

<<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：9787302070436

10位ISBN编号：7302070431

出版时间：2003-8

出版时间：清华大学

作者：林·巴斯

页数：528

字数：696000

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<软件构架实践>>

### 内容概要

本书在第1版的基础上，根据软件生命期的特点，分预想构架、创建构架、分析构架和从一个系统至多个系统进行阐述。

本书对第1版某些内容进行了深入介绍，并增添了一些新内容：ATAM、质量需求、构架重构、用UML对构架编档和CBAM等。

此外，本书还对一些新案例进行了分析，以帮助理解软件构架如何适应商业需求。

本书是卡内基·梅隆大学软件工程研究所推荐教材，荣获美国权威的“软件开发”杂志第九届图书效率大奖。

本书可作为软件学院及高校相关专业本科生和研究生的教材，也适合业界人士研究参考。

## <<软件构架实践>>

### 作者简介

作者：（美国）林·巴斯（Len Bass）（美国）保罗·克莱门茨（Paut Clements）（美国）瑞克·凯兹曼（Rick Kazman）林·巴斯（Len Bass），软件工程研究所（SEI）的一名高级软件工程师。

他已经编著了5本书籍，并发表了大量关于软件工程、人机交互的论文。

他曾经领导一个小组为飞行控制模拟器开发软件构架。

目前，该构架已经被用作美国空军标准。

保罗·克莱门茨（Paut Clements），软件工程研究所（SEI）的一名高级技术人员，其工作职责是开发软件构架和设计产品线。

他已经发表了30多篇关于软件设计和实时系统的论文。

瑞克·凯兹曼（Rick Kazman），软件工程研究所（SEI）的一名高级软件工程师，负责构架权衡分析工作，是沃特鲁大学和多伦多大学的副教授。

他已经发表了50多篇关于软件工程、人机交互和信息检索的论文。

## &lt;&lt;软件构架实践&gt;&gt;

## 书籍目录

## 第i部分 预想构架

## 第1章 构架商业周期

- 1.1 构架的产生
- 1.2 软件过程和构架商业周期
- 1.3 什么样的构架才算好
- 1.4 小结
- 1.5 讨论题

## 第2章 什么是软件构架

- 2.1 软件构架概念的澄清
- 2.2 其他观点
- 2.3 构架模式、参考模型和参考构架
- 2.4 为什么说软件构架非常重要
- 2.5 构架结构和视图
- 2.6 小结
- 2.7 可进一步参阅的文献
- 2.8 讨论题

## 第3章 a-7e航空电子系统：使用构架结构的案例分析

- 3.1 与构架商业周期的关系
- 3.2 需求与质量
- 3.3 a-7e航空电子系统的构架
- 3.4 小结
- 3.5 可进一步参阅的文献
- 3.6 讨论题

## 第ii部分 创建构架

## 第4章 理解质量属性

- 4.1 功能和构架
- 4.2 构架和质量属性
- 4.3 系统质量属性
- 4.4 实践中的质量属性场景
- 4.5 其他系统质量属性
- 4.6 商业质量
- 4.7 构架质量
- 4.8 小结
- 4.9 可进一步参阅的文献
- 4.10 讨论题

## 第5章 实现质量属性

- 5.1 策略介绍
- 5.2 可用的策略
- 5.3 可更改性策略
- 5.4 性能策略
- 5.5 安全性策略
- 5.6 可测试性策略
- 5.7 可使用性策略
- 5.8 策略与构架模式的关系
- 5.9 构架模式和样式

## &lt;&lt;软件构架实践&gt;&gt;

- 5.10 小结
- 5.11 讨论题
- 5.12 可进一步参阅的文献
- 第6章 空中交通管制系统：高可用性设计案例分析
  - 6.1 与构架商业周期的关系
  - 6.2 需求与质量
  - 6.3 构架解决方案
  - 6.4 小结
  - 6.5 可进一步参阅的文献
  - 6.6 讨论题
- 第7章 设计构架
  - 7.1 生命期中的构架
  - 7.2 设计构架
  - 7.3 形成团队结构
  - 7.4 创建骨架系统
  - 7.5 小结
  - 7.6 可进一步参阅的文献
  - 7.7 讨论题
- 第8章 飞行模拟：构架可集成性案例分析
  - 8.1 与构架商业周期的关系
  - 8.2 需求与质量
  - 8.3 构架解决方案
  - 8.4 小结
  - 8.5 可进一步参阅的文献
  - 8.6 讨论题
- 第9章 软件构架编档
  - 9.1 构架文档的使用
  - 9.2 视图
  - 9.3 选择相关视图
  - 9.4 视图编档
  - 9.5 跨视图文档
  - 9.6 统一建模语言
  - 9.7 小结
  - 9.8 可进一步参阅的文献
  - 9.9 讨论题
- 第10章 软件构架重构
  - 10.1 介绍
  - 10.2 信息提取
  - 10.3 数据库构造
  - 10.4 视图融合
  - 10.5 重构
  - 10.6 示例
  - 10.7 小结
  - 10.8 可进一步参阅的文献
  - 10.9 讨论题
- 第iii部分分析构架
- 第11章 atam：构架评估的综合方法

## &lt;&lt;软件构架实践&gt;&gt;

- 11.1 atam中的参与者
- 11.2 atam的结果
- 11.3 atam的阶段
- 11.4 完美的系统：应用atam的案例分析
- 11.5 小结
- 11.6 可进一步参阅的文献
- 11.7 讨论题
- 第12章 cbam：制定构架设计决策的定量方法
- 12.1 制定决策的环境
- 12.2 cbam的基础
- 12.3 实现cbam
- 12.4 案例分析：nasaecs项目
- 12.5 使用cbam方法的结果
- 12.6 小结
- 12.7 可进一步参阅的文献
- 12.8 讨论题
- 第13章 万维网；可互操作性案例分析
- 13.1 与构架商业周期的关系
- 13.2 需求与质量
- 13.3 构架解决方案
- 13.4 通过abc的另一个周期：基于web的电子商务构架的演变
- 13.5 实现质量目标
- 13.6 目前的构架商业周期
- 13.7 小结
- 13.8 可进一步参阅的文献
- 13.9 讨论题
- 第iv部分 从一个系统到多个系统
- 第14章 软件产品线：重用构架资产
- 14.1 概述
- 14.2 软件产品线行之有效的原因
- 14.3 范围
- 14.4 产品线的构架
- 14.5 使用软件产品线的困难之处
- 14.6 小结
- 14.7 可进一步参阅的文献
- 14.8 讨论题
- 第15章 celslusteeh：产品线开发案例分析
- 15.1 与构架商业周期的关系
- 15.2 需求与质量
- 15.3 构架解决方案
- 15.4 小结
- 15.5 可进一步参阅的文献
- 15.6 讨论题
- 第16章 j2ee/ejb：工业标准计算基础结构的案例分析
- 16.1 与构架商业周期的关系
- 16.2 需求与质量
- 16.3 构架解决方案

<<软件构架实践>>

- 16.4 系统部署决策
- 16.5 小结
- 16.6 可进一步参阅的文献
- 16.7 讨论题
- 第17章 luther构架：使用j2ee的移动应用案例分析
- 17.1 与构架商业周期的关系
- 17.2 需求与质量
- 17.3 构架解决方案
- 17.4 luther构架如何实现其质量目标
- 17.5 小结
- 17.6 可进一步参阅的文献
- 17.7 讨论题
- 第18章 用商业组件构建系统
- 18.1 组件对构架的影响
- 18.2 构架失配
- 18.3 基于组件设计的搜寻
- 18.4 aseilm示例
- 18.5 小结
- 18.6 可进一步参阅的文献
- 第19章 未来的软件构架
- 19.1 重新认识构架商业周期
- 19.2 创建构架
- 19.3 生命期中的构架
- 19.4 商业组件的影响
- 19.5 小结
- 缩略语表
- 参考文献
- 索引

## 章节摘录

版权页：插图： The Architecture Enables More Accurate Cost and Schedule Estimates. Cost and schedule estimates are an important management tool to enable the manager to acquire the necessary resources and to understand whether a project is in trouble. Cost estimations based on an understanding of the system pieces are, inherently, more accurate than those based on overall system knowledge. As we have said, the organizational structure of a project is based on its architecture. Each team will be able to make more accurate estimates for its piece than a project manager will and will feel more ownership in making the estimates come true. Second, the initial definition of an architecture means that the requirements for a system have been reviewed and, in some sense, validated. The more knowledge about the scope of a system, the more accurate the estimates.

ARCHITECTURE AS A TRANSFERABLE, RE-USABLE MODEL The earlier in the life cycle re-use is applied, the greater the benefit that can be achieved. While code re-use is beneficial, re-use at the architectural level provides tremendous leverage for systems with similar requirements. Not only code can be re-used but so can the requirements that led to the architecture in the first place, as well as the experience of building the re-used architecture. When architectural decisions can be re-used across multiple systems, all of the early decision consequences we just described are also transferred.

Software Product Lines Share a Common Architecture. A software product line or family is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Chief among these core assets is the architecture that was designed to handle the needs of the entire family. Product line architects choose an architecture (or a family of closely related architectures) that will serve all envisioned members of the product line by making design decisions that apply across the family early and by making other decisions that apply only to individual members late. The architecture defines what is fixed for all members of the product line and what is variable. Software product lines represent a powerful approach to multi-system development that shows order-of-magnitude payoffs in time to market, cost, productivity, and product quality. The power of architecture lies at the heart of the paradigm.



版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>