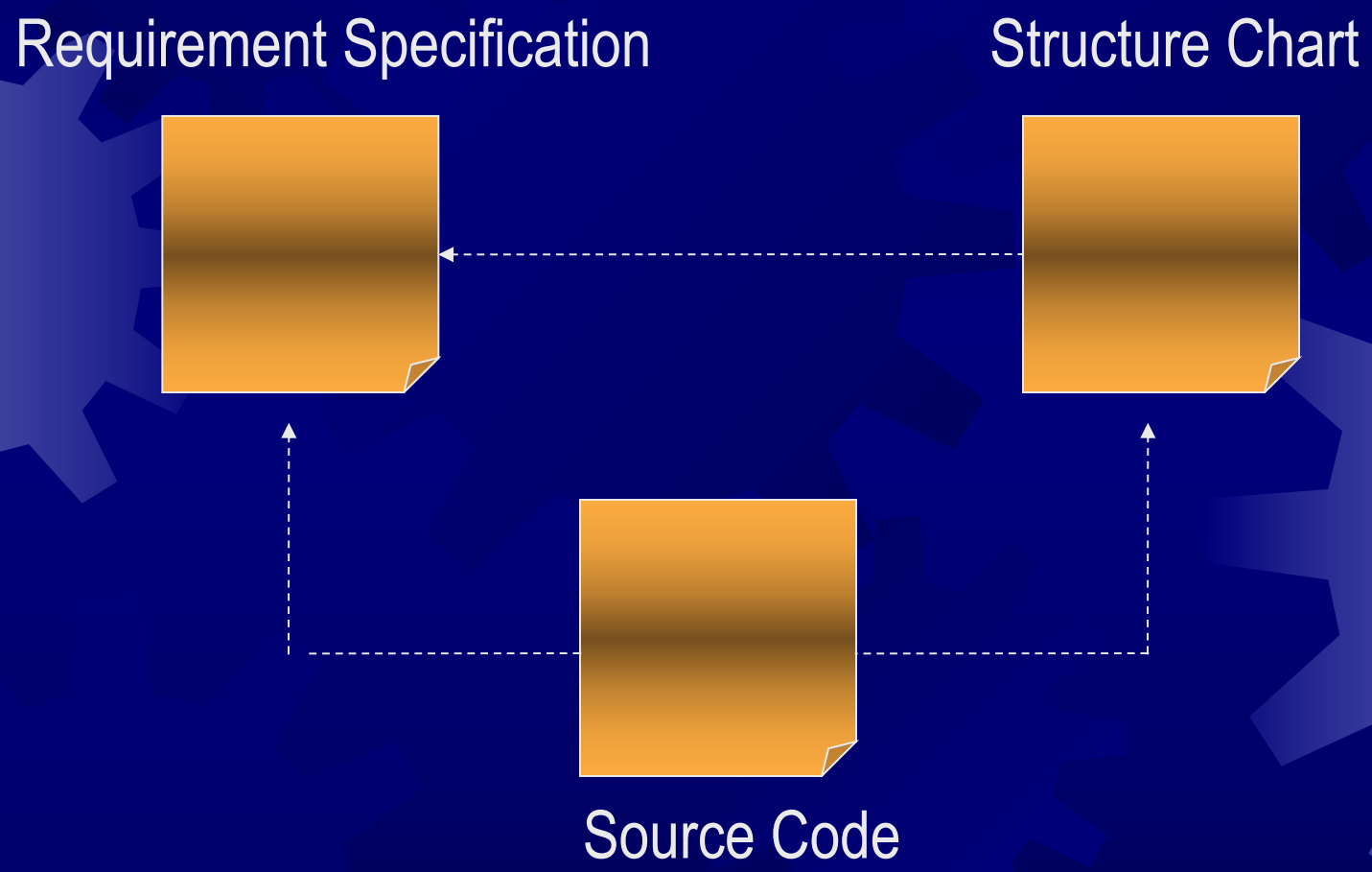
Introduction to UML

- Software development
  Wide sense / narrow sense
- The process of software development



4

A more complex process



5

Model

- Definition: the intermediate descriptions (documents) produced in the process.
- Features:
  - Abstract view of a system.
  - Easy to understand.
  - A valuable means for communication.

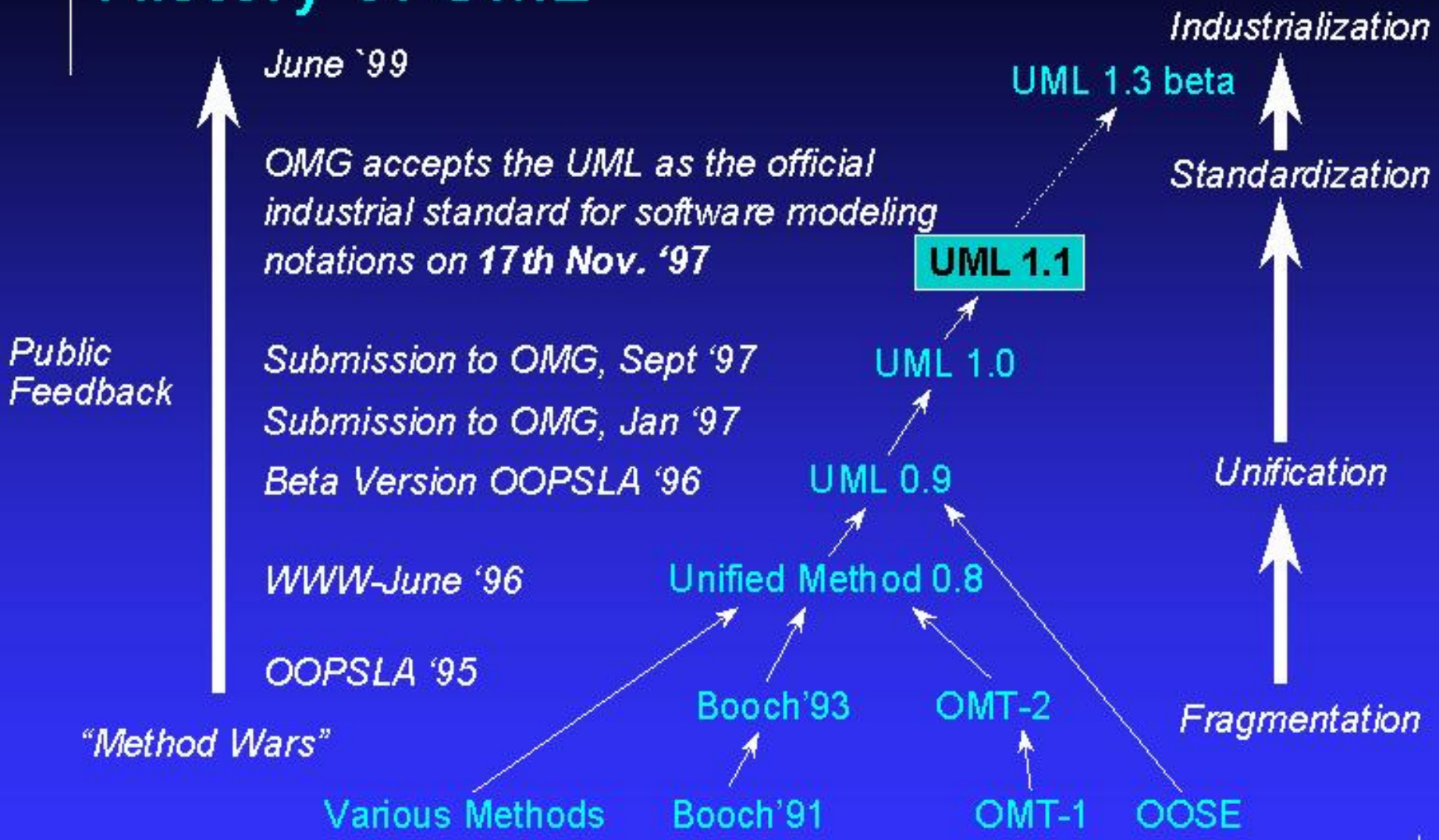
6

Classes of methodologies

- Structured methods
  - Models
    a collection of data +
    functions external to the data
  - Notations: data flow diagrams
- Object-oriented methods
  - Models: see the following example
  - Notations
    Unified Modeling Language (UML)

7

History of UML



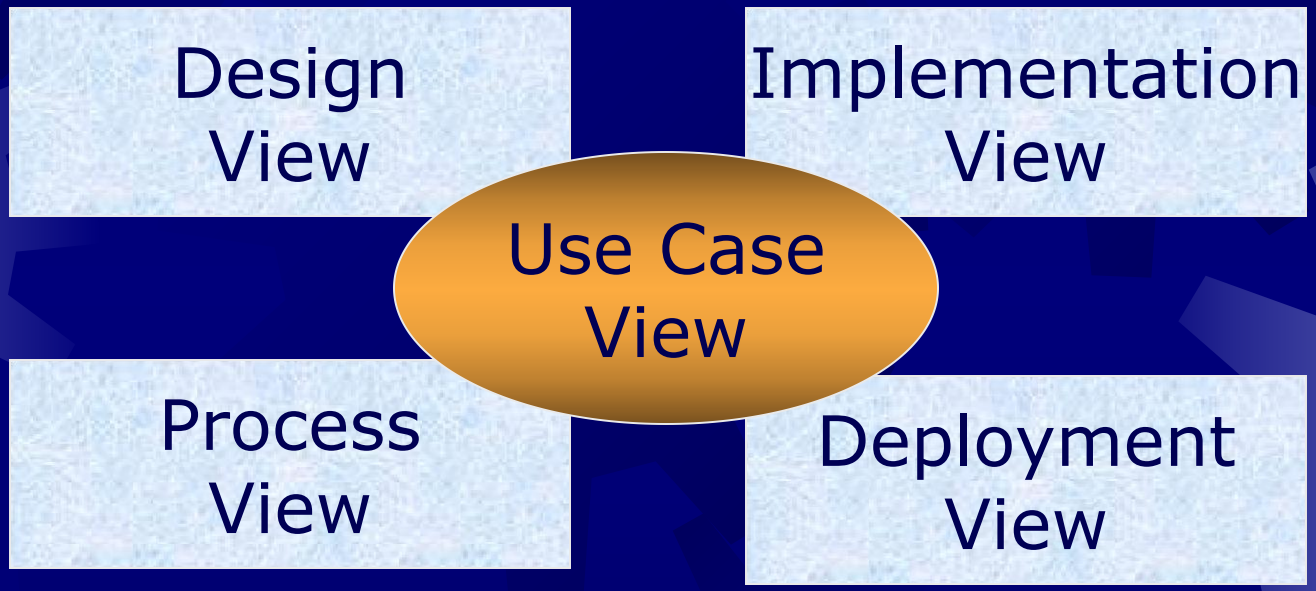
Taken from http://ias.uni-klu.ac.at/projects/uml/OTG99/sld006.htm

A brief Introduction to UML

- The dominant language in OO modeling
- It can be used with a wide range of software processes
- Views
- Models/Diagrams

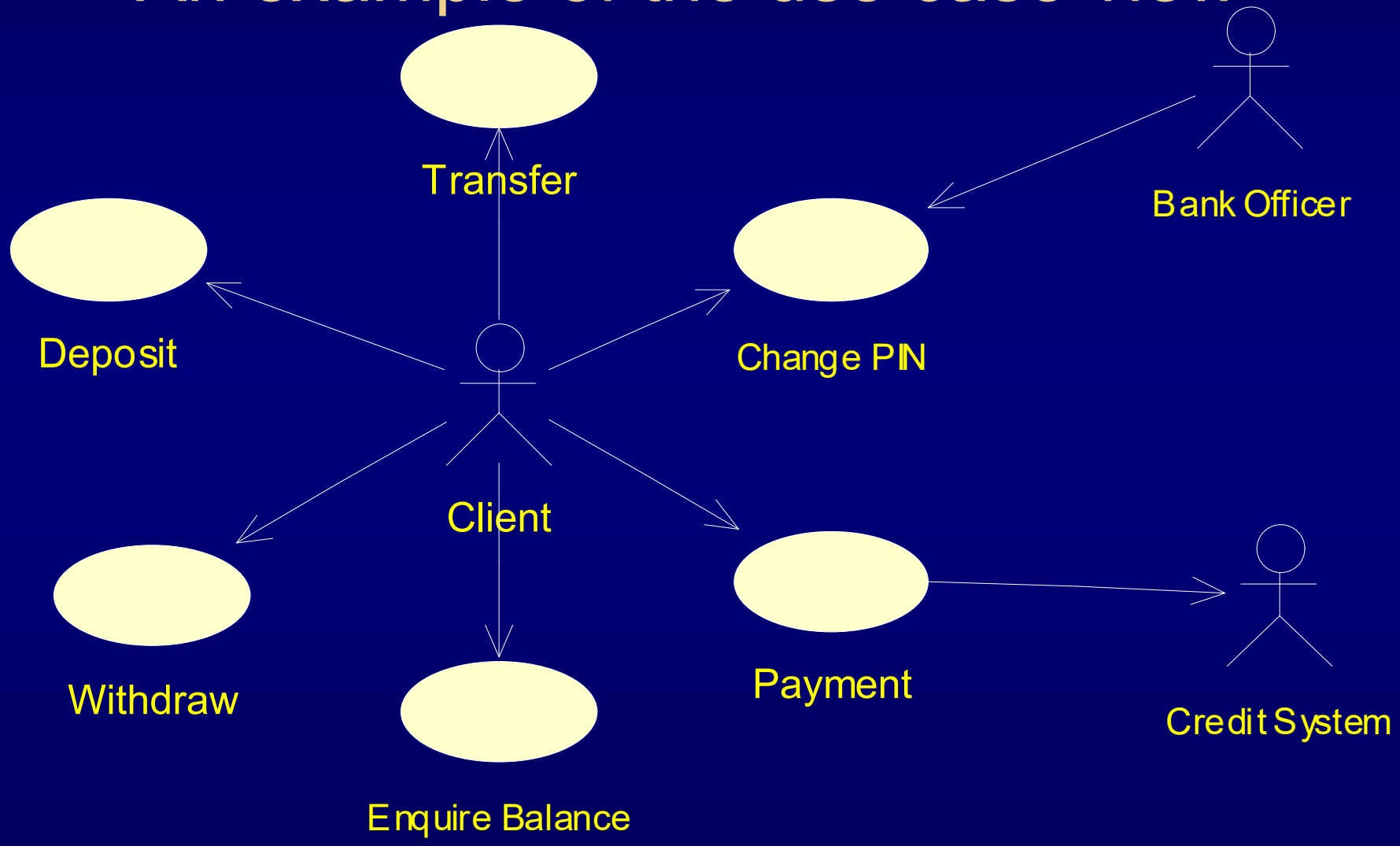
9

Views

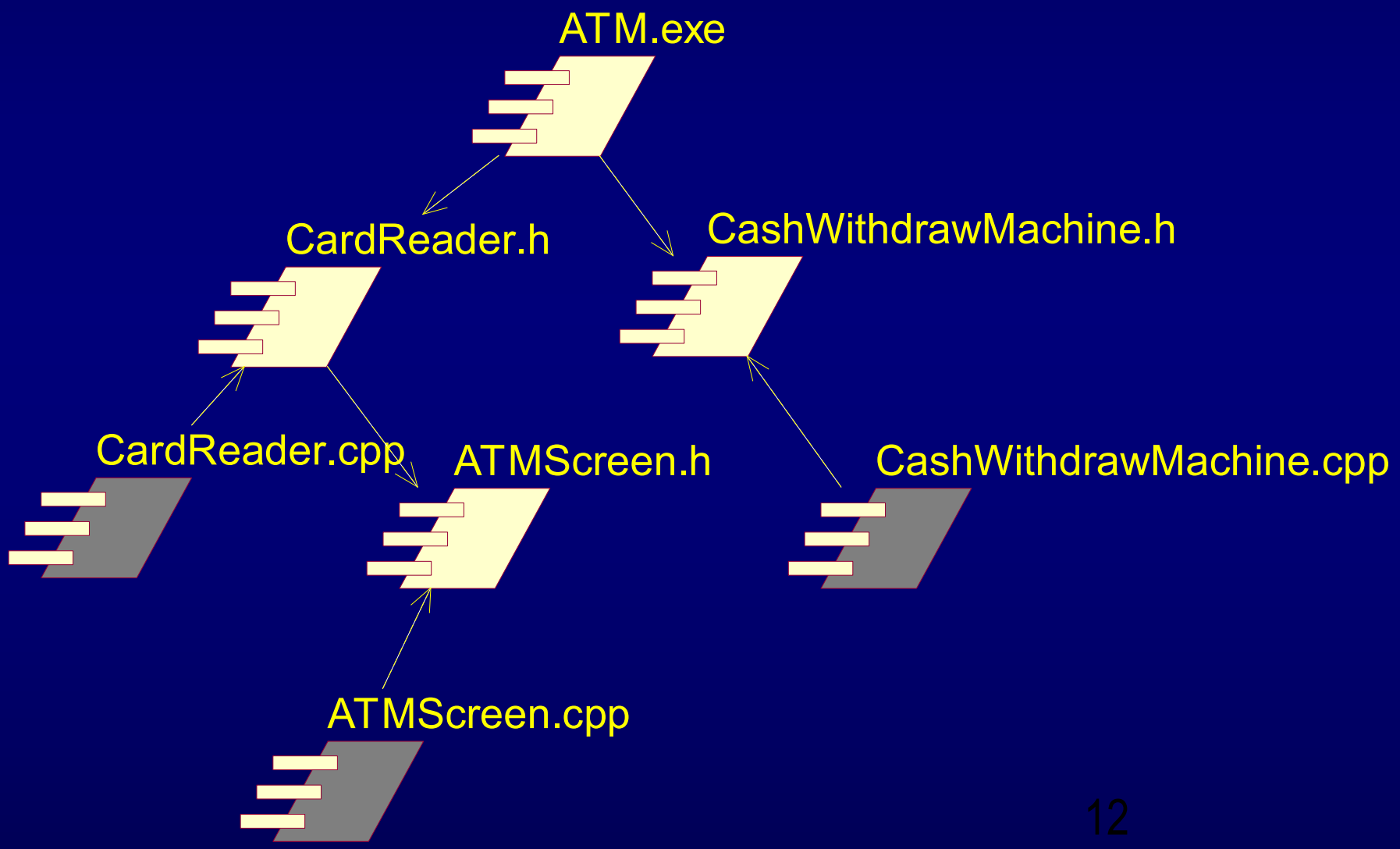

Design View
Implementation View
Use Case View
Process View
Deployment View

10

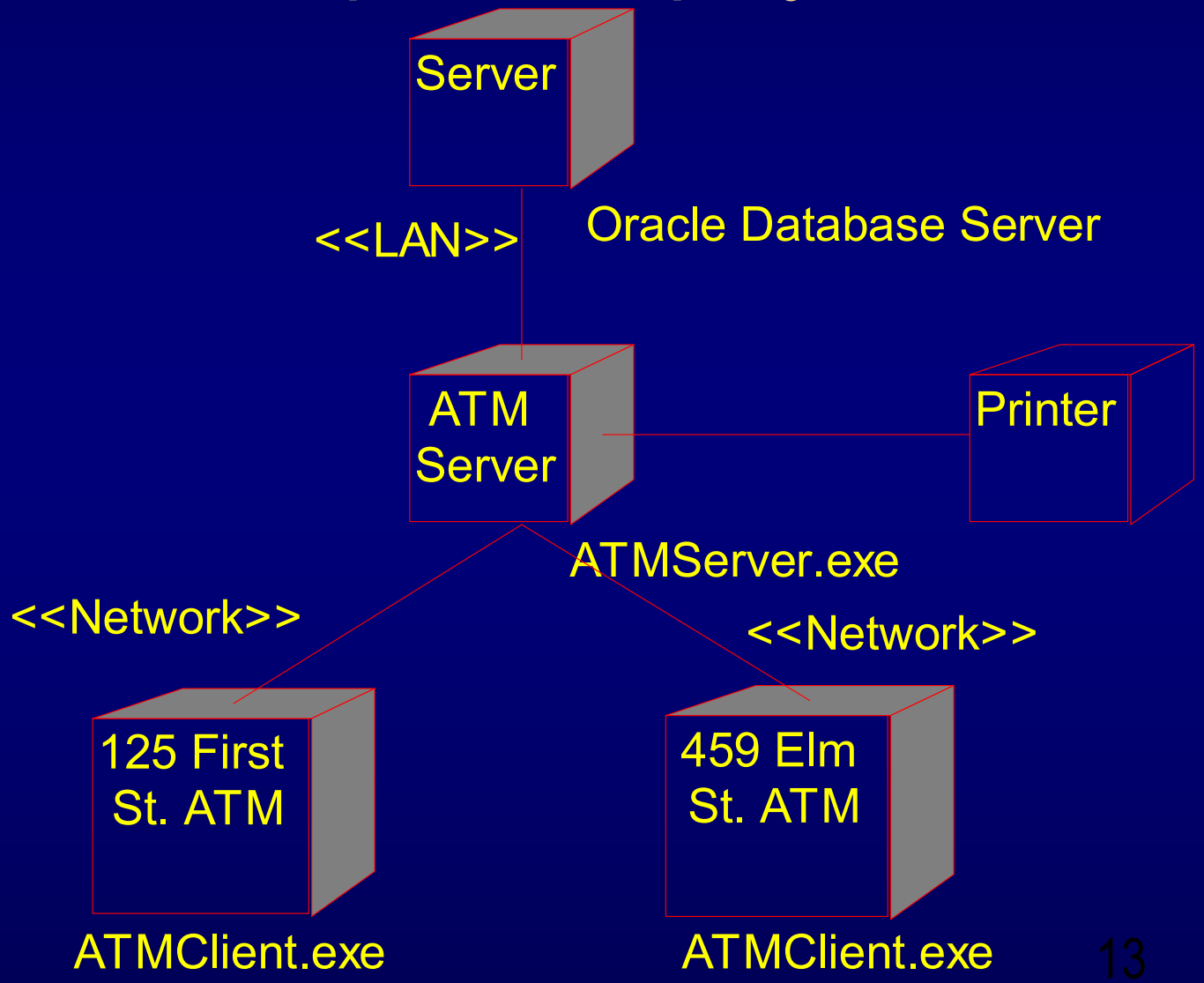
An example of the use case view



An example of the implementation view



An example of deployment view



Diagrams

- A diagram presents certain aspect of the underlying system model. Diagrams are just like programs, but more abstract (object structure).
- Do not be overwhelmed by the details of the diagrams.
- Some types of diagrams can be used in both use case and design views

14

UML's diagram types

Diagram	View
1 Use case diagram	Use case view
2 Object diagram	Use case & design view
3 Sequence diagram	Use case & design view
4 Collaboration diagram	Use case & design view
5 Class diagram	Design view
6 Statechart diagram	Design view
7 Activity diagram	Design view
8 Component diagram	Implementation view
9 Deployment diagram	Deployment view

15

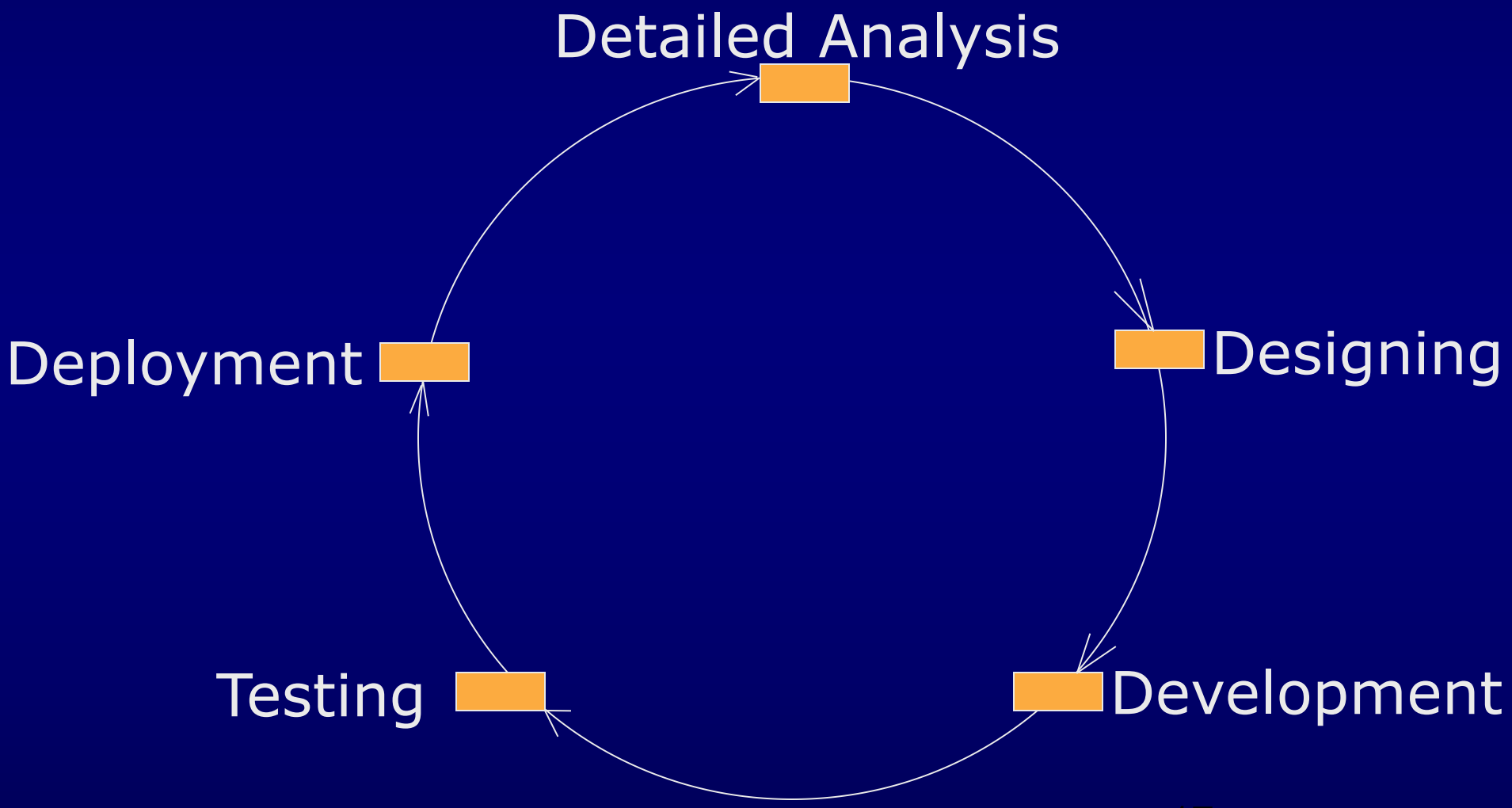
The software development process

- Linear or waterfall model

Analysis → Designing → Development → Testing → Deployment

16

Iterative model



Chapter 2: Modeling with Objects

- The object model
  - Data + Operations
  - Execution of a program: a dynamic network of intercommunicating objects.
    Nodes (object) + links (sending messages)
  - The semantic foundation for UML's design models
- The stock control example

18

Objects

- Design: how to split up a system's data and overall functionality.
- A frequently used rule: real-world objects

myScrew:Part
name="screw" number=28834 cost=0.02

19

Object Properties

- State: the aggregate of the data values contained in an object's attributes
- Behavior

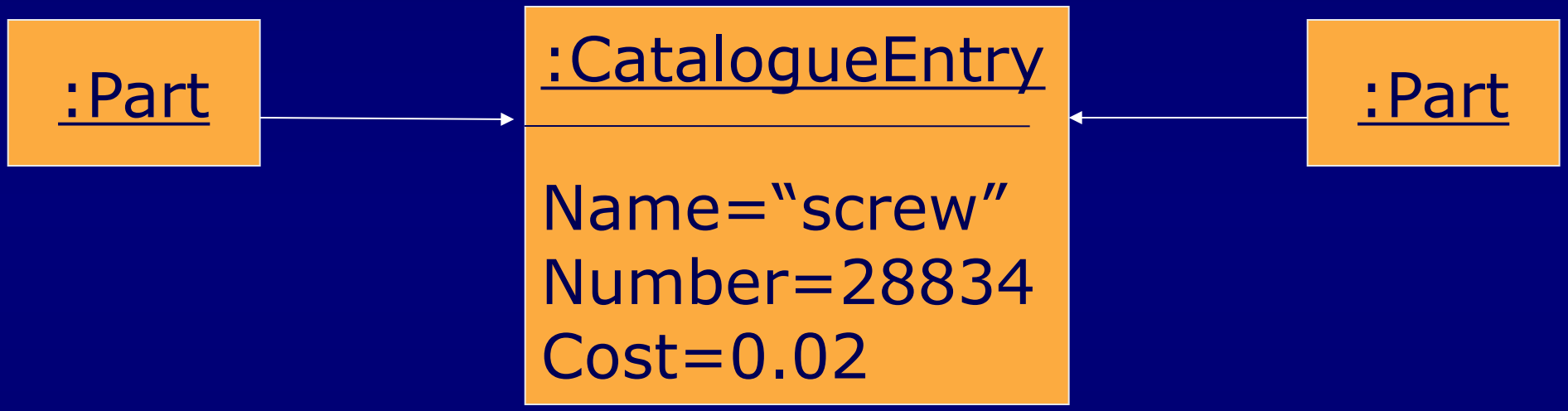
  shown in the class diagram
- Identity: address in memory
- Object names: a convenient alias for its identity.
  - Sometimes these corresponds to object names in program source, but not always, such as objects in a vector.

20

Avoiding data replication

- Data in the previous object model is replicated
  - Waste storage
  - Difficult to consistently update all objects
- Links

21



Information: attributes & links

```
#include <iostream>
using namespace std;
class CatalogueEntry {
public:
   CatalogueEntry(string p_name, long
   p_number, double p_cost){
        name   = p_name  ;
        number = p_number;
        cost   = p_cost  ;
   }
   string getName()   { return name;};
   long   getNumber() { return number; };
   double getCost()   { return cost;};
```


23

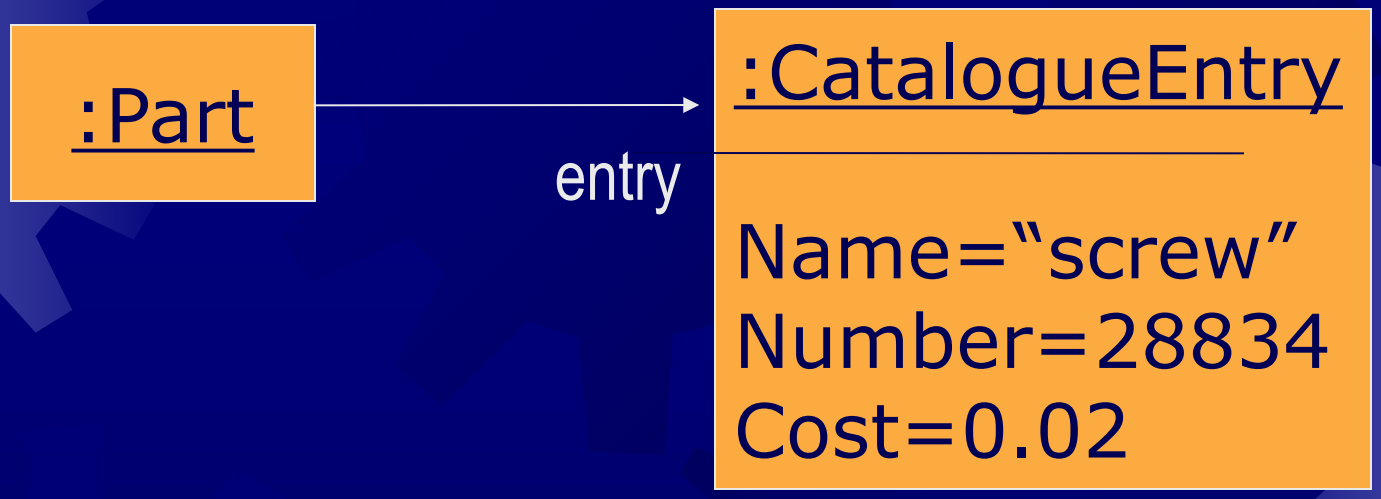
```
private:
    string name;
    long number;
    double cost;
};

class Part {
public:
    Part(CatalogueEntry * e){
        entry = e;
    }
private:
    CatalogueEntry * entry;
}
```


24

```
int main()
{
   CatalogueEntry *screw = new
         CatalogueEntry("screw",28834,0.02);
   Part * screw1 = new Part(screw);
}
```

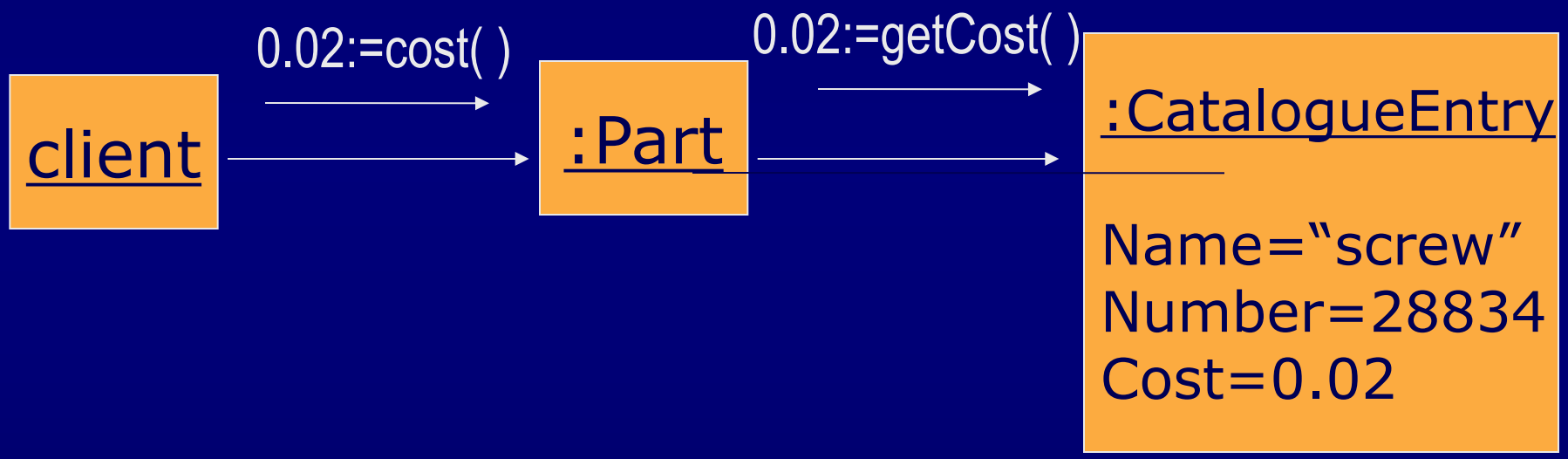

25

Navigability

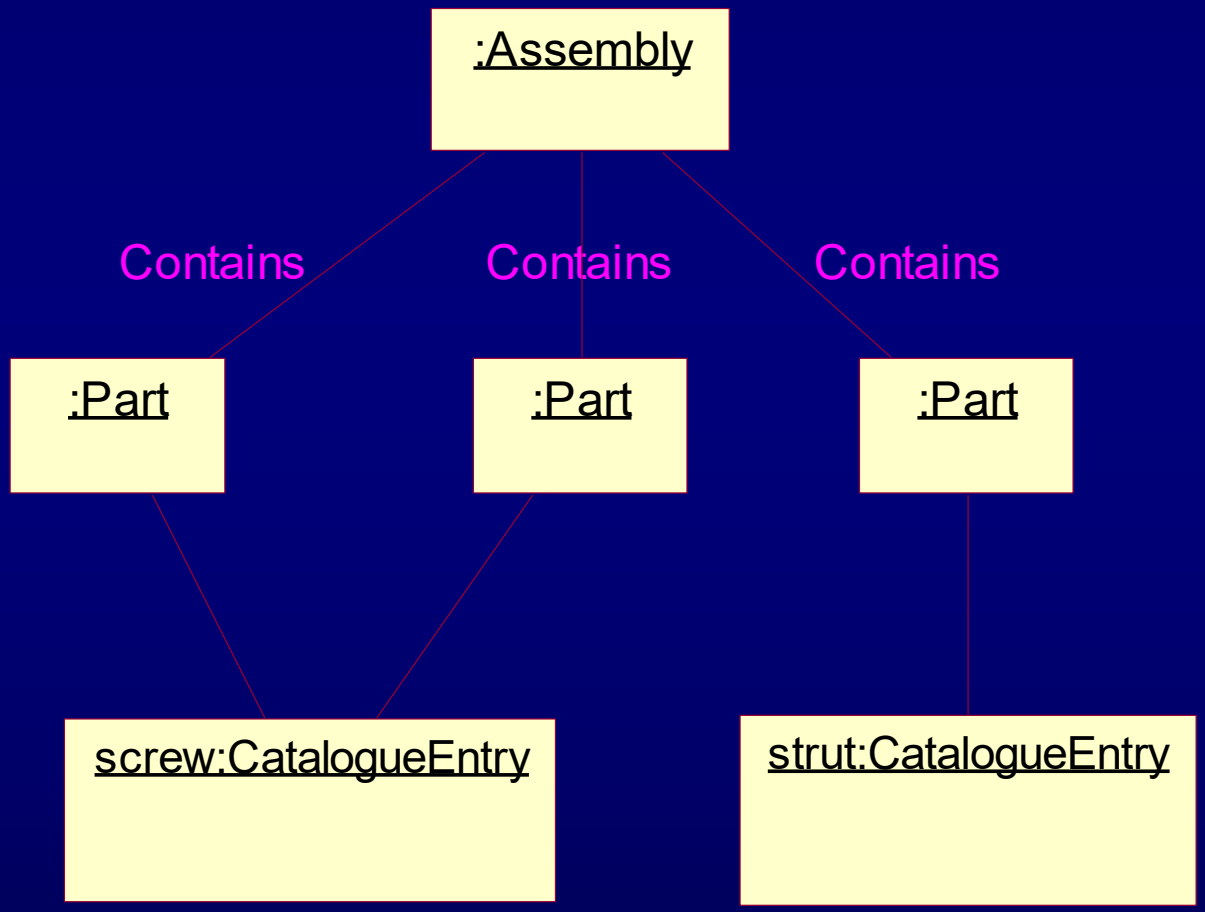


26

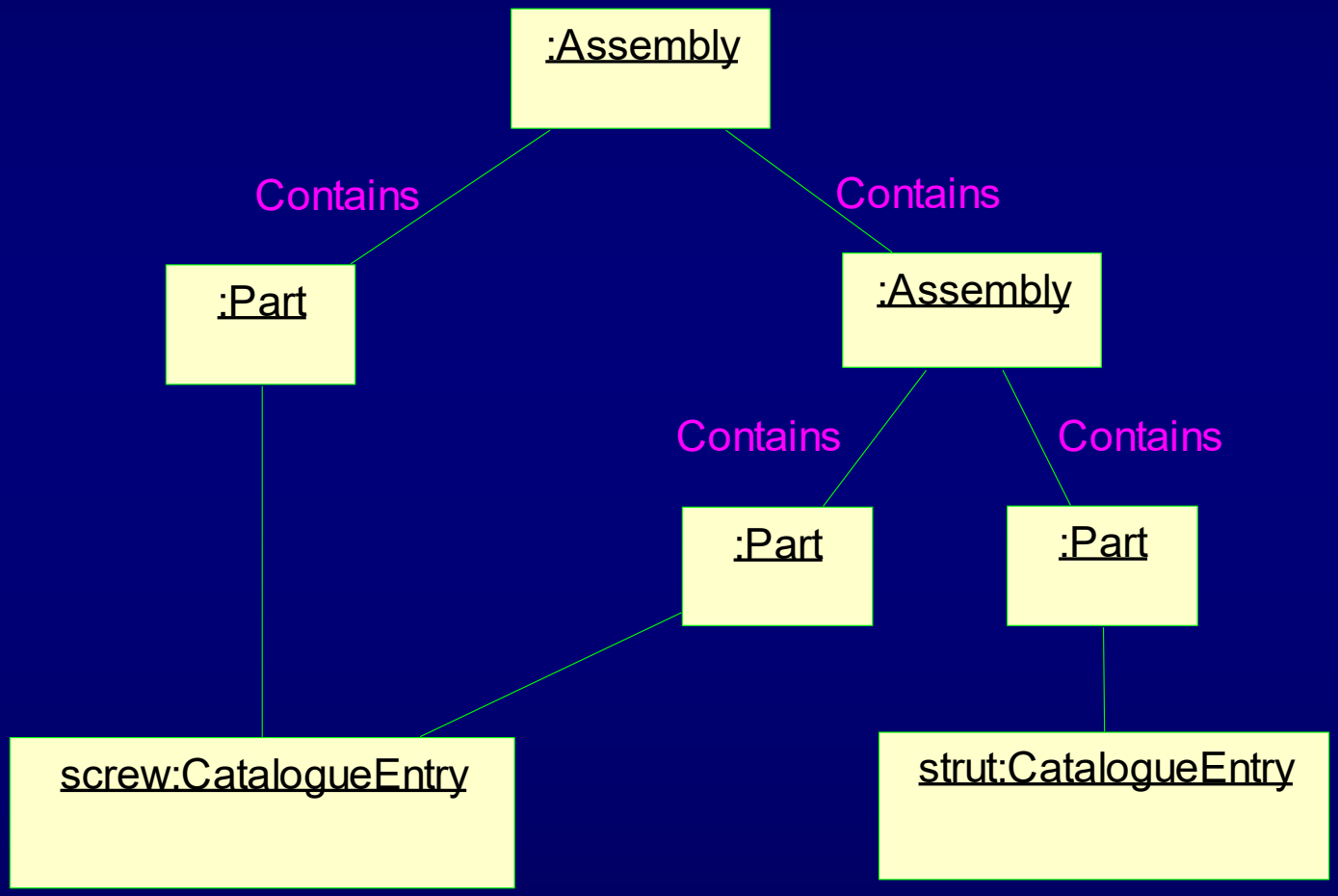
Message Passing



A simple structure

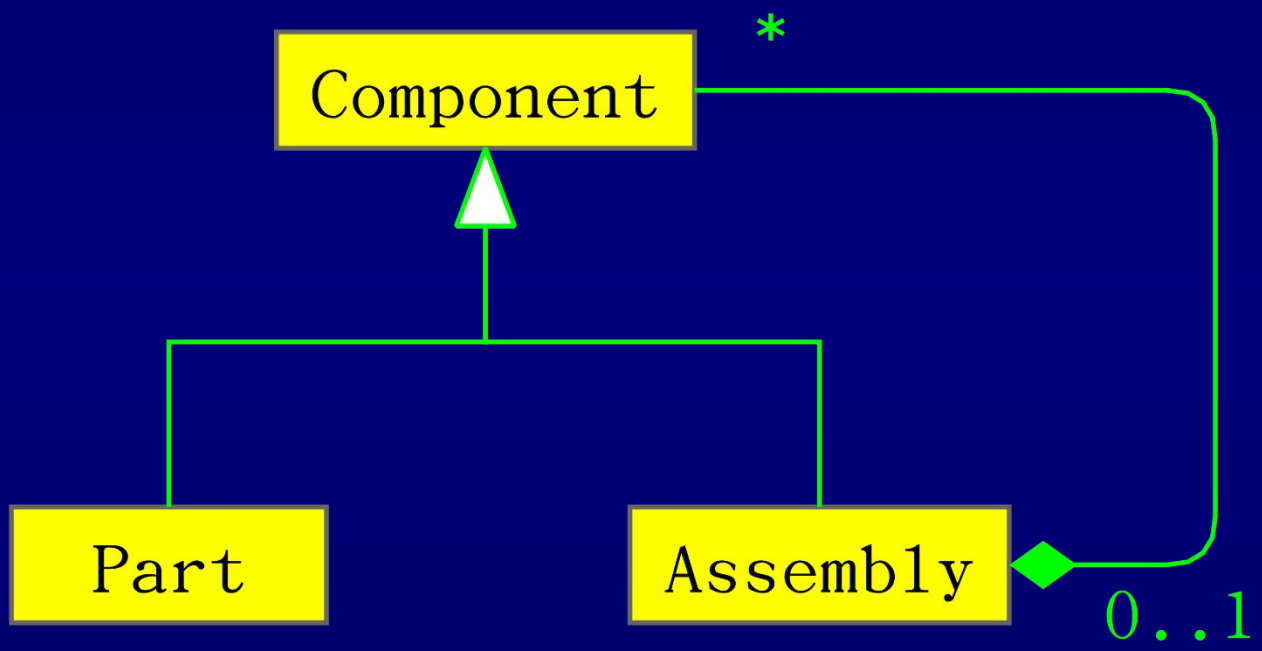


A more realistic structure



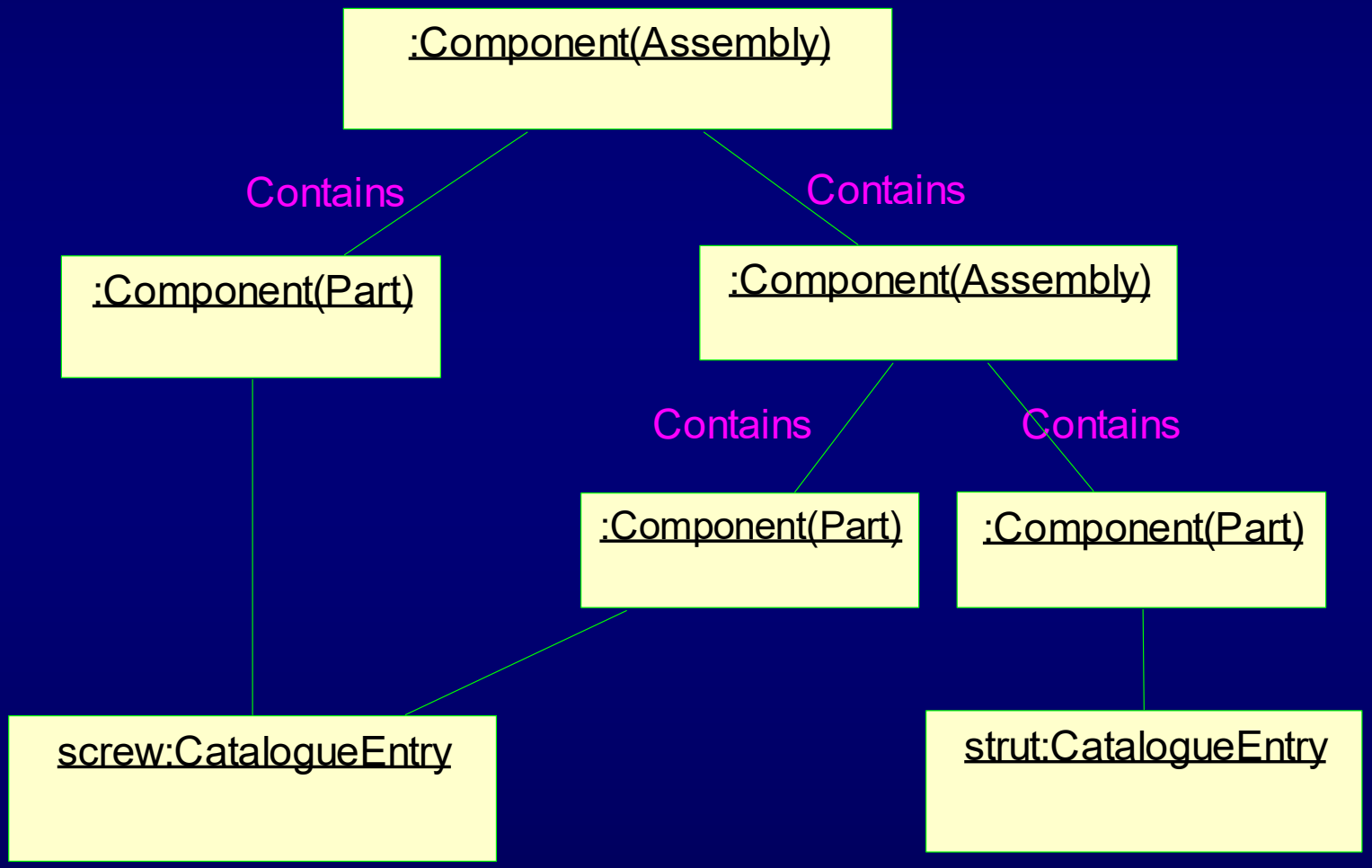
Impossible for implementation!

Use of abstract class



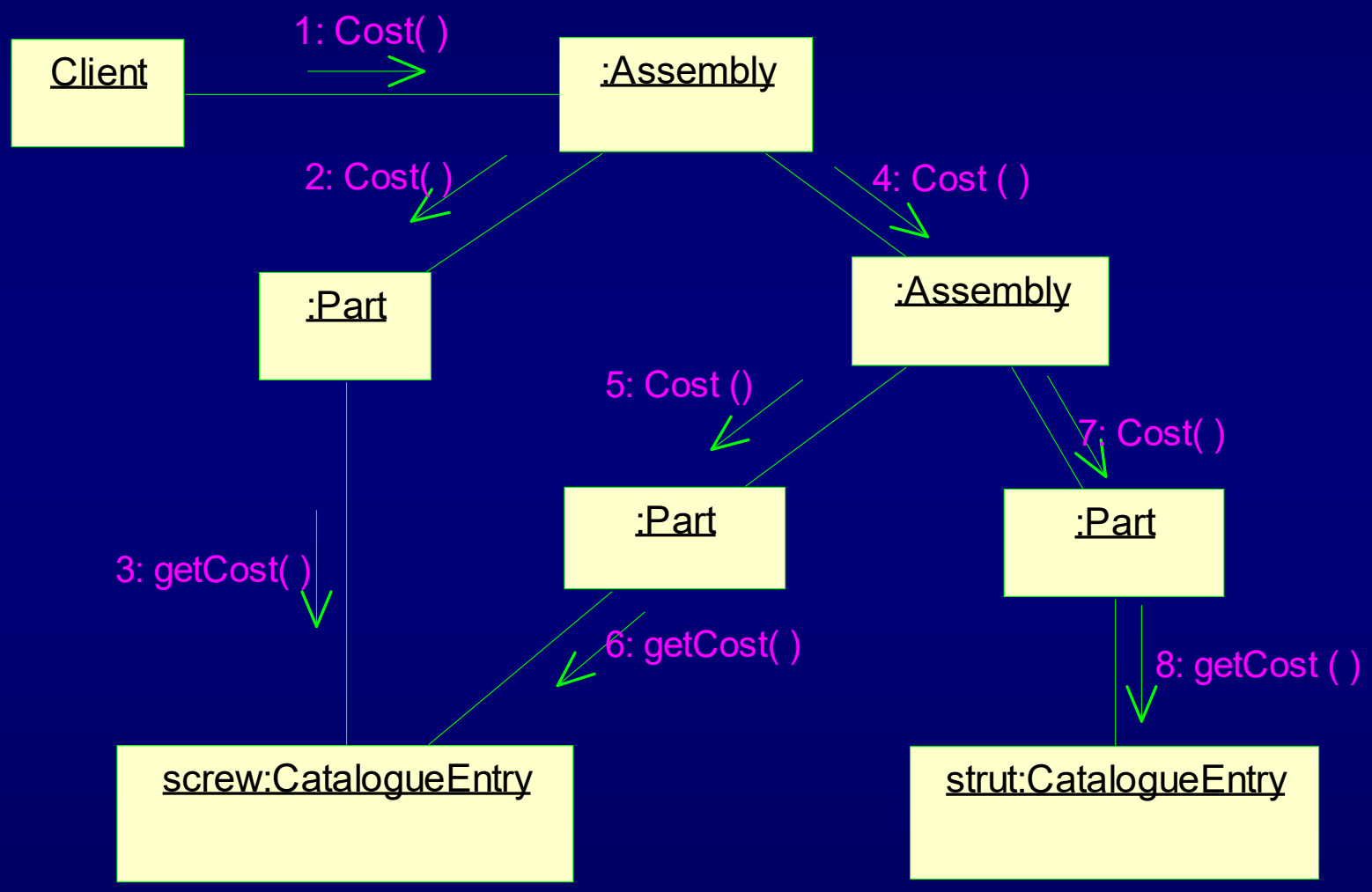
30

Object diagram



31

Late Binding & Polymorphism



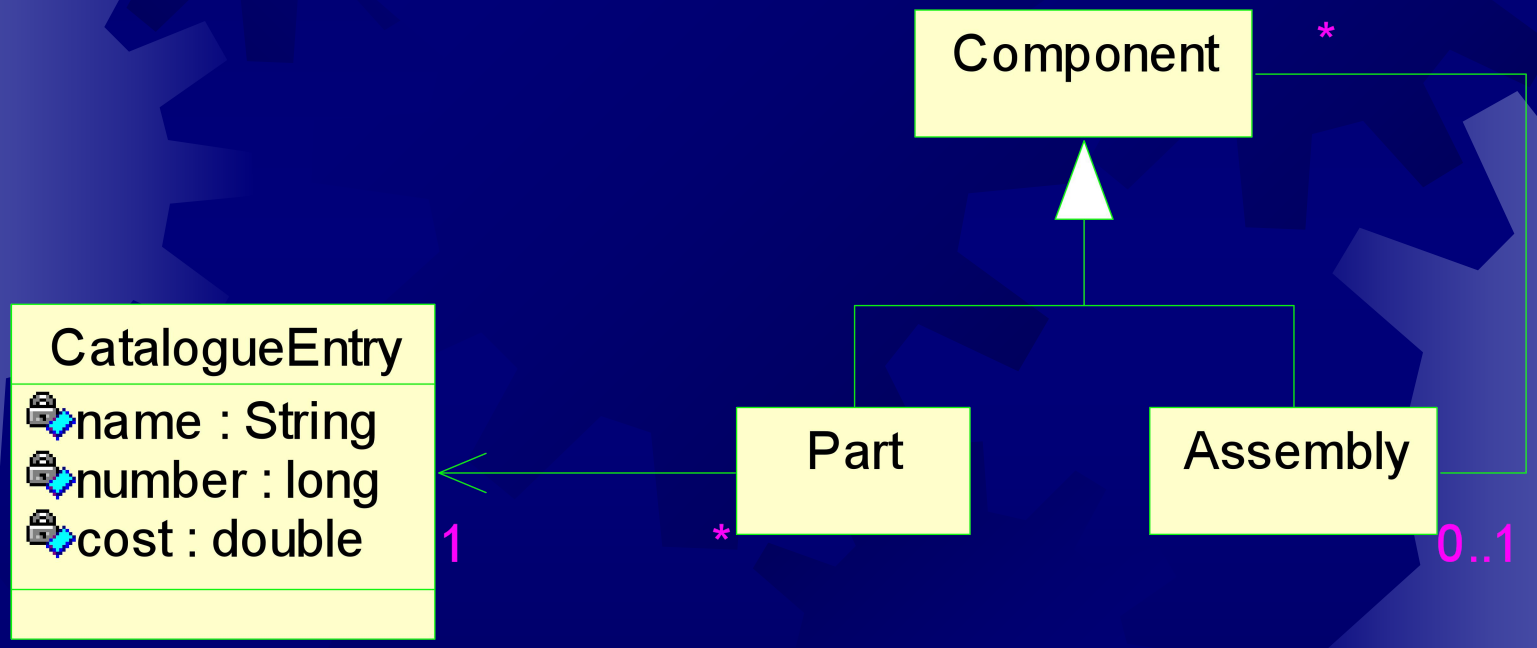
32

Class Diagrams

- Object diagrams: can only show a small subset of a program's possible states.
- We need to describe the software system in a more abstract level: class diagrams.
  - Similar to object diagram
  - Types of the attributes are shown
  - Association (linkage counter)
  - Inheritance

33

An example of class diagram



34

The applicability of the object model

- Often: object (real world) object model
- Exceptions: the example above
- Message passing
  - Real world events are often treated as messages.
  - Objects + events Objects + messages
- Strength of OO
  - Localization of data and operations with objects
  - In most cases, the following mapping holds

    real world object object model

35