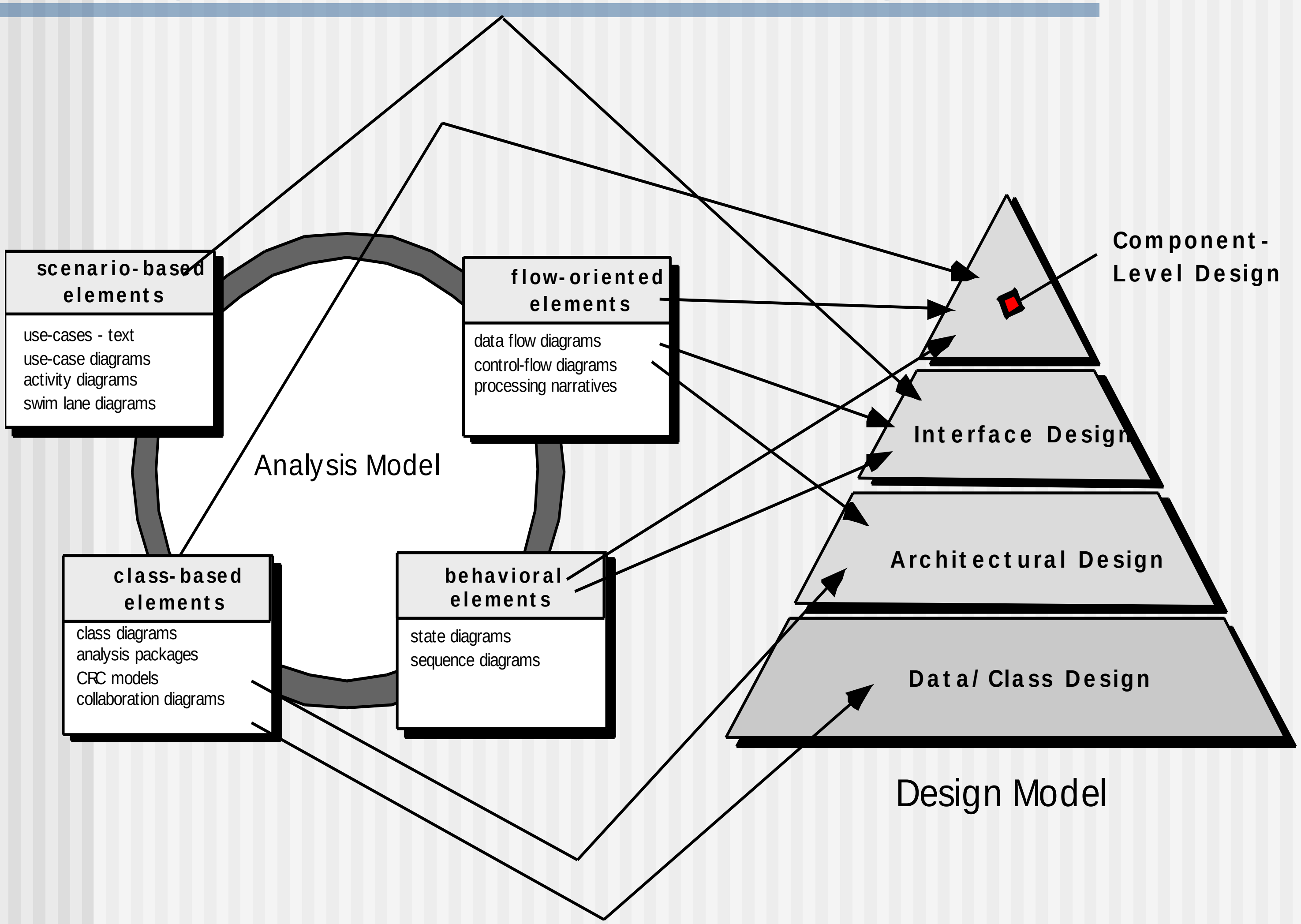
Chapter 6

- **Design Concepts**

Design

- Mitch Kapor, the creator of Lotus 1-2-3, presented a “software design manifesto” in *Dr. Dobbs Journal.* He said:
  - Good software design should exhibit:
  - *Firmness:* A program should not have any bugs that inhibit its function.
  - *Commodity:* A program should be suitable for the purposes for which it was intended.
  - *Delight:* The experience of using the program should be pleasurable one.

Analysis Model -> Design Model



设计的目标：质量

协和式飞机

瑞典瓦萨战舰

戴高乐机场

Fundamental Concepts

- Abstraction—data, procedure, control
- Architecture—the overall structure of the software
- Patterns—”conveys the essence” of a proven design solution
- Separation of concerns—any complex problem can be more easily handled if it is subdivided into pieces
- Modularity—compartmentalization of data and function
- Hiding—controlled interfaces
- Functional independence—single-minded function and low coupling
- Refinement—elaboration of detail for all abstractions
- Aspects—a mechanism for understanding how global requirements affect design
- Refactoring—a reorganization technique that simplifies the design
- OO design concepts—Appendix II
- Design Classes—provide design detail that will enable analysis classes to be implemented

Functional Independence

- Functional independence is achieved by developing modules with "single-minded" function and an "aversion" to excessive interaction with other modules.
- *Cohesion* is an indication of the relative functional strength of a module.
  - A cohesive module performs a single task, requiring little interaction with other components in other parts of a program. Stated simply, a cohesive module should (ideally) do just one thing.
- *Coupling* is an indication of the relative interdependence among modules.
  - Coupling depends on the interface complexity between modules, the point at which entry or reference is made to a module, and what data pass across the interface.

模块独立

模块的独立程度可以由两个定性标准度量

- **耦合**

  模块之间的相对独立性的度量

- **内聚**

  模块功能强度的度量

Coupling

耦合性是程序结构中各个模块之间相互关联的度量它取决于各个模块之间接口的复杂程度、调用模块的方式以及那些信息通过接口。

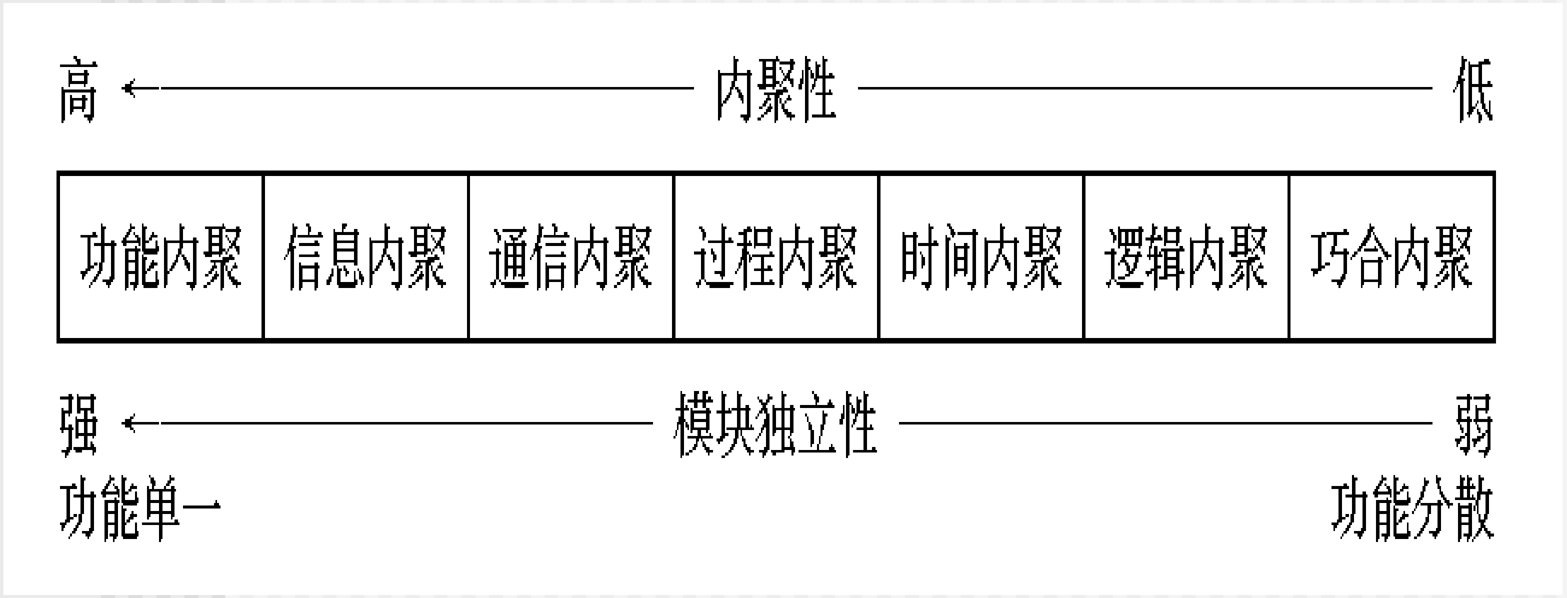
低 $\xrightarrow{\text{耦合性}}$ 高

非直接耦合	数据耦合	标记耦合	控制耦合	外部耦合	公共耦合	内容耦合

强 $\xleftarrow{\text{模块独立性}}$ 弱

Cohesion

- 一个模块内部元素在功能上相互关联的强度



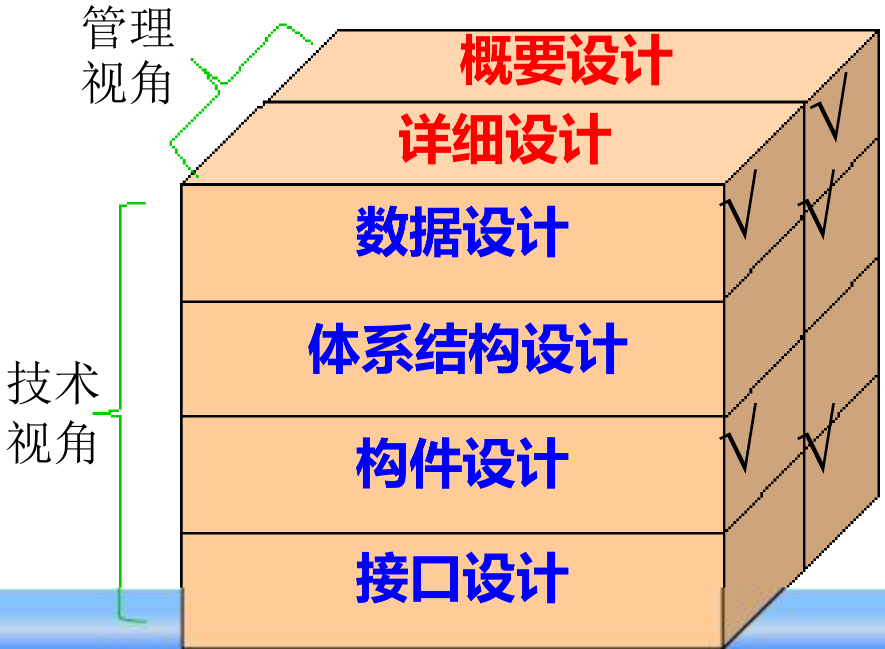
设计目标：高内聚，模块在软件过程中完成单一的任务

Refactoring

- Fowler [FOW99] defines refactoring in the following manner:
  - "Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure.”
- When software is refactored, the existing design is examined for
  - redundancy
  - unused design elements
  - inefficient or unnecessary algorithms
  - poorly constructed or inappropriate data structures
  - or any other design failure that can be corrected to yield a better design.

软件设计的两大阶段

- 从工程管理的角度看，软件设计包括：
  - 概要设计：将软件需求转化为数据结构和软件的系统结构
  - 详细设计：即构件设计，通过对软件结构表示进行细化，得到软件的详细的数据结构和算法



本章小结

- 设计是软件工程技术核心
- 数据结构、体系结构、接口和软件组件的过程细节在设计中逐步细化、开发、评审和记录
- 模块化（包括程序和数据）和抽象概念能够使设计人员简化和重用软件组件
- 细化提供了详细表示各顺序功能层的机制
- 程序和数据结构有助于建立软件架构的整体视图，而过程提供了算法实现必要的细节
- 信息隐藏和功能独立为实现有效模块化提供了启发

Questions

- 1 What is the software design manifesto?
- 2 What are the models of software design?