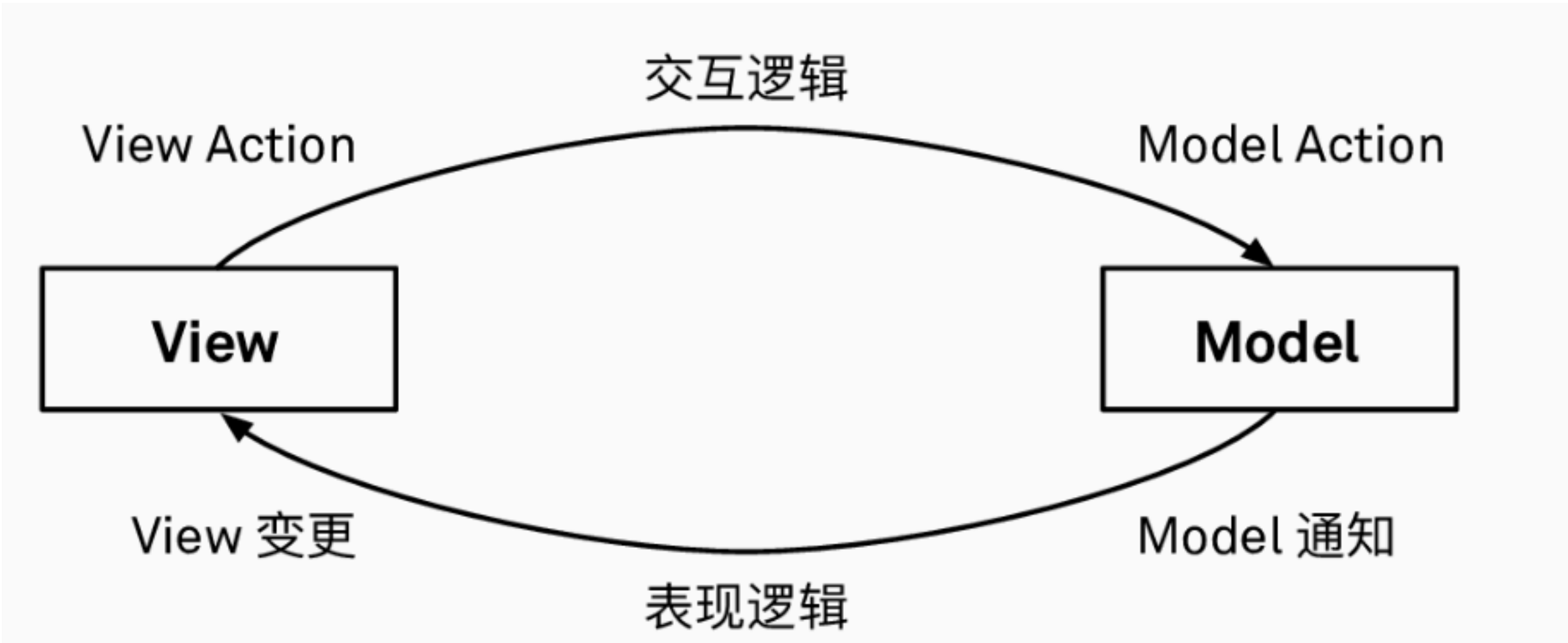
我们需要决定在 app 中如何执行下列任务：

构建 — 谁负责构建 model 和 view，以及将两者连接起来？
更新 model — 如何处理 view action？
改变 view — 如何将 model 的数据应用到 view 上去？
view state — 如何处理导航和其他一些 model state 以外的状态？

App 的本质是反馈回路

“View 层和 model 层需要交流。所以，两者之间需要存在连接。假设 view 层和 model 层是被清晰地分开，而且不存在无法解耦的联结的话，两者之间的通讯就需要一些形式的翻译：”



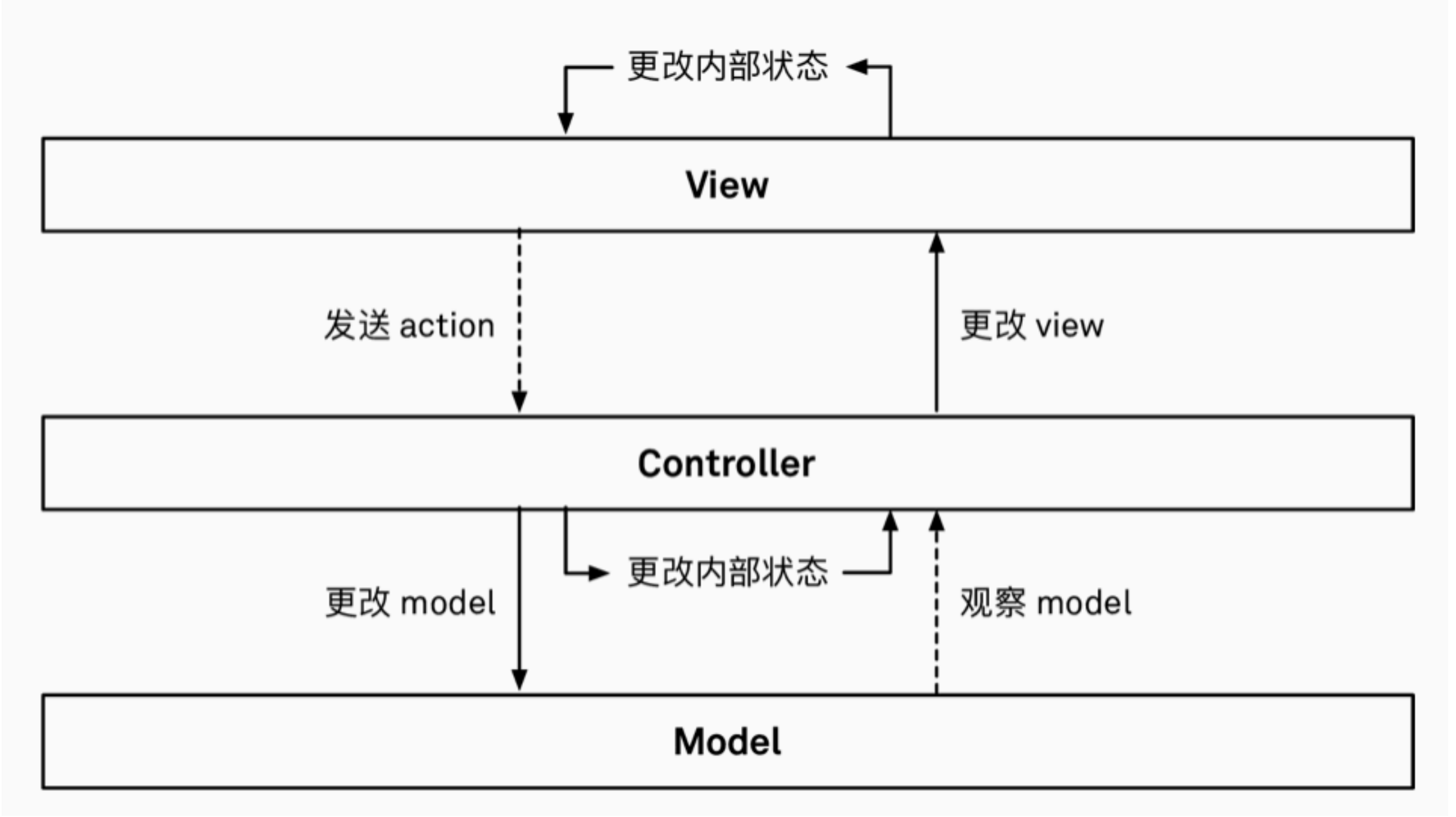
“当一个 view action 被送到 model 层时，它会被转变为model action (或者说，让 model 对象执行一个 action 或者进行更新的命令)。这种命令也被叫做一个消息 (特别在当 model 是被 reducer 改变时，我们会这么称呼它)。将 view action 转变为 model action 的操作，以及路径上的其他逻辑被叫做交互逻辑。”

“当 view 依赖于 model 数据时，通知会触发一个 view 变更，来更改 view 层中的内容。这些通知可以以多种形式存在：Foundation 中的 Notification，代理，回调，或者是其他机制，都是可以的。将 model 通知和数据转变为 view 更改的操作，以及路径上的其他逻辑被叫做表现逻辑。”

MVC

“MVC 的核心思想是，controller 层负责将 model 层和 view 层撮合到一起工作。Controller 对另外两层进行构建和配置，并对 model 对象和 view 对象之间的双向通讯进行协调。所以，在一个 MVC app 中，controller 层是作为核心来参与形成 app 的反馈回路的：”

“图中的虚线部分代表运行时的引用，view 层和 model 层都不会直接在代码中引用 controller。实线部分代表编译期间的引用，controller 实例知道自己所连接的 view 和 model 对象的接口”



MVC 中有两个最常见的问题

- 观察者模式失效
  第一个问题是，model 和 view 的同步可能失效。当围绕 model 的观察者模式没有被完美执行时，这个问题就会发生。常见的错误是，在构建 view 时读取了 model 的值，而没有对后续的通知进行订阅。另一个常见错误是在变更 model 的同时去更改 view 层级，这种做法假设了变更的结果，而没有等待 model 进行通知，如果 model 拒绝了这个变更的话，就会发生错误。这类错误会使得 view 和 model 不同步，奇怪的行为也随之而来。
- 肥大的 View Controller
  View controller 需要负责处理 view 层 (设置 view 属性，展示 view 等)，但是它同时也负责 controller 层的任务 (观察 model 以及更新 view)，最后，它还要负责 model 层 (获取数据，对其变形或者处理)。结合它在架构中的中心角色，这使得我们很容易在不经意间把所有的职责都赋予 view controller，从而迅速让程序变得难以管理。

MVC构建

1. 构建
   App 对象负责创建最顶层的 view controller，这个 view controller 将加载 view，并且知道应该从 model 中获取哪些数据，然后把它们显示出来。Controller 要么显式地创建和持有 model 层，要么通过一个延迟创建的 model 单例来获取 model。在多文档配置中，model 层由更低层的像是 UIDocument 或 NSDocument 所拥有。那些和 view 相关的单个 model 对象，通常会被 controller 所引用并缓存下来。
2. 更改 Model
   在 MVC 中，controller 主要通过 target/action 机制和 (由 storyboard 或者代码进行设置的) delegate 来接收 view 事件。Controller 知道自己所连接的 view，但是 view 在编译期间却没有关于 controller 接口的信息。当一个 view 事件到达时，controller 有能力改变自身的内部状态，更改 model，或者直接改变 view 层级。
3. 更改 View”
   “在我们所理解的 MVC 中，当一个更改 model 的 view action 发生时，controller 不应该直接去操作 view 层级。正确的做法是，controller 去订阅 model 通知，并且在当通知到达时再更改 view 层级。这样一来，数据流就可以单向进行：view action 被转变为 model 变更，然后 model 发送通知，这个通知最后被转为 view 变更。”
4. View State
   View state 可以按需要被 store 在 view 或者 controller 的属性中。相对于影响 model 的 view action，那些只影响 view 或 controller 状态的 action 则不需要通过 model 进行传递。对于 view state 的存储，可以结合使用 storyboard 和 UIStateRestoring 来进行实现，storyboard 负责记录活跃的 controller 层级，“而 UIStateRestoring 负责从 controller 和 view 中读取数据。”

MVVM

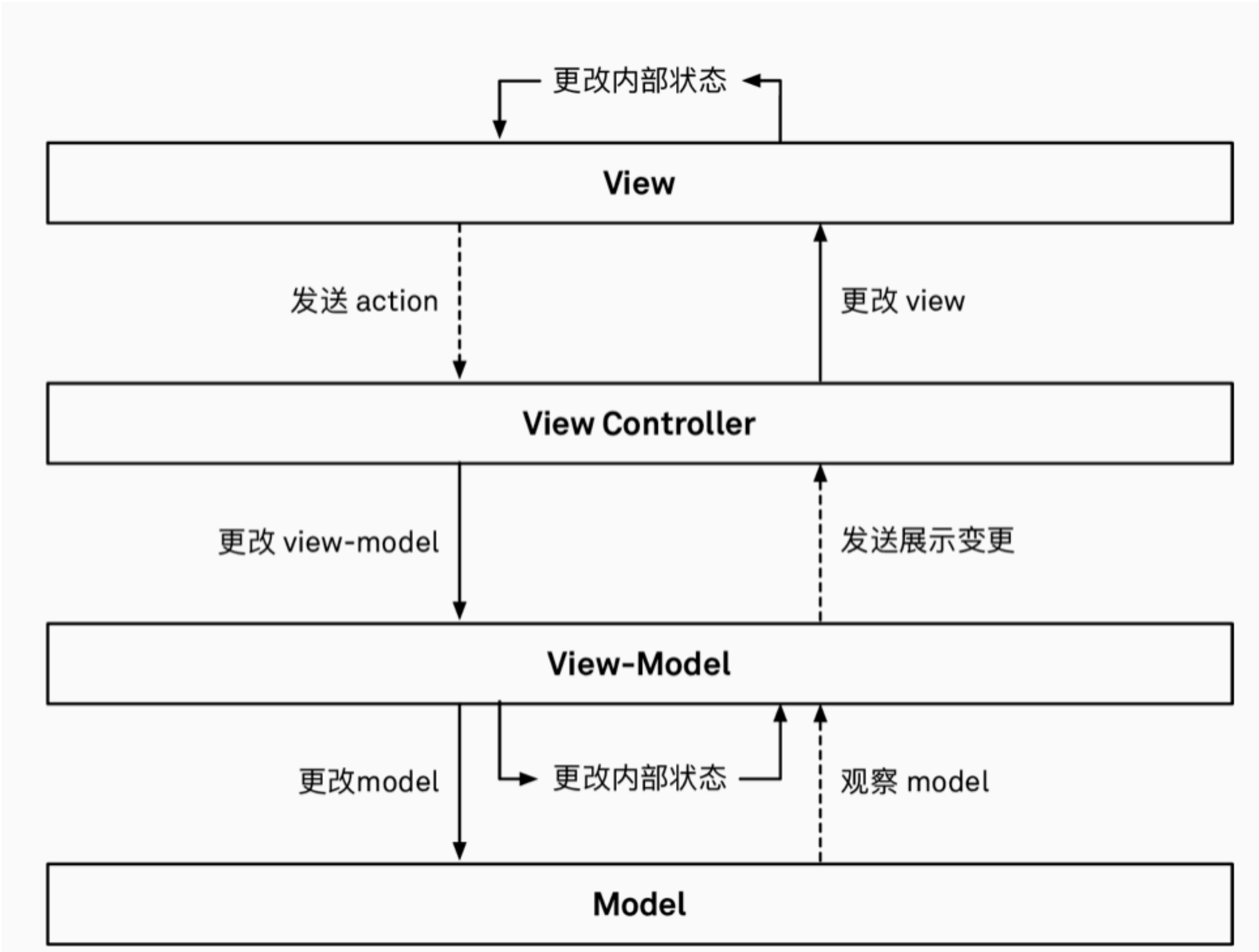
“MVVM 构建的方式和 MVC 的模式很相似：controller 层充分了解程序的结构，它使用这些认知来对所有部件进行构建和连接。
MVVM 与 MVC 最大的区别可能在于 view-model 中对响应式编程的使用了，它被用来描述一系列的转换和依赖关系。通过使用响应式编程来清晰地描述 model 对象与显示值之间的关系，为我们从总体上理解应用中的依赖关系提供了重要的指导。
具体主要有三个不同：

1. 必须创建 view-model。

2. 必须建立起 view-model 和 view 之间的绑定。
3. Model 由 view-model 拥有，而不是由 controller 所拥有。”

“MVVM 通常要求 controller 必须非常简单 (甚至简单到无需考虑)。另外，controller 必须尽可能地使用库提供的绑定方法。在这样的规则的保证下，理想情况中我们就不需要测试 controller 了，因为它没有包含我们自己的任何逻辑。究竟应该在 controller 中留存多少逻辑，根本上来说是掌握在程序员手上的。”

“MVVM 通过将 model 观察的代码以及其他显示和交互逻辑移动到围绕着数据流构建的隔离的类中，解决了 MVC controller 里不规则的状态交互所带来的有关问题。因为这是 MVC 中最显著的问题，而且会随着 Cocoa controller 的增大而恶化，这个变化在很大程度上缓解了 MVC 中 controller 肥大的问题。但是还有其他一些因素会使得 controller (以及 view-model) 变大，所以为了可持续发展，重构依然还是有需要的”



MVVM的构建

1. 构建
   和 MVC 不同的是，view controller 不再直接为每个 view 获取和准备数据，它会把这项工作交给 view-model。View controller 在创建的时候会一并创建 view-model，并且将每个 view 绑定到 view-model 所暴露出的相应属性上去。
2. 更改 Model
   在 MVVM 中，view controller 接收 view 事件的方式和 MVC 中一样 (在 view 和 view controller 之间建立连接的方式也相同)。不过，当一个 view 事件到达时，view controller 不会去改变自身的内部状态、view state、或者是 model。相对地，它立即调用 view-model 上的方法，再由 view-model 改变内部状态或者 model。
3. 更改 View
   和 MVC 不同，view controller 不监听 model。View-model 将负责观察 model，并将 model 的通知转变为 view controller 可以理解的形式。View controller 订阅 view-model 的变更，这通常通过一个响应式编程框架来完成，但也可以使用任意其他的观察机制。当一个 view-model 事件来到时，由 view controller 去更改 view 层级。
   为了实现单向数据流，view-model 总是应该将变更 model 的 view action 发送给 model，并且仅仅在 model 变化实际发生之后再通知相关的观察者。
4. View State
   View state 要么存在于 view 自身之中，要么存在于 view-model 里。和 MVC 不同，view controller 中不存在任何 view state。View-model 中的 view state 的变更，会被 controller 观察到，不过 controller 无法区分 model 的通知和 view state 变更的通知。当使用协调器时，view controller 层级将由协调器进行管理。
5. 测试

因为 view-model 和 view 层与 controller 层是解耦合的，所以可以使用接口测试来测试 view-model，而不需要像 MVC 里那样使用集成测试。接口测试要比集成测试简单得多，因为不需要为它们建立完整的组件层次结构。
为了让接口测试尽可能覆盖更多的范围，view controller 应当尽可能简单，但是那些没有被移出 view controller 的部分仍然需要单独进行测试。在我们的实现中，这部分内容包括与协调器的交互，以及初始时负责创建工作的代码。”

关于view-model

- 实现状态恢复数据
  在状态恢复的方法上，MVVM 中为各个 controller 所存储的数据来源于 view-model，而非像 MVC 那样来源于 controller 本身，除此之外，两者所使用的策略大抵相同。
- View-model 虽然名字里既有 view 又有 model，但是它所扮演的其实是不折不扣的类似 controller 的角色。
- View-model 从 view controller 和 view 中独立出来，也可以被单独测试。同样，view controller 也不再拥有内部的 view state，这些状态也被移动到了 view-model 中。在 MVC 中 view controller 的双重角色 (既作为 view 层级的一部分，又负责协调 view 和 model 之间的交互)，减少到了单一角色 (view controller 仅仅只是 view 层级的一部分)。
- “View-model 在编译期间不包含对 view 或者 controller 的引用。它暴露出一系列属性，用来描述每个 view 在显示时应有的值。把一系列变换运用到底层的 model 对象后，就能得到这些最终可以直接设置到 view 上的值。实际将这些值设置到 view 上的工作，则由预先建立的绑定来完成，绑定会保证当这些显示值发生变化时，把它设定到对应的 view 上去。响应式编程是用来表达这类声明和变换关系的很好的工具，所以它天生就适合 (虽说不是严格必要) 被用来处理 view-model。在很多时候，整个 view-model 都可以用响应式编程绑定的方式，以声明式的形式进行表达。
- “在理论上，因为 view-model 不包含对 view 层的引用，所以它是独立于 app 框架的，这让对于 view-model 的测试也可以独立于 app 框架。”

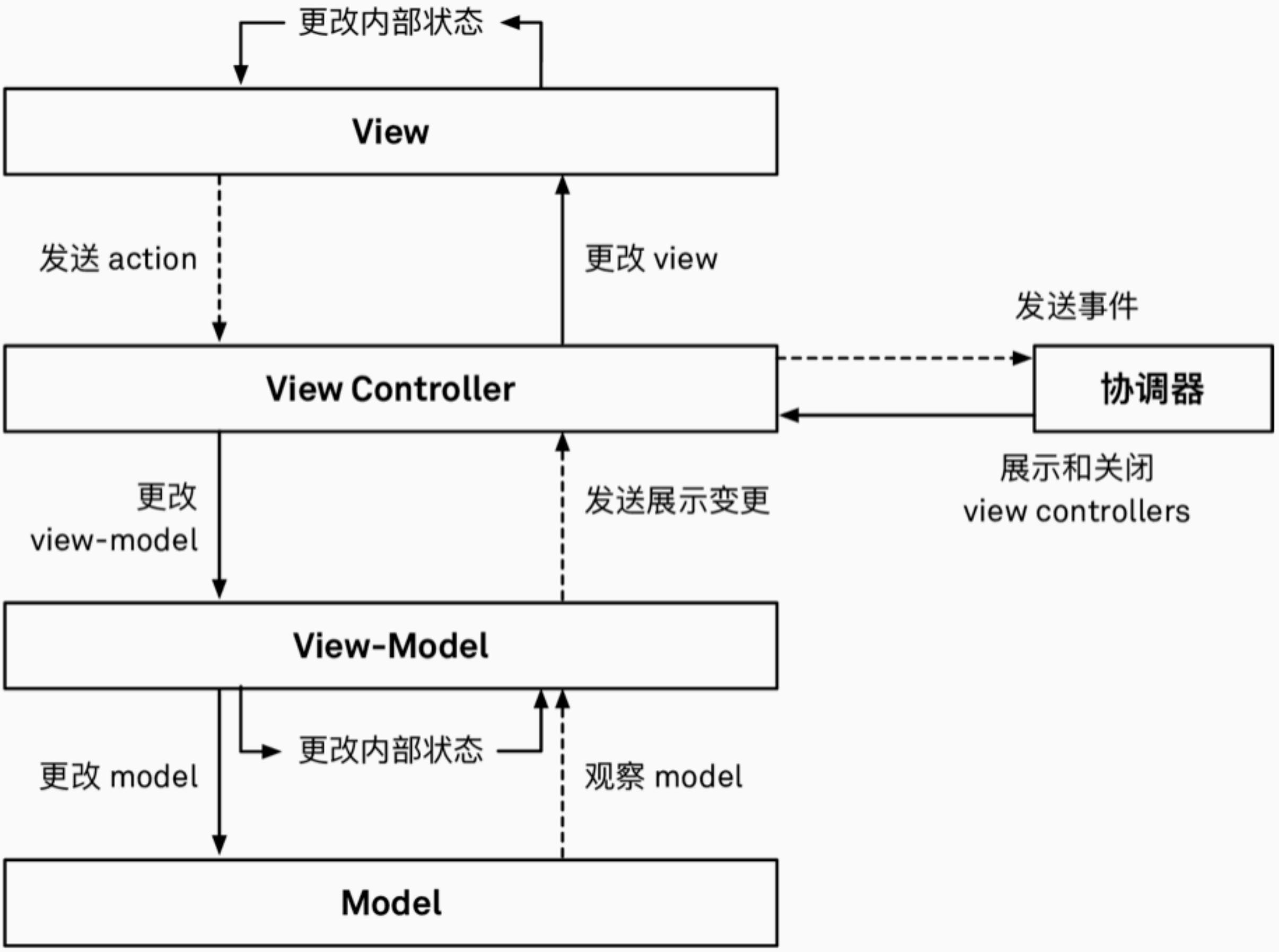
MVVM-C

“在理论上，因为 view-model 不包含对 view 层的引用，所以它是独立于 app 框架的，这让对于 view-model 的测试也可以独立于 app 框架。

由于 view-model 是和场景耦合的，我们还需要一个能够在场景间切换时提供逻辑的对象。在 MVVM-C 中，这个对象叫做协调器 (coordinator)。协调器持有对 model 层的引用，并且了解 view controller 树的结构，这样，它能够为每个场景的 view-model 提供所需要的 model 对象。”

和 MVC 不同，MVVM-C 中的 view controller 从来都不会直接引用其他的 view controller (所以，也不会引用其他的 view-model)。View controller 通过 delegate 的机制，将 view action 的信息告诉协调器。协调器据此显示新的 view controller 并设置它们的 model 数据。换句话说，view controller 的层级是由协调器进行管理的，而不是由 view controller 来决定的。如果我们忽略掉协调器，那么这张图表就很像 MVC 了，只不过在 view controller 和 model 之间加入了一个阶段。MVVM 将之前在 view controller 中的大部分工作转移到了 view-model 中，但是要注意，view-model 并不会在编译时拥有对 view controller 的引用。

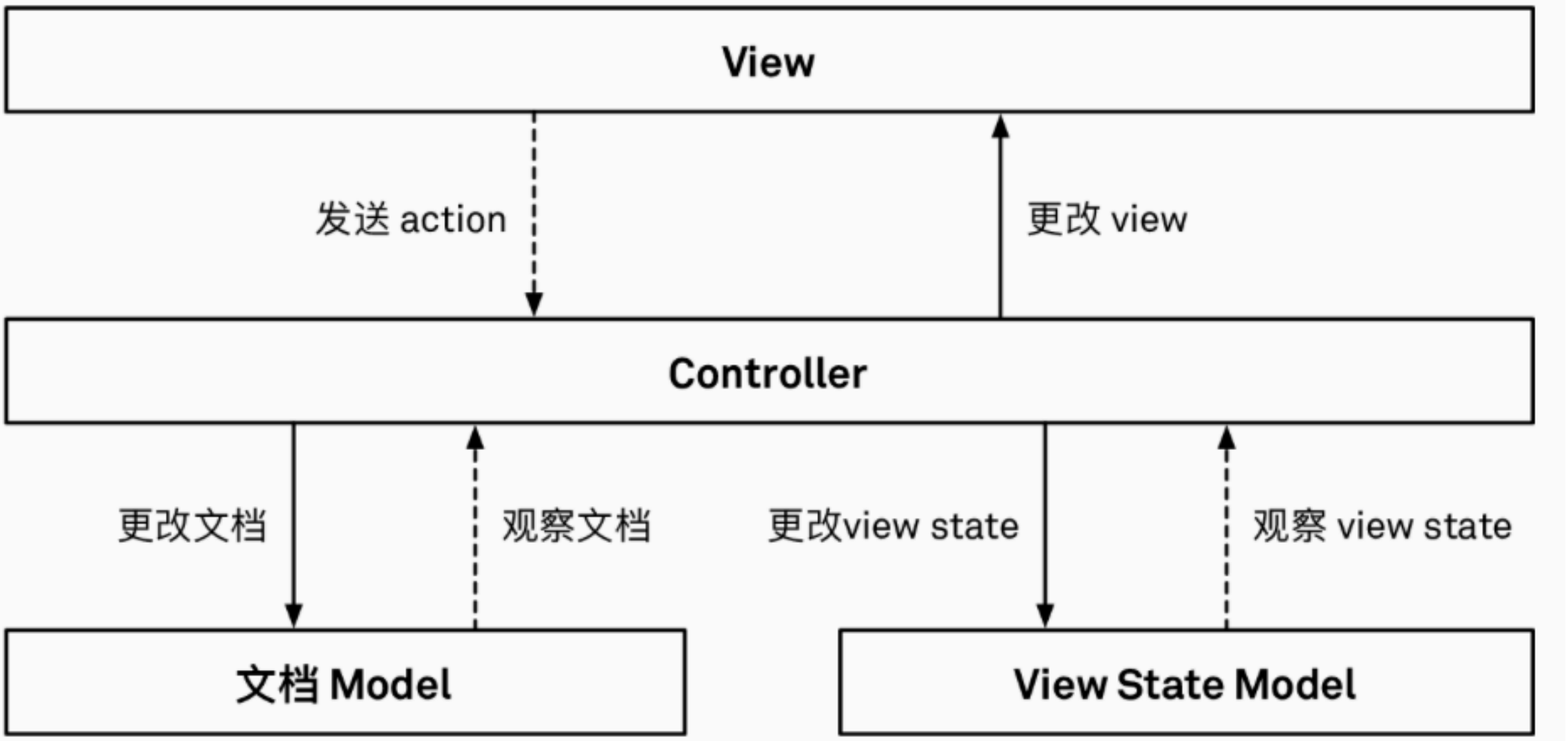
“初步印象来说，因为 MVVM-C 加入了额外的一层来进行管理，看起来是比 Cocoa MVC 模式更加复杂。不过，在实现的层级，如果你能够始终如一地贯彻这个模式，代码会变得更简单一些。啊，这里说的简单并不意味着容易，只有当你对常见的响应式代码变形熟悉以后，才不会对书写代码感到无从下手，才不会对调试问题感到懊恼沮丧。不过，从令人高兴的一面来说，精心设计的数据管道通常不容易产生错误，在长期来看维护也更容易一些。”



“iOS 中的协调器是一种很有用的模式，因为管理 view controller 层级是一件非常重要的事情。协调器在本质上并没有和 MVVM 绑定，它也能被使用在 MVC 或者其他模式上。”

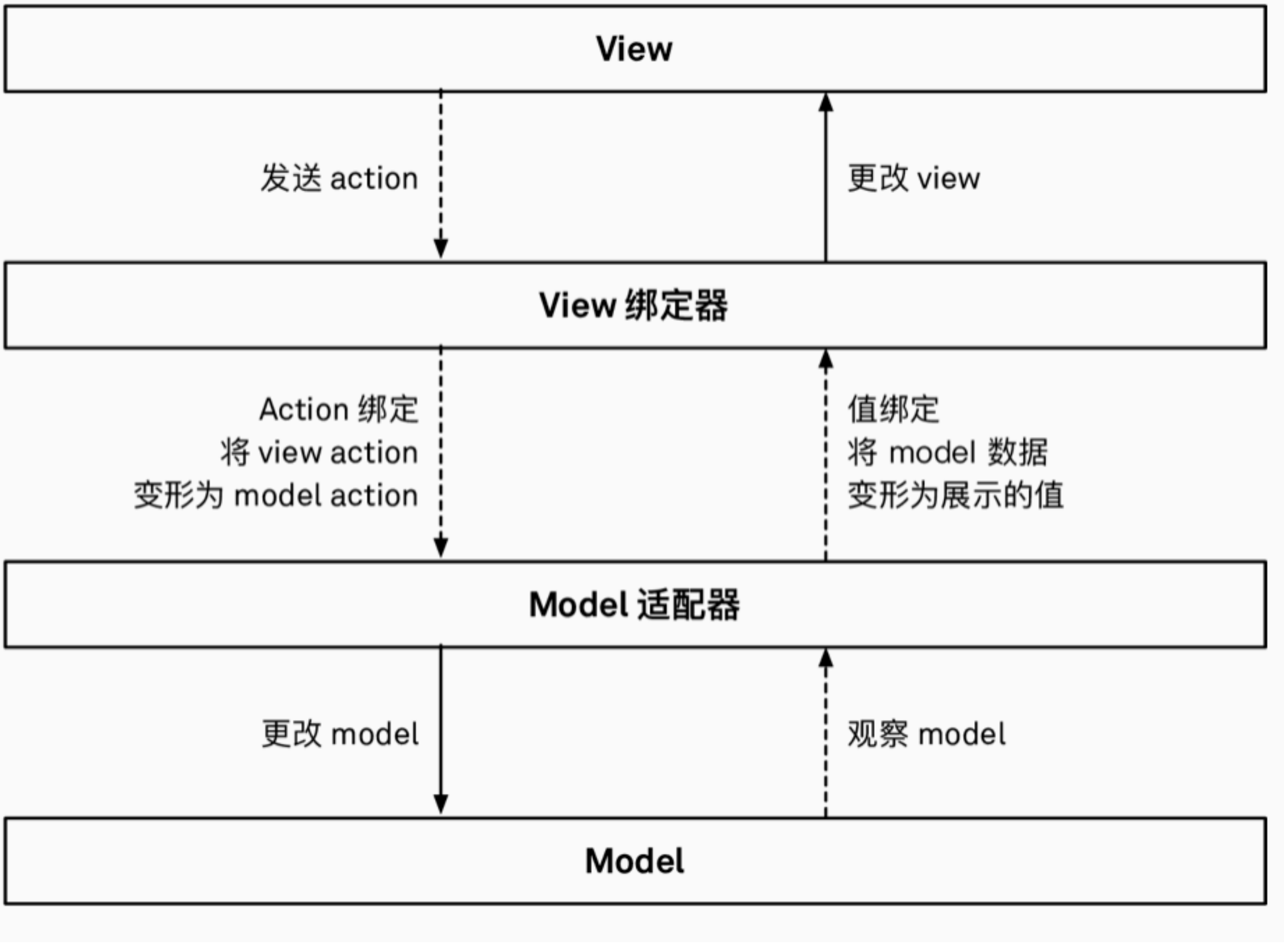
协调器为每个 controller 的 view-model 设置初始的 model 对象。
View-model 将设定值和其他 model 数据及观察量进行合并。
View-model 将数据变形为 view 所需要的精确的格式。
Controller 将准备好的值绑定到各个 view 上去。

MVC-VS



MAVB

“model 适配器 - view 绑定器 (ModelAdapter-ViewBinder, MAVB)”



Elm 架构 (TEA)

MVP，VIPER，Riblets

其他

响应式编程

> “响应式编程是一种用来交流变更的工具，不过和通知或者 KVO 不同的是，它专注于在源和目标之间进行变形，让逻辑可以在部件之间传输信息的同时得以表达。”
> “响应式编程是一种用来描述数据源和数据消费端之间数据流动的模式”
> “数据变形的部分是响应式编程所能带来的最大优势，但同时它也是学习曲线最为陡峭的部分。”