# 《恰如其分的软件结构》读书笔记

### 书籍概览

《恰如其分的软件结构》是一本深入探讨软件架构设计原则和实践的书籍。它不仅提供了软件架构的理论基础，还结合实际案例，展示了如何在现实世界中应用这些原则。这本书对于那些希望提升软件设计技能的开发者和架构师来说，是一本不可多得的资源。

## 核心理念

### 架构的重要性

这本书让我深刻认识到，“软件架构不仅仅是代码的组织方式，更是确保软件系统能够满足业务需求、技术要求和未来可维护性的关键。”架构设计的好坏直接影响到项目的成败。

### 恰如其分的设计

作者强调“恰如其分”的设计，意味着在正确的时候做出恰当的设计决策，既不过度设计，也不忽视潜在的问题。这种平衡感是架构师需要不断修炼的技能。

## 架构原则

### 模块化

模块化是软件架构中的一个核心概念。书中通过多个案例展示了如何通过模块化来提高代码的可维护性和可扩展性。我特别喜欢作者提到的“高内聚，低耦合”原则，这让我在设计系统时更加注重组件之间的边界和交互。

### 可测试性

可测试性是衡量软件架构好坏的一个重要标准。书中提供了多种策略来提高系统的可测试性，包括设计可测试的接口、使用模拟对象等。这些策略让我意识到，“好的架构设计应该从测试的角度出发，以确保软件质量。”

## 实践应用

### 风险驱动模型

在阅读过程中，我对风险驱动模型感触颇深。这个模型强调了“架构设计的度应该与项目中面对的风险成正比。”这意味着，如果项目失败的风险很高，那么我们在架构设计上就应该投入更多的精力。这种思想让我在面对复杂项目时，能够更加有针对性地分配资源和注意力。

### 螺旋模型与风险驱动模型的区别

我之前对螺旋模型和风险驱动模型有所混淆，但书中清晰地区分了两者。螺旋模型用于完整的软件开发过程，而“风险驱动模型仅用于设计阶段”，这让我意识到，即使在敏捷开发的环境中，也可以有效地应用风险驱动模型。

### 架构与设计细节

书中提到，“一份有关架构的详细说明可能是宏观与微观细节的混合物，只要细节关乎系统的整体质量，它就可能属于架构层面的内容。”这让我认识到，架构设计不仅仅是宏观层面的规划，也包括对关键细节的深入考量。

### 推定架构

了解本领域的推定架构可以避免一些在设计上的无用功，避免过度专注于细节。这一点对我来说尤为重要，因为我经常沉迷于细节中无法自拔。通过理解推定架构，我可以更有效地识别和专注于那些对项目成功至关重要的设计决策。

### 技术选择

书中还讨论了技术选择的重要性。有些风险可以通过多种技术去缓解，而另一些风险甚至需要立即发明一些技术才能解决。这让我意识到，在选择框架和技术时，不应该只满足于可用性，而是要与本来可以用到的其他技术做比较，否则会浪费机会成本。

## 个人感受

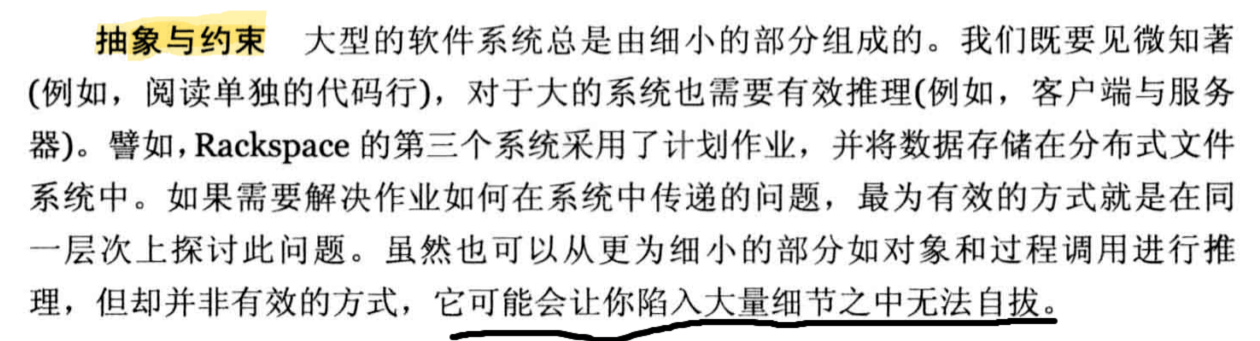
读这本书的过程中，我不断地反思自己过去的项目经验。我发现，很多时候我们过于关注功能的实现，而忽视了架构设计的长远影响。这本书提醒我，“作为一名软件工程师，我们需要有更广阔的视野，不仅要关注代码，还要关注代码背后的架构和设计。”

## 结语

《恰如其分的软件结构》是一本值得一读再读的书。它不仅提供了丰富的理论知识，还通过实际案例教会我们如何在实际工作中应用这些知识。作为一名软件从业者，我深感架构设计的重要性，并且这本书给了我很多启发和指导。我推荐每一位对软件架构感兴趣的人都应该读一读这本书。

## 附录

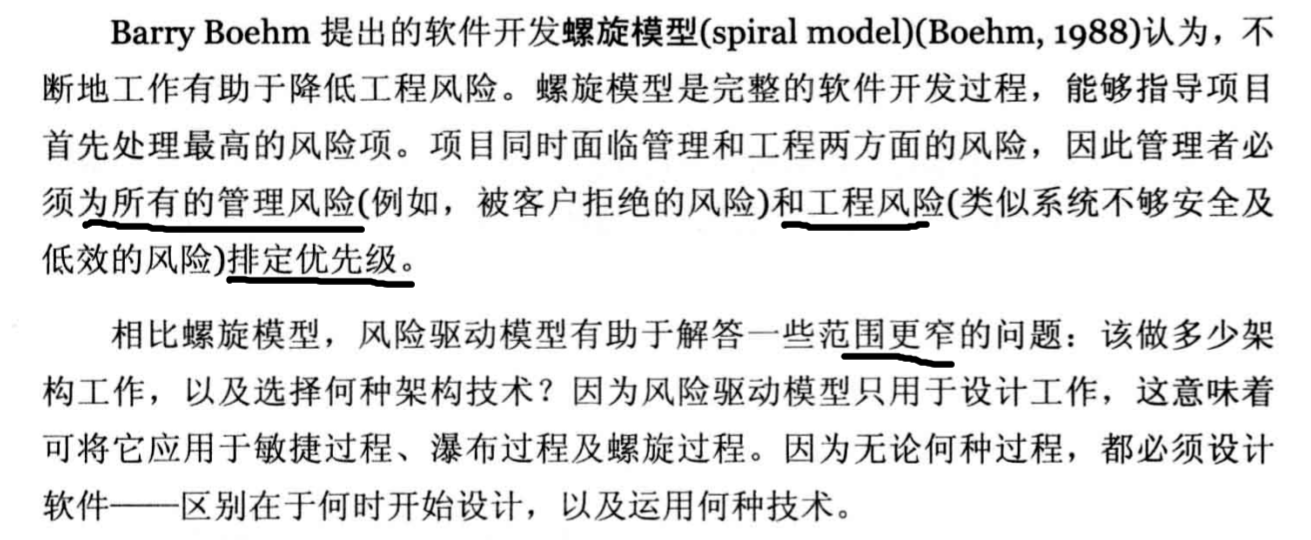
此处附一些比较杂的个人摘抄：



这里感触很深，我就经常沉迷于细节中无法自拔。

**风险驱动模型：**

为了把握架构的度（既不过度使用，也不完全忽略架构），可以用风险来权衡：失败的风险有多高，我们付出的努力就有多大。



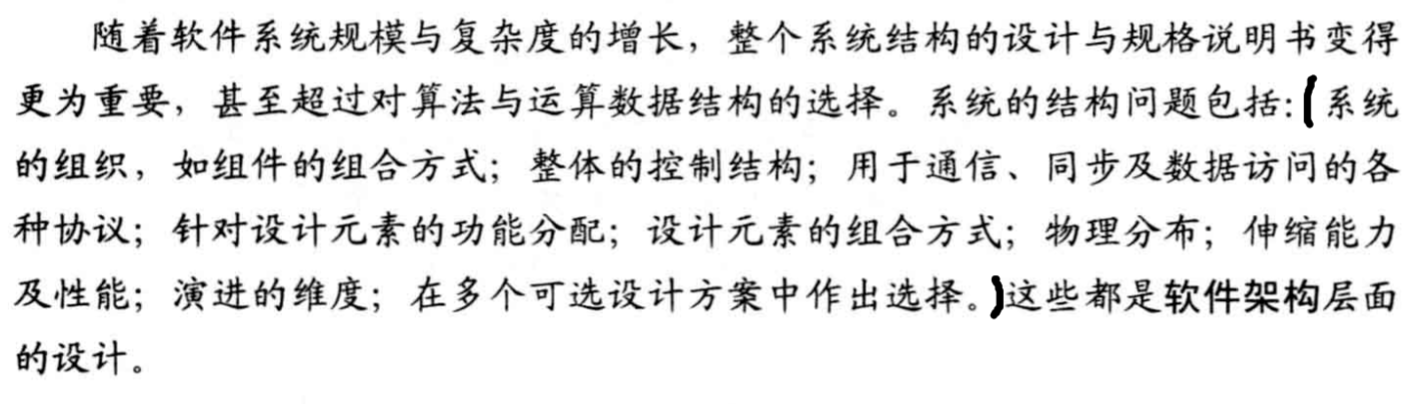
螺旋模型与风险驱动模型的区别：

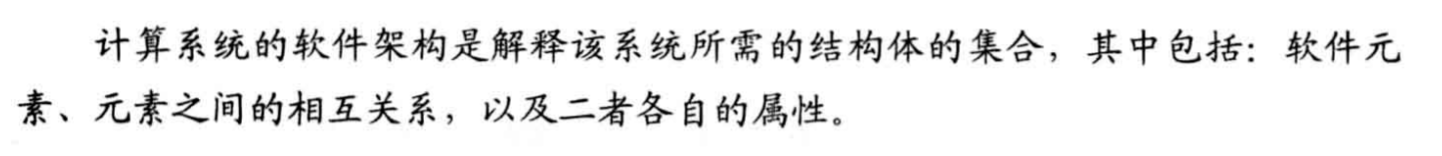
前者用于完整的软件开发过程，“风险”包括管理和工程两方面。

后者只用于设计阶段，和其他过程模型并不冲突，喜欢用敏捷开发的程序员也可以使用！

**软件架构**

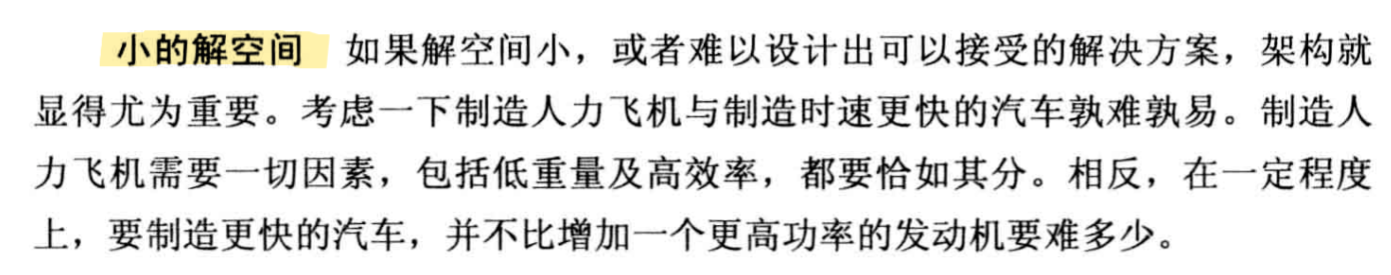
架构与设计细节：

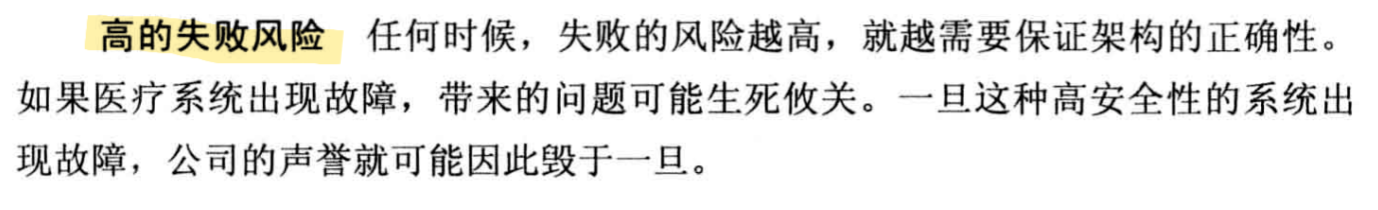




一份有关架构的详细说明可能是宏观与微观细节的混合物，只要细节关乎系统的整体质量，它就可能属于架构层面的内容。

**架构何时重要？**

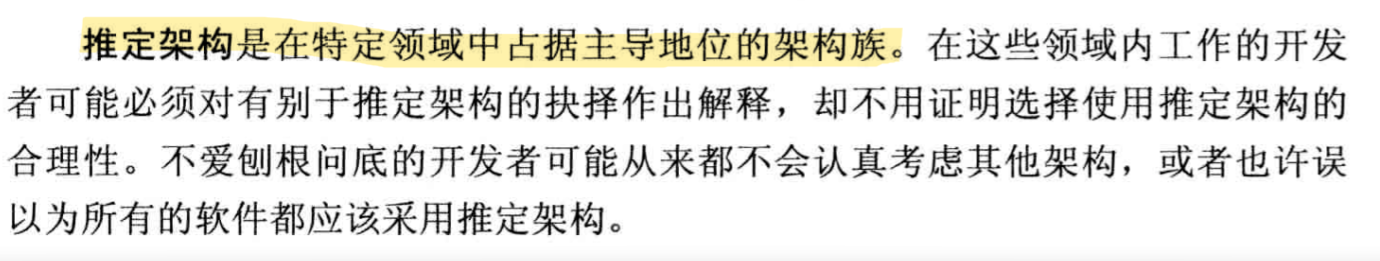




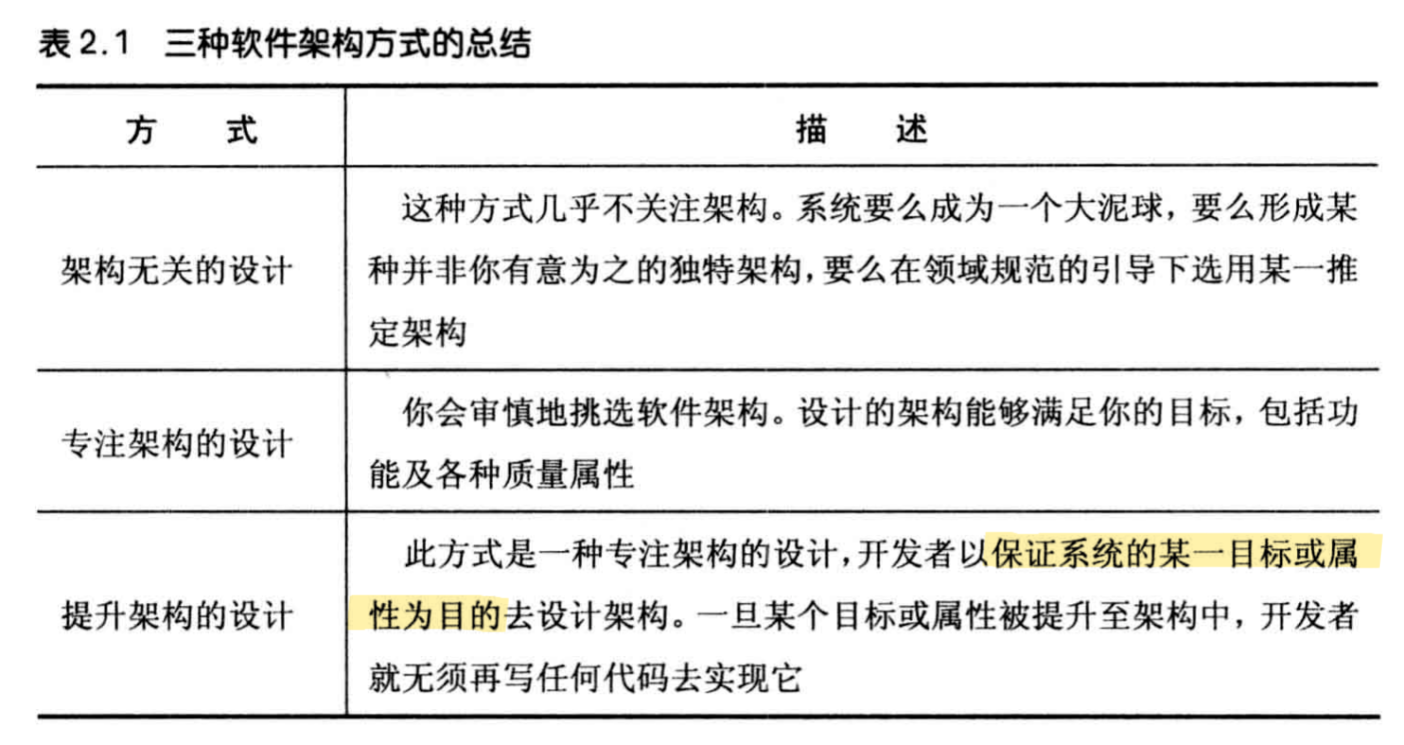




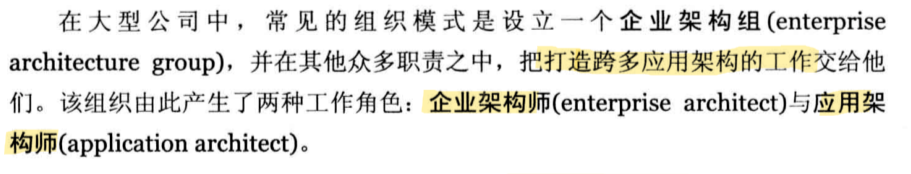
**推定架构**



了解本领域的推定架构可以避免一些在设计上的无用功，避免过度专注于细节。



注意，提升架构的设计可以用于任何软件开发过程的设计阶段，它是与设计模块、对象或数据结构类似的工程任务。

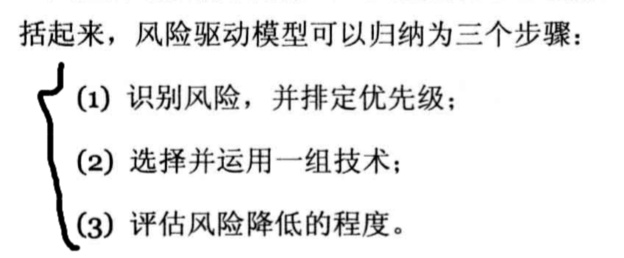






**风险驱动模型**

它的本质思想在于:你设计软件架构所付出的精力应与你在项目中面对的风险成正比。

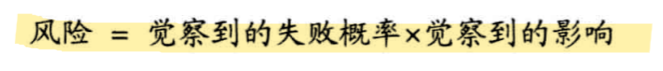


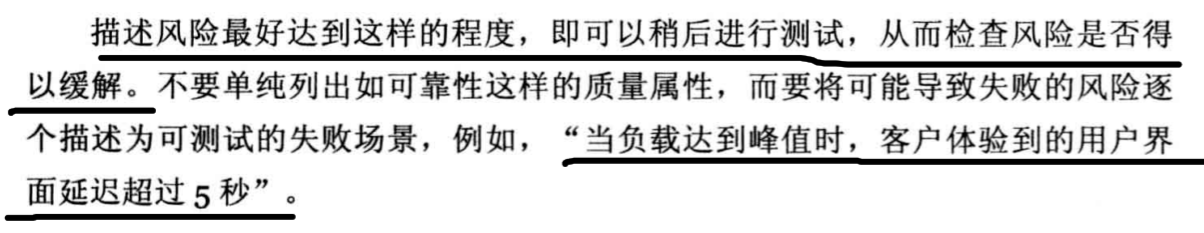
1. 需要以风险或特征为中心
2. 每个开发者对于风险的认知是主观的，但通过这个模型可以产出一系列可供评估的论据。

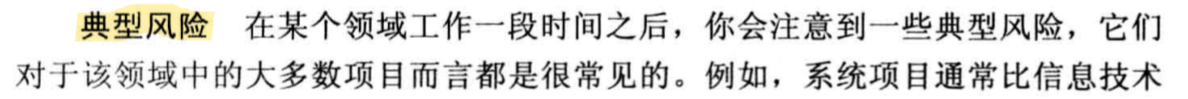
检验自己是否采用了风险驱动模型：能够将自己所面对的风险以及采用的对应技术罗列出来。

并且，采用风险驱动模型的系统应该①有多种技术选择②避免滥用标准化的过程或模板。

**3.3风险**

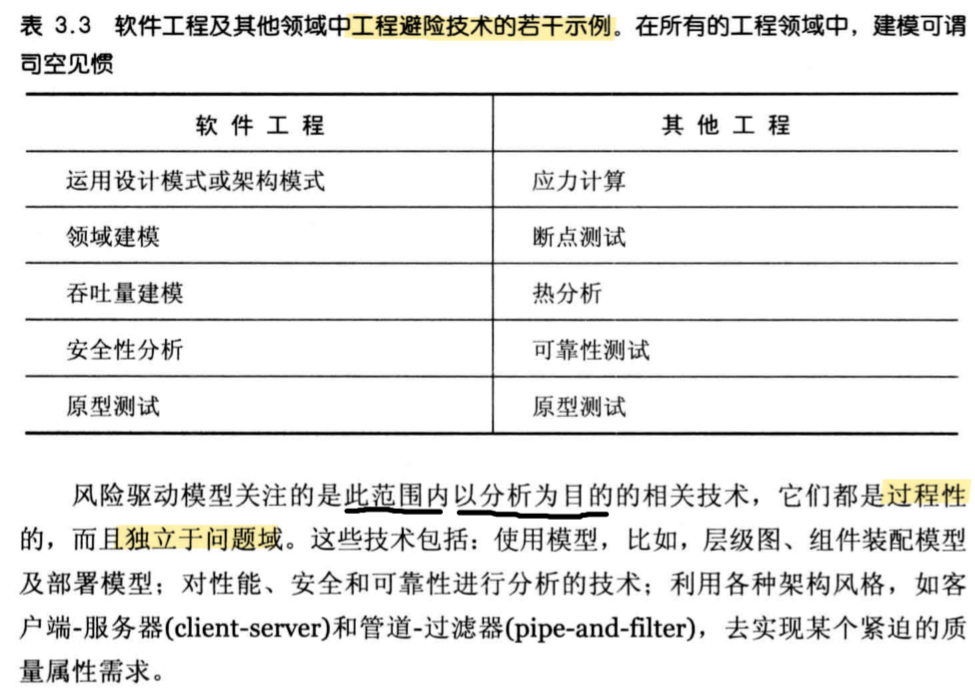






存在疑问，“利益相关者”指什么？

**技术**

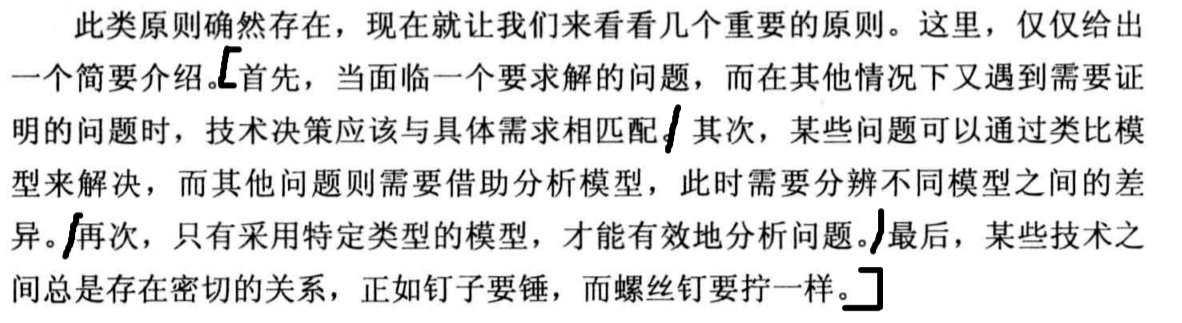


有些风险可以通过多种技术去缓解，而另一些风险甚至需要立即发明一些技术才能解决。

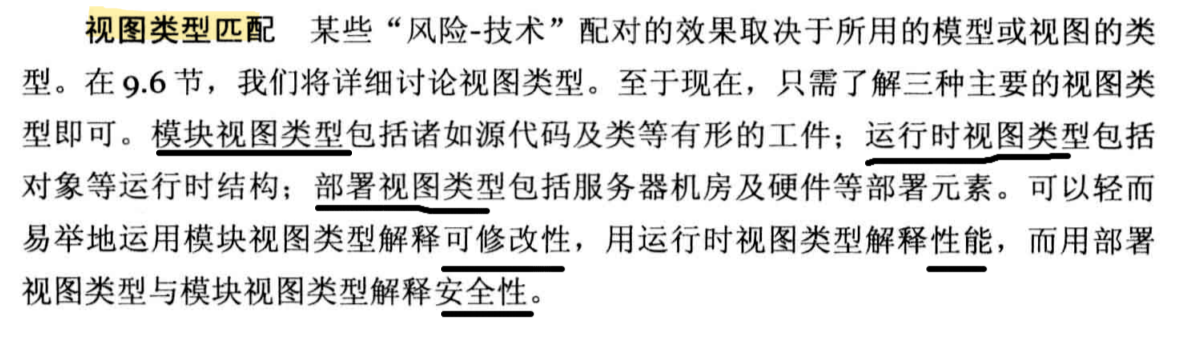
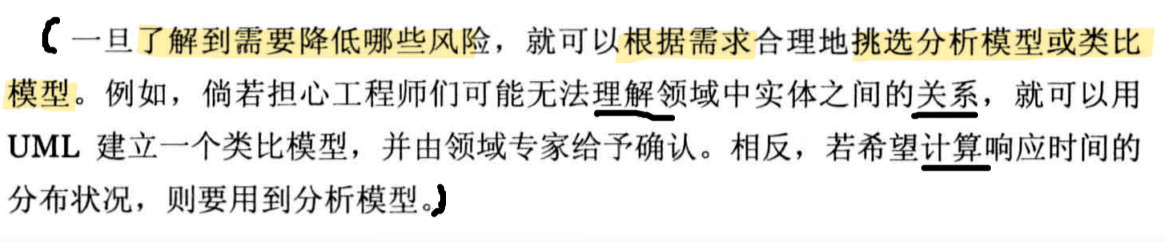
选择框架的时候不应该只满足于可用性，而是要与本来可以用到的其他技术做比较，否则会浪费机会成本。

风险是无法根除的，因为在规避风险的同时必须顾及时间与成本。

**选择技术的指导原则**



很多时候只需要设计（找到）一个解决方案，而无需证明这个设计是最优的。



**何时停止**

