面向对象分析与设计
Object-Oriented Analysis and Design

陈昊鹏
chen-hp AT sjtu.edu.cn

Fall-2011

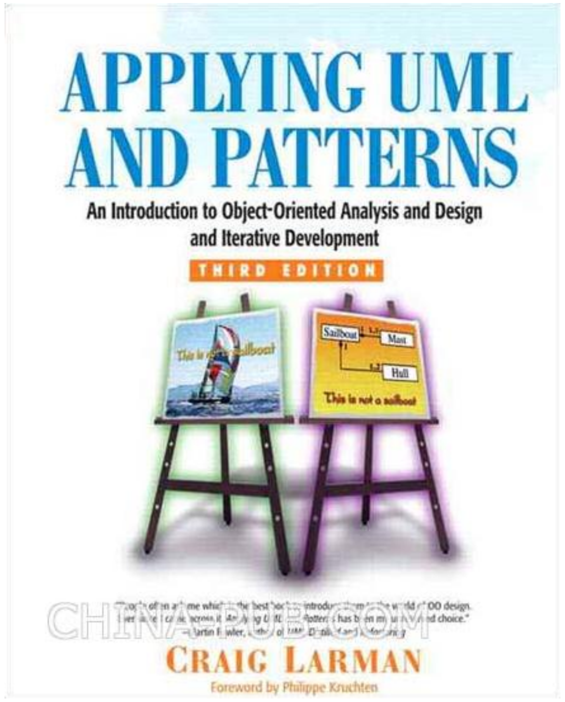
面向对象分析与设计

Object-Oriented Analysis and Design

第6章 基于设计实现系统

Chapter Six

From Design to Implementation


APPLYING UML AND PATTERNS
An Introduction to Object-Oriented Analysis and Design and Iterative Development
THIRD EDITION
This is not a sailboat
Sailboat
Mast
Hull
This is not a sailboat
CRAIG LARMAN
Foreword by Philippe Kruchten

Agenda

- **Implementation Model and UML**
  - **Deployment Diagram**
- **Forward, Reverse, and Round-Trip Engineering**
- **Code and Test**

面向对象分析与设计

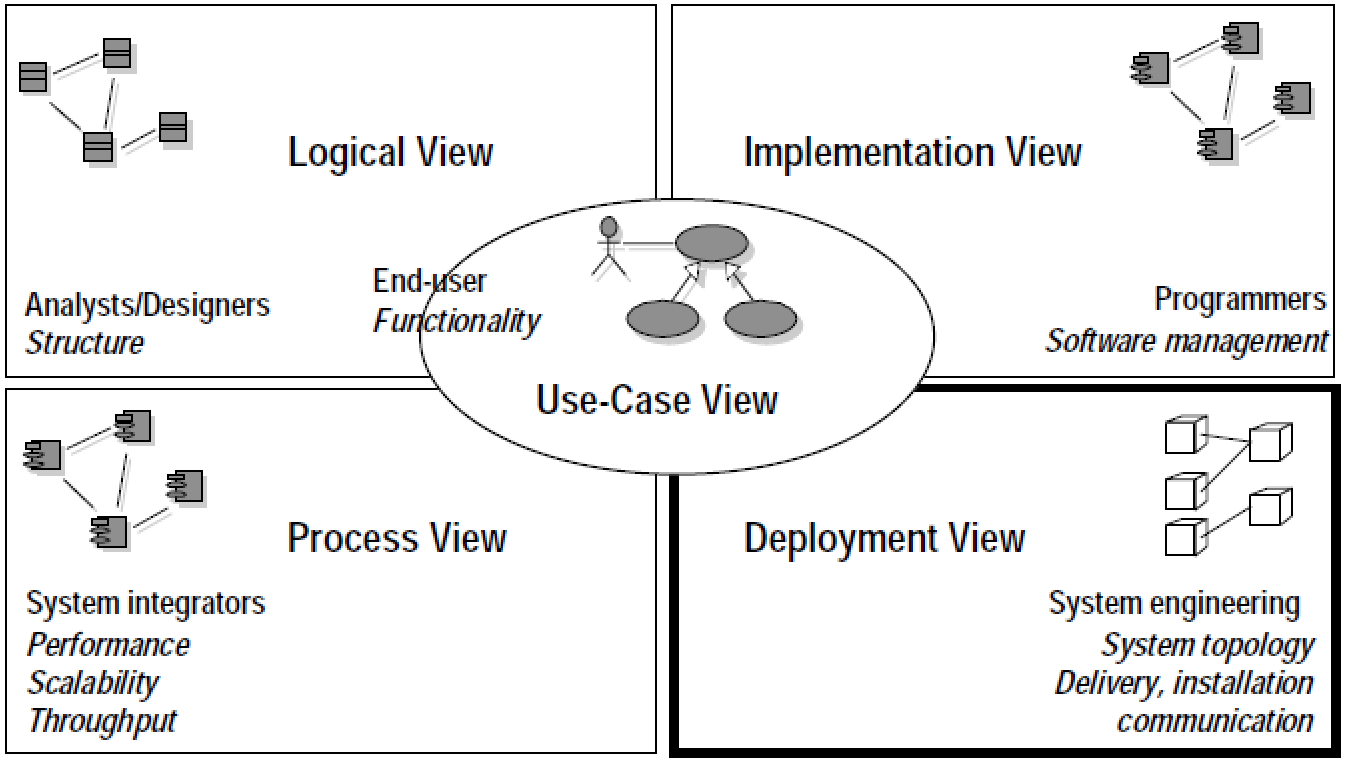
Object-Oriented Analysis and Design

Implementation Model and UML Deployment Diagram

Implementation Model and UML Deployment Diagram

- **The Deployment View**
- **UML Deployment Diagram**
  - **What is Node?**
  - **What Is Connection?**
  - **What Is Artifact ?**

Key Concepts: The Deployment View

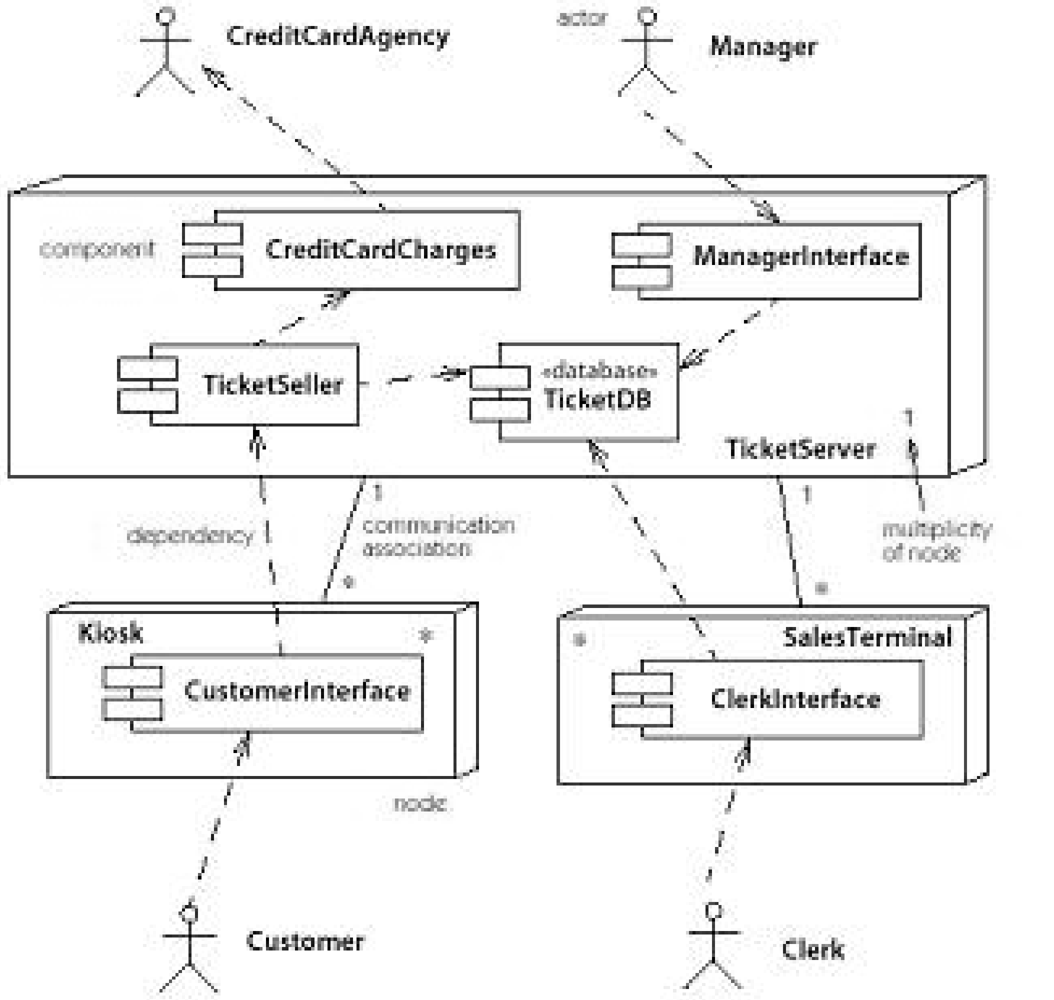


The Deployment View is an “architecturally significant” slice of the Deployment Model.

UML Deployment Diagram

- **A deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them.**
- **Captures the topology of a system's hardware**
- **Built as part of architectural specification**
  - **Purpose**
  - **Specify the distribution of components**
  - **Identify performance bottlenecks**
- **Developed by architects, networking engineers, and system engineers**

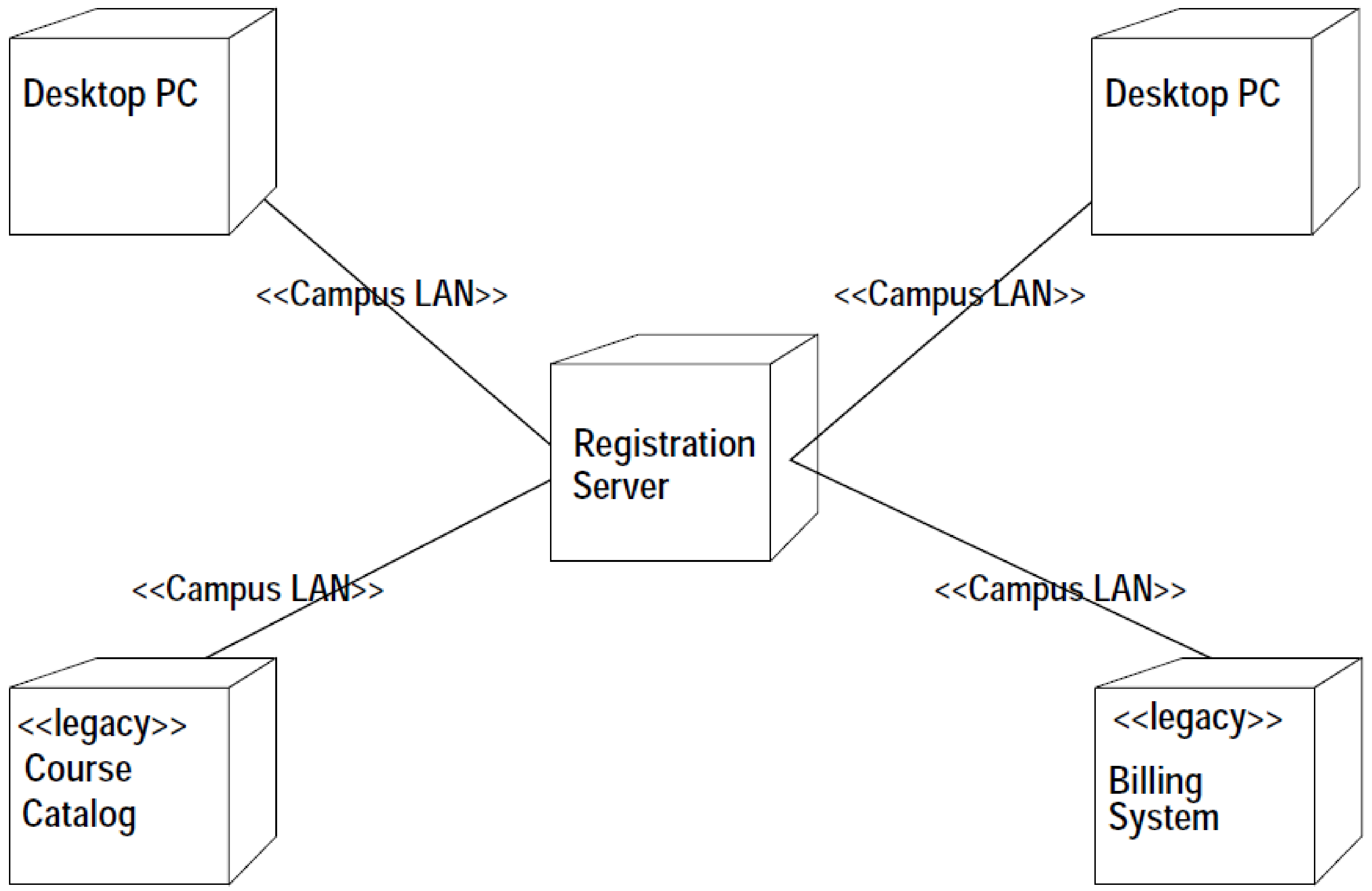
UML Deployment Diagram (1.x) - Example



Deployment diagram (descriptor level)

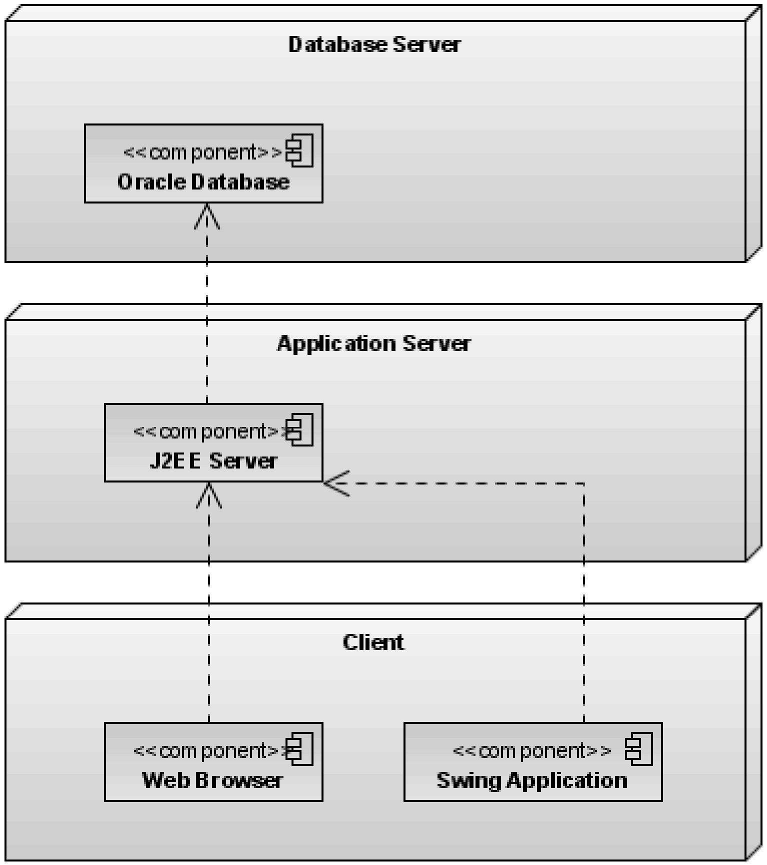
from Prof. Rao Ruonan

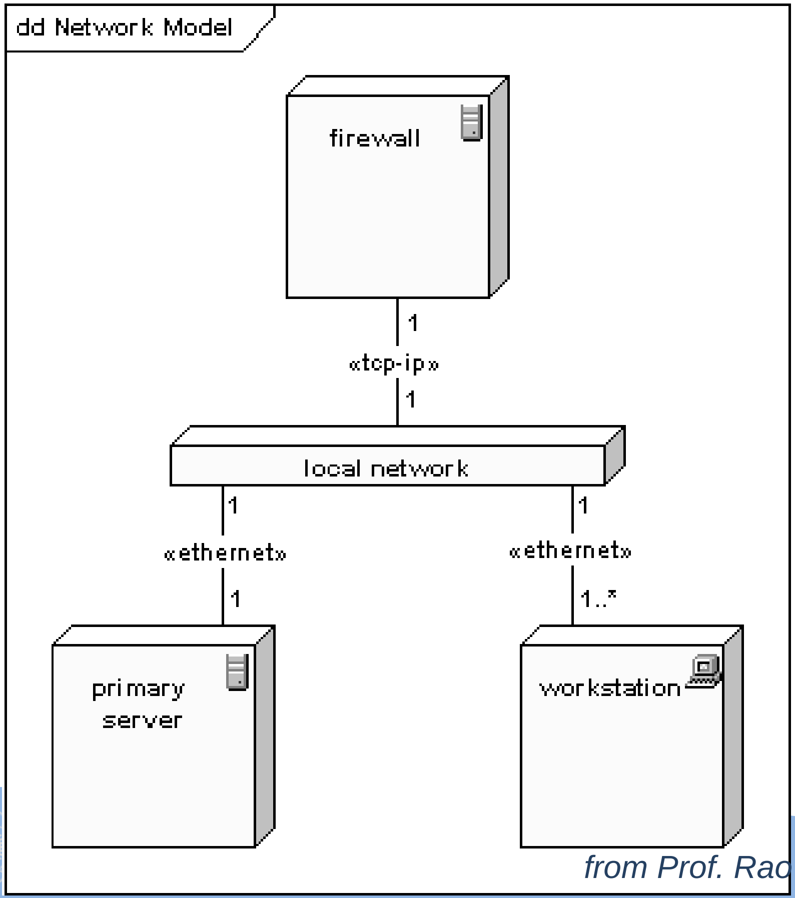
UML Deployment Diagram (1.x) - Example



UML Deployment Diagram (2.x)

- Models the run-time architecture of a system
- A diagram that shows the configuration of run time processing nodes and the artifacts that live on them.

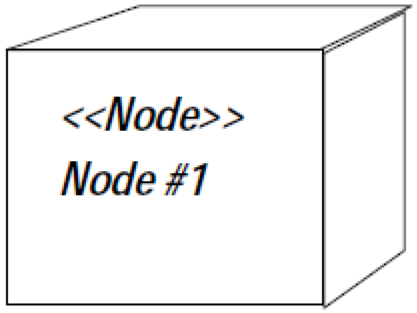




from Prof. Rao Ruonan

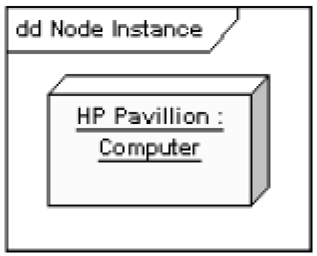
What is Node?

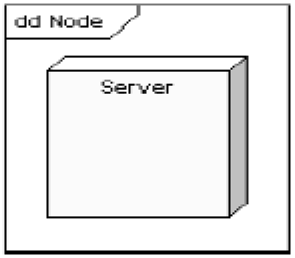
- A node is a physical element that exists at run time and represents a computational resource, generally having at least some memory and, often, processing capability.
- A set of components may reside on a node and may also migrate from node to node.
- Graphically, a node is rendered as a cube, usually including only its name.



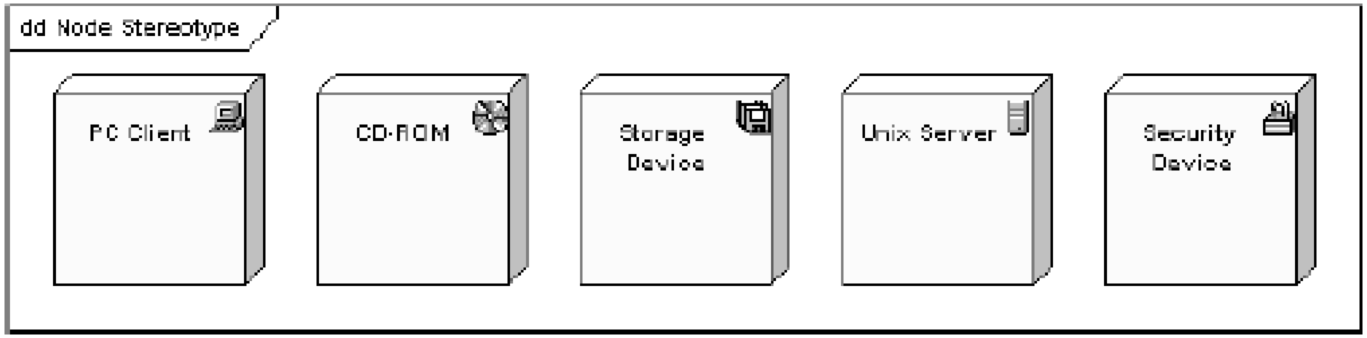
Deployment Diagram (2.x) - Node

- **Node Instance**





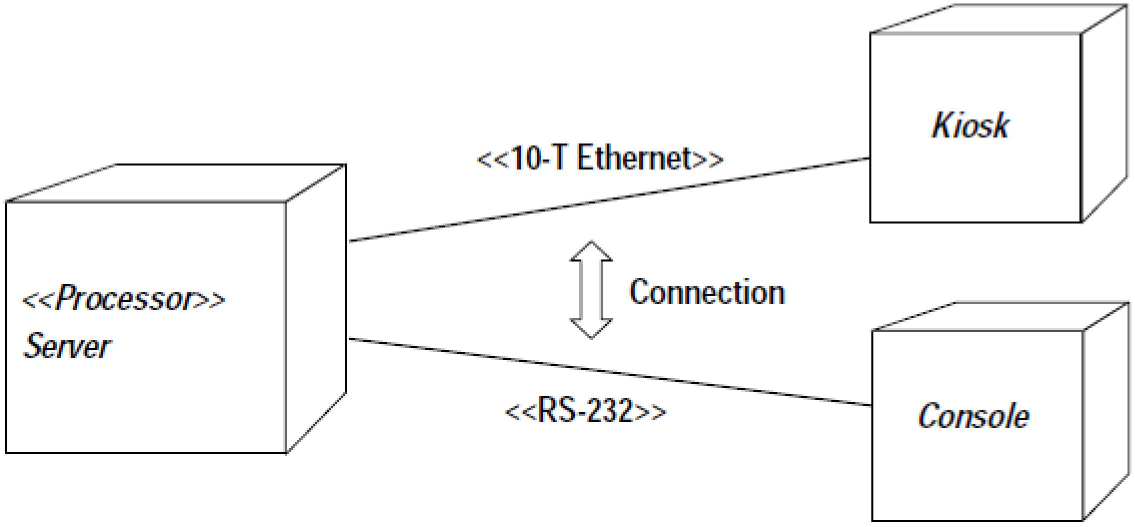
- **Node Stereotypes**
  - **A number of standard stereotypes are provided for nodes,**
  - **namely «cd-rom», «computer», «disk array», «pc», «pc client», «pc server», «secure», «server», «storage», «unix server», «user pc»**



- **Association**
  - **In the context of a deployment diagram, an association represents a communication path between nodes**

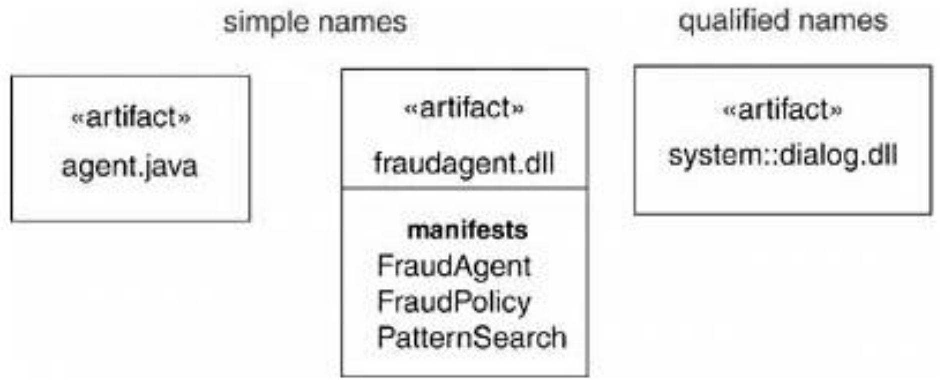
What Is a Connection?

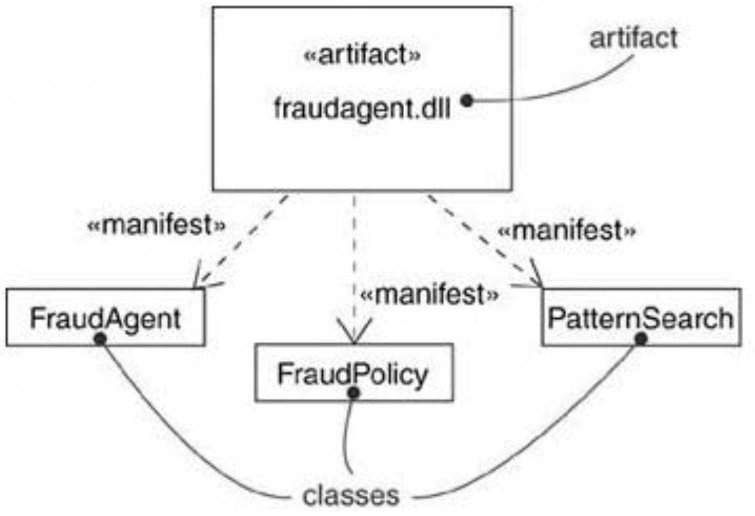
- A connection represents a:
  - Communication mechanism
    - Physical medium
    - Software protocol



Deployment Diagram (2.x) - Artifact

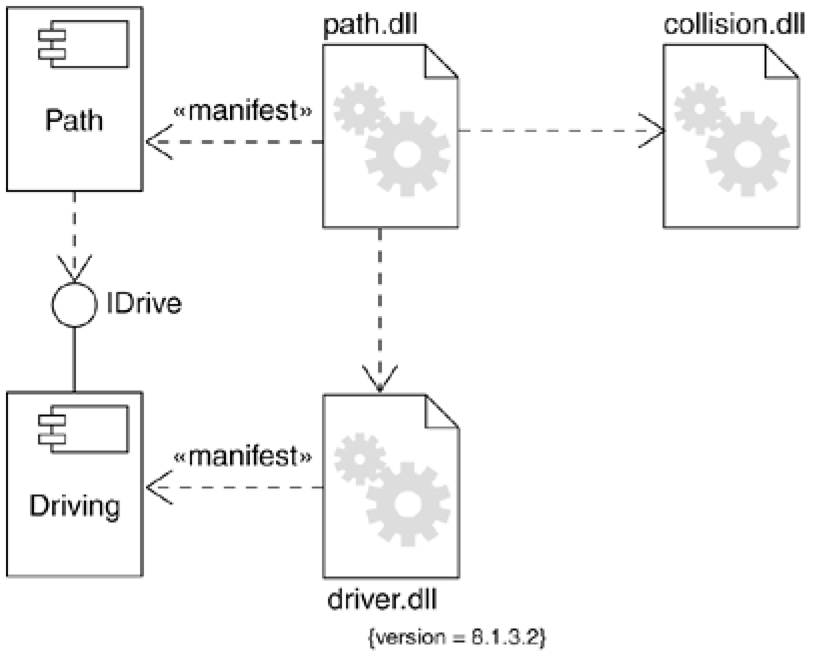
- **Artifact**
  - **A physical part of a system that exists at the level of the implementation platform.**
  - **Graphically, an artifact is rendered as a rectangle with the keyword «artifact».**





Deployment Diagram (2.x) - Artifact Diagram

- **Artifact Diagram**
  - **A variety of deployment diagram**
  - **shows a set of artifacts and their relationships.**
  - **commonly contain**
    - **artifacts**
    - **dependency, generalization, association, and realization relationships**



面向对象分析与设计

Object-Oriented Analysis and Design

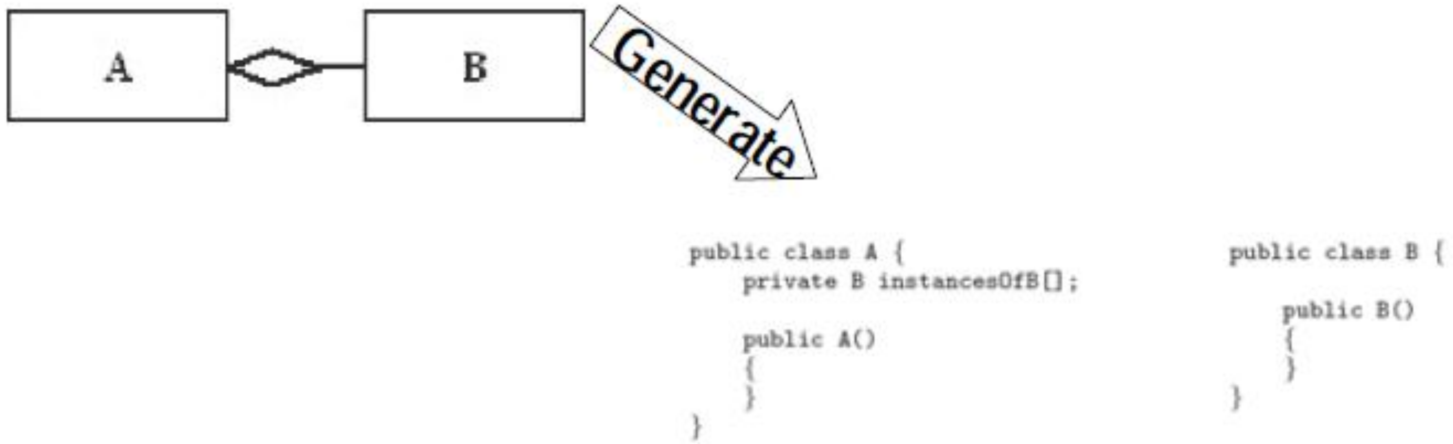
Forward, Reverse, and Round-Trip Engineering

Forward, Reverse, and Round-Trip Engineering

- **Forward Engineering**
- **Reverse Engineering**
- **Round-Trip Engineering**

Forward Engineering

- **Forward engineering means the generation of code from UML diagrams**
- **Many of the tools can only do the static models:**
  - **They can generate class diagrams from code, but can't generate interaction diagrams.**
  - **For forward engineering, they can generate the basic (e.g., Java) class definition from a class diagram, but not the method bodies from interaction diagrams.**
- **Demo**

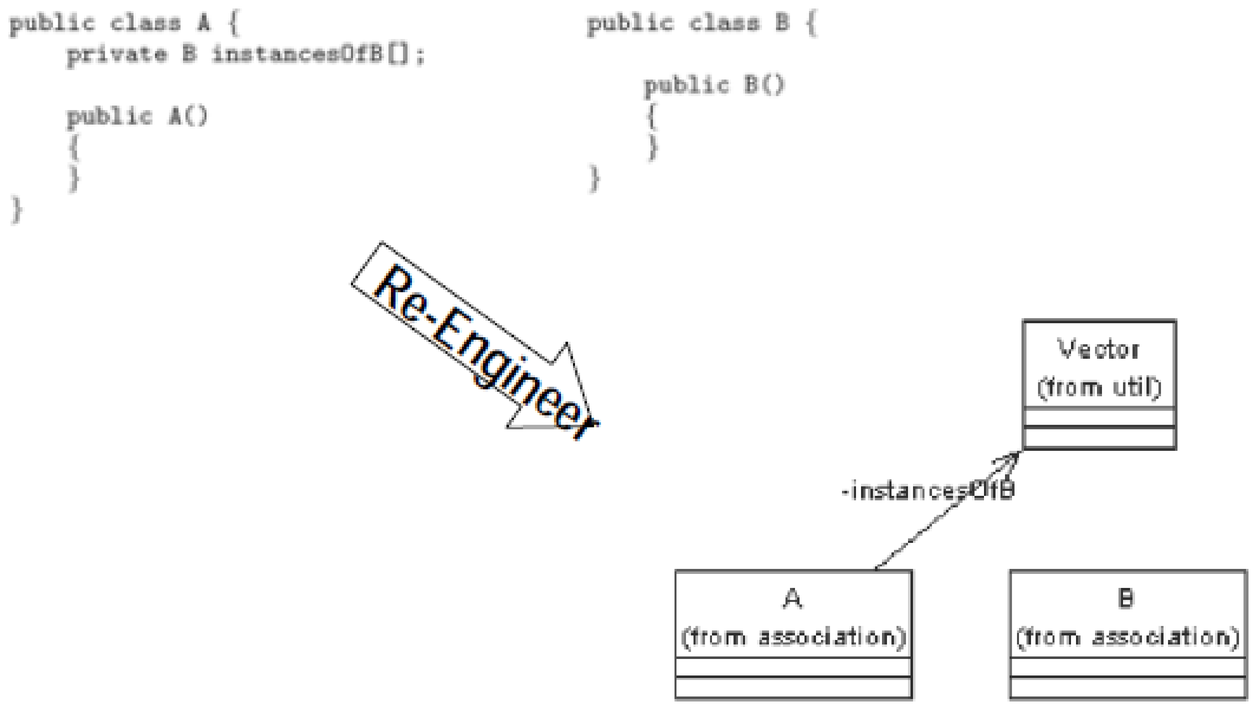


```
public class A {
    private B instancesOfB[];

    public A()
    {
    }
}
```

```
public class B {

    public B()
    {
    }
}
```

Reverse Engineering

- **Reverse engineering means generation of UML diagrams from code**
- **Demo**

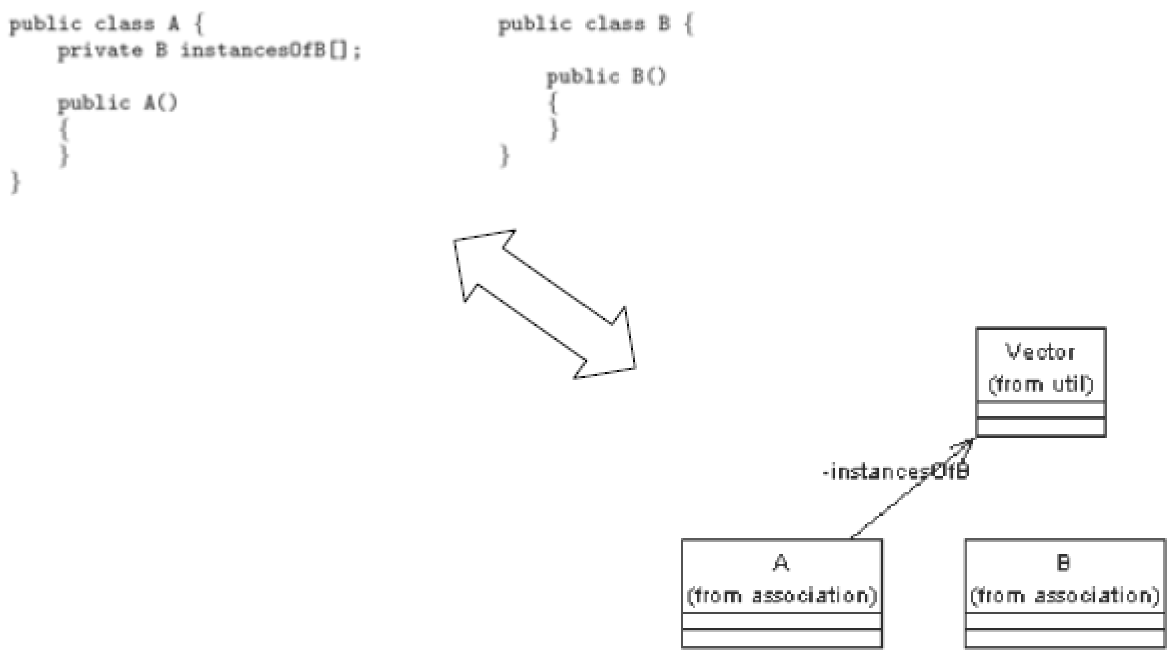


```
public class A {
    private B instancesOfB[];

    public A()
    {
    }
}
```

```
public class B {

    public B()
    {
    }
}
```

Round-Trip Engineering

- Round-trip engineering closes the loop
  - the tool supports generation in either direction and can synchronize between UML diagrams and code, ideally automatically and immediately as either is changed.
- Demo



```
public class A {
    private B instancesOfB[];

    public A()
    {
    }
}
```

```
public class B {

    public B()
    {
    }
}
```

面向对象分析与设计

Object-Oriented Analysis and Design

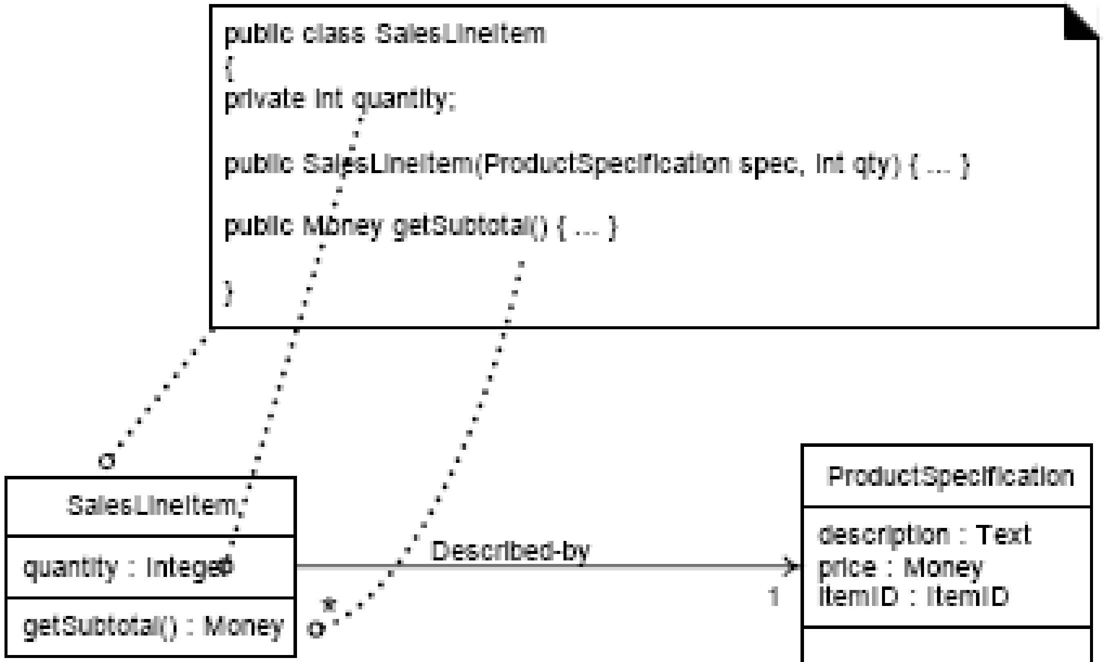
Code and Test

Code and Test

- Creating Class Definitions from Class Diagram
- Creating Methods from Interaction Diagrams
- Collection Classes in Code
- Test-Driven Development
- Refactoring

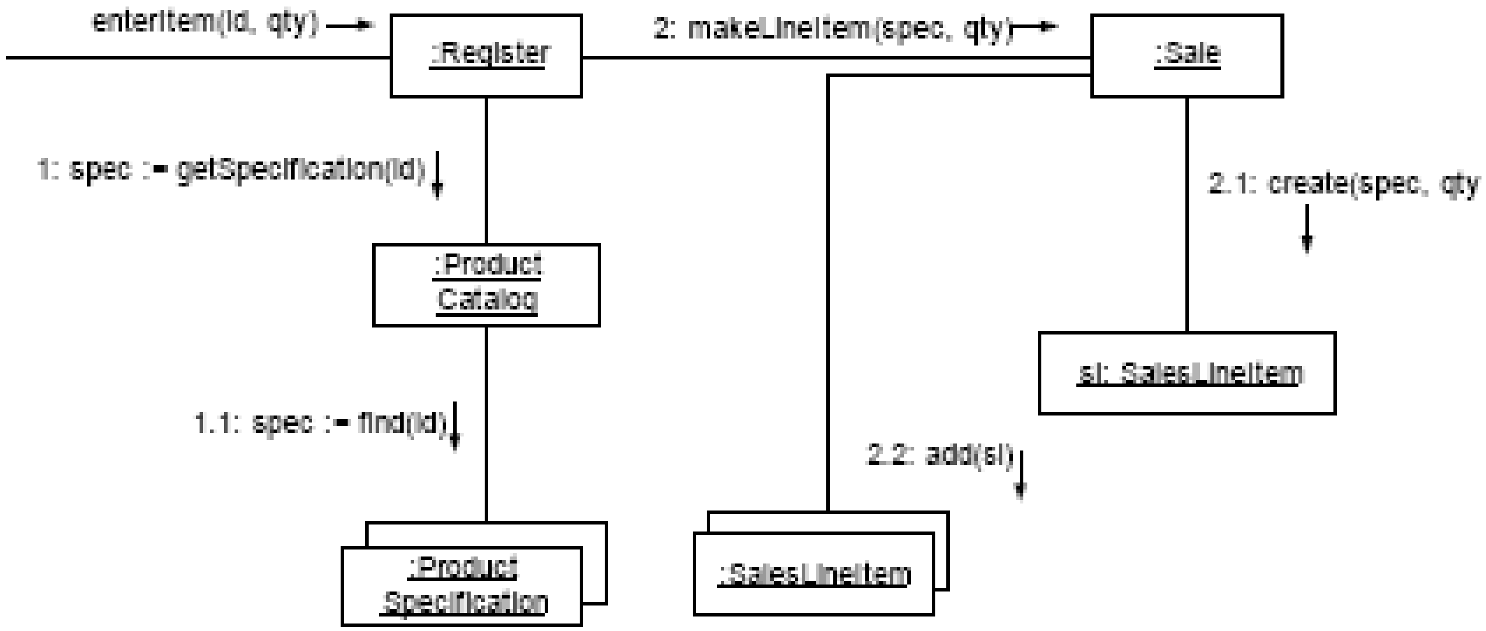
Creating Class Definitions from Class Diagram

➢ **Defining a Class with Method Signatures and Attributes**

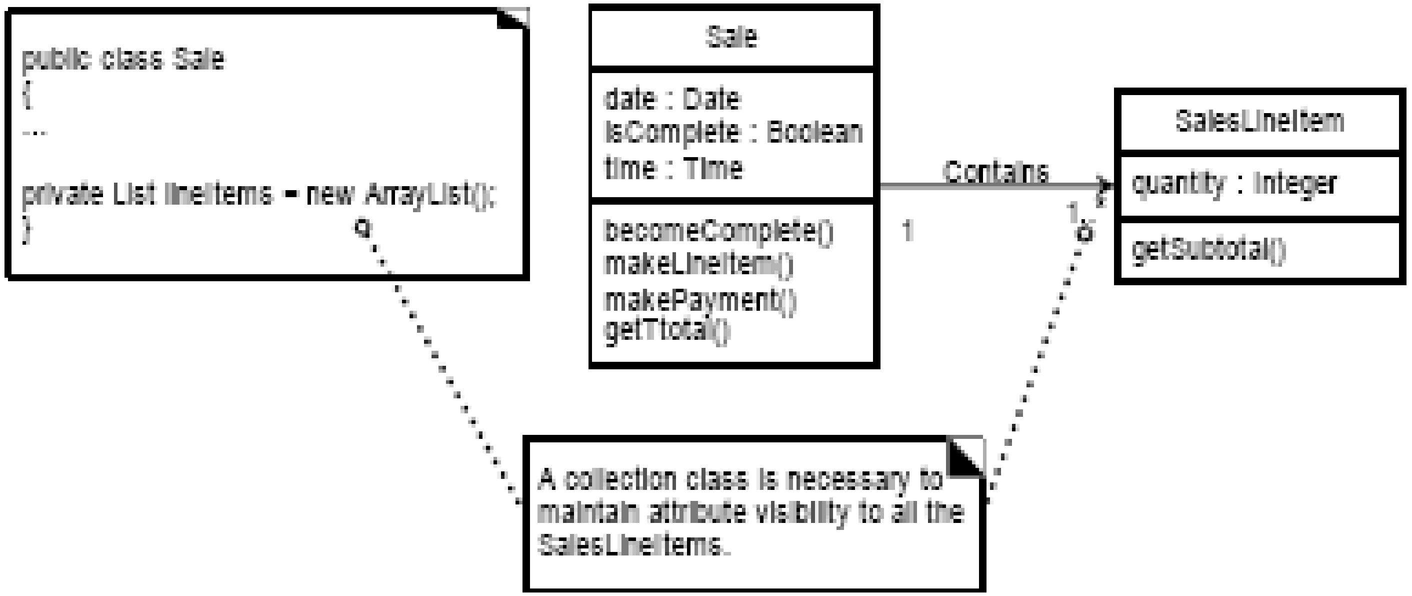


```
public class SalesLineItem
{
private int quantity;

public SalesLineItem(ProductSpecification spec, int qty) { ... }

public Money getSubtotal() { ... }

}
```

Creating Methods from Interaction Diagrams



Collection Classes in Code



```
public class Sale
{
...

private List lineItems = new ArrayList();
}
```

A collection class is necessary to maintain attribute visibility to all the SalesLineItems.

Test-Driven Development

- **An excellent practice promoted by the iterative and agile XP method, and applicable to the UP, is test-driven development (TDD).**
  - **It is also known as test-first development**
- **In OO unit testing TDD-style, test code is written before the class to be tested and the developer writes unit testing code for nearly all production code.**
- **Unit testing framework**
  - **The most popular unit testing framework is the xUnit family (for many languages)**
    - **For Java, the popular version is JUnit.**
    - **There's also an NUnit for .NET**

Refactoring

- **Refactoring is a structured, disciplined method to rewrite or restructure existing code without changing its external behavior,**
  - **applying small transformation steps combined with re-executing tests each step.**
- **Continuously refactoring code is another XP practice and applicable to all iterative methods**
- **Code that's been well-refactored is short, tight, clear, and without duplication it looks like the work of a master programmer.**
  - **Code that doesn't have these qualities smells bad or has code smells.**

面向对象分析与设计

Object-Oriented Analysis and Design

下课！