**软件架构实践（第四版）\_读书笔记\_3**

**第二部分 质量属性**

第三章·理解质量属性

*质量从来都不是偶然的；它一直是高度的意图、真诚的努力、明智的方向和熟练的执行力的结果。*

—William A. Foster 威廉·福斯特

（1）系统架构中的质量要素比功能更为关键，它们是系统能力、服务和行为的基本描述。功能是系统的基本能力，但质量要素如可维护性、可移植性、可扩展性和安全性等，对系统的长期成功至关重要。系统重设计通常不是因为功能缺陷，而是因为维护困难、移植性差、扩展性不足、性能低下或安全性问题。架构是软件创建过程中解决质量需求的关键环节，系统功能与软件结构的映射决定了架构对质量的支持。

质量属性是可度量或可测试的系统属性，它们指示系统在满足利益相关者需求方面的表现。质量属性可以被视为衡量产品在特定维度上对利益相关者的“效用”。

1. 功能性：

①功能性的定义：功能性是系统完成预期工作的能力。

功能性描述了系统的目的，而质量属性描述了系统实现这些目的的方式。功能需求与质量属性需求之间的区别有时模糊，因为某些功能实现可能与质量属性（如性能、安全性）紧密相关。

②为什么功能需求具有模糊性？这是因为功能需求的定义不明确，因为功能本身是一个模糊的概念。国际标准ISO 25010将功能适用性定义为软件产品在特定条件下使用时，能够提供满足规定和隐含需求的功能的能力。

③ 功能性与架构的定义：功能性不决定架构。给定一组功能，可以有多种架构实现这些功能。功能性独立于架构，但通过分配职责给架构元素来实现。架构设计不仅是为了满足功能，还为了支持其他质量属性，如可理解性和可维护性。

（3）质量属性场景：

①场景的六个部分：

触发事件（Stimulus）：到达系统或项目的事件，如性能事件、用户操作、安全攻击等。

触发源（Stimulus source）：触发事件的来源，影响系统如何处理事件，如受信任用户与不受信任用户。

响应（Response）：系统或开发人员对触发事件的反应，如处理事件、实现修改等。

响应测量（Response measure）：响应的可测量性，用于测试场景，如延迟、吞吐量、修改所需时间等。

环境（Environment）：场景发生的条件，如系统状态、运行模式等。

工件（Artifact）：触发事件的目标，可以是系统的一部分或整个系统。

（4）通过架构模式和策略实现质量属性

①实现质量属性的技术：

策略：影响质量属性响应的设计决策，直接影响系统对触发事件的响应。

架构模式：描述特定设计环境中反复出现的问题，并提供经过验证的解决方案。

②策略与架构模式的关系：

架构模式通常包括多个设计决策和多种质量属性策略，常在质量属性之间进行权衡。

③重构：重构是扭转架构债务的手段，可以因多种原因进行，如提高信息安全性、性能或可修改性。重构包括将共同功能分解到自己的模块中，提高内聚并减少未来更改请求时需要更改的位置数量。

④代码重构与架构重构：代码重构是敏捷开发项目的主要实践，确保没有生成重复或过于复杂的代码。架构重构适用于架构元素，以改善质量属性。

1. 使用策略进行设计：

系统设计由控制质量属性和实现系统功能的决策组成。策略和模式是架构师常用的设计技术，本书对它们进行了分类和描述，而非发明。

①关注策略的原因：

补充模式：当没有合适的模式时，策略可以增强现有模式或填补空白。

构建设计：在缺乏模式的情况下，策略允许基于第一原则构建设计。

系统化设计分析：策略提供了一种系统化的方法，使设计和分析更加有序。

②策略的重要性：策略的强大之处在于它们的普遍性和在不同设计方面的重要性。了解策略对于架构师来说至关重要，因为它们可以成为设计过程中的有力工具。