实时设计模式

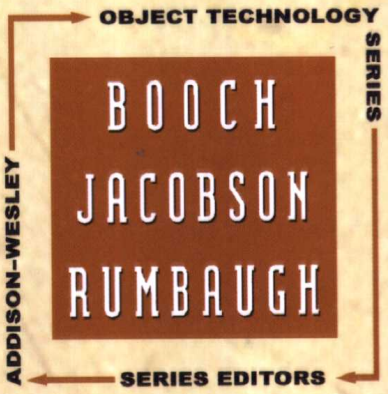
—实时系统的强壮的、可扩展的体系结构

Real-Time Design Patterns

Robust Scalable Architecture for Real-Time Systems

[美] Bruce Powel Douglass 著

麦中凡 陶 伟 译



北京航空航天大学出版社

实时设计模式

——实时系统的强壮的、可扩展的体系结构

Real-Time Design Patterns
Robust Scalable Architecture for Real-Time Systems

[美] Bruce Powel Douglass 著

麦中凡　陶伟　译

北京航空航天大学出版社

北京市版权局著作权登记号:图字:01—2003—0420

图书在版编目(CIP)数据

实时设计模式:实时系统的强壮的、可扩展的体系结构/(美)道格拉斯(Douglass, B. P.)著;麦中凡等译.
北京:北京航空航天大学出版社,2004.5

书名原文:Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems

ISBN 7-81077-421-2

Ⅰ.实… Ⅱ.①道…②麦… Ⅲ.面向对象语言,UML—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2004)第 013753 号

实时设计模式

——实时系统的强壮的、可扩展的体系结构

[美] Bruce Powel Douglass 著

麦中凡 陶伟 译

责任编辑 胡晓柏

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083) 发行部电话:(010)82317024 传真:(010)82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

涿州新华印刷有限公司印装 各地书店经销

*

开本:787 mm×960 mm 1/16 印张:26 字数:582 千字

2004 年 5 月第 1 版 2004 年 5 月第 1 次印刷 印数:5 000 册

ISBN 7-81077-421-2 定价:56.00 元

译者序

本书是美国 Addison-Wesley 公司出版的对象技术(最新 2003 年)丛书之一,旨在用面向对象技术开发实时和嵌入式系统软件。

众所周知,实时和嵌入式系统软件由于其开发层次较低,直接在硬件上开发时间紧要、安全紧要、高可靠性的系统,传统上是 C 语言和汇编语言的天下。传统的软件工程技术着重软件的可移植、可复用、可伸缩,易维护、低成本,借助于一个良好的平台,快速交付,支持业务过程的快速变革的高适应性系统。软件工程使平台以上的系统日臻完善,发展了面向对象技术、构件技术乃至于直接使用软件服务。"相信"实现服务的构件、支持构件实现的平台(操作系统、网络软件及数据库软件)都是最优或较优的。显然,与实时嵌入式系统要处处操心每个软件元件的性能、可靠性、安全性走的是两条路,他们"不相信"臃肿的通用平台能解决他们在有限资源下做出高性能系统所遇到的各种问题。所以,实时和嵌入式系统的业者一般不关心软件工程技术的最新进展,也很少在他们的工作中采用软件工程新技术,停留在提供完善的模块和子系统层次上,强耦合的过程式模块开发还是主流。

从系统的观点来看,实时嵌入式系统也是系统,特别是硬件技术的快速发展,实时嵌入式软件也有快速适应硬件型号升级问题,也有业务快速变更要求可伸缩、可修改(例如,军用导弹、飞机型号升级,民用产品换代)可复用问题。1975 年,美国国防部开发的 Ada 语言系统就是为了在主机上开发适应变化的系统。通过交叉编译生成机载、弹载的目标系统嵌入到应用环境中,一切系统维护、伸缩均在 Ada 源代码级软件上完成。在某种意义上,今日的面向对象技术得益于 Ada 的数据抽象和模块封装。由于 Ada 为保证军用软件可移植性管得过严,面向对象技术蓬勃发展在 C++上得以体现。于是实时嵌入式系统自然转向底层是 C 的 C++。但对象化对实时性、可靠

性并没有直接的好处，所以在小型实时嵌入式应用中，面向对象依然不占上风，仍然是过程式开发方法最后穿上C++外衣。尽管如此，有了C++这个桥，为实时嵌入式系统再一次和当前软件工程技术合流打下了良好的基础。

面向对象封装带来的松耦合，使它成为分布式可伸缩系统的首选技术，且随着当今网络的普及，对象成了应用的第一类公民，网络是对象交互和通信的世界。打包也好，请代理也好，不是对象只能窝在本机上做点实际工作。

为了参与网络上提供实时服务，为了支持应用系统的快速变更，快速提供嵌入式产品，大幅度降低成本提供标准化、构件化产品，实时嵌入式系统的对象化、构件化势在必行！当然，也应该看到，CPU速度大幅提高，减缓这类系统的开发者因实时对标准化、构件化的抵制，使他们乐于参加现代技术大合奏。

重新认识软件是有体系结构的。软件开发是一个软件过程，是近十年软件技术最重要的成就。过去的开发着眼于功能性能，无形中完成了一个软件过程，得到一个有体系结构的产品。当时并不在意过程好坏，体系结构是什么样的。通过验收能够满足使用的就是好产品。但情况并非如此顺利，“树欲静而风不止”，使用中不断发现新bug，用户一再要求改进（受其他新技术影响或总结出使用经验）。过程和体系结构在软件可适应、易维护性上显现出极大的威胁力。不良的体系结构使此处bug消除，彼处新bug又冒出，甚至导致整个系统崩溃。不良的过程找不到资以分析的文档、数据，一个不大的改进每步都从头试起，迟迟调不出来，大量窝工，成本飚升。于是，人们把软件开发从以功能性能为中心转而为以软件体系结构为中心，精心改进软件过程，从而软件在改变之中依然能保证符合需求的功能，并能保证高质量和低成本。

体系结构从总体上决定了软件可能达到的各项质量指标。例如，别墅、板楼、塔楼，在通风、抗震、舒适程度、方便性、成本方面各不相同，一经确定则影响终身。例如楼层太矮、电梯过少、没有绿地等。软件过程通过精心安排软件制作的各项活动切实保证软件质量。该有的活动没有或不能即时到位都会造成低质量、高成本，甚至项目失败。过程是产品质量最直接的保证。所以ISO9000系列就是所有人工制品的（过程）质量标准。美国CMM标准化更是以过程成熟度来衡量软件企业有无资质承担软件开发。软件体系结构和软件过程相辅相成，一为软件内在质量保证，一为过程质量保证。所以，以体系结构为中心的最优过程开发已为世界广泛接受。

过程是以时间为进程的各项软件活动。开发活动完成都凝聚成有体系

结构的软件，即它的后果（体系结构）是可分析、可测试度量的。优秀的体系结构如前所述，正是软件设计的目标。于是人们研究分析体系结构并寻找设计实现的途径。

体系结构不外乎是实现各项功能、性能的元构件有机组合的集合。在面向对象的背景下，它们都是类对象，为了设计（有机组合）方便，把对象组成惯用的、不易出错的、可靠的、接近标准化的形式，则为模式（pattern）。最小的模式是几个类对象组成的构件，最大的模式是若干构件组成的子系统。

最早也是最基础的设计模式是 Liskov 提出的七种基本模式，并提出对象构成模式的五条基本准则，为面向对象设计模式奠定了理论基础。随后是“四人帮”（Gamma 等四人）提出了常用的 25 种设计模式，为以模式设计软件体系结构提供了工程实践的基础。不过，“四人帮”讨论的模式作用域是局部的，我们把它们叫做机制式的设计模式（mechanistic design pattern），因为它们为对象的协作定义了各种机制。这种模式有较大的局限，只在单个的协作内。本书并不讨论机制式的设计模式，本书讨论的是体系结构设计模式（architecture design pattern）。

当然，各软件应用领域也都有本领域的模式，如果总结出来作为准规划，则本领域软件开发速度会成倍提高，而软件质量也较容易得到保证。本书就为实时嵌入式领域总结出了 54 种模式。

本书作者 Bruce Powel Douglass 博士在实时嵌入式系统领域工作已有 26 年，是世界公认的领袖人物，目前是本书推荐的 ROPES（嵌入式系统的快速面向对象过程）的创始人。他也即时转入到面向对象领域，成为 I-Logix 公司的首席技术指导，实时系统开发工具的领导者，也是 UML 规范嵌入系统会议顾问委员会的成员。Douglass 博士为多个公司和组织做过咨询，包括 NASA。他还是 OMG 的 RTDA（Real-Time Analysis and Design Working Group）的副主席和面向对象杂志的专栏作家。

本书可以作为实时嵌入式系统的设计人员和开发人员的技术参考书，也可以作为计算机、自动化、机电一体化等专业的大学高年级学生学习操作系统、计算机控制系统等课程辅助教材。同时，本书对正在学习 UML 建模的设计人员也是一个很好的范例。此外，本书可以作为软件开发组织知识共享的模板，参考本书描述模式的方式把技术人员解决问题的经验抽象化、形式化，在组织范围内共享。

译者积极翻译并向业界推荐本书还有一个原因，这就是本书以对象模式的观点把计算机最底层的操作功能实现整个演示了一遍，如并发进程、内

存管理、资源锁定、进程分布、安全性和可靠性。这对于欲深入理解计算实现本质的读者(不一定去做实时嵌入式应用),是一本很好的教材。在翻译过程中为减少混乱,我们厘定某些术语的译法,也算对业界的建议,如有不妥之处,欢迎指正。

Broker 译为“中介”,当前 Broker、Proxy、Agent、Delegation 均为“代理”,而 Broker、Proxy 在本书中也常出现在一页上,故 Broker 为“中介”(无授权),Proxy 为“代理”(有授权),而 Delegation 干脆是“委派”, Agent 在本书中出现极少,且不易冲突仍沿用“代理者”。

本书的前言、序和第 1 章至第 3 章由麦中凡教授翻译,第 4 章至第 9 章和附录由陶伟翻译。在成书过程中,吕庆中、程勇、胡斌、李晔参与了部分工作。麦中凡教授对全书做了细致的审校。

最后,我们感谢北航出版社在本书的成书过程中提供的大力帮助。

我们的电子邮件地址是:

mids@buaa.edu.cn

译者于北航

2003 年 6 月

前 言

Bruce Douglass 在本书中第一次阐明如何将当代两个重要的软件工程进展——模式(pattern)和 UML(Unified Modeling Language)——成功地应用于主流实时软件传统上使用的概念和技术中。许多有关软件模式的出版物(如文献[1]中列举的)涉及实时软件本身都是浅尝辄止,要不就只讨论像实时中间件(middleware)那样面窄而又陈旧的主题(文献[2]),或者就是特定域已有的应用(文献[3])。

本书为实时计算提供了极为有益的良策,因为软件模式和 UML 可潜在地降低许多系统软件的费用。实时软件种类繁多,几乎占据复杂性和费用的整个领域。在有些实时系统中,软件既小又简单,而硬件相当复杂且费用巨大,以至于软件费用只占系统整个费用很小的一部分(如激光陀螺仪中的软件)。而在另一些实时系统中,软件大,而且复杂到硬件费用可以忽略不计的地步,软件费用是整个系统费用的绝大部分(如军用或民用飞机中的软件)。Barry Boehm 在其新版的《无所不在的软件费用模型 COCOMO》(文献[4])一书中,为后一类软件的生命周期所有阶段都乘以 1.71 的系数(一般最大是 1)。这是根据项目环境条件和"标准的"软件比较得来的,这个系数使得软件预估更容易。多数实时系统是处于这两种极端情况之间,故多数主流实践者都将从本书中获益。

历史上,实时软件的开发者在使用最流行的软件工程方法学上,总要比其他开发者晚上一步,其原因在于:

一是如上所述,有些实时软件非常简单,只需要最基本的方法学也就够了。

较为普遍的原因是,许多还不算小的实时系统受制于硬件容量约束(由于大小、重量、功率等等)。为了可复用性、模块性、灵活性这类目的,软件结

构总是趋向于多消费时空资源。好在日常常用的系统硬件费用一直在下降而性能总在提高，多少有些补偿。此外，在许多实时系统中硬件费用仍然是比较容易量化的因素，自然就比难于量化的软件质量和费用有价值得多。

另一个原因是实时软件的实践者往往都是应用领域的专家，对于如何理解现代软件工程，以及如何正确地利用它这方面的教育接触不多。计算机科学与工程的新进展也很少进入实时领域，因为他们的受正规教育并非立足于真正大型的实时实践（实时也是计算机科学与工程惟一不足的方面）。他们学到很少一点实时理论，也都仅限于与实际工作密切相关的方面。

本书是有关软件模式和 UML 的导论，作者是本领域最有权威的撰稿人之一，旨在使本领域的实践者无需预备知识就能理解软件模式和 UML，并把它们用于主流的实时软件。只要适度地投入学习本书的材料，他们就会发现怎样才能把他们以往难以掌握的专业经验铸入一个框架中，使得他们的实时软件设计更有预断性，不仅能按时限作预断（时限的可预断性本身就是实时计算存在的理由），也能按生命周期费用预断。

实时软件设计者掌握了软件模式和 UML 还有一个好处是：在构建更大型、更复杂、更动态、分布更广的实时计算系统时，上层问题的重要性急骤增长。这种系统可为许多企业提供非常重要的（也许不一定作得出完整的评价）附加值。并因此能在实时计算系统领域接受更大的挑战，并得到经历这种开发机遇的回报。本书是这种前景的极好起点。

E. Douglass Jensen
Natick, Massachusetts
July 2002

Douglass Jensen 是知名度很高的实时计算系统的领头人之一，尤其是动态分布式实时计算系统。他因领导世界上第一个可部署的、分布式实时计算机控制系统产品的研究而获此殊荣。他做军用和民用工业的实时计算已有 30 多年，在硬件、软件、系统研究和技术方面有丰富的经验，并在卡内基·梅隆大学计算机科学系任教 8 年，现在是 MITRE 公司资源的技术领导，正在为国家战略需要的实时计算系统项目进行研究和技术培训。Douglass Jensen 的 Web 地址是：

http://www.real-time.org

参考文献

1 Boehm, Barry, Ellis Horowitz, Ray Madachy, Donald Reifer, Bradford Clark, Bert Steece, A. Winsor Brown, Sunita Chualani, and Chris Abts. Software Cost Estimation with Cocomo II. Upper Saddle River, NJ: Prentice Hall, January 2000.

2 Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley, 1995.

3 Lea, Doug. Design Patterns for Avionics Control Systems, http://st-www.cs.uiuc.edu/users/patterns/patterns.html, 1994.

4 OOPSLA 2001, Workshop on Patterns in Distributed Real-Time and Embedded Systems, ACM, October 2001.

序

目 标

实时嵌入式系统——RTE 系统(Real-Time Embedded System)运行时,比"传统"的计算机系统,如桌面和单主机,有更多的环境约束。RTE 系统必须是高效的,对它们有限的处理器和存储资源的使用进行优化,通常比有较大计算能力的系统性能要好。此外,许多 RTE 系统往往有安全紧要(safety-critical)和高可靠性(high-reliability)的重要需求,因为它们常用于航空飞行控制、核动力厂、生命支持和医疗仪器这类系统。开发 RTE 系统需满足这类特殊系统的功能和服务的质量需求,往往要求开发者有 10 年以上的丰富经验。这些开发者成年累月地一再遇到相同的问题(严格地说,不是同一问题,但有共同的线索)。一些非常优秀的开发者抽象了这些问题,并把它们的解决方案归结为泛化方法(generalized approach),并且被证实始终是有效的。这些泛化的方法被称之为设计模式(design pattern)。设计模式通常在系统层或软件体系结构上应用最适宜,而体系结构是设计决策的总和,它通常会影响到系统的基本组织。实时系统设计模式试图捕获某个地方用于开发 RTE 系统的一组体系结构(architecture)的设计模式。

读者对象

本书面向专业软件的实际开发者和计算机科学的初级或高级从业者。本书也可作为本科生或研究生的教程,但重点是实际开发而不是理论论述。本书假设读者至少精通一门编程语言,对面向对象、统一建模语言(UML)和实时系统的基本概念有初步的了解。

本书组织

第1篇由3章组成。第1章是对统一建模语言主要概念的一个非常简单的概览。第2章按照“嵌入式系统的快速面向对象过程ROPES(Rapid Object-oriented Process for Embedded Systems)”中定义的体系结构，介绍它的基本概念，包括将体系结构划分成的两个主要方面：逻辑的(设计时)和物理的(运行时)，以及体系结构的五个主要视图。第3章讨论设计模式和它们在定义体系结构时所充任的角色。由于在无过程的环境中难于讨论体系结构，引入ROPES过程和用以做优化的某些关键技术，以此作为背景，就可以在其中有的放矢地讨论设计模式了。一旦介绍完过程，下文就是设计模式。首先解释了它们的各个方面，接着是本书用以讲述设计模式的基本组织方法。最后讨论如何将设计模式应用于真实系统的开发，作为本章的结束。

第2篇中的模式围绕它们设计的体系结构概念组织。第4章致力于讨论高层结构的模式，重点围绕称之为子系统或构件的体系结构。由于并发性和资源管理对于实时嵌入式系统至关重要，第5章重点是一般的并发性模式。存储管理在本领域的许多系统中也十分重要，它就成为了第6章的主题。第7章可以看到更为通用的资源管理模式。第8章介绍了一些常见的分布式体系结构的模式，定义了对象如何跨多个地址空间和计算机分布。最后，第9章介绍了一些处理构建安全和可靠性的体系结构的模式。

本书末尾给出两个附录。第一个只是UML图形表式法的总结，第二个是模式的按名索引。

文献那一章，包含了各种论题的一些文章以及某些有用的OMG的规范。

其他信息

有关UML、面向对象技术和实时系统开发的其他信息可以从www.ilogix.com网址中找到。此外，有关UML、MDA和CORBA标准的当前进展可查www.omg.org。有关UML如何用于实时系统的更多信息，可用Addison-Wesley公司的修订版*Real-Time UML, 2nd Edition*，以及更容易理解的*Doing Hard Time: Developing Real-Time Systems with UML*，

Objects, Frameworks and Patterns。也可以参考其他一些写得很好的有关 UML 和软件工程的书。

致 谢

像本书这类书籍往往不只是由撰稿人直接写出，而是大家的努力。如 Addison-Wesley 专业编辑人员（我特别要感谢我的编辑 Paul Becker，他总是友善地催促我完成此书！），还有许多其他的人以他们自己的方式参与营造我们共同的酒吧。在 UML 工作的核心组成员 Cris Kobryn、Eran Gery、Jim Rumbaugh、Bran Selic，还有许多其他人，在为制定捕捉和操纵系统模型有用的标准语言方面，都是我必须向他们致谢的。此外，值得推崇的还有 Erich Gamma、Richard Helm、Ralph Johnson、John Vlissides 等人的优秀著作——*Design Patterns: Elements of Reusable Object-Oriented Software*。该书把设计模式的概念引入到日常使用。David Harel（状态图的发明人，状态图成为 UML 中所有行为的语义基础）和 Werner Damn 连续地对新锐技术做出了重大贡献，特别是在 UML 建模实时系统的形式验证方面。

还有我的两个儿子：Scott 和 Blake，他俩一直乖巧得令我吃惊，使我能有一个平常的心态完成这项值得一做的工作。

目 录