Chapter 8

- **Component-Level Design**

What is a Component?

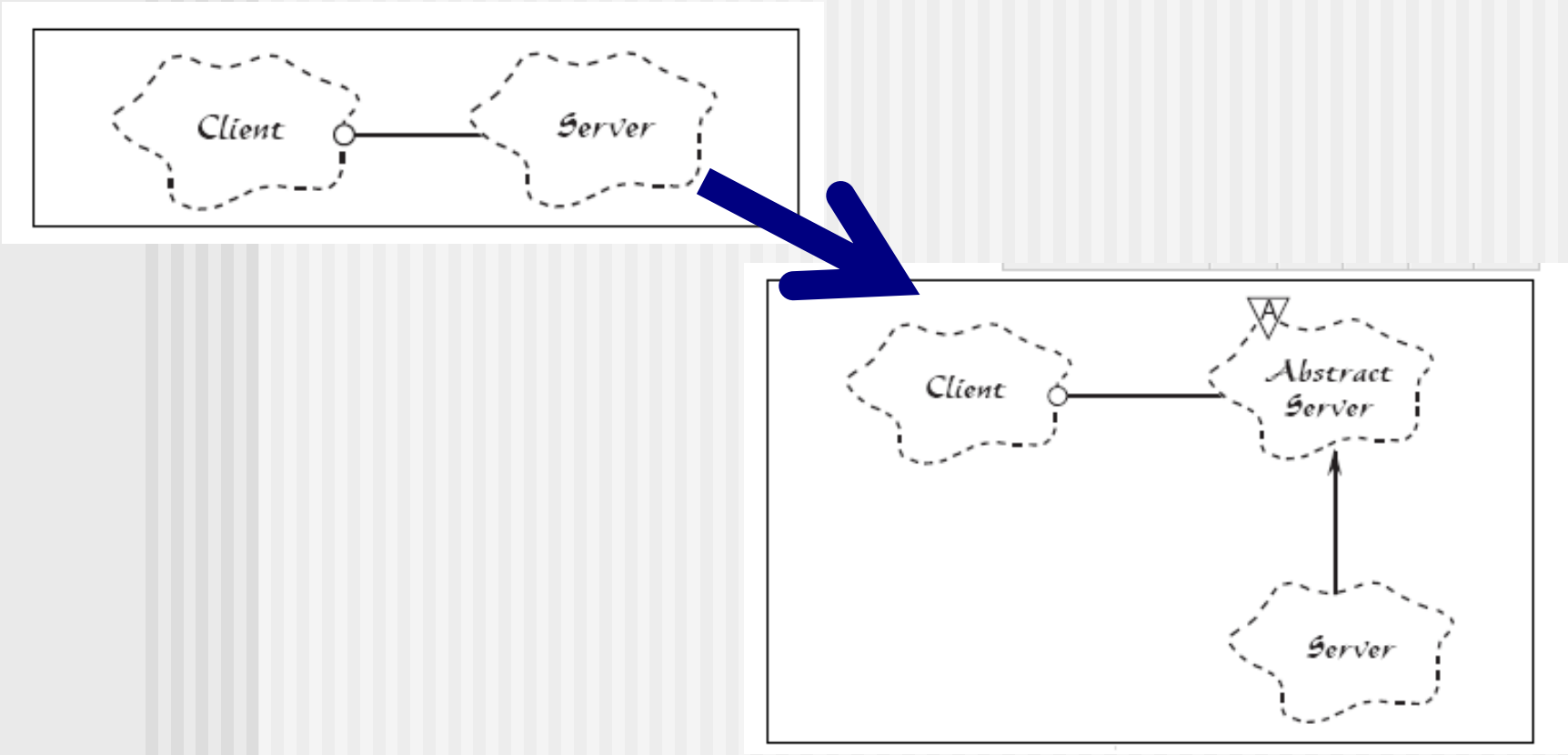
- *OMG Unified Modeling Language Specification* [OMG01] defines a component as
  - "… a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces.""
- *OO view:* a component contains a set of collaborating classes
- *Conventional view:* a component contains processing logic, the internal data structures that are required to implement the processing logic, and an interface that enables the component to be invoked and data to be passed to it.

Basic Design Principles

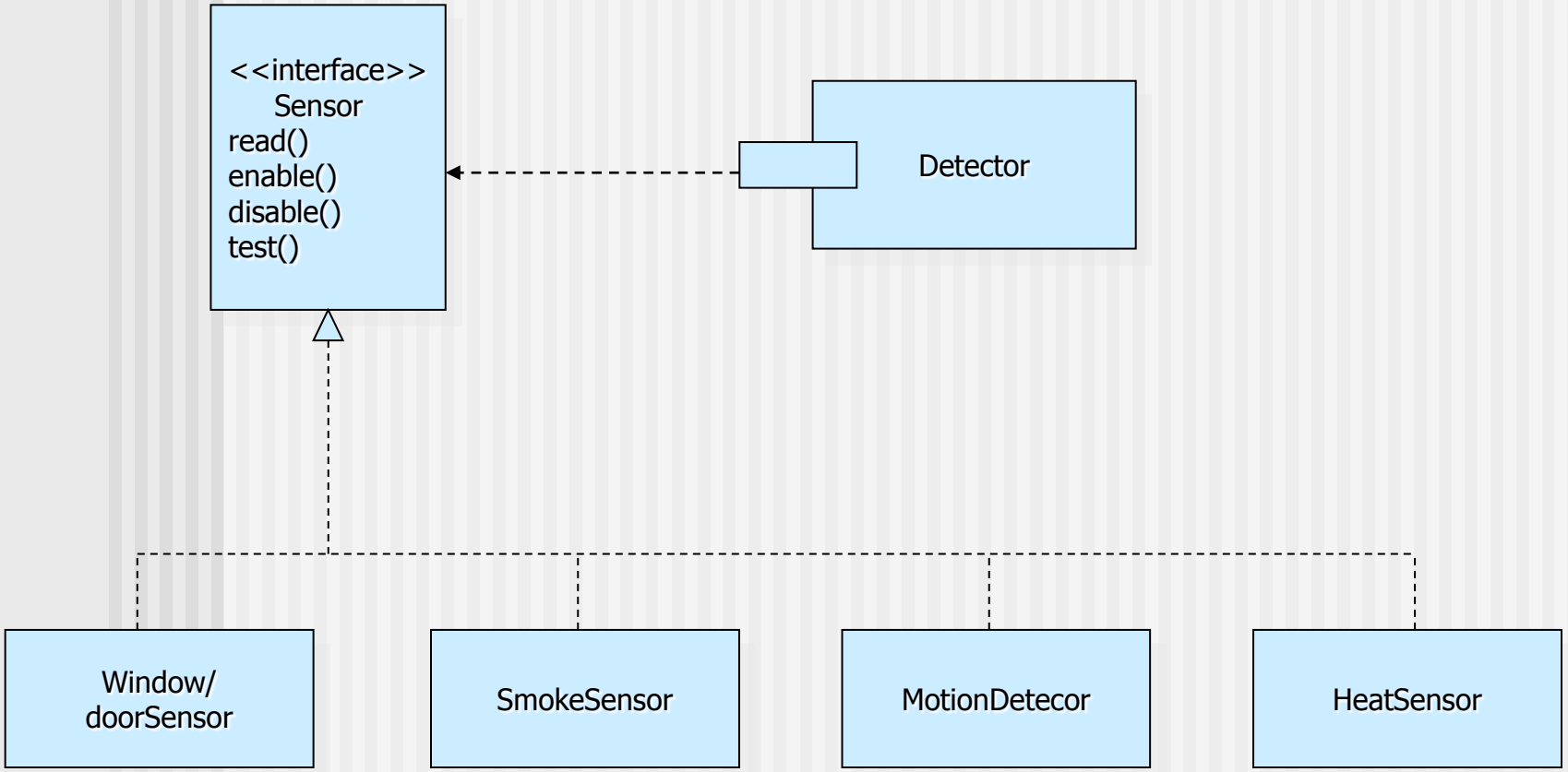
- The Open-Closed Principle (OCP). *"A module [component] should be open for extension but closed for modification.*
- The Liskov Substitution Principle (LSP). *"Subclasses should be substitutable for their base classes.*
- Dependency Inversion Principle (DIP). *"Depend on abstractions. Do not depend on concretions."*
- The Interface Segregation Principle (ISP). *"Many client-specific interfaces are better than one general purpose interface.*
- The Release Reuse Equivalency Principle (REP). *"The granule of reuse is the granule of release."*
- The Common Closure Principle (CCP). *"Classes that change together belong together."*
- The Common Reuse Principle (CRP). *"Classes that aren't reused together should not be grouped together."*

Source: Martin, R., "Design Principles and Design Patterns," downloaded from http:www.objectmentor.com, 2000.

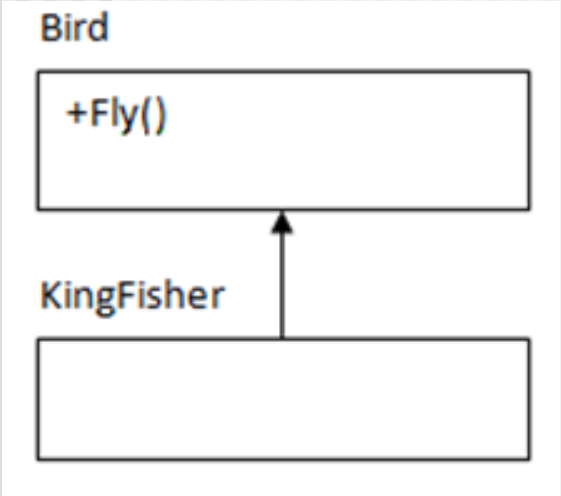
OCP (Open-Closed Principle)



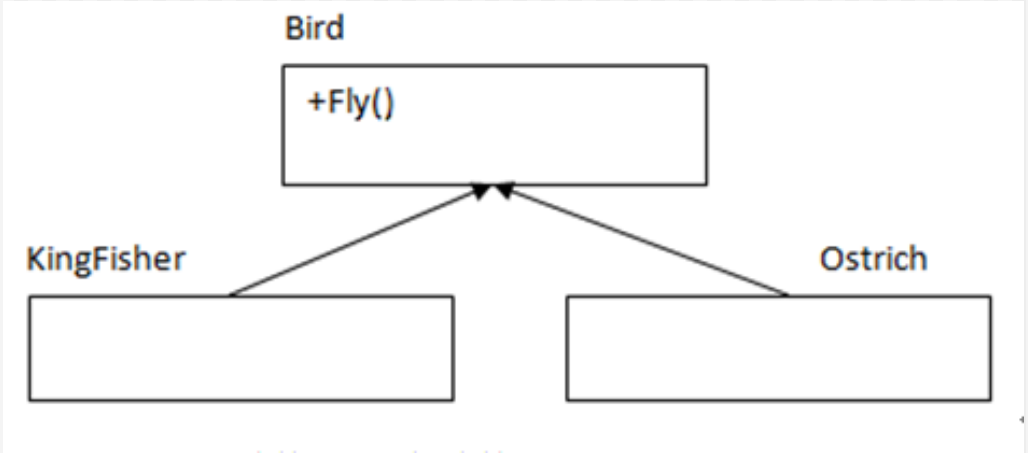
OCP (Open-Closed Principle)



LSP



LSP类结构图



违反LSP类结构图

基本设计原则（续）

- 将多个构件组织起来的原则：
  - REP发布复用等价性原则---对类打包管理，同时升级。
  - CCP共同封装原则-----一同变更的类应该合在一起。
  - CRP共同复用原则-----可能一起被复用的类才能打包到一块。

Design Guidelines

- Components
  - Naming conventions should be established for components that are specified as part of the architectural model and then refined and elaborated as part of the component-level model
- Interfaces
  - Interfaces provide important information about communication and collaboration (as well as helping us to achieve the OPC)
- Dependencies and Inheritance
  - it is a good idea to model dependencies from left to right and inheritance from bottom (derived classes) to top (base classes).

构件的UML表示

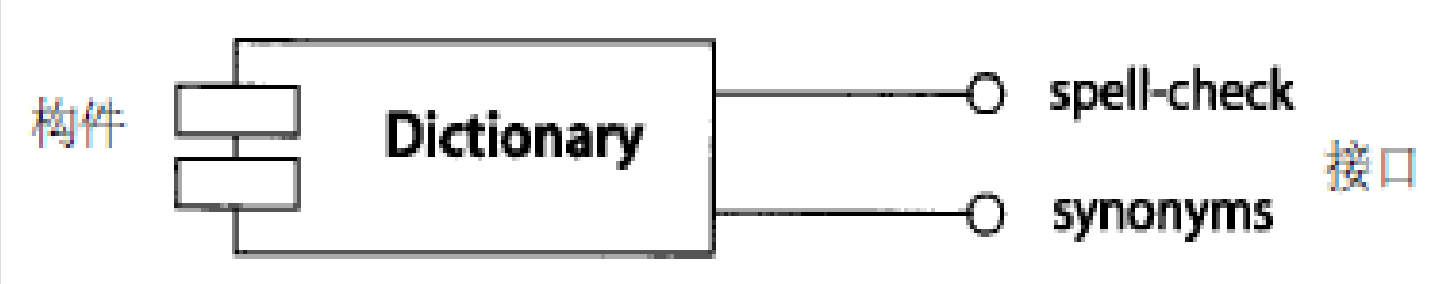


图 带接口的构件

构件具有它们支持的接口和需要从其他构件得到的接口

构件图表示了构件之间的依赖关系。
每个构件实现（支持）一些接口，并使用另一些接口。如果构件间的依赖关系与接口有关，那么构件可以被具有同样接口的其他构件替代。

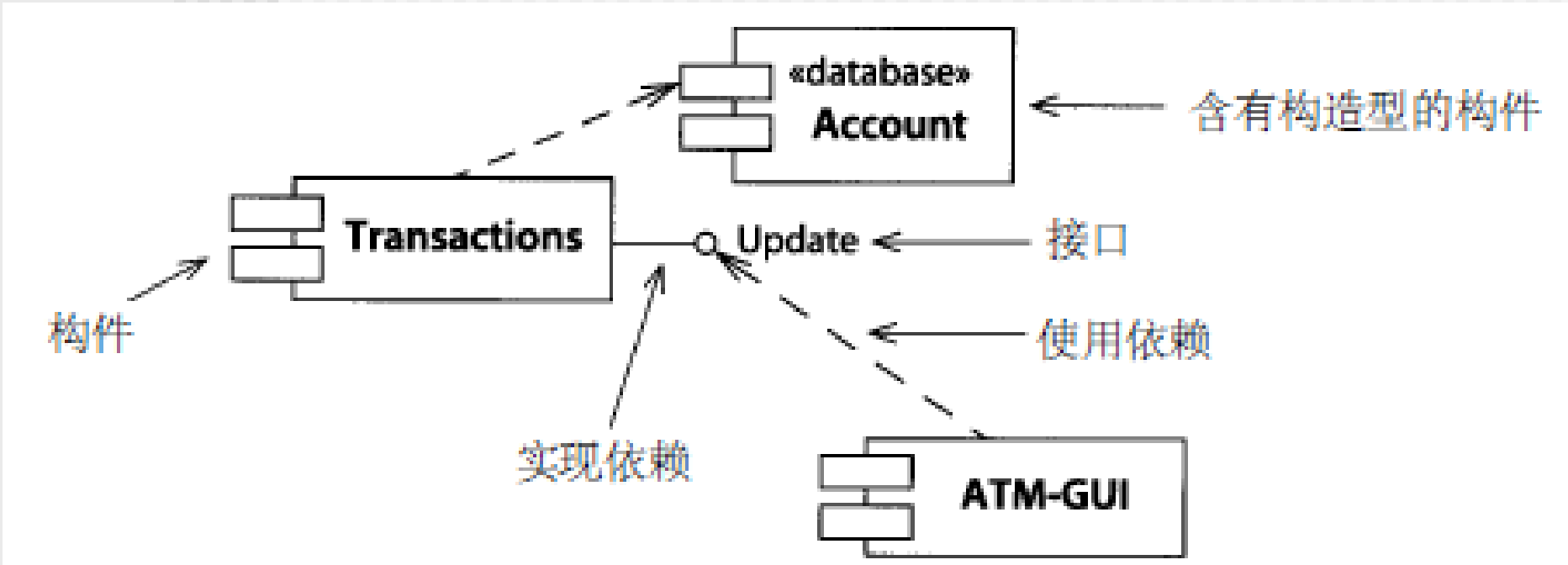


图 构件图

- 内聚与耦合密切相关，同其它模块强耦合的模块意味者弱内聚，强内聚模块意味着与其它模块间松散耦合。

  设计目标：

  力争强内聚、弱耦合

算法设计模型

- 以细节的层次表示算法，它能对质量进行评审
- 选择：
  - 图解 (例如：流程图、盒图)
  - 伪代码(例如：PDL) ... 很多选择
  - 编程语言
  - 决策表

Questions

- 1 What is a component?
- 2 What are the basic design principles of the component?