软件开发与软件架构

仲萃豪　著

科学出版社

软件开发与软件架构

仲萃豪　著

科学出版社
北京

内 容 简 介

大型应用软件的开发主要面临着开发效率低和需求适应性差两大难题，解决问题的核心是建立优秀的软件架构。本书主要探讨开发大型应用软件的理论与实践，特别是分布式系统应用软件，并以软件架构为主线对各种典型模型、方法和工具进行讨论。

本书作者是我国著名的软件工程专家。书中内容从哲理、原理、方法技术和实践四个方面展开，是作者多年科研成果、经验与感悟的总结，反映了软件工程领域技术热点与发展趋势。

本书适合作为计算机科学与技术、软件工程等专业的研究生教材，也适合相关领域的软件架构师、软件工程师和其他工程技术人员阅读。

图书在版编目(CIP)数据

软件开发与软件架构/仲萃豪著. —北京：科学出版社，2013
ISBN 978-7-03-038280-1

Ⅰ.①软… Ⅱ.①仲… Ⅲ.①软件开发 ②软件设计 Ⅳ.①TP311.5

中国版本图书馆 CIP 数据核字(2013)第 181937 号

责任编辑：任 静 张 濮 /责任校对：宣 慧
责任印制：张 倩 /封面设计：迷底书装

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

骏主印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2013 年 7 月第 一 版 开本：B5(720×1000)
2013 年 7 月第一次印刷 印张：10
字数：183 000

定价：39.00 元

(如有印装质量问题，我社负责调换)

前　言

大型应用软件的开发往往都存在着开发周期长、成本高、质量差、适应性差、维护难等诸多问题。其中，最为关键的问题是如何提高软件的开发效率，最为困难的问题是如何适应系统需求的变化。本书主要针对上述两个重要问题，探讨如何开发大型复杂应用软件，特别是分布式系统应用软件。

目前，提高软件开发效率和软件对系统需求变化的适应性主要有以下三大途径。

1)软件重用

重用技术被视为解决软件危机、提高生产率和改善软件质量的最现实可行的途径。其解决方案是近年来国际流行的基于构件技术的开发模型、基于面向服务架构的开发模型和云计算开发模型。当把应用软件开发看做产品生产时，就要采用规模化的软件生产线方式。软件生产线像建筑业、汽车业那样，将小作坊的生产方式改变成具有社会分工的基于构件技术的流水线自动化生产方式。采用软件重用技术，会大大提高软件开发效率，同时也不排斥采用自动生成技术(语言编译)和各种工具、技术、平台等开发环境。软件重用综合了上述各方面，能提高软件开发的生产力和产品质量，也能适应用户对系统需求的不断演化，从而解决软件危机。

2)自动化

全盘自动化软件开发是我们遥远的目标，但采用语言编译和用专用语言实现局部自动生成是有成效的(如报表生成)。当前的各种第四代语言比第三代语言可提高开发效率四倍以上，如 SQL、4GL、PB、VB 等;用描述语言(如 UML)编写的文档有助于开发过程规范化、过程重用和自动化;将编译语言与开发平台或开发环境结合起来，实现语法导引技术，是一种新途径，如 Shell 语言和国产“易”语言;业务流程执行语言(BPEL)及相应的各种构件库与服务库，大大提高了大型应用软件的开发效率。

3)开发工具和开发平台

为了将软件开发过程中各种方法和技术融合在一起，需要在软件开发各阶段，针对各种开发模型设计相应的开发平台。在十几年前，根据这种思想，国内外已研制过各种开发工具，并建立了开发平台，称为计算机辅助软件工程(CASE)，但成功的案例较少。经过多年的发展和努力，随着开发方法和开发模型的演变，现在已有大量的实用产品，国内几乎所有的软件开发公司都已为自己的开发领域建立了相应的平台环境。“青鸟工程”二期提出的“基于构件/构架技术的软件流水线生产”软件研发平台和北京市科学技术委员会的软件与信息服务业促进中心提出的“SOA 软件的开发和运行平台”都是极好的例子。

大型应用软件开发的核心问题是如何建立软件架构，能否很好地解决这一问题已成为关系软件开发成败的关键因素。因此，本书围绕着软件架构来介绍大型复杂应用软件的开发，并以软件架构为主线展开对各种典型模型、方法和工具的讨论。

大型应用软件架构与开发方法是当前发展最为迅速的信息技术之一。在几十年的发展过程中，涉及众多的理论、模型与工具。因此，本书在取材上不求面面俱到，而是按照时间线索详细介绍最具代表性的软件架构与开发思想、方法、技术及工具，并特别注重结合目前相关领域的研究热点，尤其是在分布式软件架构和开发方面，使读者既能够在短时间内了解这些领域的主要研究成果，又可以尽快地了解当前最新的实用方法与技术，掌握其中的精华。

本书是根据作者多年研究成果和国际上新技术的发展趋势编写而成的，收入了成熟的和有实用意义的内容，也反映了作者多年来积累的研发经验，其中所用实例都是亲身的体会，内容生动、精炼、易懂、易读，包含了作者的学术观点。书中还穿插了不少在软件发展中有意义的故事，这些故事有助于读者领会在大型应用软件设计和开发工作中蕴涵的哲理。

软件架构与开发是一门学科，可以分为哲理、原理、方法技术和实践四个方面进行论述。按照这一思路，本书的第一部分讲述软件架构与开发的哲理——软件的认知体系，并由此导出三种开发模型。第二部分针对三种模型讲解其开发原理。第三部分讲述相关的方法、技术和工具，包括不同开发阶段的平台、开发模型、架构、框架、工具、中间件等。第四部分围绕面向服务的架构，进一步分析在应用中经过验证的各种工具与平台的实例。

本书的主要内容曾在中国科学院软件研究所和计算技术研究所作为博士生学位课的授课内容，在内容上力求既保持相对成熟，又做到密切跟踪最新技术。本

书适合作为计算机科学与技术、软件工程等专业的研究生教材，也适合相关领域的软件架构师、软件工程师和其他工程技术人员阅读。

本书能得以出版，要特别感谢中国科学院软件研究所李明树所长等领导的关心、支持和资助。感谢詹少梅女士、中国科学院软件研究所吕品老师、彭军老师等在作者撰写初稿过程中给予的建议。

由于时间仓促，书中疏漏和不足之处在所难免，恳请广大读者批评指正。

目　录

第二部分　原　　理

第三部分　方法、技术和工具

第四部分　SOA 与软件开发方法

第一部分　哲　　理

第 1 章　大型应用软件架构与软件开发的认知体系

计算机科学从诞生起虽然仅有半个多世纪，但其发展速度却远远超出了人们的预计。任何一门学科从萌芽、发展到成熟，必须要解决哲理、原理、方法技术、工程管理和实践五个层次的问题，由此可以从这五个方面来衡量一个学科的成熟度。在这一过程中，任何方法、技术与工具都是由于要解决社会发展中的需求而产生的，而衡量其成熟度，就要看它是否真正解决了社会的需求，是否符合人们认识客观世界的规律，以及是否已建立了相应的理论和工程管理体系。就软件架构与开发而言，围绕上述五个层次的核心问题就是要研究软件开发中的方法和技术，并进一步解决相应的工程管理问题。因此，认知观在软件开发的发展史上起到了关键作用。不同的认知观决定了不同的软件架构与开发方法，认知观是原理和方法的出发点，是灵魂。我们将上述思想称为软件架构与开发的哲理。

软件行业是一种知识密集型的行业，软件开发长期以来存在“开发周期长、成本高、质量差、适应差、维护难”这五大难题，早期称为“软件危机”。“危机”震动了软件界，为此，学术界和产业界提出了一系列的技术和方法，而对这些技术与方法需要从一个客观、统一的视角进行审视。本章将从认知观的角度，来讨论软件开发方法和技术的发展历程，从而能够更好地帮助读者理解每种方法和技术的内涵。同时，也可以从哲学的高度，分析、评论与比较软件行业发展中的一些里程碑式的关键方法与技术。

1.1　面向过程思想的认知观

1.1.1　结构程序设计方法与软件工程

20 世纪 60 年代末到 20 世纪 70 年代初，出现了大型的软件系统，如 IBM 360 系列机操作系统和美国三大航空公司分别开发的机票预订系统等。这给程序的设计带来了新的问题，即大型系统的研制需要花费大量的财力和人力。IBM 公司当

时采用的是大兵团作战方式，类似于“大跃进”时期的人海战术。每个操作系统的版本都要花费 3000 人·年以上的工作量。可是，研制出来的产品可靠性差、错误多，维护和修改也很困难。当时，人们称这种现象为“软件危机”。

“危机”震动了软件界。一方面，程序设计的传统习惯和工作方式导致了不清晰的程序结构，很难保证程序的可靠性；另一方面，程序设计工具的严重缺乏也使大系统的开发陷入困境。追根溯源，人们开始重新研究程序设计中的一些最基本的问题。例如，程序的基本组成部分是什么？应该用什么样的方法来设计程序，并保证程序设计的正确性？程序设计的主要方法和技术应如何规范化？

正在人们束手无策的时候，荷兰科学家 Dijkstra 提出“GOTO 语句有害”(Dijkstra，1968)的论点，这一导火线引起了世界软件界的争议。接着，Dijkstra 发表了一系列的文章，提出了“结构程序设计”的思想，他认为“人的智力是有限的”。关于 GOTO 问题的这场争论持续了数年之久，直到 1974 年，Knuth 发表了文章 *Structured Programming with go to Statements*(Knuth，1974)之后，才平息了这场争论。他主张在语言控制成分中仍保留 GOTO 语句，在功能方面不加限制，但限制其使用范围，并且不必对一般用户开放。许多重要的程序设计语言，如 Pascal，即持此态度。

所谓“结构程序设计方法”，就是如何使一个庞大复杂的问题从结构上加以简化，从而得到一个结构清晰的程序。其主要思想是：通过研究程序基本数据结构和控制结构(类似于物质是由原子和分子组成的)，采取与数学求解过程极其相似的设计方法，采用逐步求精、自顶向下的数学求解过程，使人们用有限的智力去解决复杂的问题，从而确保了系统的正确性和可维护性。结构程序设计的本质思想是：软件应采用层次化的软件体系结构(也称为软件架构)。按此思想设计软件就必然导出逐步求精、自顶向下的设计方案，即如何把软件架构按层次模型的架构来设计。

结构程序设计方法的推行，意义并不止于其实用价值，更深远的意义在于它向人们揭示了研究程序设计方法的必要性。程序设计并不纯粹是一种技术性的活动，与任何学科一样，它自身也有一套基本的原理和方法。

20 世纪 70 年代，程序设计技术的发展是以方法论的研究为特征的。除结构程序设计的方法日益完善外，在数据类型抽象、程序的推导与综合、程序变换和程序自动化等方面，都取得了重要的进展。

20 世纪 80 年代初，随着对程序设计理论和方法学的不断深入研究，人们总结出一整套关于如何进行程序设计的原则和方法，这里不仅包括结构化程序设计，而且还涉及程序语义性质的表示、程序正确性的证明、程序形式化的开发等许多方面。这就使程序设计方法学作为计算机科学的一门前沿学科应运而生了。程序设计方法学的目标在于改善程序设计的过程，使之更加科学化、规范化。

到 20 世纪 80 年代后期，人们开始把结构程序设计方法推广并应用到大型软件的开发中来，称为结构化分析和结构化设计(structure analysis and structure design，SASD)，引用工程中的原理和方法，形成了“软件工程”这门学科，并研制了相应的计算机辅助设计工具环境和开发规范(简称 CASE)，在改善程序生产率和程序质量方面迈出了一大步。这两个学科奠定了软件开发最早的基础理论。

在这一时期，对于大型应用软件开发的认知观是：人的智力是有限的，要将复杂问题简化到人的智力能达到的程度。

在开发方法方面，采用结构程序设计方法(即软件层次架构的模型)，并沿用数学研究上的思维方法，用枚举、抽象、归纳、类比等方法将复杂的问题简化，使软件结构清晰，从而确保了软件的正确性。

开发模型采用瀑布模型，按照需求分析、设计、实现、测试(确认)、集成和维护坚定地、顺畅地进行。

软件架构采用层次化架构。

开发技术包括数据和控制结构、程序正确性、逐步求精、数据抽象模块、程序推导和变换等。

编程语言包括 Pascal 语言、Ada 语言、XCY 语言等。

1.1.2　对 SASD 方法的责难

软件危机的出现导致了软件工程的诞生，把软件开发作为一种生产制造过程，将软件看做工厂生产制造过程的产品，整个过程用工程原理进行管理和控制。类似于在工厂的车间中配置各种车床和工具，对软件生产来说，就应配置相应的开发平台和开发工具。在“七五”期间，为了建立攻关项目“青鸟工程”，本书作者既参加了攻关项目，负责新开发机制的研究，同时又创办了多个软件公司，从事信息管理系统(MIS)的开发。当时，国家政策规定，企业必须配有计算机管理才能被国家验收为一级企业，因此承接的项目很多。虽然出现了一系列以 SASD 方法、技术和工具为代表的成果，但软件事业还远远没有达到人们所期望的那样兴