《软件构架实践》

图书基本信息

书名：《软件构架实践》

13位ISBN编号：9787302312932

10位ISBN编号：7302312931

出版时间：2013-2

出版社：清华大学出版社

页数：589

《软件构架实践》

内容概要

《软件构架实践》

作者简介

作者:(美)巴斯、克莱门茨、凯兹曼

《软件构架实践》

书籍目录

《软件构架实践》

章节摘录

版权页： 插图： Increase Cohesion Several tactics involve moving responsibilities from one module to another. The purpose of moving a responsibility from one module to another is to reduce the llkelihnnd of side effects affecting other responsibilities in the original module. Increase semantic coherence. If the responsibilities A and B in a module do not serve the same purpose, they should be placed in different modules. This may involve creating a new module or it may involve moving a responsibility to an existing module. One method for identifying responsibilities to be moved is to hypothesize likely changes that affect a module. If some responsibilities are not affected by these changes, then those responsibilities should probably be removed. Reduce CouDlina We now turn to tactics that reduce the couoling between modules. Encapsulate. Encapsulation introduces an explicit interface to a module. This interface includes an application programming interface (API) and its associated responsibilities, such as "perform a syntactic transformation on an input parameter to an internal representation." Perhaps the most common modifiability tactic, encapsulation reduces the probability that a change to one module propagates to other modules. The strengths of coupling that previously went to the module now go to the interface for the module. These strengths are, however, reduced because the interface limits the ways in which external responsibilities can interact with the module (perhaps through a wrapper). The external responsibilities can now only directly interact with the module through the exposed interface (indirect interactions, however, such as dependence on quality of service, will likely remain unchanged). Interfaces designed to increase modifiability should be abstract with respect to the details of the module that are likely to change that is, they should hide those details.

《软件构架实践》

编辑推荐

《软件构架实践》

精彩短评

1、这本书是英文版的，却搞了个中文书名，买中文版的朋友看仔细了再买啊
2、前封面右下角上没有图上的激光防伪码
3、Software architecture in Practice, a very good book in software architecture design!
4、首先，这是一部软件架构方法学的权威书籍！2002年买了一本《软件构架实际》（第二版）的中文版，看完了，觉得写的非常好：理论方法系统，理论与实践结合紧密，方法可操作性强，难得的是有作者的研究创新内容（作者本身就是一线的高级研发人员）。第三版是2013年初才出来的最新版本，这版的内容组织上与第二版有较大的差异，但更紧凑、更系统、更完整。经过软件架构这一学科经过10多年的发展，已经比较成熟了，这本书第三版正是反应了最新的软件架构设计和分析方法学的一本比较全面和完整的书籍。加上纸张和应刷不错，物超所值！
5、详细内容上前面写的简体中文/英文，简体中文应该是优先发送的吧。可寄给我的是英文版，请问能更换吗？

《软件构架实践》

章节试读

1、《软件构架实践》的笔记-第24页

The software architecture of a system is the set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both.
开篇第一章的主要目的是初步介绍软件构架的概念，所以定义是非常重要的，起到提纲携领的作用。上段原文是作者最倾向的一种定义，第一章就围绕这个定义展开。

第一章中必须理解的两个基本概念（术语）：结构（Structure）和视图（View）。
A structure is a set of elements and the relations of them. A view a represenation of a coherent set of architectural elements, as written by and read by system stakeholders. A view is a representation of one or more structures.
结构分为三种类型：模块结构、组件-连接器结构（Component-and-Connector structures, C&C）、分配结构（Allocation structures）。

针对于三种主要的结构类型，第一章给出了详细的解释、应用的场景和一些有用的例子。比如，属于模块类型的分解结构（Decomposition structure）、使用结构（Uses structure）、分层结构（Layer structure）、类结构（Class or generalization structure）、数据模型（Data model）；属于C&C类型的服务结构（Service structure）、并发结构（Concurrency structure）；属于分配类型部署结构（Deployment structure）、实现结构（Implementation structure）、工作分配结构（Work assignment structure）等等。紧接着，基于这些结构，作者给出了它们之间的关系，并强调Fewer is Better，进而强调了选择各种结构的原则：provide insight and leverage into the system's most important quality attributes
最后，谈到了构架设计的一些可以重用的模式（Patterns）以及如何评价是不是一个“优良的”构架，当然，因为是第一章，原则性或概要性的内容较多，可能在后续章节会有更多细节展开。

其中个人比较喜欢的是，作者会插入一些注解（Side Note），形式可能是概念细节，例如，相比Software Architecture，Software Architecture和Enterprise Architecture又有什么不同？或者是小故事，比如，作者RK参与的项目架构质量评估的有趣经历。

总体来看，行文相对易懂，虽然有些专深的叙述一时不容易理解，但是随着后面的阅读，应该会逐渐明朗。

2、《软件构架实践》的笔记-第63页

书的第一部分是介绍，也就是开个题，围绕着架构这个概念作了较细致的讲解，主要是：什么是架构、架构为什么重要、架构和人以及环境的关系。在深入理解了架构之后，我们就知道架构真的是在软件开发过程中不可或缺的部分，它紧密联系着各路利益相关者和真实的软件系统，联系着软件开发的多个方面如质量、计划、人员组织和各种开发活动。恩，总之挺重要。其中的AIC(Architecture Influnce Cycle)和Stakeholders的列表让人印象深刻。

书的第二部分，重点是讲质量属性（Quality Attributes），即架构最重要的七种属性可用性、互操作性、可修改性、性能、安全、可测试性、可用性以及其他。在详细研究之前，作者给出了质量属性的定义，抄录如下：A quality attribute (QA) is a measuable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders.
应该是在第一部分曾经讲过为什么这些重要属性里面居然没有functionality，在这里又重提一下给出了解释，不过这个可以作为一个问题继续思考，为什么呢？

3、《软件构架实践》的笔记-第185页

《软件构架实践》

软件架构的基本质量属性从第4章讲起一直到第11章，讲了最核心的几种：可用性(Availability)、互操作性、可修改性、性能、安全、可测试性以及可用性(Usability)，每一章都不长，使用固定的结构讲解，即通用场景、技巧、检查列表、扩展阅读和问题讨论，作者用短短十几页来讲一种质量属性，你是不用期望能讲多细的。但是所给出的要点却是非常有用，因为这样你可以从宏观的角度把握，如果我想在系统的某个属性的某个侧面做到什么程度，我可以采用什么方法。

虽然零零散散的阅读记忆不深，但是这些质量属性在设计架构的时候要知道，然后顺藤摸瓜，自然会由浅入深，由粗见细，慢慢掌握了。

第12章讲解其他的质量属性，有可变性、可移植性、开发分布能力、扩展性、弹性、部署能力、移动性和可检测能力。

英文书读的慢，忘得快，共勉！

为了避免误导，给出英文原文：
Availability, Interoperability, Modifiability, Performance, Security, Testability, Usability, [B-list] variability, portability, development distributability, scalability, elasticity, deployability, mobility, monitorability.

4、《软件构架实践》的笔记-第137页

一路从可用性，互操作性，可修改能力，一直看到性能，确实有干货。作者的知识果然已经不是应用性的知识，而是从经验和思考里提炼出来的精华。虽然，有些看得似懂非懂，有些地方似是而非，但总体上还是有收获。凝炼。

5、《软件构架实践》的笔记-第275页

架构在敏捷项目里怎么做？多少才算足够？这一章篇幅不长，却清晰地回答了这个问题。好！

6、《软件构架实践》的笔记-第38页

第二章主要是解释了一下软件架构的重要性，作者居然列出了13条，厉害。不过看完之后，都很赞同，13条具体是啥，忘得差不多了。

7、《软件构架实践》的笔记-第250页

本章应该是本书必读章，前面的质量属性是衡量架构好坏，对设计决策作出平衡和权衡的基础，而本章的架构策略和模式则是构成了架构的基石。模式从来都不是发明出来，而是被发现出来的。本章对于《设计模式》也推崇有加，《设计模式》广泛地涵盖了系统设计不同层面的模式从而成为经典，而本章阐述和引申的都是架构设计的模式。具体的模式有多种，作者分门别类一一道来，所以基本上几十页的内容可以大致了解应用于架构的模式细节。另外，还讨论了模式和策略的关系，以及在产品设计决策时根据质量属性的要求所作的权衡。

是干货很多的一章，看得累了，所以还是去炖莲藕山药排骨汤去补一补！

8、《软件构架实践》的笔记-第45页

这一章的标题是软件构架的多种环境，具体是技术、项目生命周期、业务、专业。

技术方面主要是从架构的质量属性考量，例如容错性、扩展性、性能、安全等等方面，这些具体细节

会在第2部分着重讲解。

项目生命周期方面，先介绍了四种常见的软件开发流程：瀑布、迭代、敏捷和模型驱动开发。然后，概括了无论在哪种流程里都不可避免的7个架构活动，抄录如下：1. Making a business case for the system
2. Understanding the architecturally significant requirements
3. Creating or selecting the architecture
4. Documenting and communicating the architecture
5. Analyzing or evaluating the architecture
6. Implementing and testing the system based on the architecture
7. Ensuring that the implementation conforms to the architecture

9、《软件构架实践》的笔记-第142页

Performance Tactics on the Road 这个例子举得很好

《软件构架实践》

版权说明