西安电子科技大学
XIDIAN UNIVERSITY

计算机科学与技术学院
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY
国家示范性软件学院
NATIONAL PILOT SCHOOL OF SOFTWARE ENGINEERING

数据流体系结构风格

Data Flow Style

主讲人：蔺一帅 讲师

本节内容

1. 软件体系结构风格
2. 数据流体系结构风格
3. 批处理体系结构风格
4. 管道-过滤器体系结构风格

软件体系结构风格-从“建筑风格”开始

哥特式 源于12世纪的法国，并持续到16世纪

- 强调**垂直高度**和**石骨架结构**
  - ✓优美的彩色玻璃窗画
  - ✓直刺苍穹的尖顶
  - ✓丛生的圆柱
  - ✓飞扶壁
  - ✓有龙骨的拱顶
  - ✓尖顶形状的圆弧拱门
  - ✓强调雕塑的细节

软件体系结构风格-从“建筑风格”开始

文艺复兴式 发源于意大利弗洛伦萨，借鉴和发扬了古希腊和古罗马建筑思想 。

- 结构清晰、规则，遵循简单的形状和数学比例
- 简单的圆柱、对称，区别于日后不规则的、复杂的多面建筑
- 古典风格的圆柱、完美的几何设计和半球形的屋顶

软件体系结构风格-从“建筑风格”开始

Architectural style constitutes a mode of classifying architecture largely by morphological characteristics in terms of form, techniques, materials, etc. (**建筑风格**等同于建筑体系结构的一种可分类的模式，通过诸如外形、技术和材料等形态上的特征加以区分)。

“风格”——经过长时间的实践，被证明具有良好的工艺可行性、性能与实用性，并可直接用来**遵循与模仿 (复用)**。

软件体系结构风格-回到“软件体系结构风格”

A software architecture style (软件体系结构风格)

- describes a class of architectures（描述一类体系结构）
- independent on the problems（独立于实际问题，强调了软件系统中通用的组织结构）
- found repeatedly in practice（在实践中被多次应用）
- a package of design decisions（是若干设计思想的综合）
- has known properties that permit reuse（具有已经被熟知的特性，并且可以复用）

软件体系结构风格-定义

描述特定领域中软件系统家族的**组织方式的惯用模式**(idiomatic paradigm)，反映了领域中众多系统所共有的**结构和语义特性**，并指导如何将各个模块和子系统有效地组织成一个完整的系统。

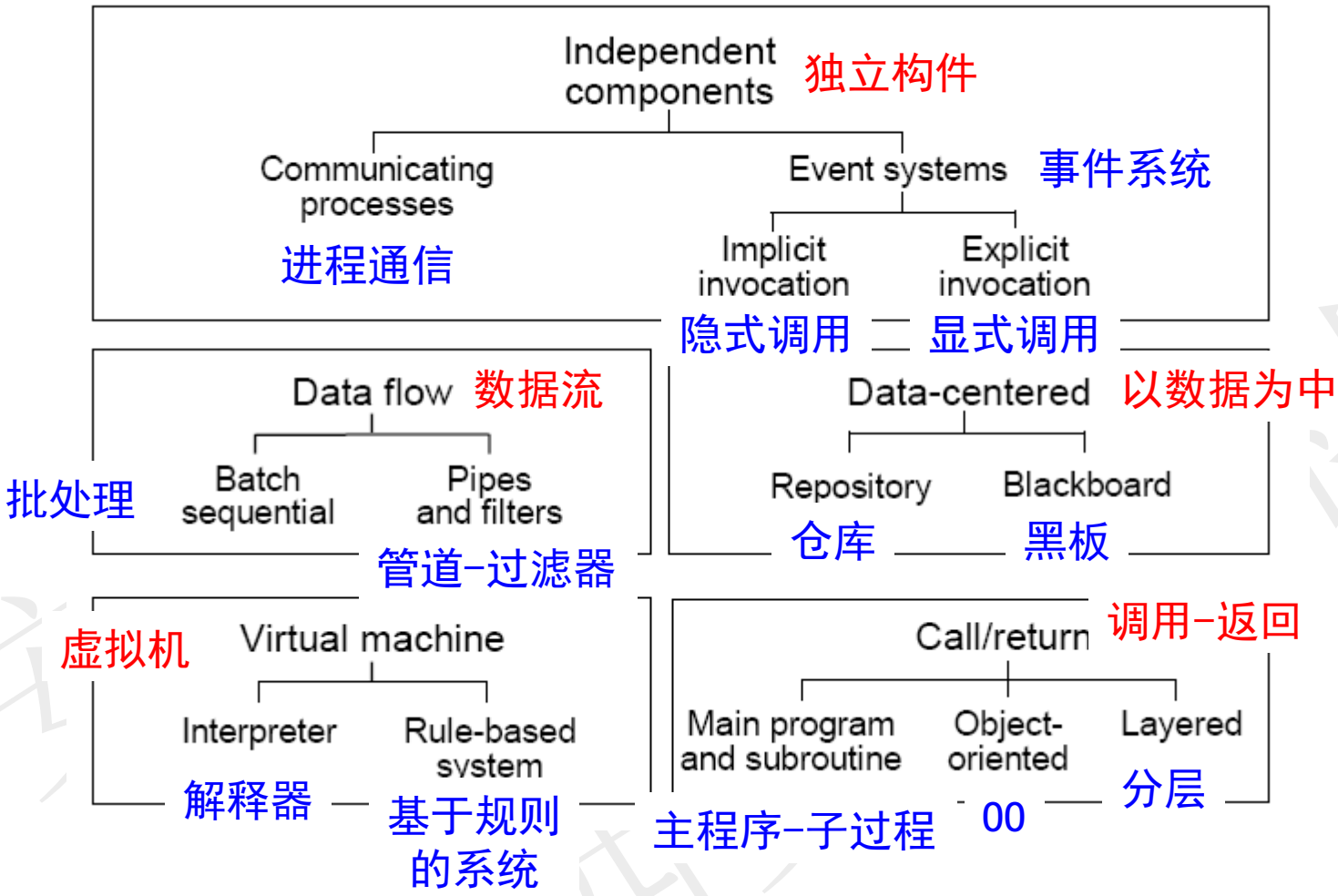
Architecture style ={

Component/Connector vocabulary,

Topology, Semantic Constraints

}

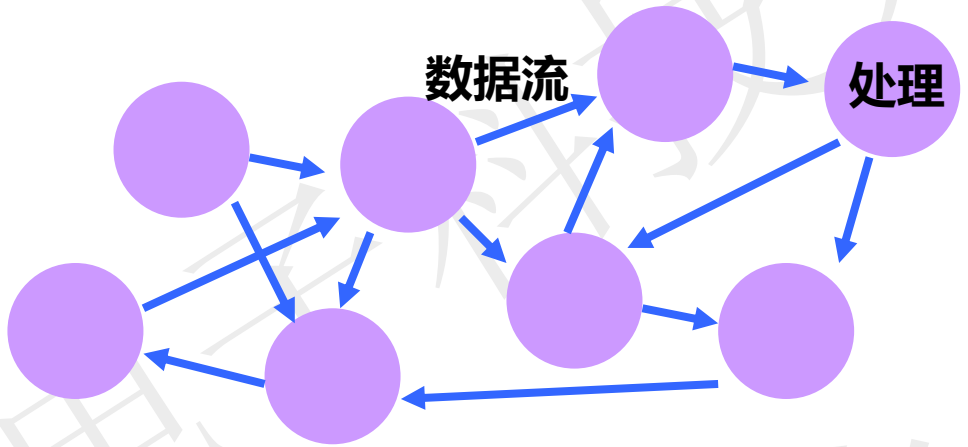
软件体系结构风格-Taxonomy of styles



本节内容

1. 软件体系结构风格
2. 数据流体系结构风格
3. 批处理体系结构风格
4. 管道-过滤器体系结构风格

数据流体系结构风格-直观理解



处理：**数据到达即被激活，**

无数据时不工作

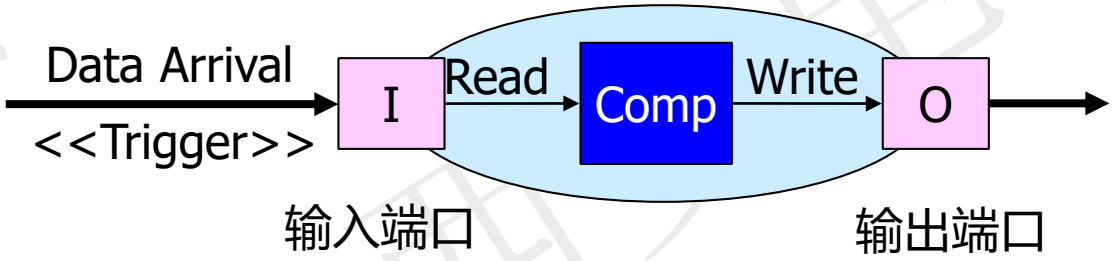
一个直观实例：在MS Excel中，改变某个单元格的值，则依赖于该单元格的其他单元格的值也会随之改变.

数据流体系结构风格-特征

- A data flow system is one in which
  - the availability of data controls the computation (数据的可用性决定着处理<计算单元>是否执行)
  - the structure of the design is decided by orderly motion of data from process to process (系统结构由数据在各处理之间的有序移动决定)
  - in a pure data flow system, there is no other interaction between processes (在纯数据流系统中，处理之间除了数据交换没有任何其他的交互)

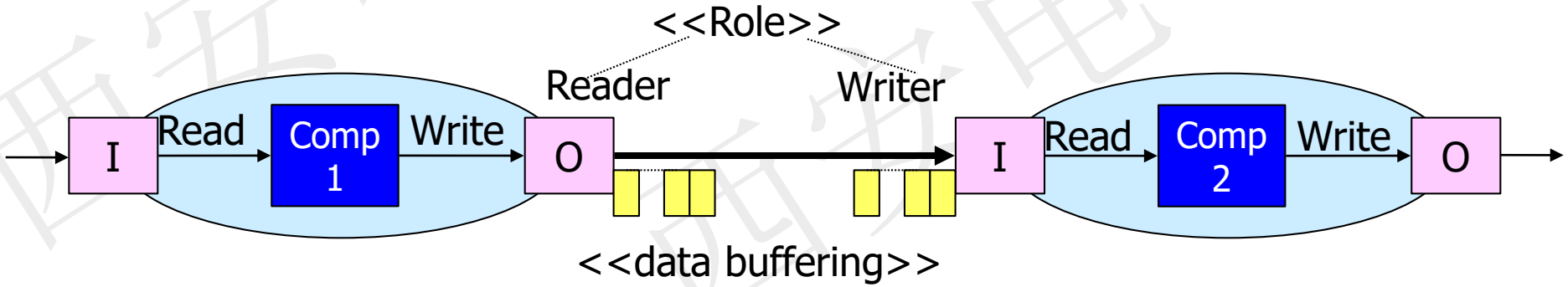
数据流体系结构风格-基本构件

- Components: data processing components(基本构件：数据处理)
  - Interfaces are input ports and output ports (构件接口：输入端口和输出端口)，Input ports read data; output ports write data
  - Computational model: read data from input ports, compute, write data to output ports (计算模型：从输入端口读数，经过计算/处理，然后写到输出端口)

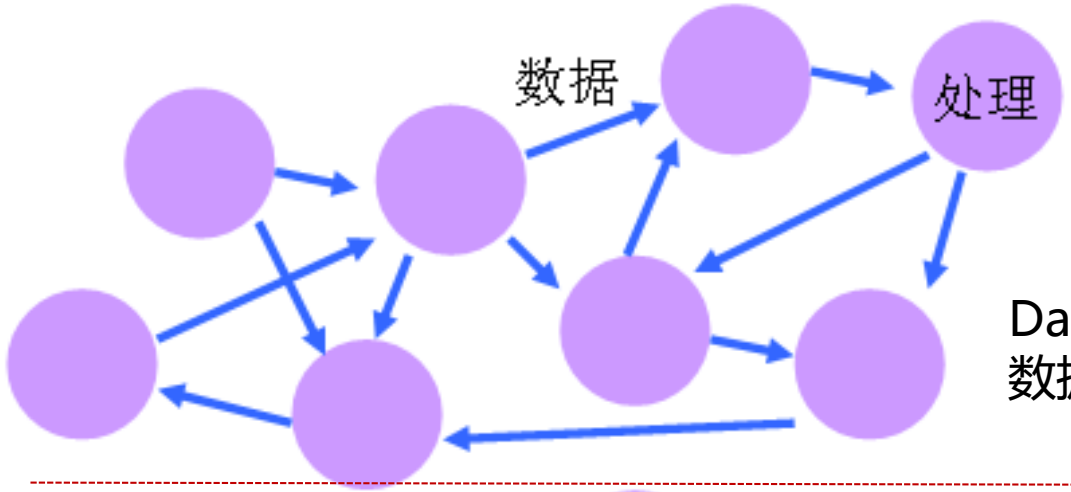


数据流体系结构风格-连接件

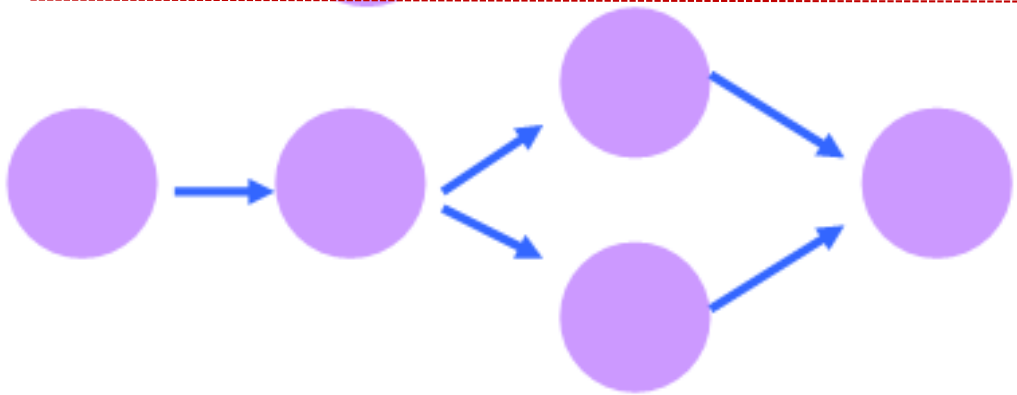
- Connectors: data flow (stream) (连接件：数据流)
  - ✓ Unidirectional, usually asynchronous, buffered (单向、通常是异步、有缓冲)
  - ✓ Interfaces are reader and writer roles (接口角色：reader和writer)
  - ✓ Computational model (计算模型: 把数据从一个处理的输出端口传送到另一个处理的输入端口)



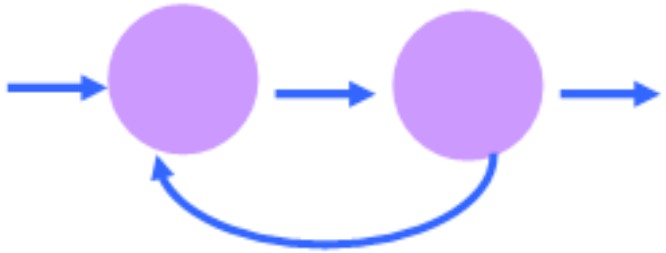
数据流体系结构风格-Patterns of Data Flow in Systems



Data can flow in free patterns
数据自由流动

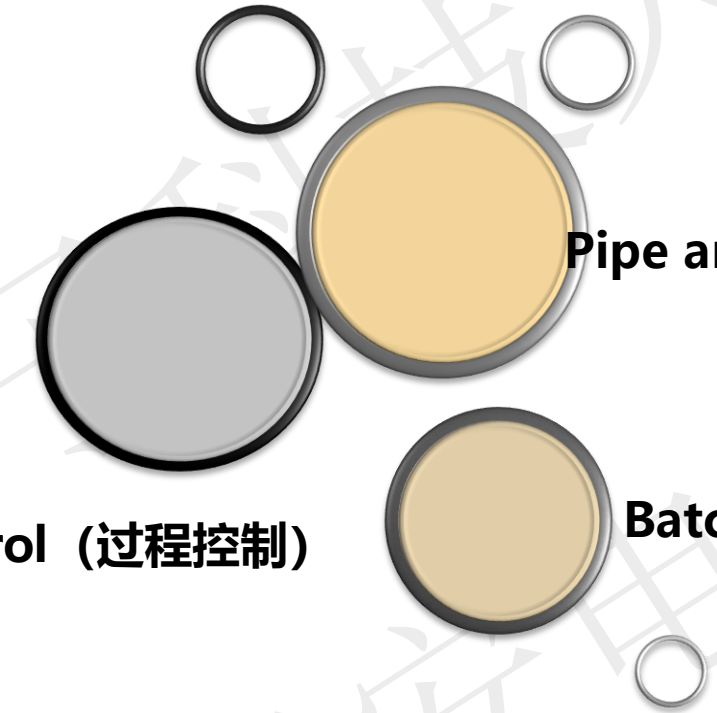


nearly linear data flow systems
近似线性数据流 √



highly constrained cyclic structures
在限度内的循环数据流 √

数据流体系结构风格-典型数据流风格



Pipe and Filter（管道-过滤器）

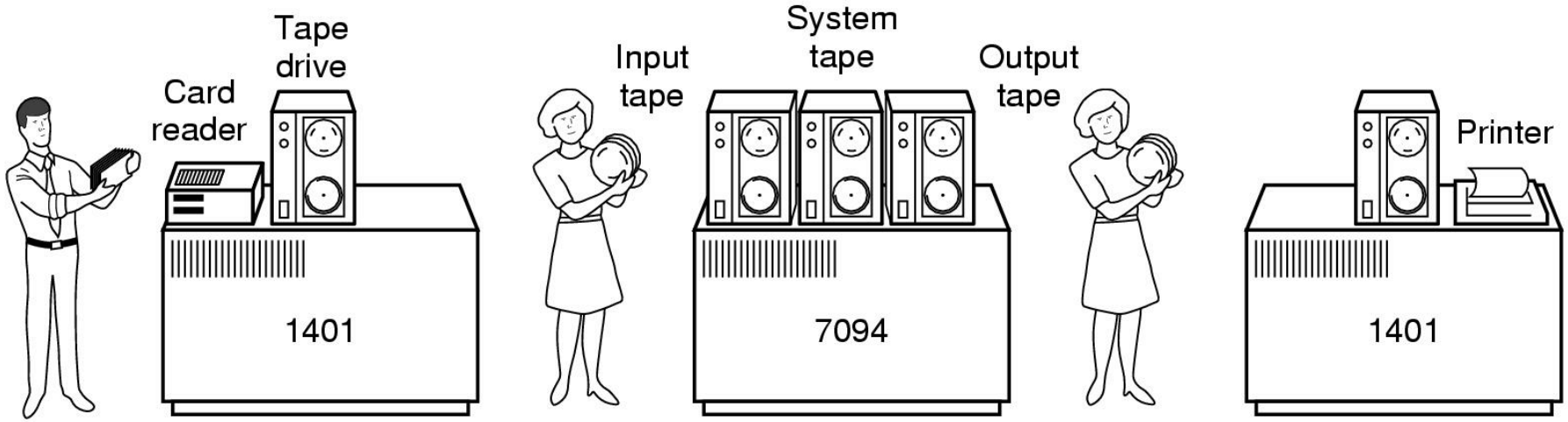
Process Control（过程控制）

Batch Sequential（批处理）

本节内容

1. 软件体系结构风格
2. 数据流体系结构风格
3. 批处理体系结构风格
4. 管道-过滤器体系结构风格

批处理体系结构风格-直观结构（1//2）

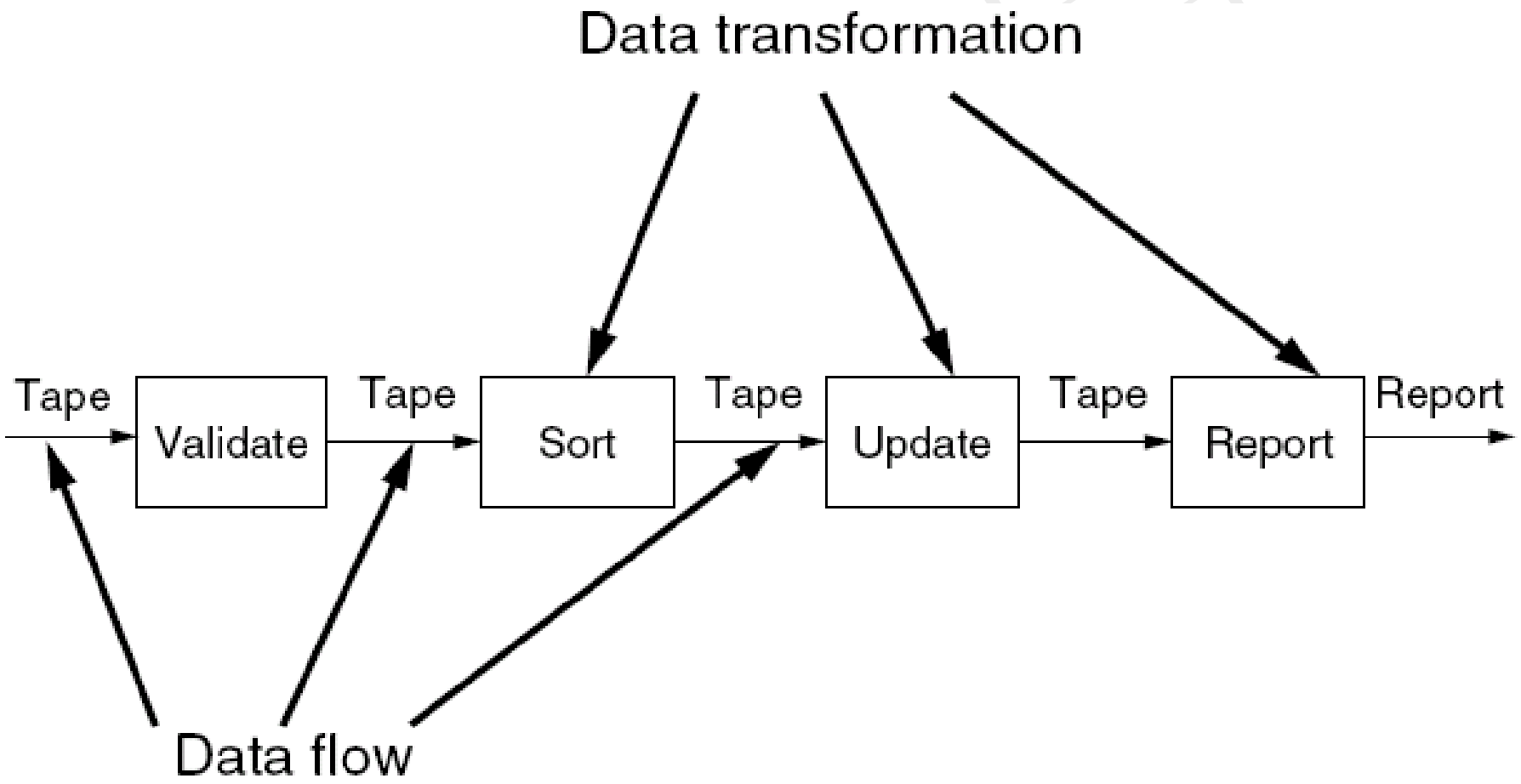


将用户输入的纸带上的数据写入磁带

将磁带作为计算设备的输入，进行计算，得到输出结果

打印计算结果

批处理体系结构风格-直观结构（2//2）



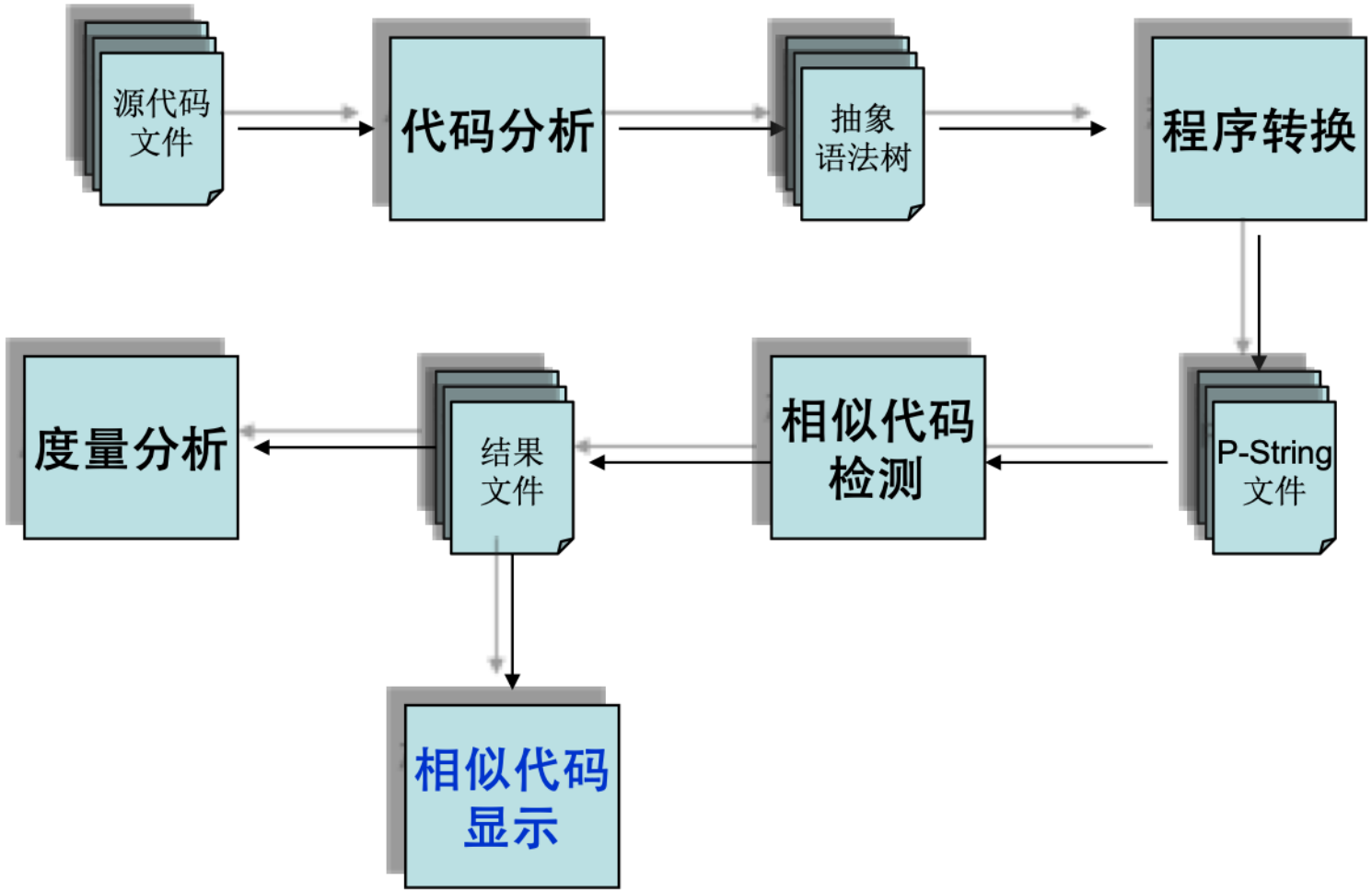
批处理体系结构风格-定义（1/2）

- Components (processing steps) are independent programs（基本构件：独立的应用程序)
- Connectors are some type of media - traditionally tape (连接件：某种类型的媒质)
- Topology: Connectors define data flow graph (连接件定义了相应的数据流图，表达拓扑结构)

批处理体系结构风格-定义（2/2）

- Processing steps are independent programs（每个处理步骤是一个独立的程序）
- Each step runs to completion before next step starts（每一步必须在前一步结束后才能开始）
- Data transmitted as a whole between steps（数据必须是完整的，以整体的方式传递）

批处理体系结构风格-应用实例（1/2）

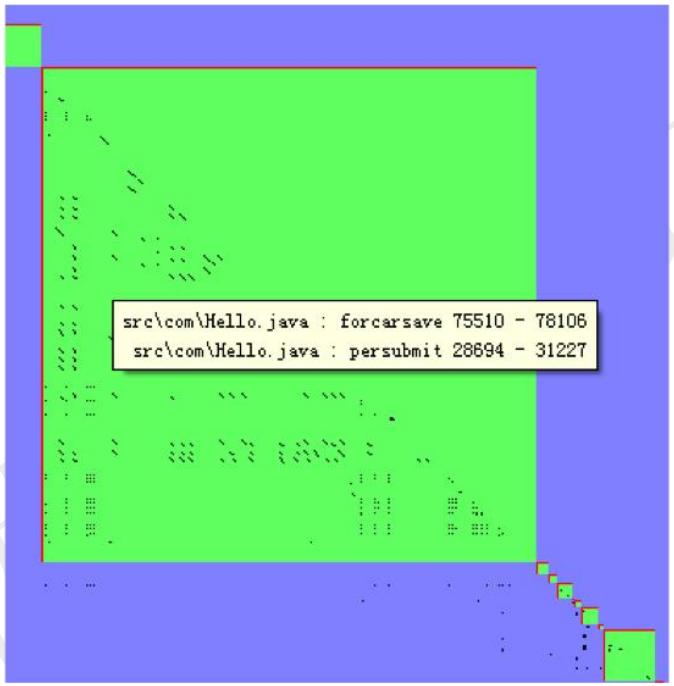


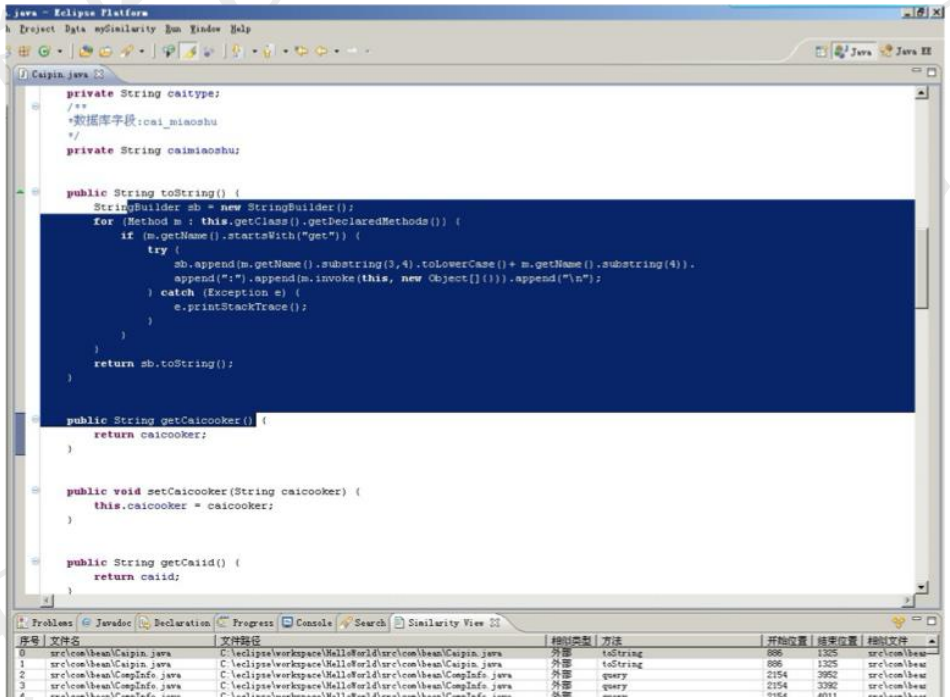
批处理体系结构风格实例：基于Eclipse的代码重复检测工具

批处理体系结构风格-应用实例（2/2）

Problems | Javadoc | Declaration | Progress | Console | Search | Similarity View

序号	文件名	文件路径	相似类型	方法	开始位置	结束位置	相似文件
0	src\com\bean\Caipin.java	C:\eclipse\workspace\HelloWorld\src\com\bean\Caipin.java	外部	toString	886	1325	src\com\bear
1	src\com\bean\Caipin.java	C:\eclipse\workspace\HelloWorld\src\com\bean\Caipin.java	外部	toString	886	1325	src\com\bear
2	src\com\bean\CompInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\CompInfo.java	外部	query	2154	3952	src\com\bear
3	src\com\bean\CompInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\CompInfo.java	外部	query	2154	3392	src\com\bear
4	src\com\bean\CompInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\CompInfo.java	外部	query	2154	4011	src\com\bear
5	src\com\bean\CompInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\CompInfo.java	外部	query	2154	3392	src\com\bear
6	src\com\bean\CompInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\CompInfo.java	外部	query	3462	4064	src\com\Hell
7	src\com\bean\RescateInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\RescateInfo.java	外部	query	1641	2869	src\com\bear
8	src\com\bean\RescateInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\RescateInfo.java	外部	query	1641	3433	src\com\bear
9	src\com\bean\RescateInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\RescateInfo.java	外部	query	1641	2869	src\com\bear
10	src\com\bean\RescateInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\RescateInfo.java	外部	query	1641	2184	src\com\Hell
11	src\com\bean\RescateInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\RescateInfo.java	外部	query	1641	2184	src\com\Hell
12	src\com\bean\RescateInfo.java	C:\eclipse\workspace\HelloWorld\src\com\bean\RescateInfo.java	外部	query	1641	2184	src\com\Hell





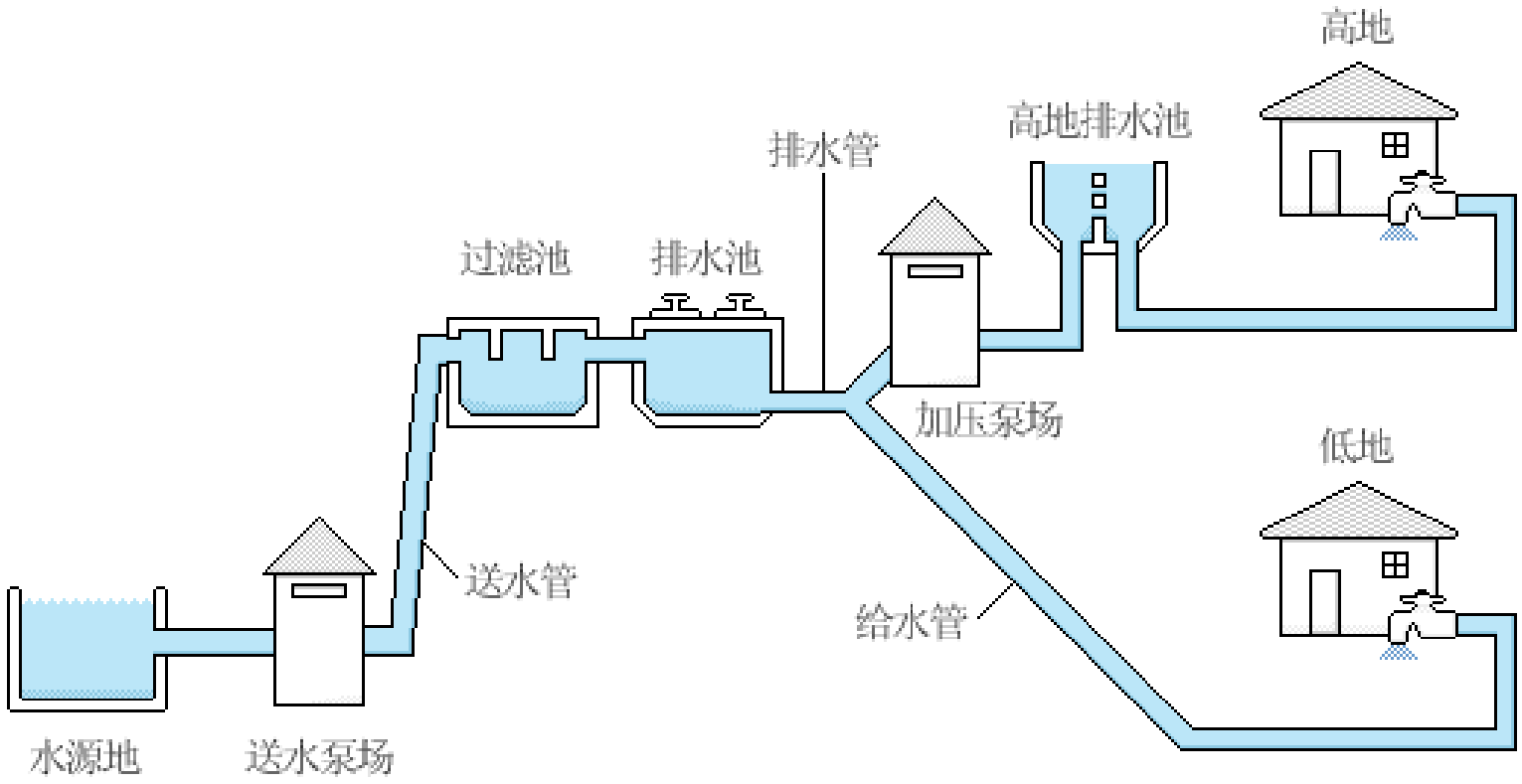
基于Eclipse的代码重复检测工具

本节内容

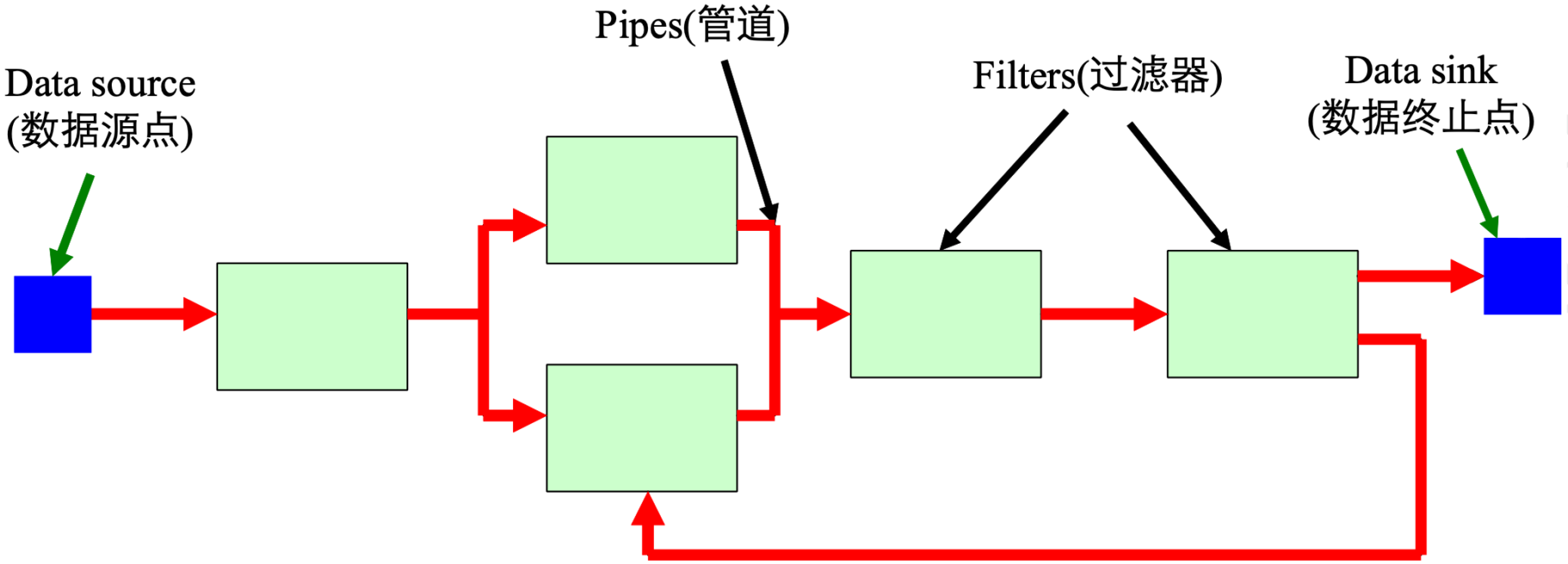
1. 软件体系结构风格
2. 数据流体系结构风格
3. 批处理体系结构风格
4. 管道-过滤器体系结构风格

管道-过滤器体系结构风格-直观结构（1//2）

自来水处理中的Pipe-and-Filter结构



管道-过滤器体系结构风格-直观结构（2//2）



管道-过滤器体系结构风格-定义（1//3）

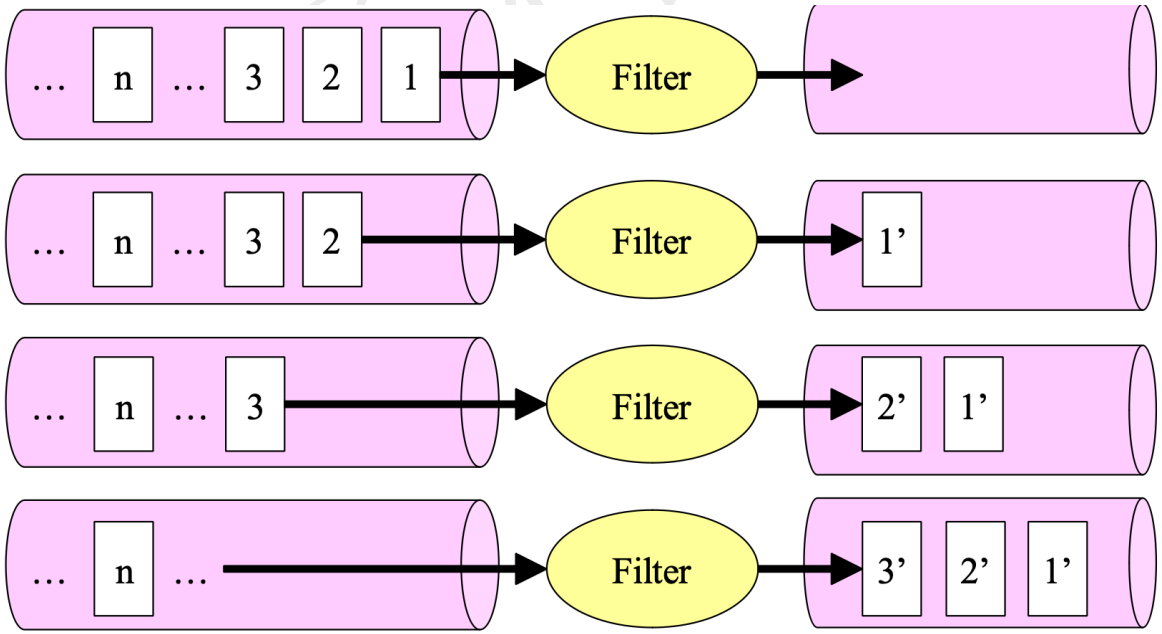
- 场境：数据源源不断的产生，系统需要对这些数据进行若干处理(分析、计算、转换等)。
- 解决方案：
  - ✓ 把系统分解为几个序贯的处理步骤，这些步骤之间通过数据流连接，一个步骤的输出是另一个步骤的输入；
  - ✓ 每个处理步骤由一个过滤器构件(Filter)实现；
  - ✓ 处理步骤之间的数据传输由管道(Pipe)负责。
- 每个处理步骤(过滤器)都有一组输入和输出，过滤器从管道中读取输入的数据流，经过内部处理，然后产生输出数据流并写入管道中。

管道-过滤器体系结构风格-定义（2//3）

- **Components: Filters — process data streams (构件：过滤器，处理数据流)**
  - ✓ A filter encapsulates a processing step (algorithm or computation) (一个过滤器封装了一个处理步骤)
  - ✓ Data source and data end/sink are particular filters (数据源点和数据终止点可以看作是特殊的过滤器)
- **Connectors: A pipe connects a source and a end filter (连接件：管道，连接一个源和一个目的过滤器)**
  - ✓ Pipes move data from a filter output to a filter input
  - ✓ Data may be a stream of ASCII characters
- **Topology: Connectors define data flow graph**
- **Style invariants：Filters are independent**

管道-过滤器体系结构风格-定义（3/3）

Incrementally transform data from the source to the sink (递增的读取和消费数据流)，数据到来时便被处理，不是收集然后处理，即在输入被完全消费之前，输出便产生了。



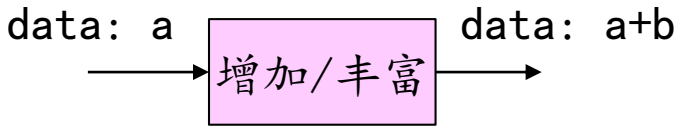
管道-过滤器体系结构风格-过滤器（1/3）

Filter（过滤器 ）

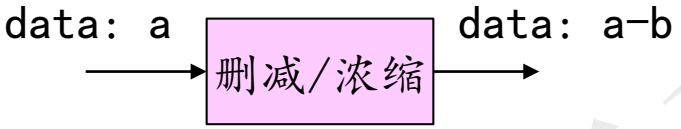
- Incrementally transform some of the source data into sink data (目标：将源数据变换成目标数据)
- Stream to stream transformation (从“数据流”→“数据流”的变换)

管道-过滤器体系结构风格-过滤器（2/3）

过滤器对数据流的五种变换类型

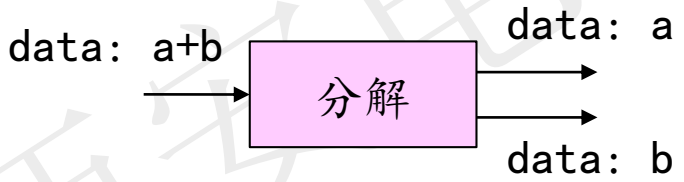


通过计算和增加信息来丰富数据

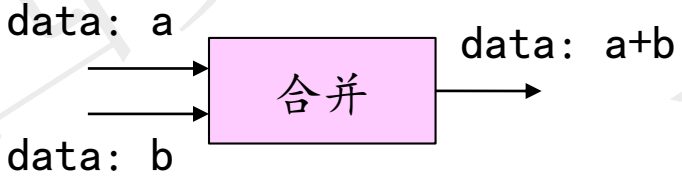


通过浓缩和删减来精炼数据

通过改变数据表现方式来转化数据



将一个数据流分解为多个数据



将多个数据流合并为一个数据流

管道-过滤器体系结构风格-过滤器（3/3）

Filters are independent entities.

- no context in processing streams (无上下文信息)
- no state preservation between instantiations (不保留状态)
- no knowledge of upstream/downstream filters (对其他过滤器无任何了解)

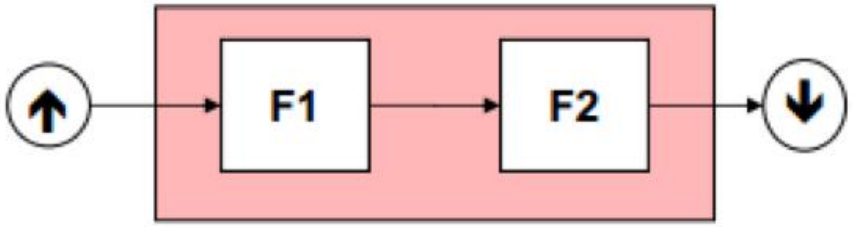
管道-过滤器体系结构风格-管道（1/2）

Pipes: move data from a filter's output to a filter's input (or to a device or file).

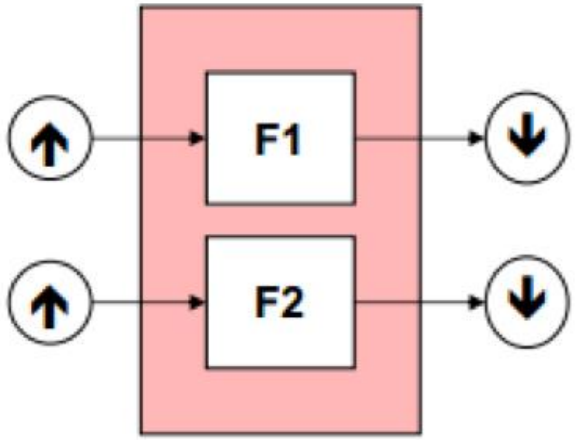
- One way flow from one data source to one data sink (单向流)
- A pipe may implement a buffer (可能具有缓冲区)
- Pipes form data transmission graph (管道形成传输图)
- 不同的管道中流动的数据流，可能具有不同的数据格式，数据在流过每一个过滤器时，被过滤器进行了丰富、精练、转换、融合、分解等操作，因而发生了变化。

管道-过滤器体系结构风格-管道（2/2）

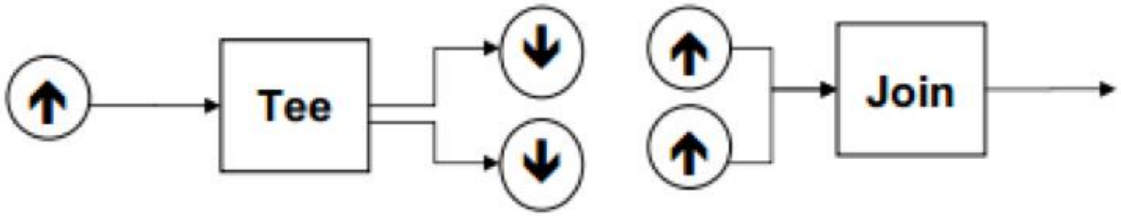
- Sequential Composition
  Unix: F1 | F2



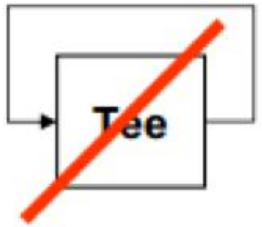
- Parallel Composition
  Unix: F1 & F2



- Tee & Join



- Restriction to **Linear Composition**



管道-过滤器体系结构风格-应用实例

- Compiler (编译器)
- Unix pipes (Unix管道)
- Image processing (图像处理)
- Signal processing (信号处理)
- Voice and video streaming (声音与图像处理)
- …

管道-过滤器体系结构风格-Advantages

- 使构件具有良好的隐蔽性和高内聚、低耦合的特点，可将整个系统的输入/输出行为看成多个过滤器的行为的简单合成；
- 支持软件复用，只要提供适合在两个过滤器之间传送的数据，任何两个过滤器都可被连接起来
- 系统维护和增强系统性能简单，新的过滤器可以添加到现有系统中来，旧的可以被改进的过滤器替换掉
- 允许对一些如吞吐量、死锁等属性的分析
- 支持并行执行：每个过滤器是作为一个单独的任务完成，因此可与其它任务并行执行。

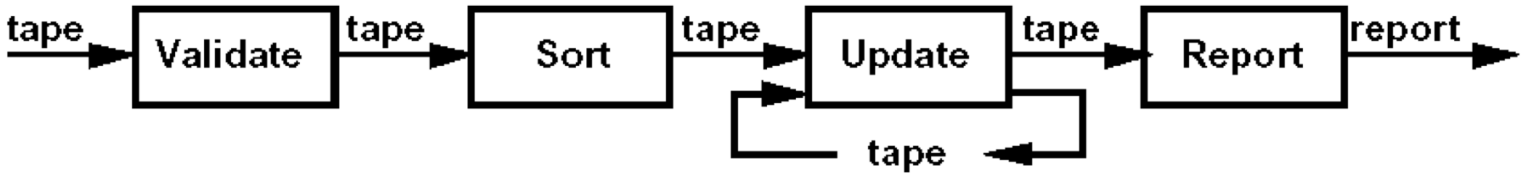
管道-过滤器体系结构风格-Disadvantages

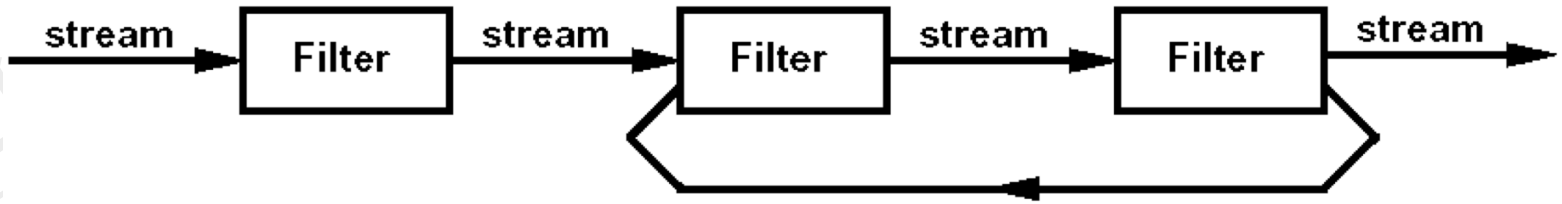
- **不适合处理交互的应用**
  - ✓在早期对交互应用需求不高的情况下提出
  - ✓当需要增量地显示改变时，这个问题尤为严重
- **系统性能不高，并增加了编写过滤器的复杂性**
  - ✓数据传输缺乏通用标准，每个过滤器都增加了解析和合成数据的工作
  - ✓绝大部分处理时间消耗在格式转换上
  - ✓不适用于需要大量共享数据的应用设置

批处理 VS 管道-过滤器

把任务分解成为一系列固定顺序的计算单元 &

彼此间只通过数据传递交互





批处理 VS 管道-过滤器

Batch Sequential	Pipe-and-Filter
total(整体传递数据)	incremental(增量)
coarse grained(构件粒度较大)	fine grained (构件粒度较小)
high latency (延迟高，实时性差)	results starts processing (实时性好)
no concurrency (无并发)	concurrency possible (可并发)

西安电子科技大学
XIDIAN UNIVERSITY

计算机科学与技术学院
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY
国家示范性软件学院
NATIONAL PILOT SCHOOL OF SOFTWARE ENGINEERING

计算机科学与技术学院微信

蔺一帅 讲师 硕导

邮箱：yslin@xidian.edu.cn

主页：https://web.xidian.edu.cn/yslin/

课程讨论群