世界著名计算机教材精选

PEARSON Prentice Hall

软件体系结构

Mary Shaw
David Garlan 著
牛振东 江鹏 金福生 编译

SOFTWARE ARCHITECTURE
PERSPECTIVES ON AN EMERGING DISCIPLINE

PEARSON Education
清华大学出版社

EISBN: 0-13-182957-2

图书在版编目(CIP)数据

软件体系结构/（美）肖（Shaw, M.），（美）加兰（Garlan, D.）著；牛振东等编译. —北京：清华大学出版社，2007.3
（世界著名计算机教材精选）
书名原文：Software Architecture
ISBN 978-7-302-14550-9

Ⅰ. 软…　Ⅱ. ①肖…　②加…　③牛…　Ⅲ. 软件-系统结构-教材　Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2007）第 011990 号

责任编辑：龙啟铭
封面设计：何凤霞
责任校对：张　剑
责任印制：李红英
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　　**邮购热线**：010-62786544
投稿咨询：010-62772015　　**客户服务**：010-62776969
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185×260　**印张**：15.5　**字数**：186 千字
版　　次：2007 年 3 月第 1 版　　**印次**：2007 年 3 月第 1 次印刷
印　　数：1～3000
定　　价：29.80 元

前言
FOREWORD

软件体系结构作为从软件设计抽象出来的一门新兴学科，目前已经成为软件工程的一个重要研究领域，合适的软件体系结构是软件系统成功开发的重要保障。“Software architecture: perspective on an emerging discipline”一书的作者 Mary Shaw 与 David Garlan 作为软件体系结构早期的研究者，在体系结构领域做出了大量先导性的工作，上述著作是软件体系结构领域的重要经典著作之一。

本书共有 8 章，第 1~5 章主要是从 Shaw 和 Gralan 的上述著作翻译而来，第 6~8 章是由编译者结合多年的软件体系结构教学经验并参考大量文献基础上编写的。本书的第 1 章和第 5 章对照原著增加了部分新内容。本书的第 6 章和第 7 章是在参考了相关研究文献基础上，结合作者在数字图书馆领域的亲身实践编写的。

本书各章的内容如下：

第 1 章　软件体系结构概述
第 2 章　软件体系结构风格
第 3 章　案例研究
第 4 章　共享信息系统
第 5 章　软件体系结构描述
第 6 章　软件体系结构的分析与评估
第 7 章　特定领域的软件体系结构
第 8 章　流行的软件体系结构

本书可以作为计算机专业研究生和高年级本科生的软件体系结构课程的教材或参考书，也可作为软件开发人员的参考手册。

本书主要由牛振东、江鹏、金福生等编著，其整体结构和内容由牛振东教授结合软件体系结构课程教学经验制定。其中，第 1 章由牛振东、江鹏编写和翻译，第 2 章由江鹏、林俊宏、杨青、牛振东翻译，第 3 章由江鹏、崔玉、张继、牛振东翻译，第 4 章由江鹏、吕虎、金福生翻译，第 5 章由江鹏、杨青、林俊宏编写和翻译，第 6 章由杨可观、金福生、牛振东、王磊杰编写，第 7 章由杨可观、王磊杰、金福生编写，第 8 章由江鹏编写。北京理工大学数字图书馆与数字化教育技术实验室、计算机软件

实验室的张春霞、彭学平、施重阳、刘扬等多名教师和研究生认真阅读了本书的草稿，并提出许多宝贵意见。最后，牛振东、金福生、江鹏统一审校了全书。

十分感谢清华大学出版社对本书出版给予的大力支持，感谢龙啟铭老师等对本书出版付出的辛勤劳动，感谢实验室全体老师和同学的支持。在编译过程中，我们参考了大量的资料，所涉及的文献及作者无法全部列举，也特此一并感谢。

虽然我们对书中出现的所有术语和翻译过程中出现的难词、难句都作了仔细的推敲和研究，但由于我们水平有限，写作时间仓促，加之软件体系结构自身的发展又非常迅速，疏漏和争议之处在所难免，恳请广大读者批评指正。

目 录
CONTENTS

第 1 章 绪论 1

1.1 什么是软件体系结构 1
软件设计层次 5
1.2 软件体系结构研究的内容和范畴 6
1.2.1 体系结构研究领域 6
1.2.2 风格、设计模式、框架 7
1.3 体系结构设计原则 8
1.4 软件体系结构研究的现状 10
1.5 全书的安排 11

第 2 章 体系结构风格 13

2.1 体系结构风格 13
2.2 管道过滤器 15
2.3 数据抽象和面向对象组织结构 18
2.4 事件驱动，隐式调用 19
2.5 分层系统 21
2.6 知识库 22
2.7 解释器 24
2.8 过程控制 25
2.8.1 过程控制范例 25
2.8.2 过程控制软件范例 28
2.9 其他常见的体系结构 30

2.10 异构体系结构 31

第3章 案例研究 33

3.1 上下文关键字 33
3.1.1 解决方案1：使用共享数据的主程序/子程序 35
3.1.2 解决方案2：抽象数据类型 36
3.1.3 解决方案3：隐式调用 37
3.1.4 解决方案4：管道过滤器 38
3.1.5 各种方案的比较 39
3.2 仪器软件 40
3.2.1 面向对象模型 41
3.2.2 分层模型 42
3.2.3 管道过滤器模型 43
3.2.4 改进后的管道过滤器模型 43
3.2.5 专用化模型 44
3.2.6 总结 45
3.3 移动机器人 45
3.3.1 设计考虑因素 46
3.3.2 解决方案1：控制环路 47
3.3.3 解决方案2：分层体系结构 48
3.3.4 解决方案3：隐式调用 51
3.3.5 解决方案4：黑板体系结构 53
3.3.6 各种方案的比较 55
3.4 定速巡航控制 56
3.4.1 定速巡航控制的面向对象观点 58
3.4.2 定速巡航控制的过程控制观点 59
3.4.3 分析和讨论 64
3.4.4 总结 67
3.5 复合混合风格的三个案例 68
3.5.1 具有多种风格的分层设计 68
3.5.2 使用不同构件风格的解释器 71
3.5.3 一个黑板风格的解释器 74

第4章 共享信息系统 77

4.1 共享信息系统概述 77
4.2 数据库集成 78
4.2.1 批序列 79
4.2.2 简单知识库（Repository） 82
4.2.3 虚拟知识库 85
4.2.4 多级分层结构 87
4.2.5 业务数据处理的共享信息系统的演化 89
4.3 软件开发环境集成 89
4.3.1 批序列 90
4.3.2 从批序列到知识库的转化 90
4.3.3 知识库 93
4.3.4 多级分层结构 94
4.3.5 软件开发环境的共享信息系统的演化 95
4.4 建筑设计集成 96
4.4.1 知识库 97
4.4.2 智能控制 99
4.4.3 建筑设计的共享信息系统的演化 101
4.5 共享信息系统的体系结构 101
4.5.1 各种数据流系统的比较 102
4.5.2 各种知识库系统的比较 103
4.6 结论 105

第5章 软件体系结构描述 107

5.1 综述 107
IEEE 软件体系结构描述框架标准 108
5.2 体系结构描述语言（ADL） 110
5.2.1 典型的软件体系结构描述语言简介 117
5.3 体系结构形式化模型和规格说明 126
5.3.1 体系结构形式化的价值 126
5.3.2 形式化一个特殊系统的体系结构 127
5.3.3 形式化体系结构风格 130

5.3.4 Z 标记形式化描述语言 137
5.4 使用 UML 描述体系结构 142

第 6 章 软件体系结构的分析与评估 145

6.1 体系结构评估概述 145
6.2 体系结构评估方法 147
6.3 体系结构权衡分析方法（ATAM） 148
6.3.1 ATAM 评估步骤 149
6.3.2 ATAM 评估工具 168
6.4 软件体系结构分析方法（SAAM） 169
6.4.1 SAAM 评估步骤 169
6.4.2 SAAM 评估工具 176

第 7 章 特定领域的软件体系结构 181

7.1 特定领域的软件体系结构概述 181
7.1.1 特定领域的软件体系结构的定义 182
7.1.2 DSSA 的组成 182
7.2 研究 DSSA 及其开发方法的意义 183
7.3 DSSA 的螺旋型演化过程 185
7.4 基于 DSSA 的软件开发 186
7.4.1 DSSA 的建模 186
7.4.2 基于 DSSA 的开发流程 188
7.5 基于 DSSA 的 DRICSM 系统建模 189
7.5.1 面向 DRICSM 的领域模型概述 189
7.5.2 开发 DRICSM 系统的特征 191
7.5.3 DRICSMSA 的体系结构模型 192

第 8 章 流行的软件体系结构 199

8.1 概述 199
8.2 基于 CORBA 的分布式构件技术 203

8.3 基于 Java 的分布式构件技术 207
8.4 基于.NET 平台的分布式构件技术 212
8.5 面向服务的体系结构 217
8.5.1 什么是面向服务的体系结构 217
8.5.2 基于 Web 服务的 SOA 实现 220

参考文献 225

第1章

绪 论

软件体系结构对于软件工程师来说是一门重要的新兴学科。但是它并不是作为软件系统新的关注点而以一种简单的、清楚的、完善的方式形成的。软件工程师需要一种更好的视角来理解软件，并试图找到一种新方法来构建更复杂的大型软件系统，这时，软件体系结构这门学科才从软件设计中抽象出来，逐渐自然演化形成。

在这一个章节中，我们将阐述什么是软件体系结构以及软件体系结构研究的内容和范畴。在这个基础上我们将勾画出软件体系结构的现状。

1.1 什么是软件体系结构

随着软件系统的规模和复杂性不断增加，系统的全局结构的设计和规划变得比算法的选择以及数据结构的设计更加重要。这种全局结构的设计和规划问题包括全局组织结构；全局控制结构；通信和同步以及数

据存取的协议；规定设计元素的功能；设计元素的组合；物理分布；规模和性能；演化的维度；设计方案的选择等。这些就是软件体系结构所要讨论的问题。

抽象地说，软件体系结构包括构成系统的设计元素的描述，设计元素的交互，设计元素组合的模式，以及在这些模式中的约束。一般来说，一个具体的系统由构件的集合以及它们之间的关系组成。这样的系统又有可能成为一个更大的系统的组成元素。

人们已经普遍认识到：为系统设计一个合适的体系结构是系统取得长远的成功的关键因素，但是当前描述体系结构的实践一直是非形式化的。一般根据长期形成的非形式化的习惯方式来描述体系结构。通常，体系结构用方框连线图来抽象的表示，并在一旁用文字加以注释，来解释符号的含义，或者说明构件的选择以及它们之间交互的一些原理。举例来说，软件体系结构的典型描述如下：

“Camelot 基于客户机–服务器模式，它使用远程方法调用来实现应用程序和服务器之间通信”[S+87]。

“抽象分层和系统分解为客户端提供了统一的系统接口，这样就使 Helix 能够适应客户端设备的多样性。这种体系结构推荐应用系统采用客户机–服务器模式” [FO85]。

“我们已经选择了一种分布式的，面向对象的解决方案来管理信息资源”[Line87]。

“将规范的顺序的编译器改造成并发的编译器的最直接的方式是采用管线，并使各个编译阶段在多个处理器上执行。一个更有效的方式是把源代码分成许多部分，在各个编译阶段被并发地处理这些部分［通过多编译处理器］，最后再将目标代码重组成一个完整的单一的程序”[Ses88]。

这种描述非常普遍。尽管几乎完全非形式化，但是这种方式沟通起来非常有效。每个独立构件的标注对于所描述的系统来说都是独特的，甚至像体系结构模式这种对软件结构的概括也是非常随意的，不同的设计者有各自的习惯做法。

由于当前描述体系结构的实践中这种相对的非形式化和高度的抽象性，起初看来，体系结构描述好像对软件工程师没有什么太大的实质价值。但是事实并非如此，这其中有两个原因。首先，软件工程师在长期的实践中已经拥有了一套共享的，语义丰富的词典，它由软件系统的习惯用语，模式，软件系统组织结构风格组成。比如说，通过识别一个管道过滤器体系结构风格的实例，一个软件工程师传达了这样的事实：这个系统的主要功能就是进行数据流的转换，系统的主要功能由各种独立实体的过滤器分别来实现，系统的响应时间和吞吐量能用以一种直接的方式计算出来。这样，尽管这种共享的词典很大程度上是非形式化，但是它能为软件工程师之间的交流提供语义丰富的内容。

第二个原因是，虽然与元素实际计算的细节相比，软件体系结构是非常抽象的，但是这种结构为理解更大范围的，系统级的关注点，比如吞吐量，通信模式，执行控制结构，可伸缩性，以及系统演化的扩展方式，提供了一个自然的框架。这样体系结构描述的作用好像是一个框架，系统属性在这个框架下进行扩充，并且，它在考察一个系统实现其整体需求的能力中扮演了非常重要角色。

当然这并不是说体系结构描述形式化的符号和严格的分析技术是不必要的。相反，如果当前的体系结构设计的实践能够获得更好的符号、理论和分析技术支持的话，我们从中的收获将更大。在后面章节中我们将探讨软件体系结构中的具体问题。

在软件体系结构这个主题上，目前有相当多（将会更多）的工作要做，包括：模块连接语言，满足特殊领域需要的系统模板和框架，构件集成机制的形式化模型。令人遗憾的是，目前没有统一的专业术语来描述这一领域的通用元素。

我们目前还没有一个能被广泛接受的体系结构范例的分类法，更不用说一个成熟的软件体系结构理论。但是，目前我们能够明显地确定出许多体系结构描述的基本元素。另外，我们已经能够确定一系列的体系结构模式或者风格，这些模式或风格形成了软件架构师的基本技能（这些模式或风格已经在上述引用中阐述，它们也是第 2 章的主题）。

软件系统的体系结构定义系统由计算构件和构件之间的相互作用组成。构件可以是客户机和服务器、数据库、过滤器或者是在一个分层系统中的层。构件之间的相互作用在这个设计层次上可以是简单和相似的，比如过程调用，共享变量的访问。但是它们也可以是复杂和语义丰富的，比如说客户机–服务器协议（client-server protocols），数据库存取协议（database-accessing protocols），异步事件多点传送（asynchronous event multicast），和管道数据流（piped streams）等。

除了指定系统的结构和拓扑关系，体系结构还指出了系统需求和已构建系统的元素之间的对应关系，这样能为设计方案的选择提供基本原则。在体系结构的层次上，相关的系统级别的问题包括了容量，吞吐量，一致性，构件的兼容性等。

更一般的说，体系结构模型阐明了构件以及相互关系之间的结构和语义上的不同。这些体系结构模型能够被组合起来定义更大的系统。理想情况下，体系结构描述中每个独立的元素被独立地定义，这样它们可以在不同的环境中被重用。体系结构为这些独立的元素建立了规格说

明，这些元素本身被抽象成体系结构的子系统，或是被传统的编程语言实现。

软件设计层次

系统设计是在多个层次上进行。每一层要恰当地处理不同的设计关注点，所以精确地划分层次非常有用。在每个层次上我们都能找到：原始的和被组装的构件；非原始构件或系统的组装规则；为系统提供语义的行为规则（BN71,New82,New90）。这些在不同的层次上是不同的，所以在每个的层次上使用不同的符号，设计问题，分析技术。这样，每一层可以分别独立的设计。但是，并不是说层次之间是完全独立的，层次之间是有关联的，因为每一层的元素都要与低一级层次的结构相对应，并要通过低一级的层次来实现其功能。

计算机的硬件系统层次结构对于我们来说已经很熟悉，软件同样有其设计层次。我们能够确定至少三个层次：

（1）体系结构级：这个级别的设计问题包括系统性能与构件之间的整体联系。这个级别的构成元素是模块，模块通过各种方式互连；通过操作算子将子系统组装成一个系统。

（2）代码级：这个级别的设计问题包括算法和数据结构；其构成元素是编程语言原语，比如数值、字符、指针以及控制线程。基本的操作算子是算法和程序设计语言提供的数据操作原语。组装机制包括记录、数组、过程。

（3）执行级：这个级别的设计问题包括存储器的映射、数据格式配置、堆栈和寄存器的分配。其构成元素是硬件所支持的位模式。使用机器代码来描述操作和组装。

1.2 软件体系结构研究的内容和范畴

1.2.1 体系结构研究领域

优秀的体系结构通常是一个软件系统取得成功的决定性因素。虽然很多体系结构范例（比如管道线，分层系统，客户机-服务器组织等等）非常有用，但是对这些范例却没有形成一致的理解，而是以习惯的方式理解它们，并且对它们的使用也没有一致的方式。结果，软件系统设计者很难在系统体系结构中找到通用的东西，不能在多个设计方案中做出合理的选择，也不能将通用的范例应用于某一具体领域，或者将他们的设计经验传授给其他设计者。

当前，软件体系结构已经成为软件工程实践者和研究者的一个重要的研究领域。很多领域的研究工作使得关于软件体系结构的许多问题正在被解决，这些领域包括模块接口语言，特定领域的软件体系结构，软件的重用，软件组织结构模式的规范化编纂，体系结构描述语言，体系结构设计形式化的支持和体系结构设计环境。

虽然目前这一领域相当活跃，但是许多研究工作只是在一些小的机构中进行，这些研究工作并没有与其他工作相互协调和配合。目前，两个机构[GPT95,Gar95a]将很多对软件体系结构感兴趣的研究者和实践者组织起来，讨论软件体系结构理论和实践的现状。这些机构的目标是对实践现状、目前的研究发展工作的分类以及这一新兴领域重大挑战等问题建立的一个共同的理解。根据这些研究机构广泛的兴趣可以将当前软件体系结构分为 4 个研究领域：

第一类是通过提供一种新的体系结构描述语言（Architectural Description Language）解决体系结构描述问题。这种语言的目标是向实

践者提供设计体系结构更好的方法，以便设计人员相互交流，并可以使用支持体系结构描述语言的工具来分析案例。这些内容将在第 5 章介绍。

第二类是体系结构领域知识的总结性研究。这一领域关心的是工程师通过软件实践总结而来各种体系结构原则和模式的分类和阐释。

第三类是针对特定领域的框架[SEI90,Tra94]的研究。这类研究产生了针对一类特殊软件的体系结构框架，比如，航空电子控制系统、移动机器人、用户界面。这类研究一旦成功，这样的框架便可以被毫不费力实例化来生产这一领域新的产品。这部分内容将在第 7 章介绍。

第四类是软件体系结构形式化支持的研究。随着新的符号的产生，以及人们对体系结构设计实践的理解逐步深入，需要用一种严格的形式化方法刻画软件体系结构及其相关性质。这些内容将在第 5 章介绍。

1.2.2 风格、设计模式、框架

1. 体系结构风格（Architecture Styles）

体系结构风格是描述特定系统组织方式的惯用范例，强调组织模式和惯用范例。组织模式即静态表述的样例，惯用范例则是反映众多系统共有的结构和语义。通常，体系结构风格独立于实际问题，强调了软件系统中通用的组织结构，比如管道线，分层系统，客户机-服务器等等。体系结构风格以这些组织结构定义了一类系统族。第 2 章将对体系结构风格作详细的介绍。

2. 设计模式（Design Pattern）

设计模式是软件问题高效和成熟的设计模板，模板包含了固有问题的解决方案。设计模式可以看成规范了的小粒度的结构成分，并且独立于编程语言或编程范例。设计模式的应用对软件系统的基础结构没有什么影响，但可能对子系统的组织结构有较大影响[1]。每个模式处理系统

设计或实现中一种特殊的重复出现的问题。例如，Bridge 模式，它为解决抽象部分和实现部分独立变化的问题提供了一种通用结构。因此，设计模式更强调直接复用的程序结构。

3. 应用框架（Application Framework）

应用框架是整个或部分系统的可重用设计，表现为一组抽象构件的集合以及构件实例间交互的方法。可以说，一个框架是一个可复用的设计构件，它规定了应用的体系结构，阐明了整个设计、协作构件之间的依赖关系、责任分配和控制流程，表现为一组抽象类以及其实例之间协作的方法，它为构件复用提供了上下文(Context)关系。在很多情况下，框架通常以构件库的形式出现，但构件库只是框架的一个重要部分。框架的关键还在于框架内对象间的交互模式和控制流模式[2]。

设计模式是对在某种环境中反复出现的问题以及解决该问题的方案的描述，它比框架更抽象；框架可以用代码表示，也能直接执行或复用，而对模式而言只有实例才能用代码表示；设计模式是比框架更小的元素，一个框架中往往含有一个或多个设计模式，框架总是针对某一特定应用领域，但同一模式却可适用于各种不同的应用。体系结构风格描述了软件系统的整体组织结构，它独立于实际问题。而设计模式和应用框架更加面向具体问题。

体系结构风格、设计模式和应用框架的概念是从不同的目的和出发点讨论软件体系结构，它们之间的概念经常互相借鉴和引用。

1.3 体系结构设计原则

软件工程师在多年的实践中，对软件体系结构的设计总结出一些带有普遍性的原则，这些原则和策略多年来一直被广泛地运用着。它们独