软件架构：可控的灵活性 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/141/2021_2022__E8_BD_AF_E4_BB_B6_E6_9E_B6_E6_c29_141314.htm 在软件开发中，我们对于软件架构经常看到极端，要么不重视软件架构，要么过分重视以至于她成了“ 天条 ”。我甚至遇到了这样的事情，某公司强制推行某基于Struts的架构设计，然而到了项目组它却处处遭到抵制，特别是分部基本上抛弃了这个架构设计。那么，这个原因在哪里呢？为什么一个成本高昂的架构设计没有被接纳呢？ 实际上有时候一个良好的设计也未必会被接纳，特别是没有Java开发实际经验甚至缺乏软件开发经验的项目经理试图控制技术的时候更加如此。我们抛开这个可能的影响来看待这个问题。 我们发现，很多的设计人员在做软件架构设计的时候忽略了几个重要的问题： 1.软件的架构在大的方向上是固定的 这种固定依据某些基本固定的软件模式（包括设计模式、编码模式或者某些特定的约定）来强化和固定。这使得软件开发在某种程度上具有某些可以明显看得到的风格，这个风格被整个的项目贯穿和坚持，这样当我们进入通常可怕的维护或者调整的时候，我们不会害怕我们在很多种不同的实现中常常迷失了方向。 2.软件的架构在具体的应用中可以适度的可控灵活 毫无疑问，软件设计开发不可能没有例外，在某种程度上保留一些适度的灵活时必要的。一方面，它符合软件开发的实际；一方面，适度的可控灵活体现了一种对实现者的尊重和信任。然而，如何实现可控的灵活呢？通常，我认为可以考虑有限种类的设计选择并通过有色彩的图形来表明这种可选择性。但即便如此，设计者很重

要的要考虑这些不同选择之间的关系，以及这些选择和整体设计的兼容性，这个时候，面向接口的设计和适度的“粘合”设计是必要的。 来源：www.examda.com 3.架构设计的支撑架构设计通常会涉及到一些支撑设计，这些设计和实现水准直接的体现了系统的最有价值的部分，更细的划分我们应该可以看到性能和结构相关的部分，以及必要的基础类。通常，作为设计人员，我们应该能够设计并实现系统的性能、结构、扩展、粘合等部分的工作，这部分的工作通常不应该离开设计人员的控制和把握，这些代码的书写或者至少积极的关注是设计人员必要的素质.我们不应该让更多的看到我们无法写出代码来（我也反对没有分工什么都做的设计人员！），实际上，这是软件最困难也是最有乐趣的地方。特别是在测试结果中发现这些设计实现带来的性能的极大提高的时候是非常有成就感的事情。至于基础类，应该是尽可能的减少依赖和耦合的。架构设计的支撑要尽可能的对客户程序员隐藏不必要的中间细节，对基础类要尽量简单、稳定并真正需要。 通常，新兴的技术会用在架构设计的支撑设计里面并被封装以隐藏具体的实现，而通盘应用新兴技术的风险是巨大的，并且对人员的获得也是问题。有些技术是可以提供替代技术的，一些新兴的技术实现也是可以参考的（但未必采用她的实现）。要记住软件开发是需要速度、质量、可维护而不是技术表演，这个度应该由分析设计人员负责任的来把握。 