第7章 软件设计——面向对象方法

7.2 人机交互部分的设计

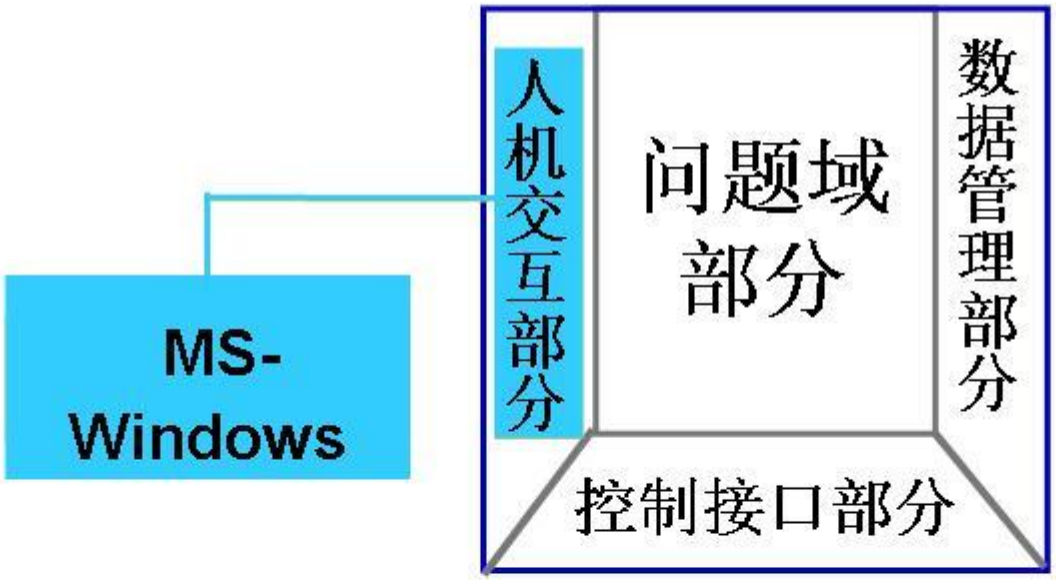
刘其成

计算机与控制工程学院

ytliuqc@163.com

2018-09

什么是人机交互部分



7.2.1概述

- 人机交互部分是面向对象设计模型的组成部分之一
- 把人机交互部分作为系统中一个独立的组成部分，进行分析和设计，有利于隔离界面支持系统的变化对问题域部分的影响。
  - 当界面支持系统变化时，问题域部分可基本保持不变。
- 人机交互部分包含的对象构成了系统的人机界面，称做界面对象。

- 人机交互部分的友好性对用户情绪和工作效率产生重要影响。
  - 交互界面设计得好，则会使系统对用户产生吸引力，能够激发用户的创造力，提高工作效率；
  - 相反设计得不好则会使用户感到不方便。

- 软件系统大多采用图形方式的人机界面
  - 有形象、直观、易学、易用等特点。
  - 但是图形用户界面的开发工作量也很大，在系统开发成本中占有很高的比例。

界面支持系统

- 支持图形用户界面开发的软件系统
  - 窗口系统（如Windows）
  - 图形用户界面（Graphics User Interface，GUI）系统（如Motif）
  - 与编程语言结合为一体的可视化编程环境（如Visual C++）。
- 利用界面支持系统，图形用户界面的开发效率可得到显著提高，因此应用系统的人机界面开发大多依赖某种界面支持系统。

- 人机界面的开发除了软件的知识以外，还需要心理学、艺术等许多其他学科的知识。
  - 在心理学的指导下，一种人机界面的设计才会使人感到舒适、振奋、兴趣盎然、给人以正确的启发，而不是烦躁、颓丧、索然无味以及引起误导。
  - 同时，界面设计的总体布局以及各个局部的形状、尺寸、图案、纹理、色彩、变幻等要达到美观而协调的效果，需要有美术人员的参与，并且要借鉴心理学、统计学等方面的研究成果。

7.2.2可视化编程环境下的人机界面设计策略

- 采用可视化编程环境作为界面支持系统时，人机界面的设计相对比较简单。
  - 针对可视化编程环境的设计策略，减少或简化了许多工作内容。
  - 设计者利用可视化编程环境及其类库，降低了设计的工作强度，简化了设计文档；
- 但是需要掌握所依赖的环境和类库，并在设计中把复用类库中提供的类作为基本出发点。

1. 掌握可视化编程环境及其类库

- 软件设计人员不负责系统实现，但是为了使设计能与环境的实际情况相吻合，必须掌握或了解实现设计的语言、类库、编程环境等软件。
  - 正如一个建筑设计师，虽然不负责施工，却必须了解施工所用的材料、机械和建筑工艺。

- 在人机界面的设计中，要掌握可视化编程环境及其类库。
  - 因为不同的编程环境和类库所支持的界面对象不相同，功能、风格都有不少差异，设计者必须根据这些具体特点来进行设计，才能使设计与实现很好地衔接。
- 需要掌握的重点是：

- （1）该环境对各种界面对象所采用的术语及其含义。
- （2）类库中提供的界面对象对应的类。
- （3）各个界面对象类的属性与服务，包括从上一层的父类继承来的属性与服务；但不必太关心描述对象外观的属性。
- （4）各个类所创建的界面对象的外观，以及它们在人机交互中所适合的输入与输出；
- （5）各个类之间的继承层次。
- （6）各个界面对象涉及的事件及其事件处理机制。
- （7）比较复杂的界面对象，可以采用整体-部分结构由多个界面对象而形成组合对象。

2. 根据人机交互需求选择界面元素

- 根据人机交互需求，在可视化编程环境所能支持的界面元素中进行选择。
- 必要时，设计者应该在环境中实际操作和演示一下准备选择的各种界面元素，以决定哪些元素最适合本系统的人机交互。

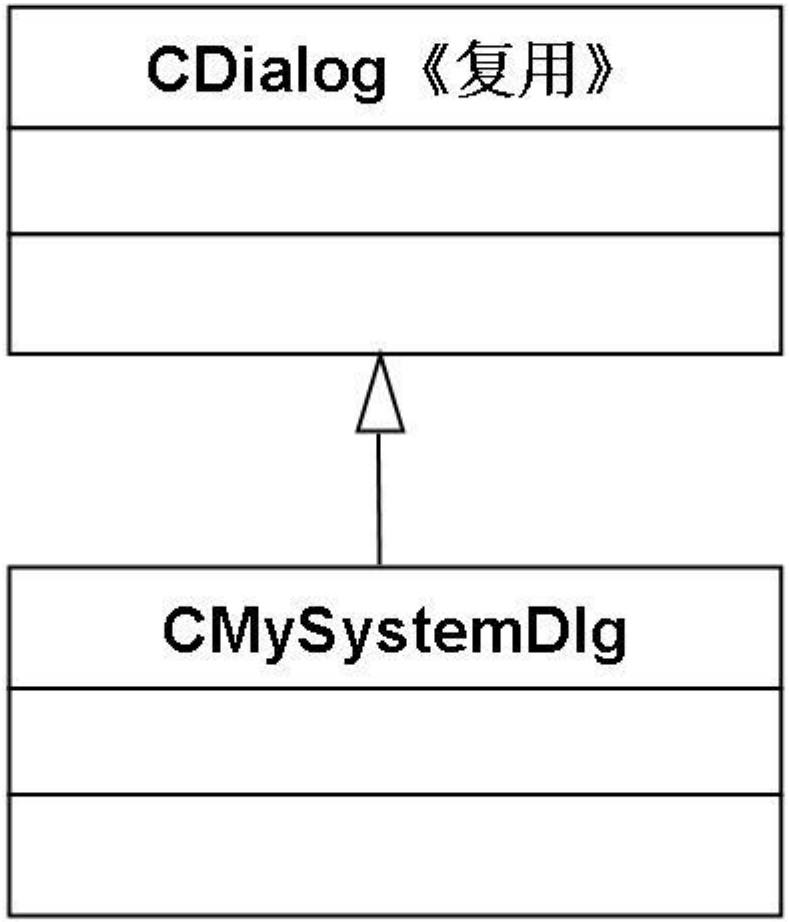
3. 类图的设计

- 1）根据复用原则确定类的选择
- 确定类图中表示界面对象的类时，应首先使用环境及其类库所提供的可复用类。
  - 这些类通常能满足应用系统的大部分人机交互需求，充分复用这些类会大大简化界面的设计和实现。
  - 所以，除非所需要的界面对象的功能或风格很特殊，在类库中也没有提供相应的类，否则都要首先想到复用已有的类。
- 可以直接复用类库中的类，通过将可复用类某些参数具体化（例如属性的初始值）对可复用类进行定制。

直接复用

- 如果所需要的界面对象的功能或风格有特殊，需要对类库中的可复用类进行改进，应用系统可以对可复用类的定义进行扩充。
  - 例如在对话框中添加编辑框控件实现定制的对话框，这是对类库中 Dialog类的扩充，定制的对话框在Dialog类的基础上添加了其他控件作为自己的组成部分。
- 这实际上是通过对可复用类的继承，定义了本系统中的一个新类。

通过继承复用



- 面向对象设计中这种类图不需要填写被复用类的属性和服务，也不需要画出比它层次更高的一般类。
  - 类库中提供的界面对象对应的类通常有许多属性和服务，在类库中还有很多层被它继承的一般类。
  - 这些信息都是现成的，不需要在应用系统中实现。
- 这样可以简化面向对象设计的文档，把主要精力用于解决本系统的问题。同时，对实现者也没有影响
  - 如果在应用系统的类图中全部表示这些信息，会使类图很庞大，是一项沉重的负担，而且意义也不大

2) 定义表示逻辑特征的属性，忽略物理特征的属性

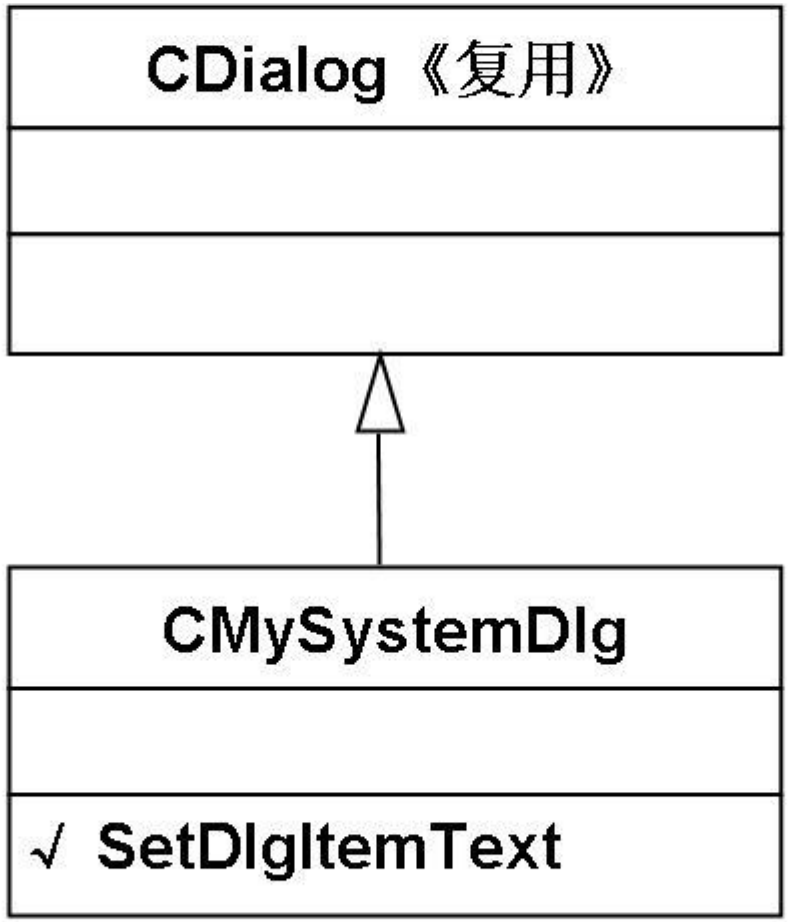
- 通过继承可复用类而定义的新类，可以对继承来的属性设置不同的初始值，也可以在新类中增添新的属性。
- 设计人员主要定义那些描述界面对象逻辑特征的属性，特别是表现命令的组织结构、界面元素之间组成关系的属性。
  - 例如在一个菜单类中，每个选项表示一条命令。要用属性来描述菜单的每个选项，属性的名称应该与它所对应的命令相符。
  - 又如在一个对话框中包含若干控件，应该用属性表示这个对话框含有的控件对象。

- 设计阶段不关心描述界面对象物理特征的属性（大小、形状、位置、颜色、边框、底纹、图案式样、二维效果），这些属性由实现人员来处理。
  - 实现人员以可视化的方式定制界面对象的这些特征，效果会更好，效率也会更高。
  - 如果有美工人员参加实现，那么在艺术效果方面还会取得更好的效果。

3）特别标注从高层类继承的服务

- 面向对象设计中界面对象对应的类在类图中不填写被复用类的属性和服务，也不需要画出比它层次更高的一般类。这样简化了面向对象设计的文档。
- 但是，这样的类图表示不出一个特殊类中含有一个通过继承得到，并提供其他对象使用的服务，因而也难以表达其他对象请求该项服务的消息。
  - 解决方案是在本系统定义的特殊类中显式地表示它从类库中的类继承的，并且将在本系统中被使用的服务。

特别标注从高层类继承的服务



- 在类图的服务栏填写该服务的名称，并且做一个“√”标记，表明这个服务是通过继承得到的，不需要在本系统中实现。
- 同时，在类的描述模板中服务说明条目中指出该服务是从类库中哪个类继承来的。

- 界面类库中一些属性和服务被继承之后并不真正被使用；另有许多属性和服务是由编程环境自动使用的，应用开发者不必关心。
- 只有在手工产生的程序代码中需要使用的那些服务，才是设计人员和实现人员必须了解的，在特殊类中显式地以“√”符号表示。
- 继承来的属性一般不需要关注，当某些情况下需要关注时，也采用这种表示法 。

4) 用整体-部分结构表示界面的组织结构和命令层次

- 要正确区分对象的普通属性和它的部分对象。
- 当一个界面对象带有内部组织结构时，可视化编程环境对不同的情况有不同的定义方式。
  - 有些组成部分（例如下拉菜单的选项、窗口的边框）被作为对象的一个普通属性
  - 有些组成部分（例如对话框的一个下拉菜单或按钮）则被作为一个部分对象。
- 环境类库对这种组成部分给出相应的类定义，则建立整体-部分结构。否则只用属性表示，不建立整体-部分结构。

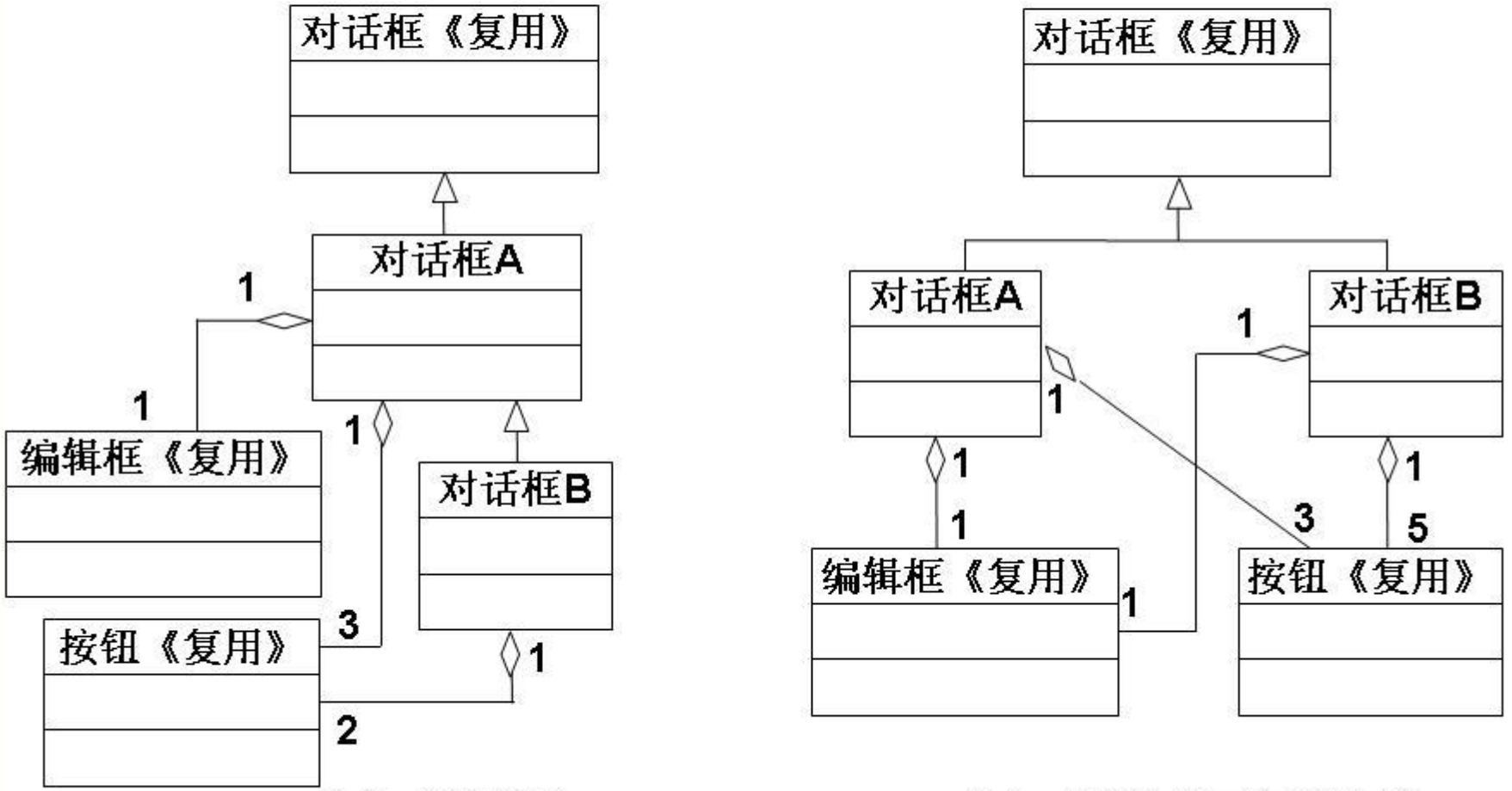
- 但如果一个整体对象的部分对象具有以下的简单性，整体-部分结构的表示法可以简化。
  - (1) 这种界面对象在本系统中总是依附于整体对象而存在，系统中不会单独地创建这样一个独立存在的对象实例。
  - (2) 该对象无论从物理上还是从逻辑上都不再包含级别更低的部分对象。
  - (3) 该对象没有或者只有一个表示逻辑特征的属性。例如，按钮对象只需要一个名称属性来表示其逻辑特征，其他属性都只表示其物理（外观）特征。
  - (4) 不需由程序员通过手工编程实现该对象与其他对象间的消息。

- 在这种简单的情况下可以隐式地表示部分对象：
  - 类图中不画出部分对象的类，只是在整体对象中通过一个属性表示整体对象拥有这样一个部分对象，
  - 用这个部分对象的类名作为该属性的类型，并在类描述模板的属性说明中加以说明。
- 当类图很庞大时，采用这种策略可以使之简化。但是这种策略失去了一些直观性，在类图不很庞大时还是显式地画出部分对象类为好。

5）采用一般-特殊结构从可复用类直接继承

- 如果一个应用系统中使用的两个或两个以上界面对象有许多共同特征，按照通常的设计策略是运用一般-特殊结构，在一般类中定义共同拥有的属性和服务，特殊类继承一般类，从而简化设计和实现。

- 例如，系统中使用了A，B两个对话框类，其中B拥有A的所有特征，只是比A增加了两个按钮；那么，在一般的界面支持系统下可以设计如图a的所示的一般-特殊结构，使A继承可复用类，B又继承A，从而简化了这两个对话框类的设计和实现。
- 但是在可视化编程环境下，若按图a的结构基于自定义的A类对话框来实现B，则可能缺乏环境支持，或者环境虽然有支持但操作比较复杂，需要有较高的应用技巧。
- 采用如图b所示的一般-特殊结构更便于实现，因为在环境支持下，直接基于可复用类来定制B类对话框是很方便的。



（a）普通策略

（b）直接继承可复用的类

6）表达手工编程实现的消息，忽略自动实现的消息

- 界面类库中的每个类都定义了许多服务，应用系统在这些类的基础上定制的类提供的服务也很多，每个服务都对应着一种消息。
- 但有大量的消息，特别是处理界面对象常规操作的消息，是在可视化编程环境支持下生成应用程序代码时自动实现的。
  - 例如改变窗口的大小或位置，在对话框中将焦点转移到其中的某个控件，滚动条的上下、左右滚动等
- 这些消息不需要程序员通过手工编程去实现，因此设计类图时可以忽略对这些消息的表示。

- 设计者需要关注并且要在类图和类描述模板中表达的，是那些必须由程序员通过手工编程来实现的消息，包括
  - 1）接收界面操作事件的界面对象，通过它的一个服务向对该事件进行实际处理的功能对象发送的消息。
  - 2）从要求在人机界面上进行输入/输出的功能对象，向提供这种输入/输出服务的界面对象发送的消息。
- 表示方式和通常的做法一样，在类图中画出消息发送者与接收者之间的消息连接符号，并在发送者一端的类描述模板中进行相应说明。

7.2.3 界面类与问题域类间通讯的设计

- 有些界面对象要与问题域中的对象进行通讯，所以要对二者之间的通讯进行设计。
- 设计时应注意以下三点：
- （1）人机界面负责输入与输出和窗口更新这样的工作，并把所有面向问题域部分的请求转发给问题域部分，在界面对象中不应该对业务逻辑进行处理。

- （2）问题域部分的对象不主动发起与界面部分对象之间的通讯，只是对界面部分对象进行响应。

  把界面对象向问题域部分对象传输的信息或发布命令看作是请求，而把从问题域部分对象向界面部分对象传输的信息看作是回应。

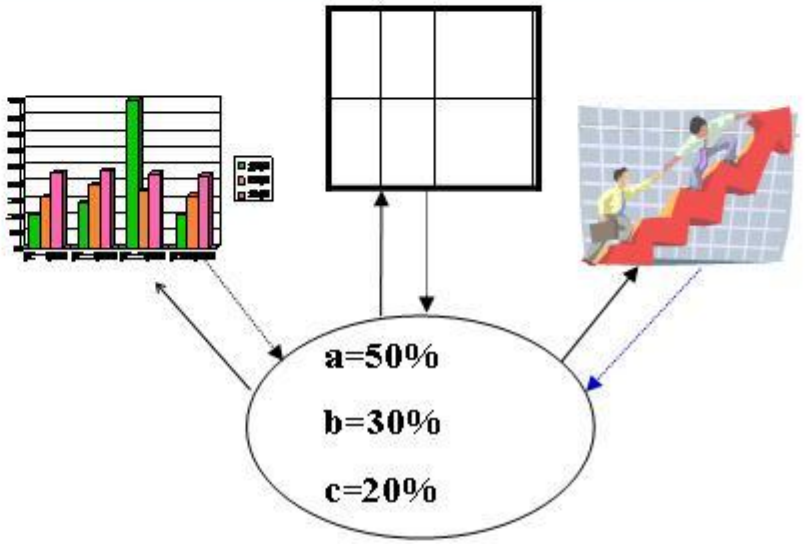
- （3）尽量减少界面部分与问题域部分的耦合。

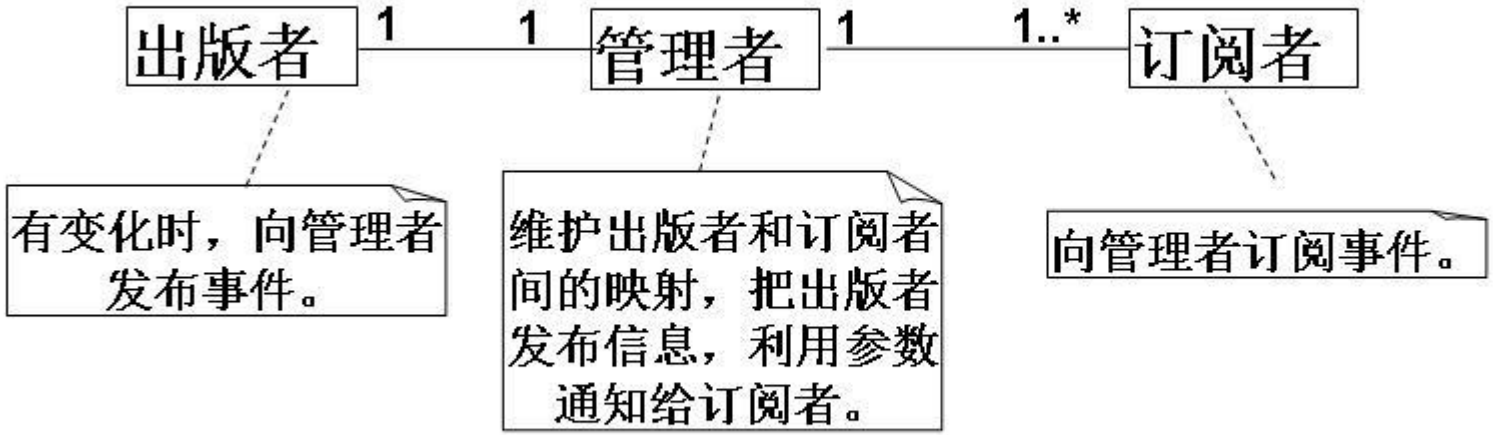
由于界面是易变的，从易于维护和易于复用的角度出发，问题域部分和界面部分应是低耦合的。

可以通过在人机交互部分和问题域部分之间增加控制器或协调类的方式解决这种问题。

如采用观察者模式等相关设计模式。

例题：出版-订阅模式（观察者模式）

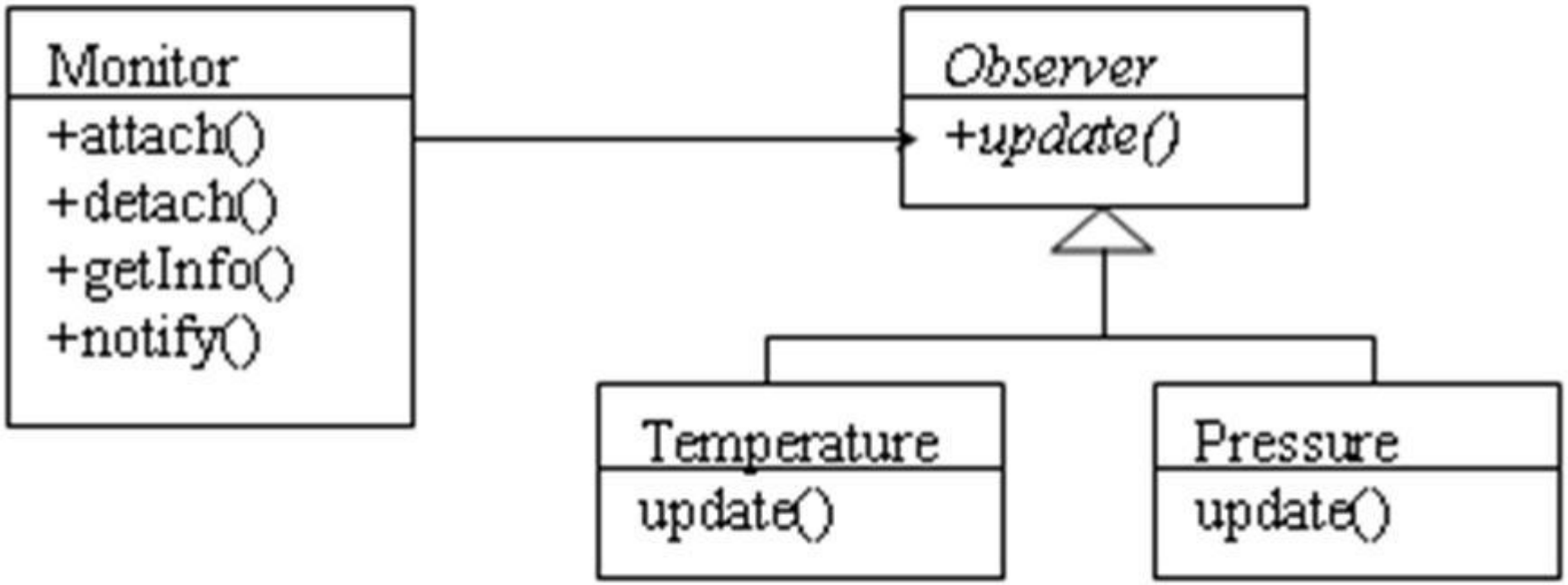


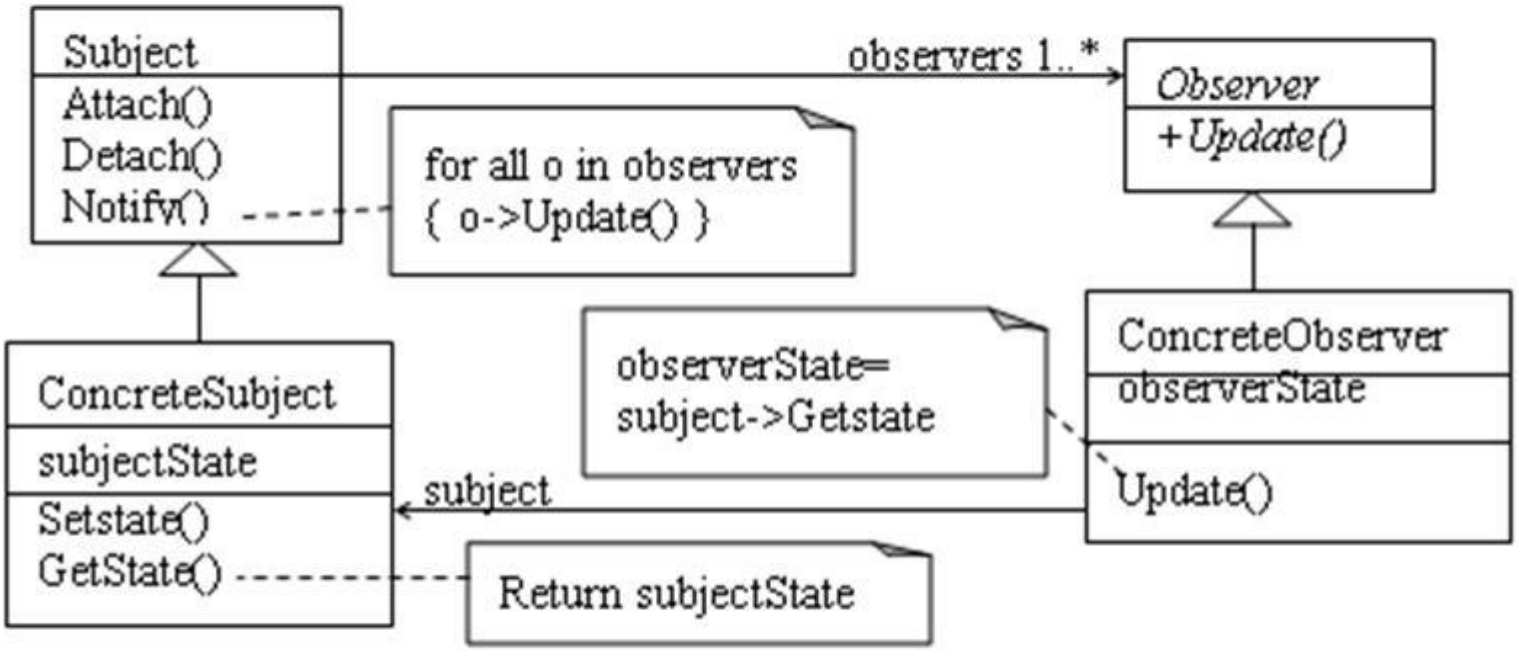


解决问题域中的对象对界面层对象的访问问题

- **Observer (观察者)模式**用于定义对象间的一对多的依赖关系，当一个对象发生变化并对外发布消息时，所有依赖它的对象都将得到通知并可进行更新。后一句话中的那组对象是观察者，它们要在发布消息的对象中进行登记（订阅），以便在发布消息时能找到它们。该模式也称为**发布—订阅（Publish-Subscribe）模式**。

- 采用观察者模式，要在发布消息的对象中设立增减观察者对象的操作，对于观察者对象要建立一个统一的用于接收消息的接口。
- 益处：观察者发生变化时，只须在发布消息的对象中增减对象标识即可，而不需进行其他改动


Monitor
+attach()
+detach()
+getInfo()
+notify()
Observer
+update()
Temperature
update()
Pressure
update()



Observer模式的通用结构图

谢谢

2018年11月27日