案例编号：

案例名称：实验室设备管理系统基于数据流的模型抽象和分解

案例目标：

理解基于数据流的设计模型的抽象和分解

案例描述：

在本案例中主要是针对实验室设备管理系统进行基于数据流的设计，抽象和分解出实验室设备管理系统的组件图。不同的用户具有不同的功能和流程。设备负责人主要负责设备状态变更的通知和确认以及更新设备使用人和授权，设备管理人主要是负责设备状态变更通知和确认以及更新责任人和授权，设备使用人主要负责更新设备的状态，设备审计人主要负责提取设生成设备状态列表。

Keyword：

设备、设备负责人、设备管理人、设备使用人、设备审计人

系统描述：

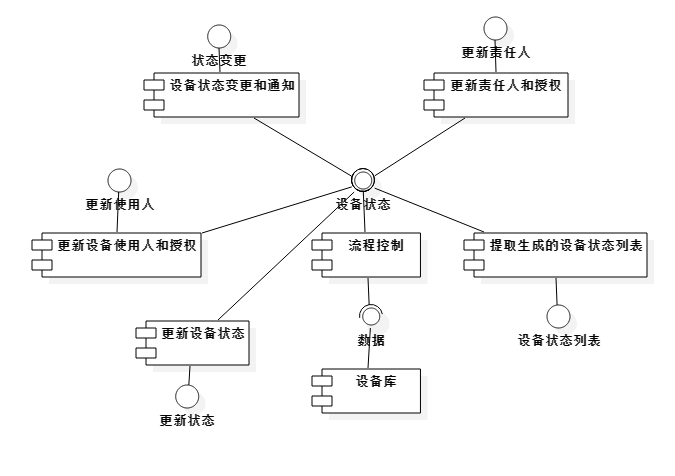
实验室设备管理系统主要是对实验室的设备进行管理的，设备具有很多种状态，还包括了设备负责人、设备管理人、设备使用人和设备审计人。不同的用户具有不同的功能和流程。设备负责人主要负责设备状态变更的通知和确认以及更新设备使用人和授权，设备管理人主要是负责设备状态变更通知和确认以及更新责任人和授权，设备使用人主要负责更新设备的状态，设备审计人主要负责提取设生成设备状态列表。

案例建模过程：

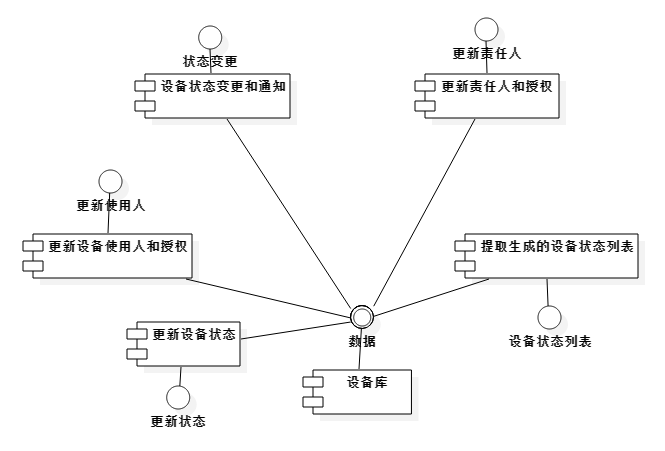
1. 在采用基于数据流的抽象和分解时，首先一定要有数据的收集。
2. 对收集好的数据进行处理。
3. 处理完的数据结果需要表示给用户。
4. 最后就需要有数据存贮，然后对数据进行管理（涉及数据库的设计）。
5. 在这整个过程还需要考虑系统的可用性和性能。比如：数据的处理过程是怎么样的，又如何去展现你的数据呢？体现出一个系统的可用性。

案例结果：

1. 正面



1. 反面



案例总结：

设备有很多状态，还具有设备负责人、设备管理人、设备使用人和设备审计人。不同的用户具有不同的功能和流程。根据设计原则可以得到一个案例结果中的组件图。有设备状态变更通知和确认组件、更新设备使用人和授权组件、更新设备状态组件、提取生成的设备状态列表组件、更新责任人和授权组件。我们从反面设计的组件图中可以知道，每个组件都有自己的控制逻辑，都要通过设备库访问数据，但是设计存在一个问题：组件很多地方都有相近的逻辑，而我们的每个组件都需要查看设备的状态，都需要做流程的控制。把这些共同点抽象放在一起的话，进一步改进可以得到正面设计的组件图，组件图中包含流程控制的组件，让所有组件都和流程控制交互，此时其它组件通过流程控制组件和设备库进行交互。这样设计的优点有：

1. 可以减少组件对数据的破环，即保证了数据的安全和完整性。如果数据库发生变化，只有流程控制发生变化，别的不受影响。
2. 从系统性能角度来看，可以提升数据库的性能，当数据量很大时，可以优化流程控制，比如cache。
3. 系统涉及权限、角色、流程等的访问控制，可以进行规则的配置。

所以在基于数据流的抽象和分解设计过程中需要对系统进行全面的考虑，如果系统比较小，流程图等就可以满足系统的设计需求，但是如果系统比较大时需要达到很大的用户量，规模时，需要严肃的思考。比如：淘宝双十一具有百万条规模的数据，多人在线时，业务量比较大，绝不能说让每个模块都去访问数据库，否则系统就会崩溃，合理的设计模块才能让系统更好的运行。

比如对于更新设备状态这个功能，我们应该怎么设计？设计可以从以下几个方面考虑：

* 人机交互（Web页面？）--浏览器端
* 数据库访问（访问方式？）--服务器端
* 数据确认与更新—浏览器/服务器端
* 浏览器端与服务器端的连接和通讯控制

设备变更通知和确认这个功能可以从这这个方面考虑：

* 人机交互（Web页面？）--浏览器端
* 数据库访问（访问方式？）--服务器端
* 角色流程控制（首先通知负责人，确认后再通知管理员进行确认）
* 浏览器端与服务器端的连接和通讯控制

其他的几个模块也大抵如此。把浏览器端与服务器端的一些业务请求，交给流程控制。