iOS 常用架构模式分析

张　帅　张　愉　尚兆洁

长安大学，陕西 西安 710064

摘要：在软件开发中，架构设计是体现一份代码是否具有高质量的绝对标准。因为如果不注意架构的设计，面对浩如烟海的类和 bug，我们就会束手无策。我们不可能记住每一个类，许多细节也很容易忘记。即使我们已经遵循了苹果的 MVC，但依然会存在问题。在 iOS 开发中，常用的架构模式有 MVC、MVP、MVVM。本文主要介绍以上几种架构模式的特点和适用情况。

关键词：iOS 开发；架构模式；MVC

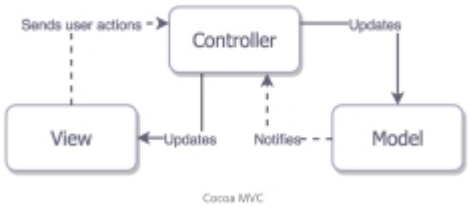
中图分类号：TP316　　文献标识码：A　　文章编号：1671-5586（2016）2-0159-01

1 优秀架构的标准

一个优秀架构的标准应该时这样的：1）每个实体都有严格的角色确定，他们的任务分配是很明确而均匀的；2）可测性好；3）易于使用，易于维护。

2 MVC

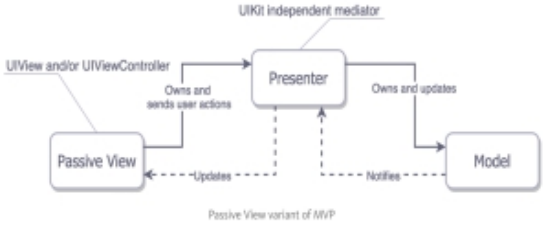
在 iOS 开发中的 MVC 如下：



控制器是模型和视图的中介，而模型和视图互相不知道彼此的存在。最不具复用性的是控制器，但是我们必须有这样一个地方来存放不能放在模型里的复杂逻辑。理论上看起来非常直接，但是也会觉得有点问题。Cocoa MVC 模式促使你把视图控制器越写越大，因为在视图的生命周期中它们是密不可分的，“为视图控制器减负”成为了 iOS 开发者的一个重要话题。即使你可以把一些逻辑和数据转换卸载给模型，但最终视图控制器变成了一个全世界的数据源和代理大管家，还经常负责分发网络请求以及各种乱七八糟的操作。当遇到单元测试的时候，问题就更加明显了。鉴于视图和控制器是高度耦合的，很难去做测试。因为必须非常小心地去把复杂逻辑和视图显示的代码分开，来模拟视图的生命周期。

这样看来，Cocoa MVC 似乎是个落后的模式。现在让我们从刚才提到的好的设计模式的三个特点的角度来考察一下它：分配 — 视图和模型是分开的，但是视图和控制器是耦合的；可测性 — 鉴于分配实现得不好，只能测试单独的模型；易用性 — 对比其他模式来说，代码量是最少的。并且，大家都很熟悉它，即便是不太有经验的程序员也可以轻松维护它。如果没有太多时间来打磨精修代码的结构，或者其他的模式维护成本过高，那么应该选择 Cocoa MVC。

3 MVP — 实现 Cocoa MVC 的理想



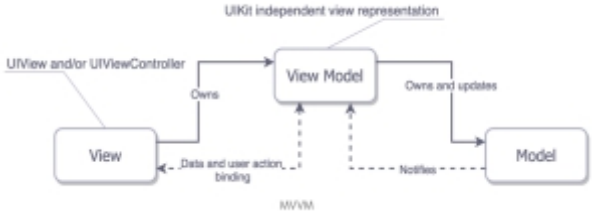
MVP（Passive View variant）的模式图如上所示。它不同于 MVC 的视图和控制器的耦合，在 MVP 中的中介—表达者（Presenter）和视图控制器的生命周期没有任何关系，并且视图可以轻易地被模拟，因此提出者里不用写任何和布局（layout）有关的代码，它负责视图的数据和状态更新。

对于 MVP，UIViewController 的子类实际上是视图而不是表达者。这个区别极大提升了可测性，同时会牺牲一定的开发速度，因为需要手动管理数据和时间之间的关系。

现在看一下 MVP 的特点：分配——我们把大部分事情分给了表达者和模型去做，视图则非常简单；可测性——因为视图的设计很简单，我们可以去测试大部分的逻辑；易用性——代码量也达到了 MVC 模式的两倍。但是 MVP 的思想表达的很清楚。

4 MVVM — MV（X）家族最新最先进的成员

MVVM 是 MV（X）家族的最新成员，理论上 MVVM（Model-View-ViewModel）模型看上去很完美。视图和模型我们都很熟悉了，这里的媒介改由 View Model 充当。



它和 MVP 很相似:MVVM 也把 view controller 当做视图；视图和模型之间没有耦合。另外，它也像监控 MVP 那样做绑定，但是这次不是在视图和模型之间，是在视图和视图模型之间。

现在看一下 MVP 的特点：分配—在 MVVM 中，视图比 MVP 中的视图有更多的责任担当，因为 MVVM 的视图，它的状态更新是由视图模型设置绑定实现的，而 MVP 中的视图只是把事件传递给表达者，自己不做更新；可测性—视图模型不知道视图，这使得我们可以方便地检测它。视图也可以被检测，不过它是依赖于UIKit 的;易用性—在我们的例子中它与MVP 拥有相同的代码量，但是在实际的 app 中，MVP 的代码量更多，因为你需要手动把事件从视图传递到表达者，而 MVVM 可以用捆绑做到。MVVM 很有吸引力，因为它结合了之前模型的优点，并且不需要为视图的更新增加冗长的代码，并且可测性也还不错。

5 结语

我们介绍了几个架构模型，最后我们发现没有全能的最优架构，选择架构是一个在不同需求间找平衡的过程。因此，在一个应用中混合使用不同的架构是很正常的。比如：你从 MVC 开始，然后你发现某个界面用 MVC 不够高效，于是采用了 MVVM，但只是对这个特定的界面。如果其他的界面用 MVC 用得好好的，没有必要把它们也重构成 MVVC，因为它们可以很好地共存。我们追求简单的做法，但不是为了简单而简单。

参考文献

[1]王虹，刘景文，李学斯，等.基于 iOS 设备的移动 PACS 应用开发[J].中国数字医学，2014（1）：50-51.