高等学校计算机专业“十一五”规划教材

软件体系结构实用教程

付　燕　主　编
龚尚福　龙熙华　副主编

西安电子科技大学出版社
http://www.xduph.com

高等学校计算机专业“十一五”规划教材

软件体系结构实用教程

付　燕　　主　编
龚尚福　龙熙华　副主编

西安电子科技大学出版社
2009

内容简介

本书对软件体系结构的理论基础、研究内容、当前研究现状和实践应用进行了系统的介绍。通过本书，读者可以了解到软件体系结构的基本概念、风格、描述方法、设计方法、评估方法和集成开发环境等内容。

本书共分 10 章。第 1 章简单介绍了软件重用和构件技术的一些基本概念，它们是学习软件体系结构有关知识的基础；第 2 章介绍了软件体系结构的概念、发展和研究现状；第 3 章对软件体系结构的风格进行了较为详细的介绍，并给出了一些应用实例；第 4 章讨论了软件体系结构的三种描述方法；第 5 章介绍了软件体系结构设计过程中使用的一般原理和设计模式，以及常用的体系结构设计方法；第 6 章对 Bass 等人提出的一种基于体系结构的软件开发过程做了详细介绍；第 7 章介绍了软件体系结构评估方法，重点介绍 SAAM 和 ATAM 方法；第 8 章介绍了 Web 服务体系结构的有关知识，并给出了一个简单的应用实例；第 9 章对特定领域的软件体系结构进行了介绍，详细讨论了其建立过程；第 10 章介绍了软件体系结构集成开发环境的具体功能。

本书可作为计算机软件专业高年级本科生和研究生的软件体系结构教材，也可作为软件开发人员的参考书。

图书在版编目(CIP)数据

软件体系结构实用教程 / 付燕主编.
—西安：西安电子科技大学出版社，2009.9
高等学校计算机专业“十一五”规划教材
ISBN 978-7-5606-2315-3
Ⅰ. 软… Ⅱ. 付… Ⅲ. 软件—系统结构—高等学校—教材 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字(2009)第 119130 号

策　　划　陈　婷
责任编辑　陈　婷
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　邮　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　西安文化彩印厂
版　　次　2009 年 9 月第 1 版　2009 年 9 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印张 18.5
字　　数　429 千字
印　　数　1～4000 册
定　　价　26.00 元
ISBN 978 – 7 – 5606 – 2315 – 3 / TP • 1173
XDUP 2607001-1

*** * * 如有印装问题可调换 * * ***
本社图书封面为激光防伪覆膜，谨防盗版。

高等学校计算机专业“十一五”规划教材

编审专家委员会

主　任：马建峰　（西安电子科技大学计算机学院院长，教授）

副主任：赵祥模　（长安大学信息工程学院院长，教授）

余日泰　（杭州电子科技大学计算机学院副院长，副教授）

委　员：（按姓氏笔画排列）

王忠民　（西安邮电学院计算机系副主任，教授）

王培东　（哈尔滨理工大学计算机与控制学院院长，教授）

石美红　（西安工程大学计算机科学与技术系主任，教授）

纪　震　（深圳大学软件学院院长，教授）

刘卫光　（中原工学院计算机学院副院长，教授）

陈　以　（桂林电子科技大学计算机与控制学院副院长，副教授）

张尤赛　（江苏科技大学电子信息学院副院长，教授）

邵定宏　（南京工业大学信息科学与工程学院副院长，教授）

张秀虹　（青岛理工大学计算机工程学院副院长，教授，）

张焕君　（沈阳理工大学信息科学与工程学院副院长，副教授）

张瑞林　（浙江理工大学信息电子学院副院长，教授）

李敬兆　（安徽理工大学计算机科学与技术学院院长，教授）

范　勇　（西南科技大学计算机学院副院长，副教授）

陈庆奎　（上海理工大学计算机学院副院长，教授）

周维真　（北京信息科技大学计算机学院副院长，教授）

徐　苏　（南昌大学计算机系主任，教授）

姚全珠　（西安理工大学计算机学院副院长，教授）

徐国伟　（天津工业大学计算机技术与自动化学院副院长，副教授）

容晓峰　（西安工业大学计算机学院副院长，副教授）

龚尚福　（西安科技大学计算机系主任，教授）

策　划：臧延新　云立实

杨　璠　陈　婷

前　言

软件体系结构是软件系统的高层结构，是一种用于理解系统级目标的框架。随着软件系统规模越来越大、越来越复杂，整个系统的结构和规格说明就显得越来越重要。对于大规模的复杂软件系统来说，总体的系统结构设计和规格说明比对计算的算法和数据结构的选择更为重要。在这种背景下，人们认识到了软件体系结构的重要性，并认为对软件体系结构进行系统、深入的研究将会成为提高软件生产率和解决软件维护问题的新的最有希望的途径。

软件体系结构研究的主要内容包括软件体系结构描述、软件体系结构设计、基于体系结构的软件开发方法、软件体系结构评估等。

1．软件体系结构描述研究

体系结构描述语言(Architecture Description Language，ADL)是一种形式化语言，系统设计师可以利用它所提供的特性进行软件系统概念体系结构建模。ADL 提供了具体的语法与刻画体系结构的概念框架。它使得系统开发者能够很好地描述他们设计的体系结构，以便与他人交流，能够用提供的工具对许多实例进行分析。

Kruchten 提出的“4+1”模型是对软件体系结构进行描述的另一种方法，该模型由逻辑视图、开发视图、过程视图和物理视图组成，并通过场景将这 4 个视图有机地结合起来，比较细致地描述了需求和体系结构之间的关系。

Booch 从 UML 的角度给出了一种由设计视图、过程视图、实现视图和部署视图，再加上一个用例视图构成的体系结构描述模型。可以使用 UML 对构件交互模式进行静态建模和动态建模。

2．软件体系结构设计研究

体系结构设计研究的重点内容之一就是体系结构风格的研究。人们在开发不同系统时，会逐渐发现一类系统的体系结构有许多共性，于是抽取出这些共性构成一些富有代表性和被广泛接受的体系结构风格。它定义一组构件、连接件的类型以及它们之间应该如何连接的约束。虽然系统组织方式可以是无穷的，但如果能用少量的风格类型表达出较多的系统组织方式，不仅可以缩短系统分析设计的时间，还能大大提高大规模软件重用的机会。

通过对软件体系结构设计过程进行研究，总结出软件体系结构设计过程中用到的一般原理主要有以下几个：抽象、封装、信息隐藏、模块化、注意点分离、耦合和内聚、接口和实现分离、分而治之、层次化等。

在几十年的软件设计研究和实践中，设计人员和程序员积累了大量的实际经验，发现并提出了大量在众多应用中普遍存在的软件结构和结构关系，模式被用于软件体系结构设计中。一个模式关注一个在特定设计环境中出现的重现设计问题，并为它提供一个解决方案。利用设计模式可以方便地重用成功的设计和结构。

试读结束：需要全本请在线购买：　www.ertongbook.com

生成一个满足软件需求的体系结构的过程即为体系结构设计。体系结构设计过程的本质在于：将系统分解成相应的组成成分(如构件、连接件)，并将这些成分重新组装成一个系统。常用的体系结构设计方法有 4 类，分别为制品驱动(Artifact-Driven)的方法、用例驱动(Use-Case-Driven)的方法、模式驱动(Pattern-Driven)的方法和领域驱动(Domain-Driven)的方法。

3. 基于体系结构的软件开发方法

在引入了体系结构的软件开发中，应用系统的构造过程变为“问题定义→软件需求→软件体系结构→软件设计→软件实现”，可以认为软件体系结构架起了软件需求与软件设计之间的一座桥梁。而在由软件体系结构到实现的过程中，借助一定的中间件技术与软件总线技术，软件体系结构易于映射成相应的实现。Bass 等人提出了一种基于体系结构的软件开发过程，该过程包括 6 个步骤：导出体系结构需求、设计体系结构、文档化体系结构、分析体系结构、实现体系结构、维护体系结构。

4. 软件体系结构评估

由于软件体系结构是在软件开发过程之初产生的，因此好的体系结构可以减少和避免软件错误的产生和维护阶段的高昂代价。想要判断所使用的体系结构是否恰当，需要使用专门的方法对软件体系结构进行分析和评估。常用的软件体系结构评估方法有软件体系结构分析方法(Software Architecture Analysis Method，SAAM)和体系结构权衡分析方法(Architecture Tradeoff Analysis Method，ATAM)。

5. 特定领域的体系结构框架

特定领域的应用通常具有相似的特征，如果能够充分挖掘系统所在领域的共同特征，提炼出领域的一般需求，抽象出领域模型，归纳总结出这类系统的软件开发方法，就能够指导领域内其他系统的开发，提高软件质量和开发效率，节省软件开发成本。正是基于这种考虑，人们在软件的理论研究和工程实践中，逐渐形成一种称之为特定领域的软件体系结构(Domain Specific Software Architecture，DSSA)的理论与工程方法，它对软件设计与开发过程具有一定参考和指导意义，已经成为软件体系结构研究的一个重要方向。

6. 软件体系结构支持工具

几乎每种体系结构都有相应的支持工具，如 UniCon、Aesop 等体系结构的支持环境，C2 的支持环境 ArchStudio，Acme 的支持环境 AcmeStudio，支持主动连接件的 Tracer 工具等。另外，还出现了很多支持体系结构的分析工具，如支持静态分析的工具、支持类型检查的工具、支持体系结构层次依赖分析的工具等。本书通过两个较为著名的软件体系结构集成开发环境 ArchStudio 4 和 AcmeStudio，介绍了软件体系结构集成开发环境的具体功能。

本书共分 10 章。第 1 章简单介绍了软件重用和构件技术的一些基本概念，它们是学习软件体系结构有关知识的基础；第 2 章介绍了软件体系结构的概念、发展和研究现状；第 3 章对软件体系结构的风格进行了较详细的介绍，并给出了一些应用实例；第 4 章讨论了软件体系结构的三种描述方法；第 5 章介绍了软件体系结构设计过程中使用的一般原理和设计模式，以及常用的体系结构设计方法；第 6 章对 Bass 等人提出的一种基于体系结构的软

件开发过程做了详细介绍；第 7 章介绍了软件体系结构评估方法，重点介绍了 SAAM 和 ATAM 方法；第 8 章介绍了 Web 服务体系结构有关知识，并给出了一个简单的应用实例；第 9 章对特定领域的软件体系结构进行了介绍，详细讨论了其建立过程；第 10 章介绍了软件体系结构集成开发环境的具体功能。

本书由付燕任主编并编写第 5、6、8、9 章，龚尚福编写第 1、2 章，龙熙华编写第 3 章，李贵民编写第 4、7 章，黄旭、王丽雯编写第 10 章，张相绘制了书中部分插图。

编写本书时，曾直接或间接地引用了许多专家、学者的文献(详见书后参考文献)，在此向他们深表谢意。同时也感谢西安电子科技大学出版社的大力支持。

由于软件体系结构的发展非常迅速，其本身也在不断发展完善，又因为作者水平有限，虽然尽了很大的努力，书中仍难免有疏漏和不妥之处，敬请读者批评指正。

编　者

2009 年 5 月

目　录

试读结束：需要全本请在线购买： www.ertongbook.com

第 1 章　软件重用与构件技术

1.1　软件重用概述

自从 1968 年 NATO(North Atlantie Organization，北大西洋公约组织)会议提出“软件危机”以来，软件工程取得了非常大的进展。然而随着计算机应用领域的迅速延伸，软件规模不断扩大，软件复杂性不断提高，又出现了新一轮的“软件危机”。软件重用无疑是解决这一问题行之有效的方法。软件重用是软件开发中避免重复劳动的解决方案，其出发点是应用系统的开发不再采用一切“从零开始”的模式，而是在已有的工作的基础上，充分利用以前系统开发中积累的知识和经验，如需求分析结果、设计方案、源代码、测试计划及测试案例等，将开发的重点转移到现有系统的特有构成成分。通过软件重用，不仅可以提高软件开发效率，减少分析、设计、编码、测试等过程中的重复劳动，而且因为重用已经过充分测试的软件开发成果，可以避免重新开发引入的错误，从而提高当前软件的质量。

1.1.1　软件重用的定义

软件重用(Software Reuse)是一种由预先构造好的、为重用目的而设计的软件构件来建立或者组装软件系统的过程。它的基本思想是放弃那种原始的、一切从头开始的软件开发方式，而利用重用思想，通过公共的可重用构件来集成新的软件产品。

随着软件重用思想的深入，可重用构件不再仅仅局限于程序源代码，已经延伸到包括对象类、框架等在内的软件开发各阶段的成果。在位于不同抽象层次、不同大小的软件构件，以及文档交付方面，不同类型可重用构件的例子包括：应用包、子系统、数据类型定义、设计模型、规格说明、代码、文档、测试用例和测试数据。在面向对象开发中，不同类型可重用构件的例子包括应用框架、用例、高层对象类、分析和设计模型、类定义、基本对象类(如底层的类、日期类和字符串类等)、类库(如一组支持某一领域的相关类的组合——图形用户界面和数据库等)、方法(如类的服务或者类的行为)、测试包(如测试用例、测试数据和预期结果)、函数(如程序模块)、文档(如分析文档和设计文档)、项目、测试和实施计划的框架等。

1.1.2　软件重用的研究现状

最早的软件重用可以追溯到子程序库的使用，但正式提出软件重用的概念是在 1969 年举行的首次讨论软件工程的国际会议上，D. Mcilroy 发表了题为“Mass-Produced Software Components”的论文，提出建立生产软件构件的工厂，用软件构件组装复杂系统的思想。

此后 10 年中，有关软件重用的研究没有取得很大进展，直到 1979 年 Lanergan 发表论文，对其在 Raythen Missile Division 的一项软件重用的项目研究进行总结，才使得有关软件重用的研究重新引起人们的注意。据 Lanergan 称，他们分析了 5000 个 COBOL 源程序，发现在设计和代码中有 60%的冗余，此外，大部分商业应用系统的逻辑结构或设计模式属于编辑、修改、报表生成等类型。假如将以上这些冗余代码和模块重新设计，并进行标准化，那么将在 COBOL 商业应用程序中获得 15%～85%的重用率。

1983 年，由 ITT 赞助，Ted Biggerstaff 和 Alan perlis 在美国的 Newport 组织了第一次有关软件重用的研讨会。随后在 1984 年和 1987 年，国际上权威的计算机杂志 IEEE Transactions on Software Engineering 和 IEEE Software 分别出版了有关软件重用的专辑。1991 年，第一届软件重用国际研讨会(International Workshop on Software Reuse，IWSR)在德国的 Dortmund 举行，第二届 IWSR 于 1993 年在意大利的 Lucca 举行，从 1994 年的第三届 IWSR 起，软件重用国际研讨会改称为软件重用国际会议。美国国防部的 STARS 计划是较早的一个由政府资助的有关软件重用研究项目。STARS 的目标是在大幅度提高系统可靠性和可适应性的同时提高软件生产率，虽然该计划的目标是要构造一个软件开发支撑环境，但计划的重点之一是软件重用技术。

国内的相关研究也较多，如青鸟构件库管理系统(JBCLMS)是北京大学软件工程研究所在杨芙清院士领导下的研究成果，它的目标是致力于软件重用，以构件作为软件重用的基本单位，提供一种有效的管理和检索构件的工具。JBCLMS 作为企业级的构件管理工具，可以管理软件开发过程的不同阶段(分析、设计、编码、测试等)、不同形态(如需求分析文档、概要设计文档、详细设计文档、源代码、测试案例等)、不同表示(如文本、图形等)的构件，提供多种检索途径，以便于快速检索所需构件。

1.1.3 重用驱动的软件过程

1．软件重用失败的原因

尽管软件产业从本质上是支持重用的，但到目前为止，很少有成功实施重用的公司。主要原因有以下几点：

(1) 缺乏对为什么要实施重用的了解。

(2) 认为重用没有创造性。

(3) 管理者没有对重用承担长期的责任和提供相应的支持。

(4) 没有支持重用的方法学。

(5) 不愿意改变当前的工作方式。

(6) 管理者不信任重用带来的商业价值。

(7) 没有支持实施重用的工具。

(8) 没有可重用的构件，没有构件重用库。

软件重用失败的原因之一是缺少支持软件重用的软件过程，因此，为了更好地实施软件重用，需要有一个重用驱动的软件开发过程。

2．重用驱动的软件过程

一个重用驱动的软件过程描述了如何组装可重用构件来建立软件系统，以及如何建立

和管理可重用构件。如图 1-1 所示，一个重用驱动的软件开发过程可从两个视角来看待重用。

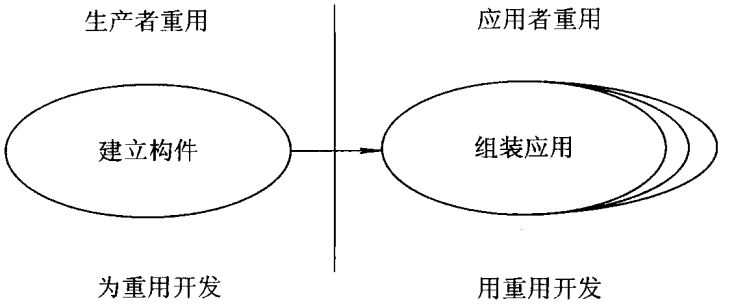

图 1-1　重用驱动的软件开发过程

(1) 应用者重用(Consumer Reuse)：使用可重用构件去建立新软件系统的活动，即用重用构造的视图。

(2) 生产者重用(Producer Reuse)：建立、获取或者重新设计可重用构件的活动，即为重用构造的视图。

应用者重用关心利用可重用构件来建立新系统，它包括以下几个步骤：

(1) 寻找候选的可重用的构件，由它们来产生软件生命周期每一阶段的交付。

(2) 对候选构件进行评价，选择那些适合于在本系统内重用的构件。

(3) 更改可重用构件或者对它们进行特殊处理，使之能满足本系统的要求。

(4) 为可重用构件提出一些更改或者改进之处，增强它们未来的可重用性。

生产者重用关心开发可重用的软件构件，它包括以下几个步骤：

(1) 识别应建立的或者准备重用的候选构件。

(2) 完成共性/差异分析，确定使构件能够重用所必须满足的要求。

(3) 建立或者准备可重用构件。

(4) 封装该可重用构件。

(5) 将可重用构件加入到重用目录或重用库中。

1.2　构件的特点和分类

一般认为，构件是指语义完整、语法正确和有可重用价值的单位软件，是软件重用过程中可以明确辨识的系统。结构上，它是语义描述、通信接口和实现代码的复合体。简单地说，构件是具有一定的功能，能够独立工作或能同其他构件装配起来协调工作的程序体。构件的使用同它的开发、生产无关。从抽象程度来看，面向对象技术已达到了类级重用(代码重用)，它以类为封装的单位。这样的重用粒度还太小，不足以解决异构互操作和效率更高的重用。构件将抽象的程度提到一个更高的层次，它对一组类的组合进行封装，并代表完成一个或多个功能的特定服务，也为用户提供了多个接口。整个构件隐藏了具体的实现，只用接口对外提供服务。

构件是组成软件的基本单位，它包含以下 3 个内容：

(1) 构件是可重用的、自包含的、独立于具体应用的软件对象模块。

(2) 对构件的访问只有通过其接口进行。

(3) 构件不直接与别的构件通信。

1.2.1 构件的特点

总的说来，软件构件具有以下特点：

(1) 以二进制形式存在，软件构件一般不再以源代码方式实现重用。

(2) 可以与其他独立开发的软件构件协同工作，经过少量的修改，软件构件可以很容易地移植到其他构件生产商所生产的构件中。

(3) 软件构件具有相对独立的功能，可以顺利地将软件构件组合成一个应用系统。

(4) 与程序设计语言无关，软件构件不依赖于任何一门编程语言，这便于软件开发人员将其与别的软件构件组装。

(5) 成为其他软件构件的生成模块，软件构件和一般的对象相比，可大可小。

(6) 具有良好定义的接口。

1.2.2 构件的分类

构件种类非常丰富，从不同角度可以将构件分为很多类。

(1) 根据构件重用的方式，分为黑盒构件和白盒构件。黑盒构件是不需了解其内部结构就能通过接口从外部调用，能达到即插即用效果的构件；而白盒构件是必须经过修改才能重用的构件。

(2) 根据构件功能用途，分为系统构件、支撑构件和领域构件。系统构件是在整个构件集成环境和运行环境中都可以使用的构件；支撑构件是在构件集成环境和构件管理系统中使用的构件；领域构件是为专门应用领域制定的构件。

(3) 根据构件粒度的大小分为基本数据结构类构件(如窗口、按钮、菜单等)、功能构件(如录入、查询、删除等)和子系统构件(如人事管理、学生入学管理、图书馆信息管理等)。

(4) 根据构件重用时的形态，分为动态构件和静态构件。动态构件是运行时可动态嵌入、链接的构件，如对象链接和嵌入、动态链接库等；静态构件如源代码构件、系统分析构件、设计构件和文档构件等。

(5) 根据构件的外部形态，将构成一个系统的构件分为以下 5 类：

- 独立而成熟的构件。独立而成熟的构件得到了实际运行环境的多次检验，该类构件隐藏了所有接口，用户只需用规定好的命令使用即可，例如数据库管理系统和操作系统等。
- 有限制的构件。有限制的构件提供了接口，指出了使用的条件和前提。这种构件在装配时会产生资源冲突、覆盖等影响，在使用时需要加以测试，例如各种面向对象程序设计语言中的基础类库等。
- 适应性构件。适应性构件对构件进行了包装或使用了接口技术，对不兼容、资源冲突等进行了处理，可以直接使用。这种构件可以不加修改地使用在各种环境中，例如 ActiveX 等。
- 装配的构件。装配的构件在安装前已经装配在操作系统、数据库管理系统或信息系统不同层次上，使用胶水代码就可以进行连接使用。目前一些软件商提供的大多数软件产品都属于这一类。

试读结束：需要全本请在线购买： www.ertongbook.com