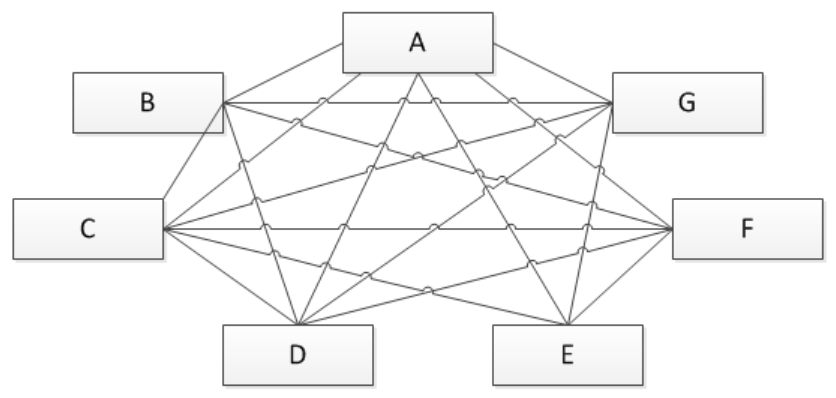
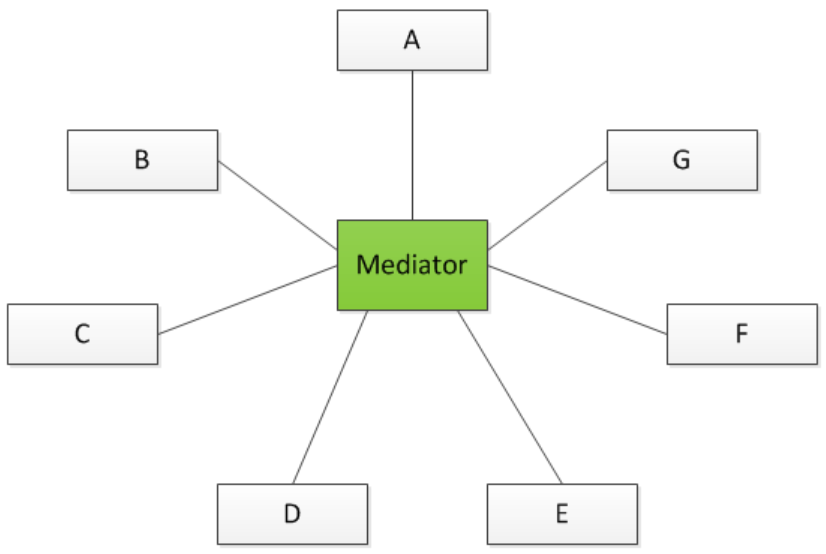
调停者模式是对象的行为模式。调停者模式包装了一系列对象相互作用的方式，使得这些对象不必相互明显引用。从而使它们可以较松散地耦合。当这些对象中的某些对象之间的相互作用发生改变时，不会立即影响到其他的一些对象之间的相互作用。从而保证这些相互作用可以彼此独立地变化。

为什么需要调停者

如下图所示，这个示意图中有大量的对象，这些对象既会影响别的对象，又会被别的对象所影响，因此常常叫做同事(Colleague)对象。这些同事对象通过彼此的相互作用形成系统的行为。从图中可以看出，几乎每一个对象都需要与其他的对象发生相互作用，而这种相互作用表现为一个对象与另一个对象的直接耦合。这就是过度耦合的系统。



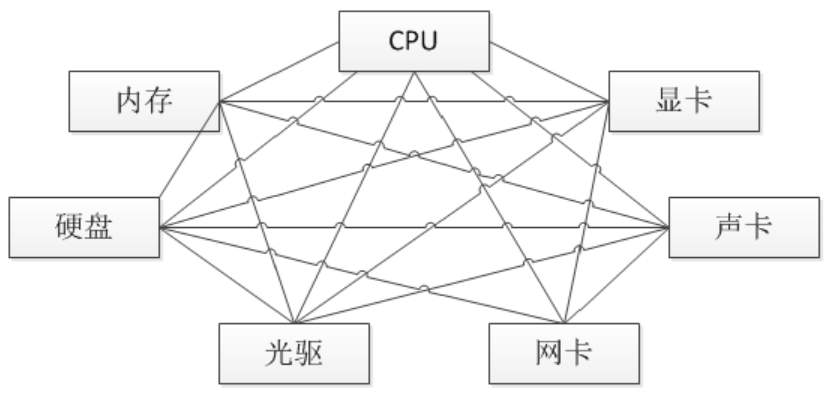
通过引入调停者对象(Mediator)，可以将系统的网状结构变成以中介者为中心的星形结构，如下图所示。在这个星形结构中，同事对象不再通过直接的联系与另一个对象发生相互作用；相反的，它通过调停者对象与另一个对象发生相互作用。调停者对象的存在保证了对象结构上的稳定，也就是说，系统的结构不会因为新对象的引入造成大量的修改工作。



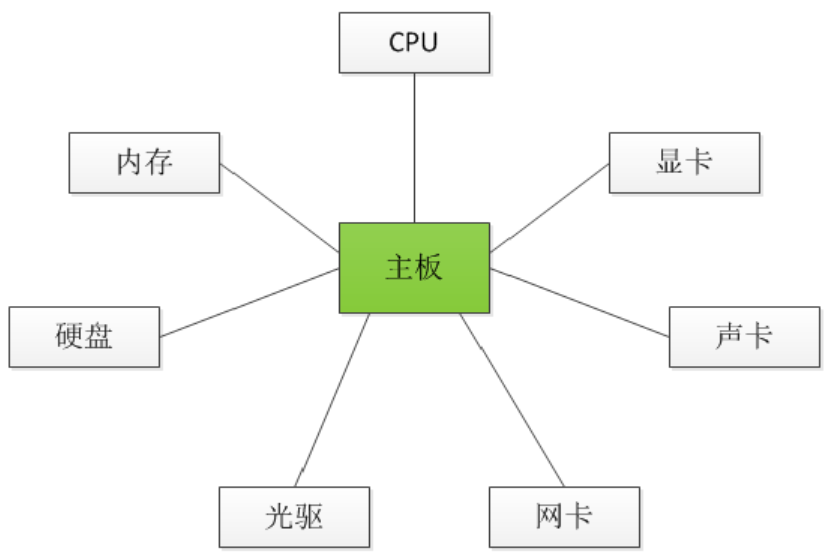
一个好的面向对象的设计可以使对象之间增加协作性(Collaboration)，减少耦合度(Couping)。一个深思熟虑的设计会把一个系统分解为一群相互协作的同事对象，然后给每一个同事对象以独特的责任，恰当的配置它们之间的协作关系，使它们可以在一起工作。

现实情形举例：

大家都知道，电脑里面各个配件之间的交互，主要是通过主板来完成的。如果电脑里面没有了主板，那么各个配件之间就必须自行相互交互，以互相传送数据。而且由于各个配件的接口不同，相互之间交互时，还必须把数据接口进行转换才能匹配上。



所幸是有了主板，各个配件的交互完全通过主板来完成，每个配件都只需要和主板交互，而主板知道如何跟所有的配件打交道，这样就简单多了。



调停者模式包括以下角色：

　　●　　抽象调停者(Mediator)角色：定义同事对象在自身改变的时候去通知调停者要使用的抽象方法。

　　●　　具体调停者(ConcreteMediator)角色：实现了抽象调停者所声明的方法。具体调停者知晓所有的具体同事类，并负责具体的协调各同事对象的交互关系。

　　●　　抽象同事类(Colleague)角色：定义出拿到调停者的接口。同事对象只知道调停者而不知道其余的同事对象。

　　●　　具体同事类(ConcreteColleague)角色：所有的具体同事类均从抽象同事类继承而来。实现自己的业务，在需要与其他同事通信时，只与持有的调停者通信，调停者来负责与其他的同事交互。

调停者模式的优点：

　　●　　松散耦合

　　调停者模式通过把多个同事对象之间的交互封装到调停者对象里面，从而使得同事对象之间松散耦合，基本上可以做到互不依赖。这样一来，同事对象就可以独立地变化和复用，而不再像以前那样“牵一处而动全身”了。

　　●　　集中控制交互

　　多个同事对象的交互，被封装在调停者对象里面集中管理，使得这些交互行为发生变化的时候，只需要修改调停者对象就可以了，当然如果是已经做好的系统，那么就扩展调停者对象，而各个同事类不需要做修改。

　　●　　多对多变成一对多

　　没有使用调停者模式的时候，同事对象之间的关系通常是多对多的，引入调停者对象以后，调停者对象和同事对象的关系通常变成双向的一对多，这会让对象的关系更容易理解和实现。

调停者模式的缺点：

　　调停者模式的一个潜在缺点是，过度集中化。如果同事对象的交互非常多，而且比较复杂，当这些复杂性全部集中到调停者的时候，会导致调停者对象变得十分复杂，而且难于管理和维护。