# **《软件架构实践》（第 4 版）读书笔记**

**刘佳鑫2022141461205**

在大三这一关键学习阶段，研读《软件架构实践》（第 4 版），犹如手握一把精准的手术刀，将复杂的软件架构领域层层剖析，让我收获颇丰。

## **核心概念**

阅读这本书对我来说最重要的就是了解了各种软件质量属性.恰好这学期我还上了黄武老师的软件设计与体系结构这门课,讲的就是软件架构.他说了一句话我记忆很深刻,他说计算机学院的学生都会写软件,但是他们写不好软件,原因就是他们没学过这门课,不了解软件里的各种质量属性,有了这些基础知识理论,才能写好软件.

这本书里设计的质量属性如下所示.

****质量属性****

* 可用性：系统在需要时正常运行的能力，涉及系统的容错、恢复机制等，对用户体验和系统稳定性至关重要。
* 可部署性：将软件系统部署到不同环境中的难易程度，包括部署的步骤、自动化程度以及对不同硬件和软件平台的适应性。
* 能源效率：关注软件系统在运行过程中的能源消耗，对于移动设备、数据中心等资源受限的环境尤为重要，体现了软件对硬件资源的优化利用。
* 可集成性：软件系统与其他系统或组件集成的能力，涉及接口的兼容性、数据格式的一致性以及交互协议的适应性等方面。
* 可修改性：系统易于修改、扩展和维护的特性，包括代码的可扩展性、模块的独立性和可替换性等，影响软件的生命周期和后续更新。
* 性能：系统在处理事务、响应请求等方面的效率，包括处理速度、吞吐量、响应时间等性能指标，直接关系到用户满意度和系统的竞争力。
* 安全性：保护系统免受未经授权的访问、攻击和数据泄露的能力，涵盖认证、授权、加密、数据保护等多方面的技术和策略。
* 防护性：类似于安全性，但更侧重于系统对外部攻击的防御能力，如网络攻击、恶意软件等，确保系统的完整性和数据的安全性。
* 可测试性：系统易于测试的程度，包括测试的便利性、测试工具的适配性以及测试覆盖的范围，是保证软件质量的重要基础。
* 易用性：软件系统的操作简便性和用户友好性，涉及界面设计、用户操作流程等，决定了用户能否方便、高效地使用系统。

虽然说我读了这本书,知道了这些质量属性,可对于目前还未经历编写过大型项目的我而言不过都只是了解,希望以后在实践编写大型工程项目的时候,能在动手过程中加深理解,把理论知识融进实践里.

## **二、架构解决方案**

在《软件架构实践》的学习历程中，接触到的各类架构解决方案，犹如为我打开了一扇通往高效软件构建世界的新大门。它们不再只是抽象概念，更像是一组组实用的工具包，助力架构师应对复杂多变的开发需求。

****软件接口****：它实实在在地充当着软件系统不同部分之间的 “翻译官”。每一次软件组件之间的信息传递、功能调用，都依赖于接口的顺畅对接。就好比搭建乐高积木，每一块积木上的凸起与凹槽，便是一种简易接口，唯有匹配精准，才能稳固拼接。设计软件接口时，数据交换格式不容小觑，格式不一致，信息传递就会乱码频发；协议的通用性若缺失，组件之间仿若操着不同 “方言”，难以交流；调用便利性不佳，更是会拖慢开发进度，让整个系统的协同运作磕磕绊绊。在小型项目里，接口问题或许还能勉强应对、逐一排查，可一旦面对大型工程，牵一发而动全身，一个接口设计失误，就可能导致连锁反应，让系统集成陷入僵局，扩展性沦为泡影。

****虚拟化****：这项技术有着神奇的魔力，它能把物理硬件资源巧妙包装，化身为灵动多变的虚拟环境。服务器虚拟化，让一台物理服务器摇身变成多台虚拟服务器，各自独立运行不同服务，资源不再闲置浪费；桌面虚拟化更是突破传统办公设备的限制，实现随时随地、多终端接入办公桌面。对于开发者而言，虚拟化意味着测试环境可以快速搭建、一键销毁重建，开发成本大幅降低；部署软件时，不用再受硬件环境的苛刻制约，大大增强了灵活性。它宛如给软件系统穿上了一件自适应外衣，不管外界硬件环境如何更迭，软件始终能找到舒适的 “运行姿势”。

****云和分布式计算****：云计算带来的是一场资源分配的革新，按需索取计算资源，如同用水用电般便捷。以往搭建服务器，前期要巨额投入硬件购置成本，还得担忧后续资源扩容难题；如今借助云平台，点点鼠标就能弹性扩容，低谷期缩减资源，节省开支。分布式计算则把庞大的计算任务拆解，分散到众多节点并行处理，恰似一群蚂蚁分工搬运重物，效率激增。二者珠联璧合，重塑了软件架构的底层逻辑。软件的部署不再局限于单机，而是能横跨多个地域、多种设备；运行时，故障转移更为灵活，某个节点 “病倒”，其他节点迅速接力；管理上，集中管控与分散运维相结合，让系统在复杂架构下依然井井有条，全方位提升可扩展性、性能表现与资源利用率。只是当下，我还缺少亲手驾驭它们的实战历练，满心期待未来大型项目里，能深度体验这两项技术的魅力与威力。

## **三、架构设计实践**

在深入钻研《软件架构实践》这本书后，属性驱动设计（ADD）犹如一盏明灯，为我照亮了架构设计这片迷雾重重的领域，让我初窥科学、系统的架构设计门道。

## **3.1 ADD 理念**

属性驱动设计（ADD）的核心在于紧扣质量属性来雕琢架构，它彻底颠覆了我以往对架构设计那种模糊且无序的认知。以往构思架构时，往往是先勾勒功能模块，至于系统能否稳定运行、高效响应，或是安全无虞，多是事后补救式的考量。而 ADD 强调打从一开始，就要把可用性、性能、安全性这类关键质量属性摆在 “驾驶座”，让它们主导设计方向。这就好比盖房子，不再是先把四面墙随意垒起来，再去想怎么遮风挡雨、方便进出，而是一开始就依据防风、采光、出入便捷这些需求去规划布局。

## **3.2 关键质量属性锚定**

开启 ADD 之旅的第一步，是精准锁定系统的关键质量属性。这绝非易事，需要与各方利益相关者深度沟通，从用户对体验流畅度的诉求、运维人员对部署简易性的期望，到企业对数据安全保密性的严格要求，都得一一梳理。例如，设计一款电商系统，用户下单支付环节的性能 —— 响应时间必须极短，不然客户流失率会飙升；同时，用户账户信息的安全性更是重中之重，一丝漏洞都可能引发信任危机。把这些关键属性挖掘出来，就等于为后续设计立下了明确的 “军令状”。

## **3.3 迭代式架构细化**

有了关键质量属性 “领航”，便要着手初始架构设计，并逐步拆解细化。先是把宏大的系统设想成一个整体，粗线条勾勒出大致框架，划分出几个核心模块。紧接着，像雕琢艺术品般，一次次精修，把模块再拆分成更细致的组件。这期间，每一刀下去，都得时刻审视是否契合既定的质量属性。拿可修改性来说，组件之间得保持 “松耦合”，方便后续升级迭代；涉及性能，就要斟酌数据传输路径是否最短、运算负荷是否均衡分布。以开发一个社交平台为例，用户动态发布模块，既要考虑内容审核的安全性，又要兼顾大量图文上传时的性能优化，不断权衡调整各组件的职责与交互方式。

## **3.4 持续评估保障**

ADD 最为严谨之处，在于全程贯穿的质量属性影响评估。每一个新的设计决策、每添加一处新功能、每调整一次组件关系，都要拉回 “质量属性” 这杆秤上称一称。绝不能埋头苦干，一路狂飙，最后才发现架构在可用性上漏洞百出，或是安全性千疮百孔。借助模拟测试、专家评审等手段，不断给半成品架构 “挑刺”，及时纠偏，直至最终的架构稳稳当当地接住关键质量属性的严苛考验。当下，虽还欠缺大规模实战检验，但 ADD 这套科学打法已融入我的设计思维，只盼着投身大型项目时，能让理论在实践中大放异彩，真正塑造出经得起推敲的优质架构。

## **四、学习感悟与收获**

### **4.1 思维蜕变：从碎片化到系统性**

初涉软件工程专业时，我对软件架构的认知犹如一盘散沙，质量属性、架构方案、设计流程各自为政地存于脑海。研读《软件架构实践》这本书后，最大的思维转变在于搭建起了一套完整的知识脉络。质量属性不再是孤立的概念，而是贯穿于架构解决方案与设计实践全程的核心脉络。意识到设计软件架构并非简单堆砌代码、拼凑功能，而是一场精心编排的 “交响乐”，每个音符 —— 无论是接口的规范、虚拟化的资源调配，还是 ADD 里每一步迭代，都要为整体的和谐共鸣负责。这种系统性思维，不仅重塑了我对专业知识的理解，更让我在面对复杂问题时，懂得如何抽丝剥茧，追溯到根源，从架构底层逻辑找解法。

### **4.2 跨学科视野拓展**

软件架构绝非只关乎代码与算法，它本质上是一个跨学科的 “集大成者” 。深入了解各种质量属性时，我发现可用性关联着人机交互与用户心理学，要让软件易用，得懂用户操作习惯与心理预期；安全性牵扯到密码学、网络攻防知识，加密算法、认证机制背后是深厚的数学与信息安全原理；能源效率更是涉足硬件底层，软件与硬件协同优化才能实现降耗节能。这驱使我主动涉猎计算机科学之外的知识领域，为成为一名全面的架构师添砖加瓦，毕竟未来复杂的项目场景，需要能横跨多学科知识版图的复合型人才。

### **4.3 挫折中沉淀耐心与韧性**

实践 ADD 等设计方法的过程，像是一场荆棘密布的冒险。最初尝试将理论套用到小型模拟项目时，状况百出：不是忽略了关键质量属性的权衡，就是架构细化到一半发现走进死胡同，推倒重来成了家常便饭。但正是这些挫折，磨砺出我的耐心。我开始习惯在反复试错里沉淀思绪，冷静复盘每一个失误环节，而非盲目蛮干。也正是这份韧性，让我更坚信实践出真知，每一次失败都是靠近成熟架构师的一步，日后面对真正大型、高压的项目，心态上已然做好了准备。

### **4.4 沟通协作意识觉醒**

书中虽未大篇幅讲团队协作，但越钻研越明白软件架构师处于项目的 “十字路口”，要和各方紧密沟通。收集质量属性需求时，得和用户、产品经理深度对话，精准捕捉需求痛点；实施架构解决方案，要与开发团队无缝对接，确保技术落地可行；评估架构质量，少不了运维、测试人员的反馈助力。明白了一己之力难成大事，学会用通俗易懂的话术阐释架构理念，凝聚团队共识，朝着共同架构愿景迈进，也是这段学习历程馈赠的宝贵收获，为未来职场协作埋下积极伏笔。