哈爾濱工業大學

软件架构与中间件

作业 2

学　　号　L170300901

学　　生　卢兑玳

设计模式(Design Patterns)

概要

▶为解决具有一般性的设计问题，将多类之间的典型协助工作模式化
▶ 提供有用的抽象画，组合中使用的几个模块（或 Object-Oriented Circles, Classes）的集合
▶ 标准问题的反复解决方案
▶ 图案的主要例子，Smalltalk 中已知的 MVC(Model-View-Controller) 模式
▶ 软件架构风格

设计模式(Design Patterns)

KWIC-INDEX

ex) 假设题目为 Software engineering should be a compulsory topic
▶ 因为有 7 个单词，所以需要 7 行
▶ 第二词"engineering"应成为第二行中的第一个字
- 即，必须'n 号 shift'
▶ 排在前面的 software 是退到第二行最后一头的

result)
Software engineering should be a compulsory topic.
Engineering should be a compulsory topic. Software
should be a compulsory topic. Software engineering
be a compulsory topic. Software engineering should
a compulsory topic. Software engineering should be
compulsory topic. Software engineering should be a
topic. Software engineering should be a compulsory1

▶ 以前生产线重复与其他生产线一起按标准词典编纂上的顺序保存，输出为 KWIC-index

▶ KWIC 指 Key Word In Context

▶ 即使只知道题目的一部分，如果使用 KWIC-index，搜索题目也会变得容易

▶ 设计解决上述问题的 SW 时，首先要决定如何分割最上面的阶段

KWIC-INDEX 的 4 种可能的架构

▶ 使用共享数据的设计

▶改变表达(Representation) 的算法时，将焦点放在使用信息隐蔽的差异

▶ 使用默认呼叫和抽象数据类型的分割方法

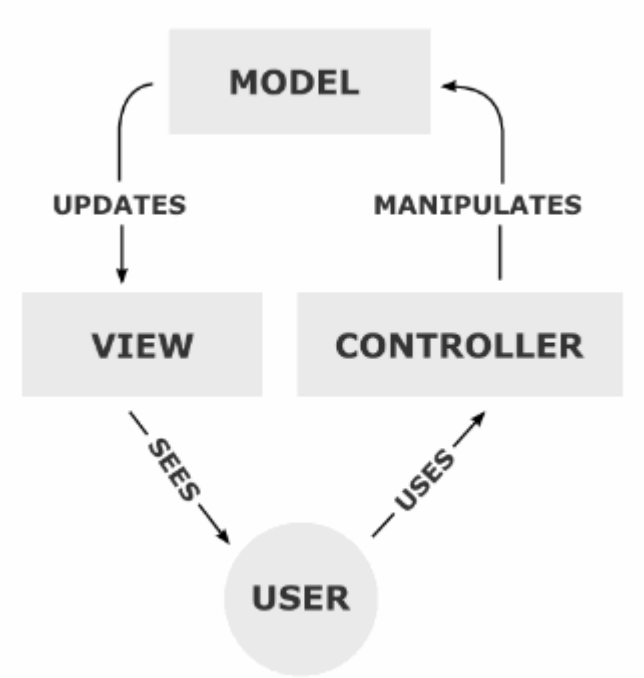
▶ 说明使用 UNIX 管道和过滤方法的分割方法

设计图案的圆形例子

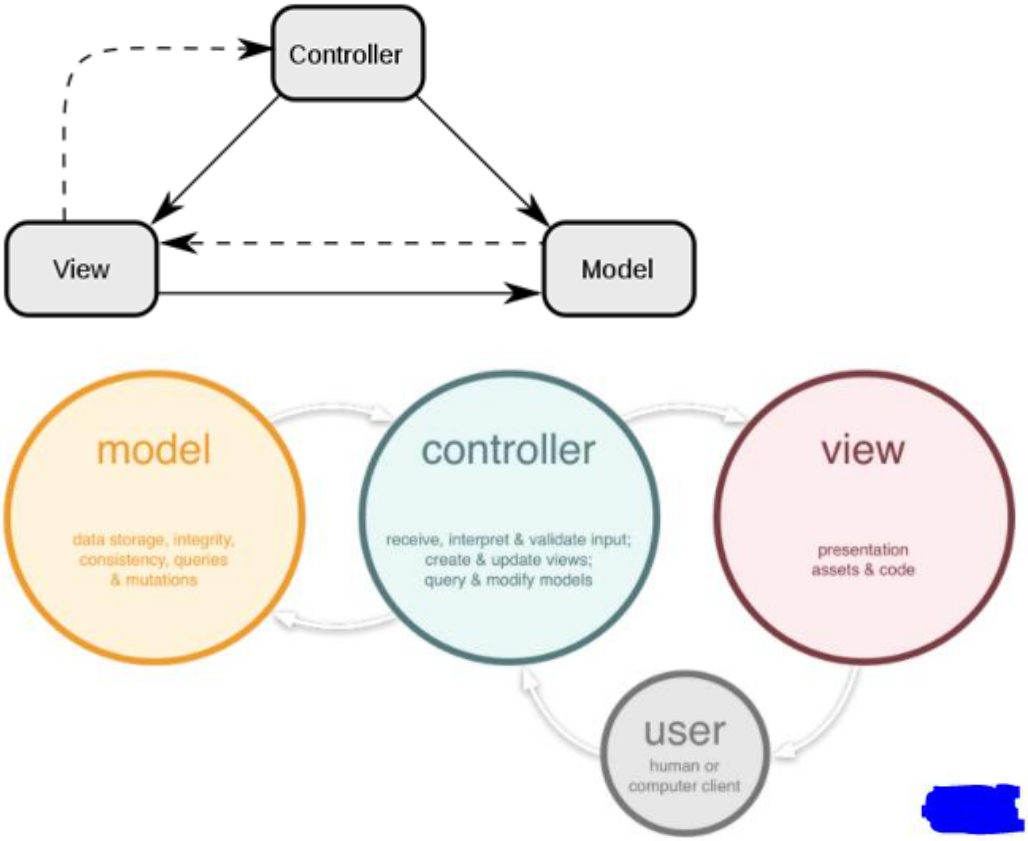
Model- View- Controller (MVC) 图案
Interactive Systems 由处理用户输入并显示数据的 Computational Elements 构成
分离处理 I/O 的计算要素的设计，从要素分离是 MVC 模式的
MVC 图案是设计图案之一。

设计模式是指，整理程序或特定开发过程中发生的问题，并根据具体情况简便应用，通过特定的"规章"制作出便于使用的形态。
要制作什么样的 APP...需要维护该应用程序，与其他人共享时，我们应该想出更简单、更利索的方法。 如果这些方法不明确的话.. 我们将不得不逐一制作出类别的函数。
库或框架就是相应的例子。
例如，如果直接使用 jQuery， 那么使用纯 Javascript 来选择 DOM("#lucid ") ，
那么应该用 document. get Elements Byid("lucid ") 长长的字体去寻找。
例如，制定什么样的 data，并制定修改该 data 的 logic。 如果制作 data 部分时，每一个 logic 都未能分离，并一起定义的话？ 以后很难维护。 为了"帮助"而设计出这种设计模式，这样"更简单、更方便"使用的方法叫做设计模式。 其设计模式有条纹图案、观察家图案等多种，其中之一就是 MVC 图案。

MVC 是 Model, View, Controller 的缩写。构成一个应用程序、项目时，将构成要素分为三个角色的模式。



如上图所示，如果用户操作 controller， controller 就会通过 model 获取数据，并根据这些信息控制视觉表达的 View 并传递给用户。 那是为了说明一个 logic 而制作的画，事实上 MVC 模式的结构更适合这幅画。 必须有 Controller 对 view 产生影响的部分。



让我们看着上面的图片，重新找到 MVC 模式是什么的感觉吧。 型号为控制器，控制器为 View，View 为用户，用户为再次向控制器前进。

▶ 模特， Model

应用程序的信息， 数据显示 。 数据库， 最初定义的常数， 初始化值， 变量等 。 也指负责这些 DATA、信息加工的编成。这个型号有以下规则。

1. 用户需要编辑的所有数据。

即，如果画面上方框中有字，则需要有方框的画面的位置信息、方框的大小信息、字体内容、字体位置、字体格式信息等。

2. 对于查看器或控制器，任何信息都不应了解。

即，当数据发生变更时， 不可拥有参照查看器的内部属性值， 以便在模型中直接调整屏幕 UI 。

3.发生变更时，应体现变更通知的处理方法。

如果更改了型号属性中的文本信息， 则需要通过发生事件的方式传递给某人，并实现当有人发送请求更改型号的活动时接收该信息的处理方法。 此外，型号需可重复使用，其他接口亦需保持不变。

▶ 视图. View

显示用户界面元素， 如 input 文本、 校验框项目等 。 换句话说，负责数据和客体的输入，以及显示的输出。 以数据为基础，用户可以看到的画面。查看时有以下规则。

1. 型号信息不能单独储存。

您将收到您所拥有的信息， 以便在屏幕上显示文字， 但您不能在任意视图中保存这些信息 。 如果您被命令绘制一个方框， 您必须只显示在屏幕上， 而不保存屏幕上所需要的信息 。

2. 模特或控制器等其他构成要素必须不知道。

除了模特等自己以外，其他因素不得参照或了解如何操作。 只要收到数据，视图就会显示在画面上。

3.发生变更时，应体现变更通知的处理方法。

像模型一样发生变更时，要体现向早些人告知变更的方法。 在视图中，如果用户在画面中更改画面中显示的内容，则需要将这些内容传达给模特，从而改变

模型。 显示更改通知以执行此任务 。 而且要设计成可重复使用，在表达其他信息时要容易设计。

▶ 控制器， Controller

起到连接数据和用户界面要素的桥梁作用。即用户点击并修改数据的"活动"处理部分。控制器也需要理解以下规则。

1. 要了解模特或视图。

模特或 View 不知道彼此的存在，只有将变更告知外部、接收的方法，控制器为了仲裁，需要了解模特及其相关的 View。

2. 要监测模型或视图的变更。

收到型号或视图的变更通知后，需要解释并通知各个组件。另外，应用软件的主要 LOGO 由控制器负责。

▶ 为什么要使用 MVC 模式呢？

用户浏览的页面、数据处理、以及从中控制这两项的控制、 创建由这三项组成的一个应用程序， 就可以集中精力进行各自负责的操作。 工厂也只负责一个角色，处理起来效率更高。 在这里也是一样。

* 事实上，铃木根和隔壁春之保根分别生产军用螺栓和军用螺母。

（柯蒂斯·勒梅，在东京大空袭之前对部下说。）

如果相互分离，集中精力开发应用程序，那么维护性、应用程序的扩张性、灵活性就会增加，重复编码的问题也会消失。 为此而准备的 MVC 模式。

*柔韧性:这里的柔韧性是指对客户端的新要求，能够以最小的费用更灵活地应对。
* 商务:程序的逻辑结构

▶ MVC 模式的意义

MVC 模式最终是对"该如何分配"的答案之一。 对于某些特定角色，在分担角色

时提出指导方针的方法之一就是 MVC 模式。如果使用此模式进行库式或框架设

计，就能体验到非常简单有趣的体验，诞生出美丽的代码。 当然，我们还要培养能够制作这种库和框架的实力。

参考资料

https://jhc9639.blog.me/221730791313