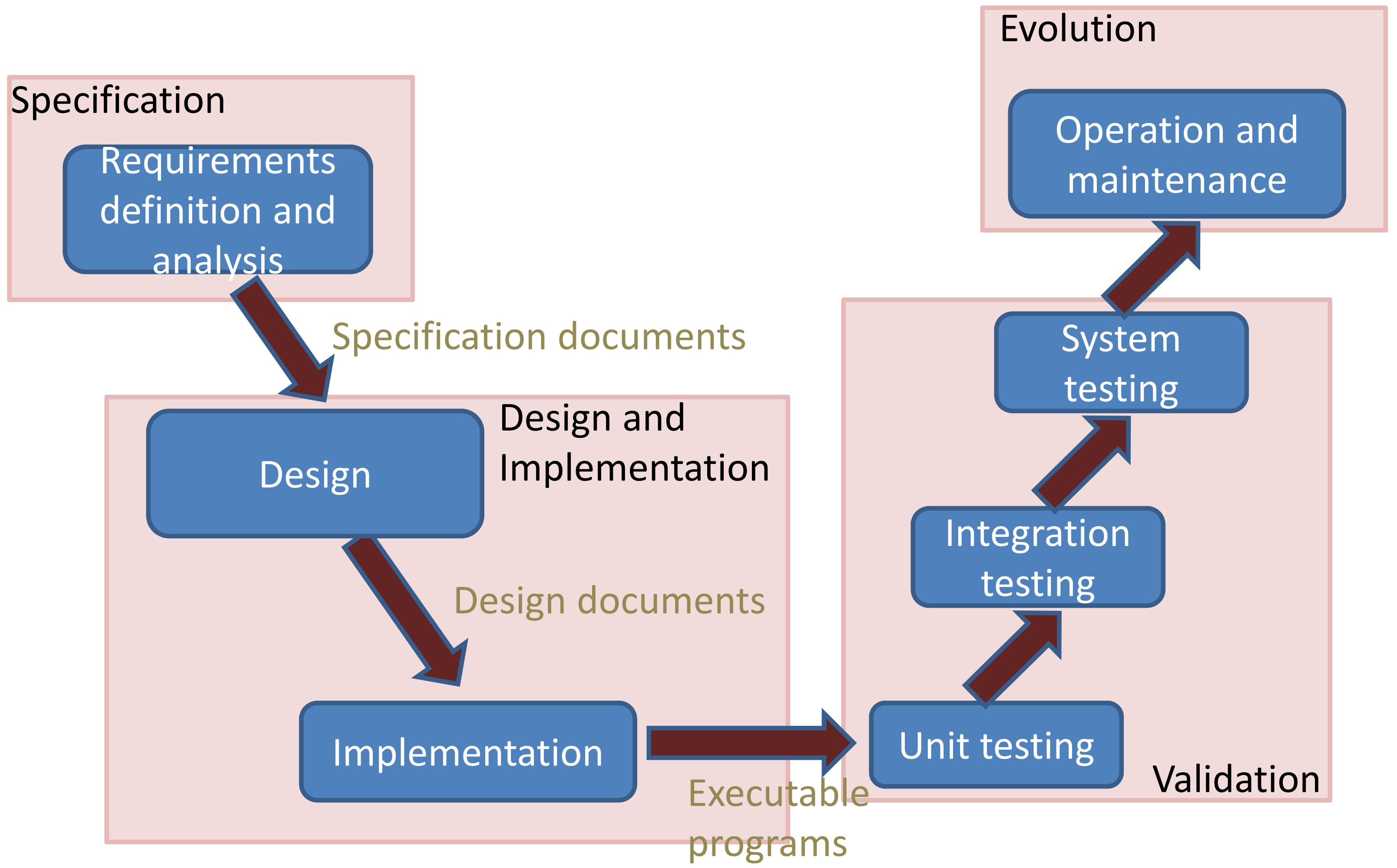
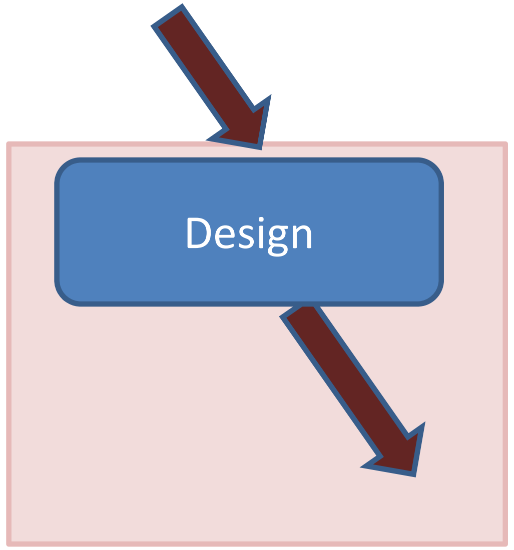
Architectural Design

Xin Feng

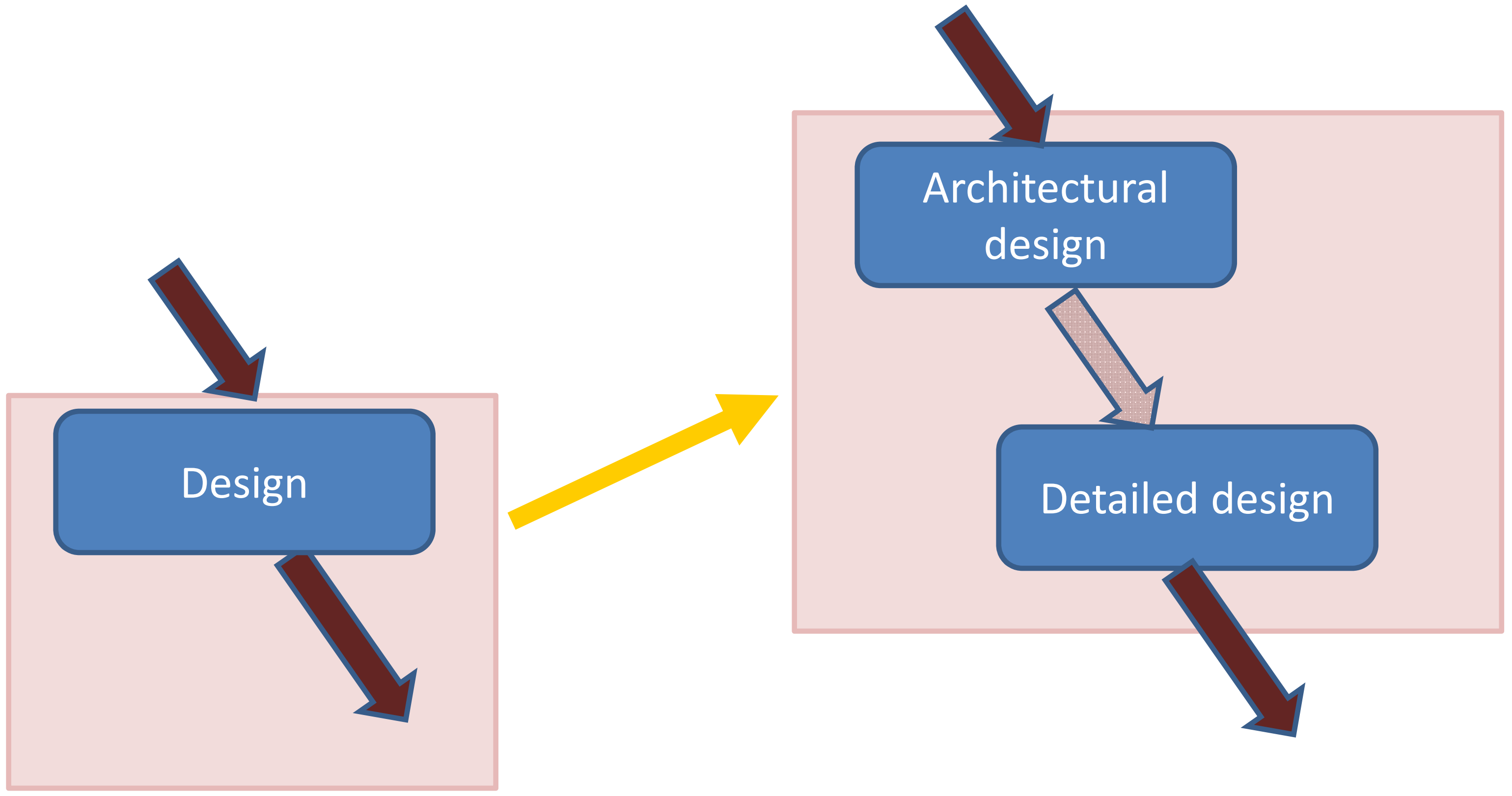
Software Development Life Cycle



Software Development Life Cycle



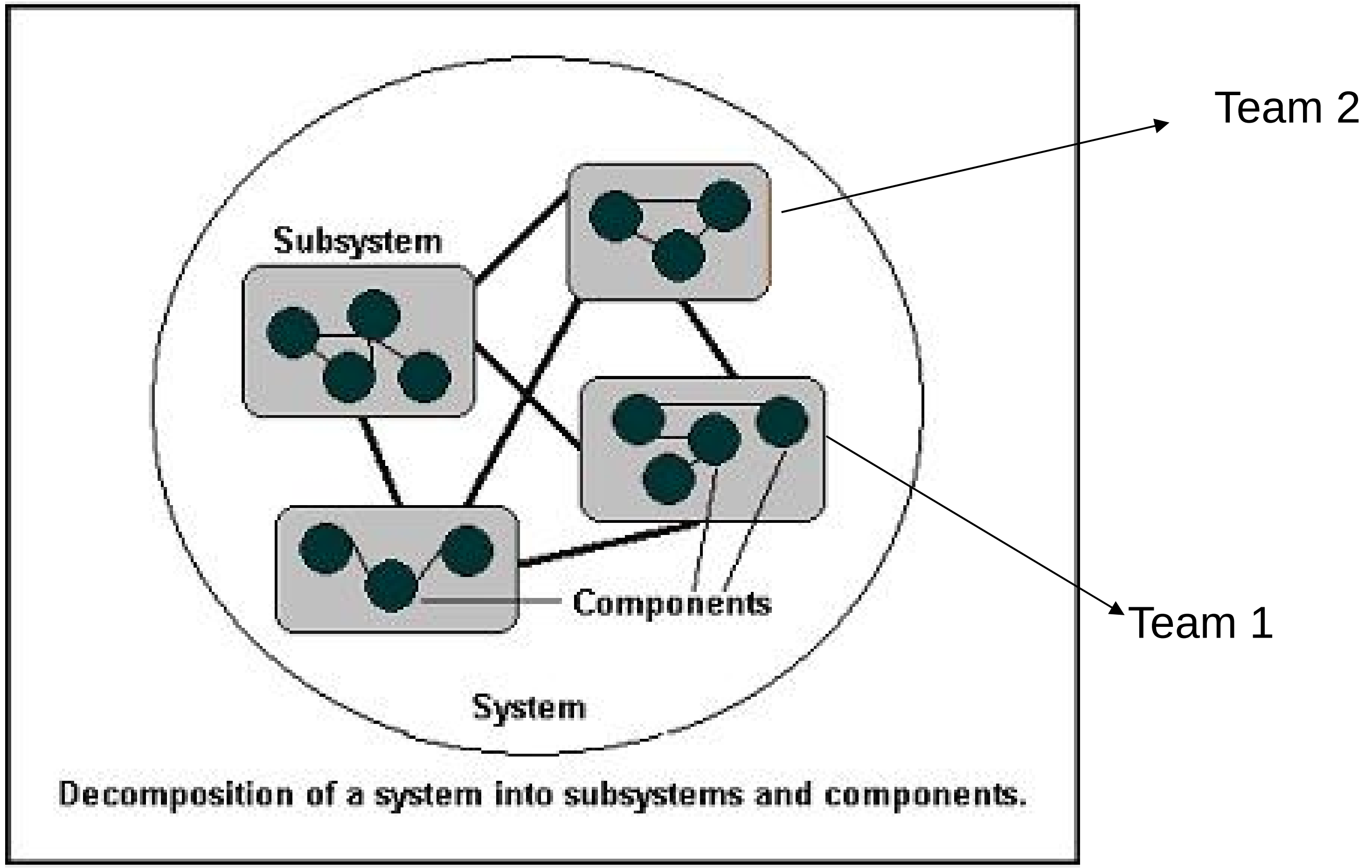
Software Development Life Cycle



Subsystems

- Decompose (分解) system into manageable （可管理的） parts
- Each subsystem can be assigned to a developer or a team and implemented independently
- The individual subsystems can be implemented concurrently （同时）
- A subsystem can be further divided into simpler subsystems
- E.g. Package in Java

System and Subsystem



Decomposition of a system into subsystems and components.

Picture resource:
h:/www.google.com/imgres?imgurl=http://www.stsc.hill.af.mil/crosstalk/2006/09/0609Butler_Fig2.jpg&imgrefurl=http://www.stsc.hill.af.mil/crosstalk/2006/09/0609Butler.html&h=290&w=525&sz=17&tbnid=g2H7MdIixghteM::&tbnh=73&tbnw=132&prev=/images%3Fq%3Dsystem%2Bsubsystem%2Bpicture&hl=en-GB&usg=__pFRPFiEjlZSw5Ufu2GgpxkNOSjg=&ei=5W6SSfXZGtW4kAXlo_H6Cw&sa=X&oi=image_result&resnum=4&ct=image&cd=1

Software Architectural Design

- Identify （标识）subsystems
  – What subsystems
- Establish （建立）relationships between the subsystems
  – How do they communicate

Is a Software Architecture Important

Picture resource: http://images.google.com/images?um=1&hl=en&newwindow=1&rlz=1T4GGLJ_en-GBCN276CN309&q=house+collapse+images+cartoon+structure

Is a Software Architecture Important

- A good software architecture can help
  - communication
    - Developers in different phases can understand
  - system analysis
    - Complexity （复杂度）analysis
  - large scale （规模）reuse
    - The reuse of subsystems

Architectural Design

- A software architectural design is an abstract decomposition of a system into large-grain （大粒）components (构件), each of which may be a substantial system in its own right （本身）.

Design Fundamentals

- Abstraction （抽象）
- Pattern （模板）
- Modularity （模块性）
- Information hiding （信息隐藏）
- Independence （独立）
- Refinement （精炼）
- Refactoring （重构）

Abstraction

- Many levels of abstraction can be posed
  - Procedural abstraction
  - Data abstraction

Book a ticket

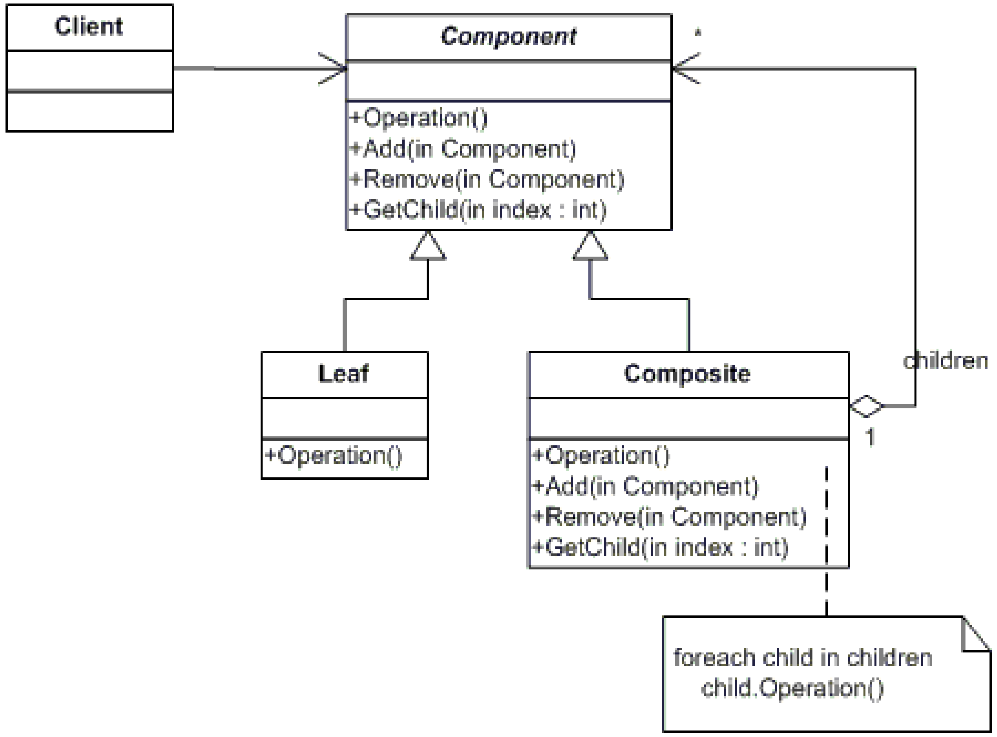
1. Browse a website
2. Search for flight info.
3. Pay the ticket

eTicket

1. Passenger
2. Date
3. Time

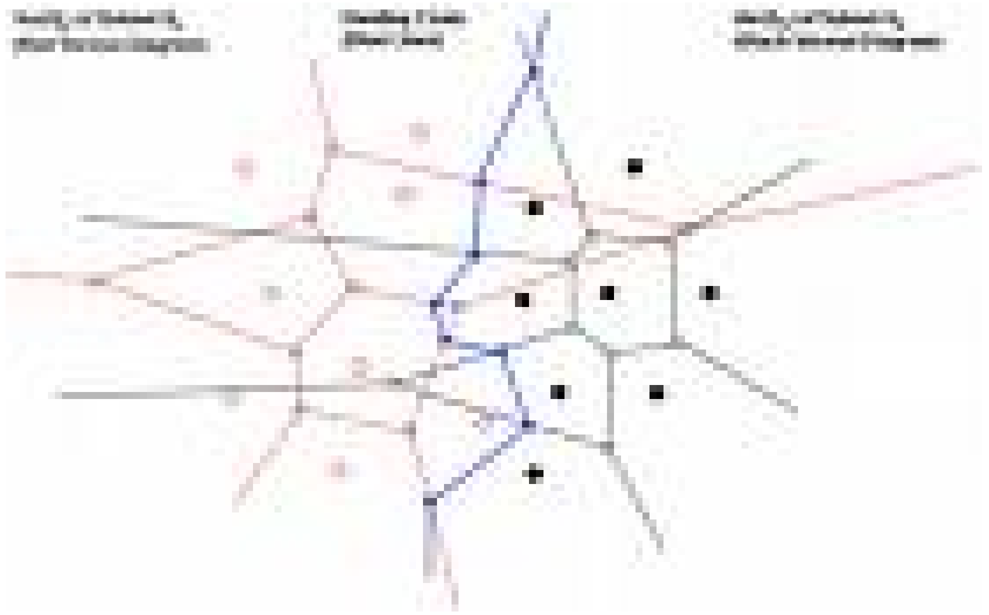
Pattern

- A design pattern （模板）describes a design structure that solves a particular design problem within a specific context (背景)
  - Architectural patterns
  - Design patterns
  - Coding patterns



Modularity

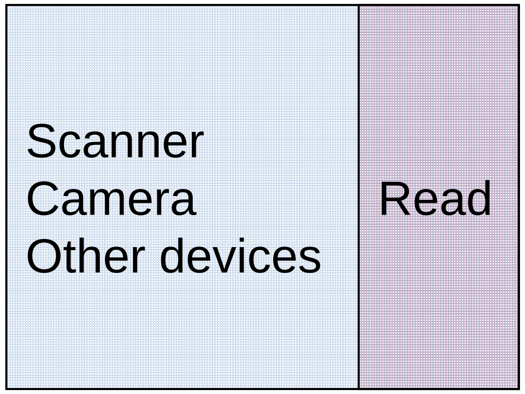
- Modularity is the single attribute of software that allows a program to be intellectually （智力上 ）manageable
  - Divide and conquer strategy （分而治之）
  - Solve the complexity



PictPicture source: http://images.google.com/images?ndsp=20&um=1&hl=en&newwindow=1&rlz=1T4GGLJ_en-GBCN276CN309&q=Complexity+divide+and+conquer+picture&start=40&sa=N

Information Hiding

- Each module is characterized （刻划）by the design decisions which are invisible （不可见的）from outside the module



Independence

- Coupling （耦合）
  - The measure of the dependencies between two subsystems
  - Looser, better
- Cohesion （凝聚）
  - The measure of the dependencies within a system
  - Higher better

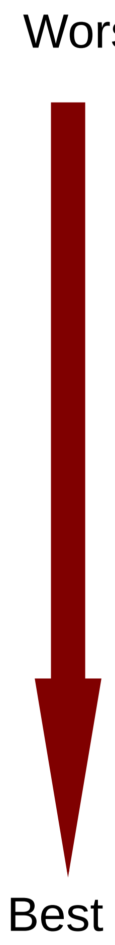
Cohesion

Worst

Coincidental cohesion	parts of a component are grouped arbitrarily
Logical cohesion	parts of a component are grouped because they logically are categorized to do the same thing
Temporal cohesion	parts of a component are grouped by when they are processed
Procedural cohesion	parts of a component are grouped because they follow a certain sequence of execution
Communicational cohesion	parts of a component are grouped because they operate on the same data
Sequential cohesion	parts of a component are grouped because the outputs are from one part
Informational cohesion	parts of a component are grouped they use they use the same data structure
Functional cohesion	parts of a component are grouped because they together perform exactly one function

Best

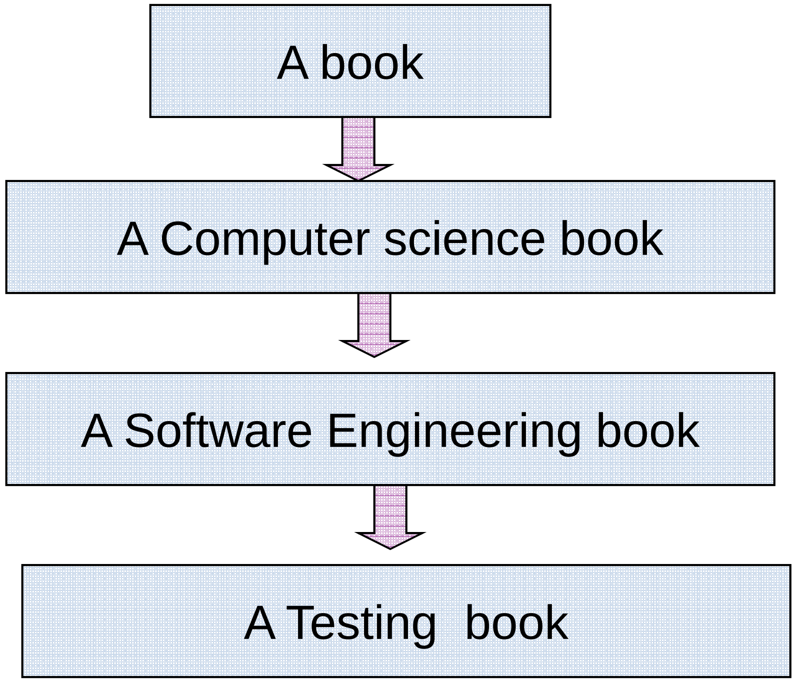
Coupling



Content Coupling	A component uses/alters (改变) data in another
Control Coupling	Two components communicating with a control flag （标志）
Global-data Coupling	Two components communicate via global （总） data
Data-structure Coupling	Communicating via a data structure passed as a parameter
Data Coupling	Communicating via parameter passing

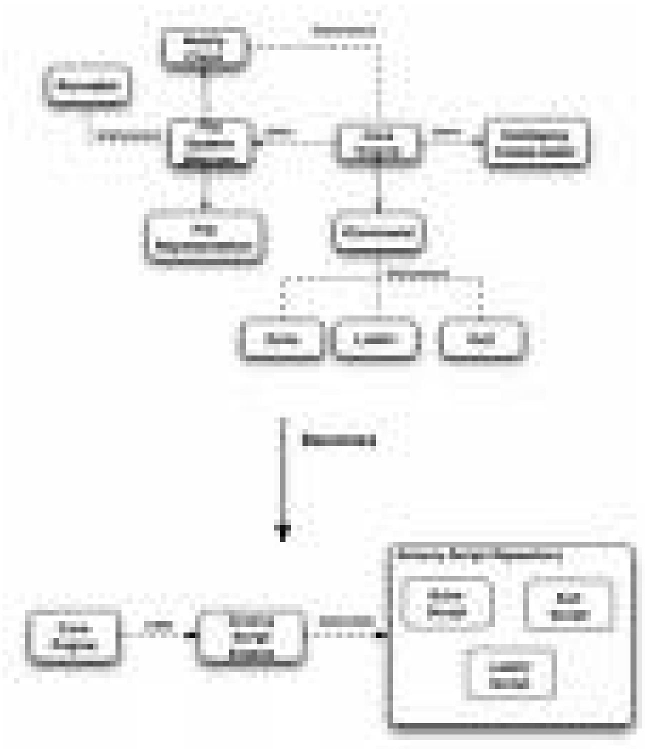
Refinement

- Stepwise （逐步）refinement （精化）is a process of elaboration (阐述). It is opposite to abstraction.



Refactoring

- It is a method that reorganizes （重组） the architecture to make the design better.



Picture source: http://images.google.com/imgres?imgurl=http://www.pabrantes.net/blog/space/start/2007-12-10/1/refactoring.png&imgrefurl=http://www.pabrantes.net/blog/space/start/2007-12-10/1&usgJZGyYh6xxJeBT2Gd7PqETejM0o8=&h=549&w=480&sz=50&hl=en&start=20&um=1&tbnid=5PopuhRVAlDaRM:&tbnh=133&tbnw=116&prev=/images%3Fq%3Drefactoring%2Bpicture%26um%3D1%26hl%3Den%26newwindow%3D1%26rlz%3D1T4GGLJ_en-GBCN276CN309%26sa%3DX

Questions to Answer in Architectural Design

1. How many processors?
2. What templates （样板） available?
3. What architectures are to be established?
4. How to decompose a system into subsystems?
5. How to decompose a subsystem into modules?
6. How are the modules controlled?
7. How good is the architecture?

Architectures from Different Perspectives(观察角度）

- Static model
  - Subsystems or components
- Dynamic model
  - Processes in run time
- Interface model
  - Services offered by each subsystem
- Relationship model
  - Calling relationships
- Distribution model
  - Distribution (分布) on different computers

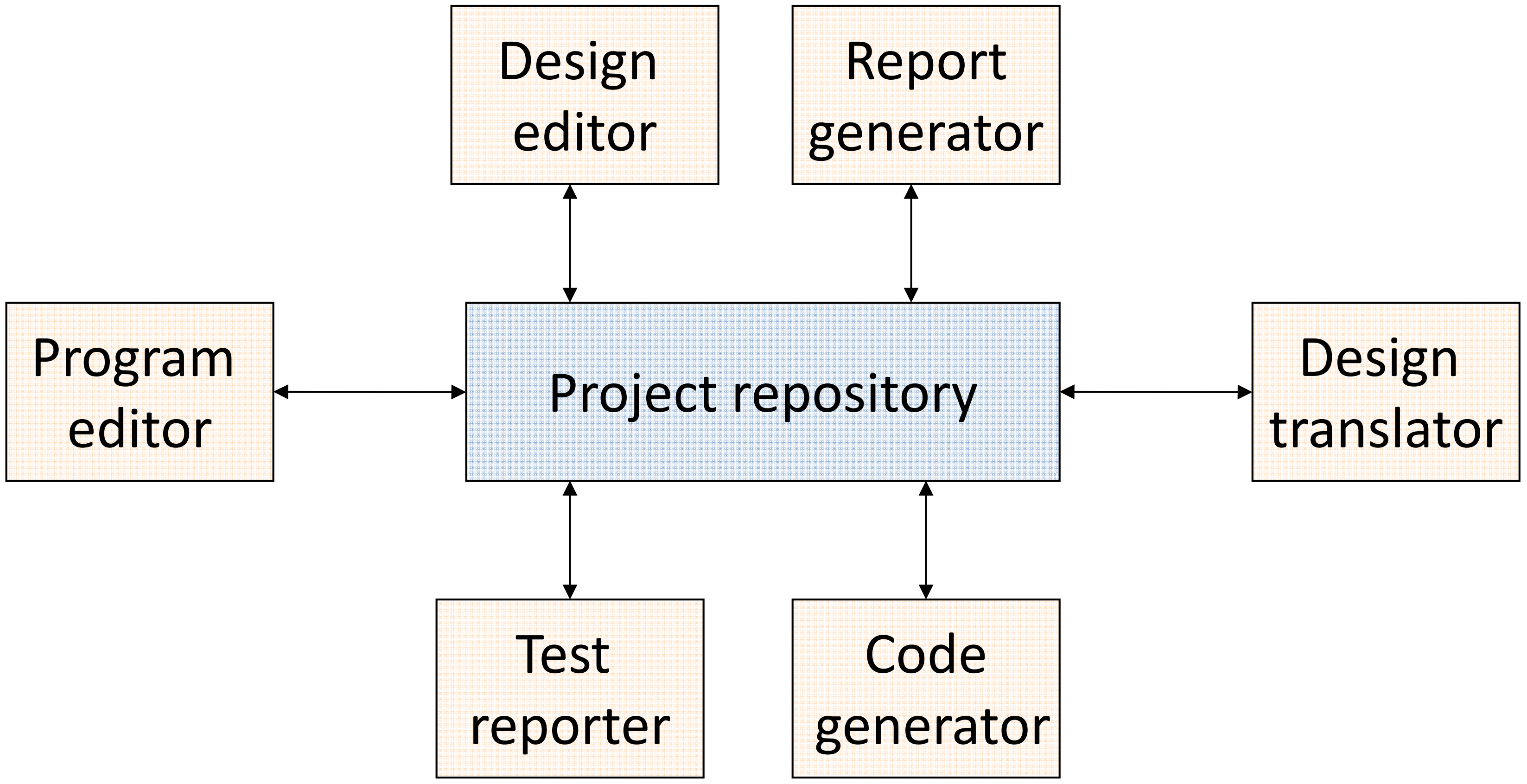
System Organization

- Three models can be used to decompose a system into subsystems
  - The repository （储存库 ） model
    - Repository is shared by the subsystems
  - The client server model
    - The services offered by servers can be used by clients through network
  - The layered model
    - The upper layers use the lower layers through defined standard interface

System Organization

- Three models can be used to decompose a system to subsystems
  - The repository model
    - Repository is shared by the subsystems
  - The client server model
    - The services offered by servers can be used by clients through network
  - The layered model
    - The upper layers use the lower layers through defined standard interface

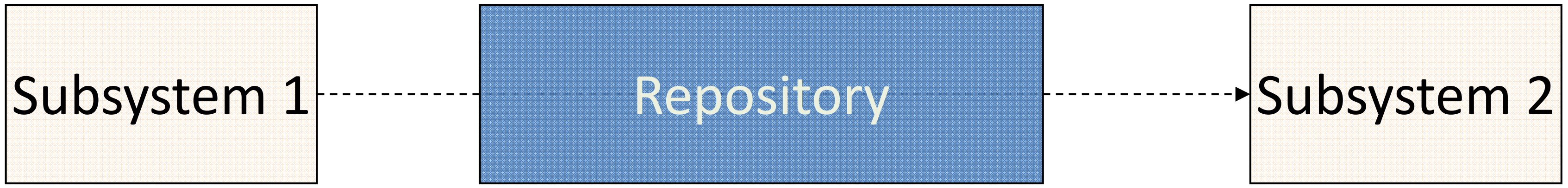
The Repository Model – An Example



CASE Toolset Architecture

Advantages and Disadvantages

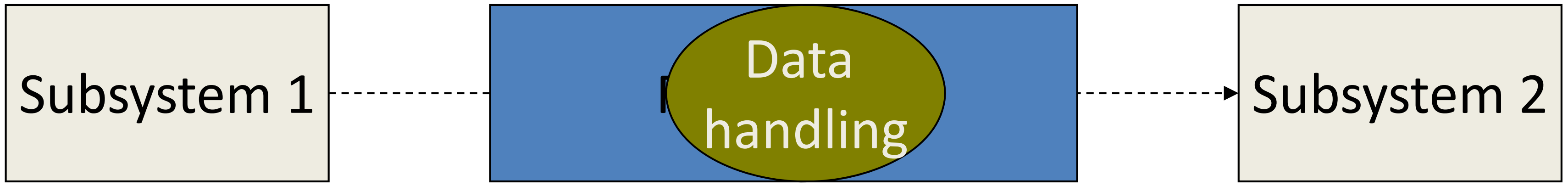
No direct transmission (传递) between two subsystems



The subsystems must follow the data model defined by repository

Advantages and Disadvantages

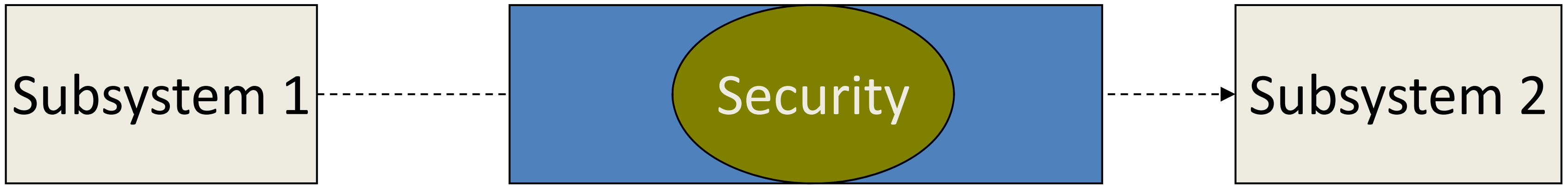
The information handling is centralized



The changes in repository can affect many subsystems

Advantages and Disadvantages

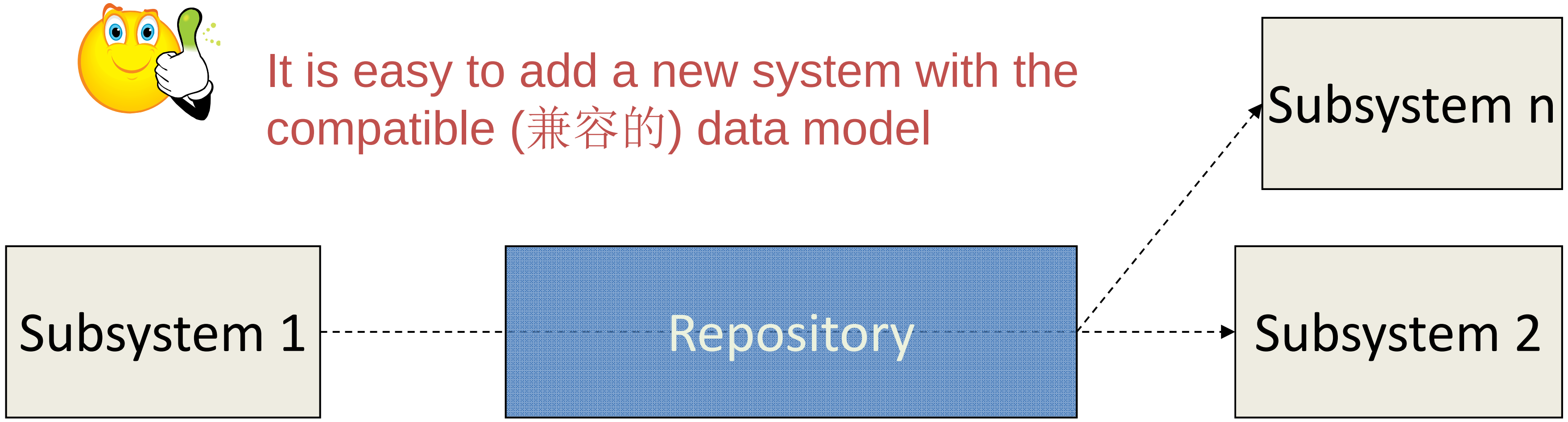
The repository manager is responsible



Subsystem cannot have their own policies

Advantages and Disadvantages

It is easy to add a new system with the compatible (兼容的) data model

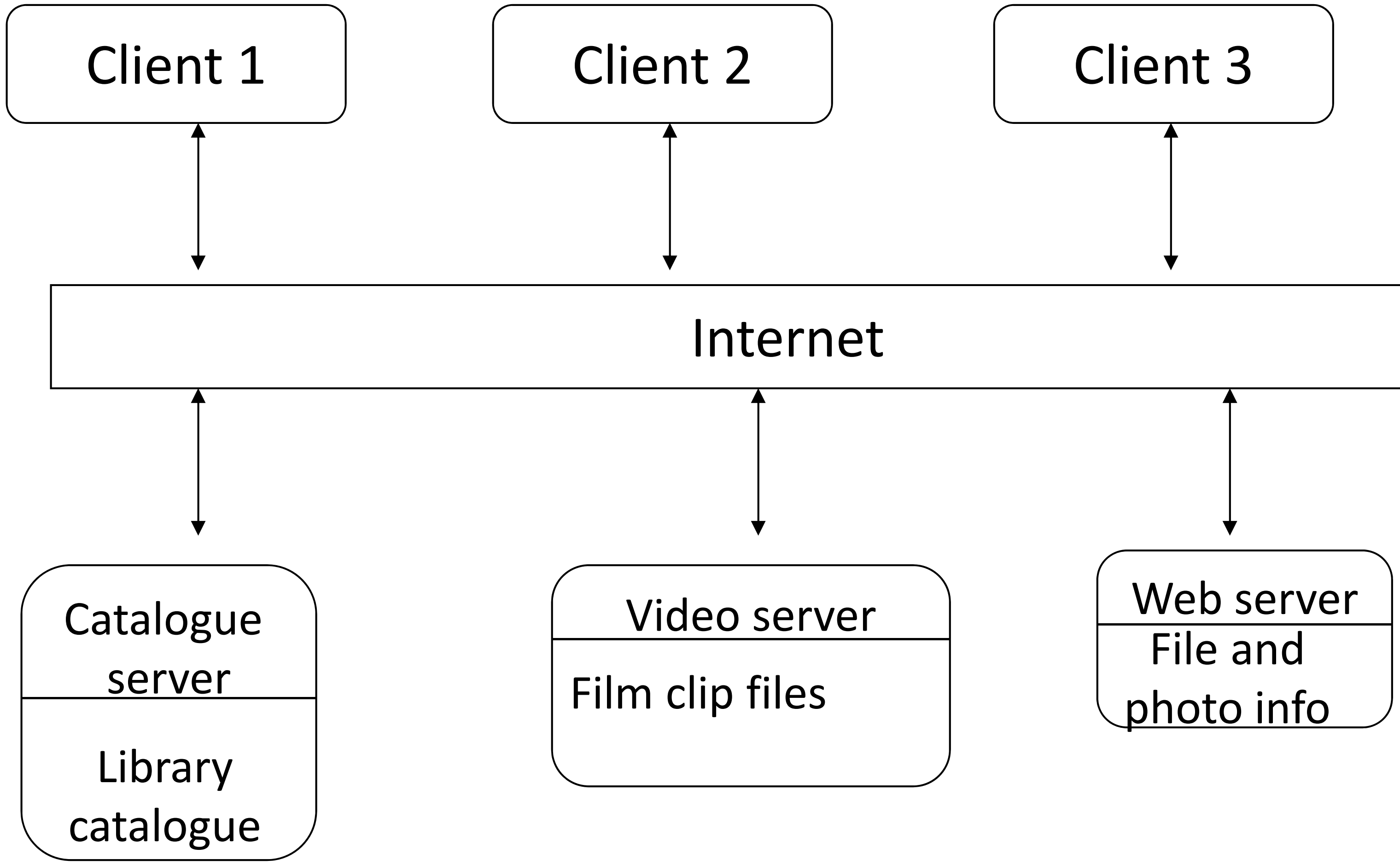


It is hard to distribute the repository to a number of machines

System Organization

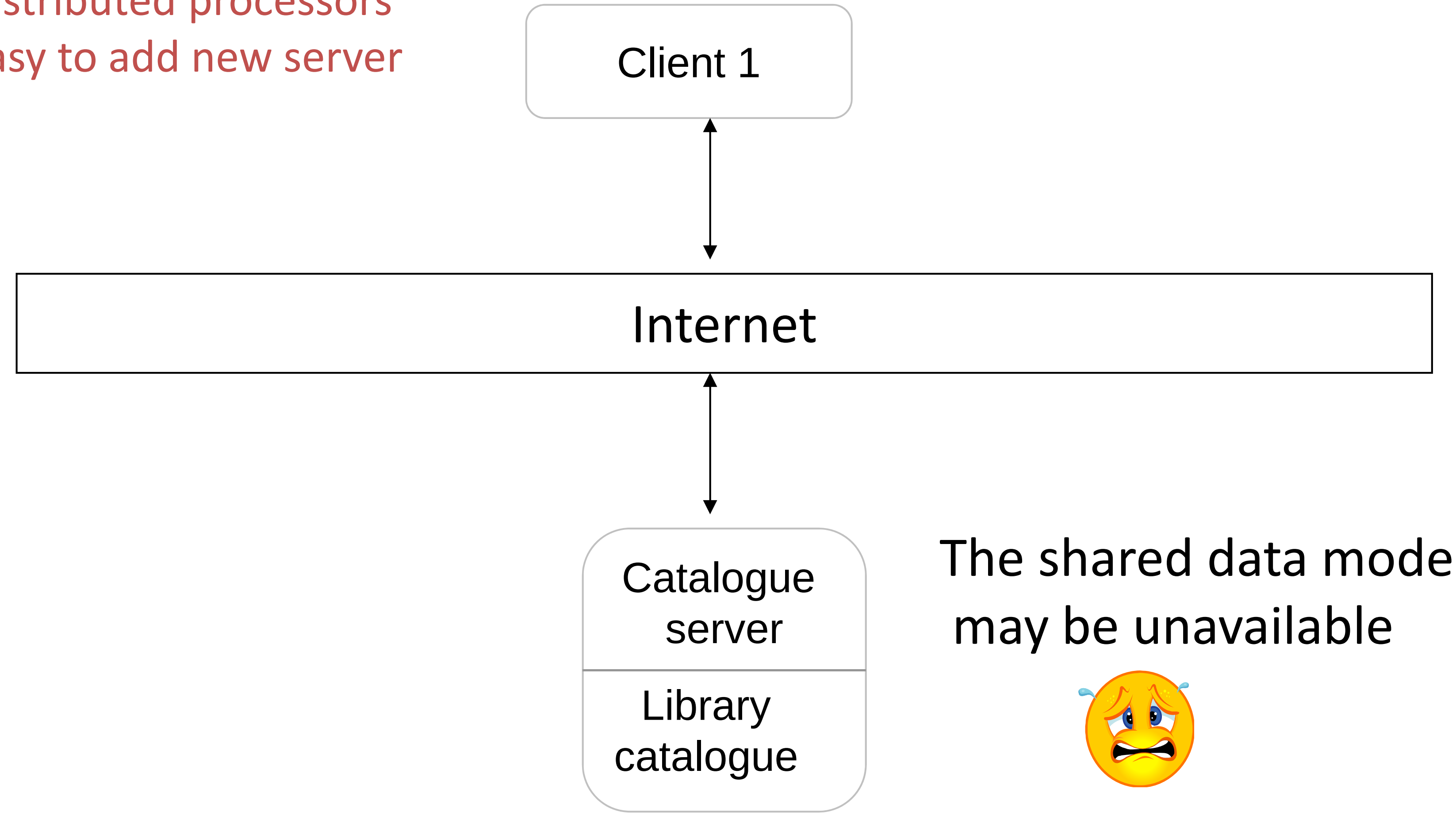
- Three models can be used to decompose a system to subsystems
  - The repository model
    - Repository is shared by the subsystems
  - The client server model
    - The services offered by servers can be used by clients through network
  - The layered model
    - The upper layers use the lower layers through defined standard interface

The Client Server Model – An Example



Advantages and Disadvantages

Distributed processors
Easy to add new server



The shared data model may be unavailable

System Organization

- Three models can be used to decompose a system to subsystems
  - The repository model
    - Repository is shared by the subsystems
  - The client server model
    - The services offered by servers can be used by clients through network
  - The layered model
    - The upper layers use the lower layers through the defined standard interfaces

The Layered model – An Example

Configuration （配置） management system layer

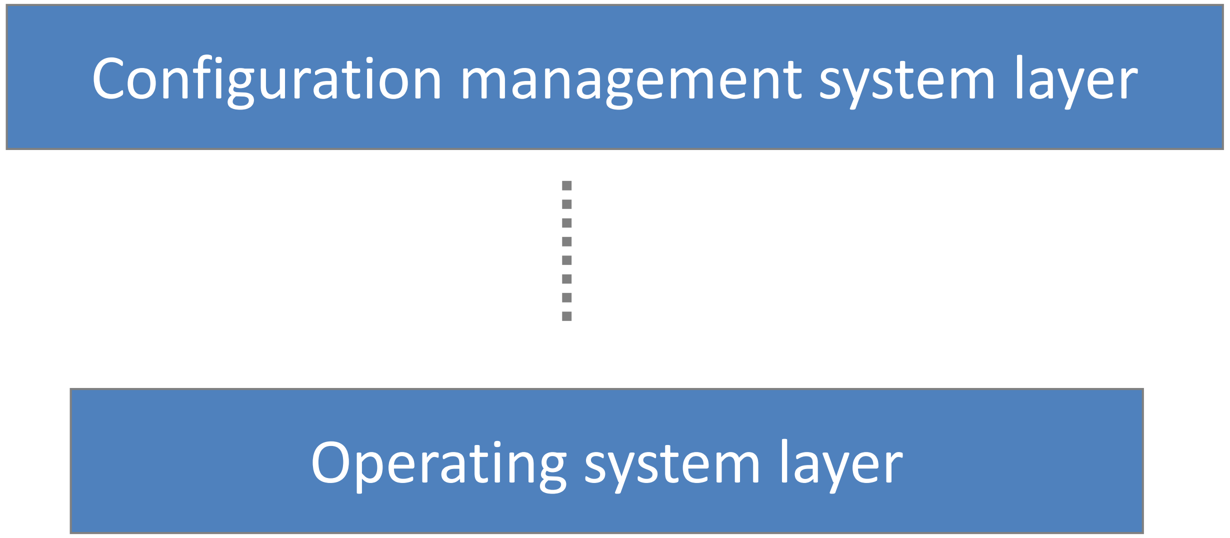
Object management system layer

Database system layer

Operating system layer

Advantages and Disadvantages

Support incremental process model



Hard to structure into layers

Advantages and Disadvantages

A layer can be replaced by another one if the interface is not changed

Configuration management system layer

Another operating system

Performance can be a problem

Component Control

- How to use a component’s service
  - Centralized （集中式）control
    - One component has the overall （全面的）control of start or stop of other components
  - Event-based （基于事件的）control
    - Each component can response to the external events

Centralized Control

- The call-return models （调用-返回）
  - Tree structure diagrams
- The manager models （中间协调）
  - A component coordinates other components in a concurrent system

Event-based Control

- Broadcast models （广播）
  - An event is broadcasted to all the components
- Interrupt-driven models （中断驱动）
  - An interrupt handler detects the external interrupts and passes them to the corresponding components.