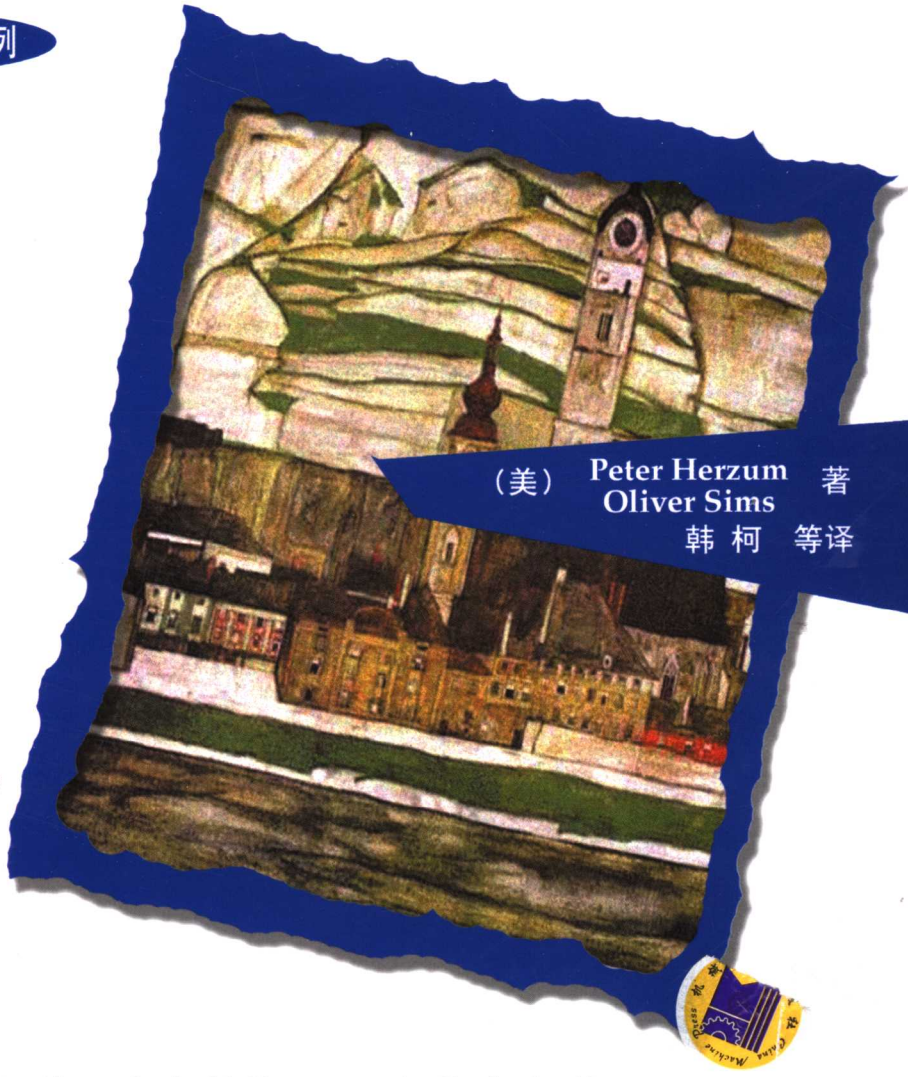
软件工程技术丛书

HZ BOOKS

基于组件的企业级开发

软件复用与构件技术系列



(美) Peter Herzum
Oliver Sims
著
韩柯 等译

Business Component Factory
A Comprehensive Overview of Component-Based Development for the Enterprise

机械工业出版社
China Machine Press
中信出版社
CITIC PUBLISHING HOUSE

软 件 工 程 技 术 丛 书

基于组件的企业级开发

软件复用与构件技术系列

(美) Peter Herzum
Oliver Sims 著
韩 柯 等译

Business Component Factory
A Comprehensive Overview of Component-Based Development for the Enterprise

机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE

本书提出了一种基于组件的完整策略，即“业务组件方法”。这种策略运用了组件思想，并将其扩展到软件系统开发、部署、运行和进化的所有方面，提出了应对基于组件开发所面临的现实挑战所需的概念，重点讨论了频繁发生变化的大规模分布式企业系统。

本书思路清晰、结构严谨，对项目经理、系统分析员、软件设计人员等有很大帮助。对于高等院校的相关专业学生来说，本书也是拓展视野的良好参考书。

Peter Herzum, Oliver Sims: Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise (ISBN: 0-471-32760-3).

Authorized translation from the English language edition published by John Wiley & Sons, Inc.

本书版权登记号：图字：01-2002-4241

图书在版编目（CIP）数据

基于组件的企业级开发 /（美）赫尔祖姆（Herzum, P.），（美）西姆斯（Sims, O.）著；韩柯等译. -北京：机械工业出版社，2005.8

（软件工程技术丛书 软件复用与构件技术系列）

书名原文：Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise

ISBN 7-111-16846-1

Ⅰ.基… Ⅱ.①赫… ②西… ③韩… Ⅲ.软件开发 Ⅳ.TP311.52

中国版本图书馆CIP数据核字（2005）第074870号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：姚 蕾 赵 俐

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2005年8月第1版第1次印刷

787mm × 1092mm 1/16 · 23印张

印数：0 001 -4 000册

定价：48.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译者序

大型软件系统的快速、经济和高质量的实现，一直是软件界努力战胜的一种挑战。其中一个主要问题，就是如何有效解决随软件规模一起快速增长的复杂性。从这个意义上说，高级程序设计语言、数据库管理系统、模块化、面向对象、代码自动生成、分布式系统、软件重用，都可以看作是这种努力的一部分。

最近十几年，组件化软件系统开发被认为是解决大规模软件系统开发复杂性问题的一种有效途径。本书作者认为目前软件业的业务推动因素、技术、开发过程、标准和日趋成熟的体系结构，已经构成建立行业大规模软件生产的基础，认为具有成本效益的软件生产不只是一个方法论或开发过程问题（方法论和开发过程只是整个问题的一部分），而是要求软件开发的各个方面要用一组协同和集成的概念、体系结构、方法论等来处理。本书把这组集成要素叫做软件开发方法，提出一种基于组件的完备方法，即“业务组件方法”。本书运用了组件思想，并将其扩展到软件系统开发、部署、运行和进化的所有方面，重点讨论了频繁发生变化的大规模分布式企业系统。

本书思路清晰，结构严谨，视点很高，相信会对项目经理、系统分析员、软件设计等人员有很大帮助。对于在校高年级学生来说，本书也是拓展视野的相当不错的参考书。

在翻译过程中，我们力求忠实原文。但由于译者的知识水平和实际工作经验有限，不当之处在所难免，恳请读者批评指正。参加本书翻译工作的还有：黄慧菊、屈健、刘芙蓉、王威、李津津、原小玲、韩文臣、耿民、付程、孟海军、杜旭涛、杜蔚轩、李宗泽等。

韩 柯

2005年6月

作者介绍

Peter Herzum是国际公认的大型分布式系统基于组件开发的先驱，作为顾问、方法论学者、首席架构师和高级经理，他参加过很多应用组件和对象技术的大规模开发项目。从1992年起，在积累了丰富的面向对象方面的经验之后，Peter开始探索企业级组件开发所面临的技术、体系结构、方法论和组织方面的问题，后来提出了“业务组件方法”。他还在开发具有成本效益的高质量软件的各个方面，培训和指导过世界各地的很多机构。Peter经常在世界各地的会议上发表演讲，是OMG一位很活跃的成员。他还是Vayda & Herzum公司（www.vaydaherzum.com）的首席技术官和创始人之一。

Oliver Sims是Genesis开发集团公司的企业组件计划部主管，被人们广泛赞誉为分布式业务对象和组件系统设计与实现的领袖。他曾经多年担任高级企业软件首席架构师，在这之前是大型业务系统的系统顾问。Oliver是OMG体系结构委员会的一名成员，自1994年以来一直积极参加OMG业务对象领域任务小组的工作。他撰写了《Business Objects》一书。

前　言

具有成本效益的软件制造已经不再是梦想，已经有一些机构有能力建立自己的软件开发体系，以可重复的方式，采用基于组件的方法，生产具有成本效益的高质量软件。今天，业务推动因素、技术、开发过程、标准和日趋成熟的体系结构，这些因素共同使软件业长期以来一直为之奋斗的工业化软件生产梦想，终于有可能变成现实。

本书试图提炼并解释应用组件进行大规模软件开发的原理、概念和复杂性度量实践。现在人们已经清楚地认识到，虽然技术是必不可少的推动因素，但是具有成本效益的软件制造并不只是一个技术问题。具有成本效益的软件制造也不只是一个方法论或开发过程问题，方法论和开发过程只是这一难题的一部分。具有成本效益的软件制造其实要求软件开发的各个方面要用一组协同和集成的概念、体系结构、方法论等来处理。我们把这组集成要素叫做软件开发“方法”。本书将提出一种基于组件的完备方法，即“业务组件方法”，这种策略运用了组件思想，并将其扩展到开发、部署、运行和进化的所有方面，特别关注经常发生变化的大规模分布式企业系统。

我们认为，业务组件方法是第一个在整个开发生命周期始终关注组件的方法和整套体系结构观点。这种方法在很大程度上基于一种我们叫做“业务组件”的新的、整体的、也许还是革命性的核心概念，这种概念关注的是分布式现实、依赖性降低、自治开发和在单一多方面软件结构中部署等问题。这种方法提供了一种概念框架，把组件思想带入可伸缩系统领域，澄清了不同的组件粒度。这种方法还提供了一种超越当前面向对象的方法论，提出了应对企业基于组件的开发所面临的现实挑战所需的概念和理念。

本书将运用业务组件方法，描述如何绘制“业务组件工厂”蓝图，即为满足企业需要，高质量、高速度、灵活地开发软件的能力。此外，本书还提供大量指南、实用技巧和体系结构模式，可帮助读者理解并缩短开发周期，有助于全面审视今天已有的各种组件技术和开发方法。

为什么要阅读本书

本书有什么不同之处？首先，**本书全篇讨论的都是组件**，讨论的是构建组件、运用组件合成系统和系统联邦所需的概念框架、开发过程、体系结构和实用技巧。其次，本书讨论的是构建大规模的业务系统，不是单独的用户界面或嵌入式系统或实时系统。本书关注的是企业业务系统。第三，本书以体系结构为核心。一度曾经有人认为，实际问题的全部就是好的方法论和过程。但是今天，如果必须在好的方法论和好的体系

结构之间进行选择，我们肯定选择的是好的体系结构。当然，没有一种好的体系结构也能够建立一种可重复的过程，只不过我们反复构建的是很差的系统。

本书不是一本市场开发用书。构建大型分布式系统是很复杂的，也许比应该具有的复杂程度还要复杂。而组件技术仍然是前沿技术。我们不打算说教：基于组件的开发会解决所有的开发问题。相反，我们会指出接受与使用组件的挑战、风险和成本。但是我们也会提出应对挑战、降低成本和风险的建议。本书给出的很多概念和指南都是很多分布式系统第一线工作人员的黄金规则和技巧。

本书也不是关于面向对象开发的专著。我们认为面向对象的开发只是构建基于组件系统所需的很多要素中的一个。但是我们将讨论面向对象对于构建组件的重要性，以及面向对象原理应该怎样运用于基于组件方法背景下的整个开发周期中。

最后，本书关注的不是具体技术。相反，本书描述的是任何大型基于组件的业务系统开发所需的原则，探索的是如何考虑这种开发以在快速变化的技术情况下保护投资。根据我们的经验，我们认为本书提出的概念可以在当前市场上的所有主要组件技术上实现。

本书针对的读者

首先，本书是针对参与、学习或对运用组件思想实现业务软件开发感兴趣，以及对支持未来软件生产率革命所需的下层体系结构、过程、设计技术和模式感兴趣的任何读者。但是本书对于那些对经济高效地生产软件和构建今天的业务解决方案感兴趣的人也很重要。

本书主要针对以下这些读者：

- 将要进行基于组件的开发，以及想了解基于组件开发的前景的读者，或当前正在分析大量采用组件实现的领域的读者。虽然不能从一本书中学到所需的一切，但是却可以采用一本书指导学习。如果读者想构建业务组件系统，那么本书将有助于组织思路，并使读者的直观感觉得到证实。
- 软件分析师、架构师、设计人员和软件工程师，将通过本书找出适用于从需求到构建，再到运营的全过程的概念框架、体系结构原则、设计模式和方法论建议。读者会发现业务组件方法不是一种要么全有要么全无的方法，本书的不同部分可以按需运用，即使只是部分地采用业务组件方法，也会使成本显著降低。
- 软件工程师，本书将讨论大家感兴趣的有关业务组件虚拟机方面的问题，将自顶向下地归纳功能开发人员所需的软件技术透明性。
- 需要宏观地了解组件如何端到端地使用的管理者、非技术人员和顾问。具体地说，这类读者应该阅读第1章“基于组件的开发”和第2章“业务组件方法”。
- 对理解完全采用组件思想构建大型分布式系统感兴趣的计算机科学系大学生和研究生会通过本书找到很有用的概念、观点和模式，这些读者会发现本书提出的概念独立于任何具体的可以购买到的组件中间件。
- 工具和中间件提供商，这些读者可以把本书看作是对其工具或中间件未来版本的需求定义的一个来源。

总之，我们希望本书对所有参与或对基于组件的开发感兴趣的读者提供帮助。

本书的组织

本书假设读者对软件工程的主要原理和面向对象方法的主要概念都已经有了一定的了解。本书分为3个部分：概念框架、建立组件工厂和制造基于组件的软件。

第一部分：组件概念。第1章到第6章将描述达到高效制造快速演化的基于组件的分布式系统目标所使用的概念。这一部分根据各种组件粒度从小到大地组织，并提供组装系统和组件联邦所需的概念。

第二部分：建立组件工厂。第7章到第10章将介绍如何运用第1部分所描述的概念建立组件工厂本身，即形成高效生产各种粒度组件的能力。这一部分以第2章给出的体系结构视点为线索组织。

第三部分：制造基于组件的软件。第11章到第13章将介绍基于组件系统的主要建模和设计考虑，在最后一章将简要讨论迁移到基于组件开发的一些主要考虑。

最后，本书还给出两个附录，第一个附录是本书所使用的命名约定，第二个附录给出的是本书所使用的术语解释。全书中有关更详细的技术讨论和作者给出的很多选择，都放在方框中。

本书的写作过程

本书的两位作者在20世纪80年代各自独立地参与了采用面向对象技术开发业务系统。面向对象的方法初看起来功能很强，但是经过一段时间之后，人们发现还遗漏了一些东西。生产率方面的预期效果、高水平的重用和掌握复杂开发的能力，都还没有真正实现。

在研究更高效的系统开发方法的过程中，Peter与很多很出色的同事一起，从1992年开始探索大规模基于组件开发的组件和方法论。这种探索包括也许是第一个组件工厂的体系结构确定、设计和开发，包括自己开发的大型基于组件系统的技术基础设施，以及可以根据这种开发的需要进行剪裁的方法论。他们还探索了运用这种方法论和技术基础设施开发和部署大规模系统。他在1994年第一次定义了业务组件概念和业务组件方法的第一个版本。1995年，他开始为撰写以后的专著搜集了很多概念。

Oliver第一次接触对象技术并不是通过语言，而是通过试图使大量早期以个人计算机为前端的分布式系统应用程序员的工作得以简化。实际上，他第一次运用面向对象思维还是在20世纪80年代后期创建真正的可插入个人计算机组件的时候。Oliver从20世纪70年代中期就开始参与分布式系统的开发工作，他把下一个考虑目标定为把分布式系统作为一个整体，最终提出了4层结构和分布式领域开发的概念。在这之后他参加了一段时间的技术开发，生产出可以在多种平台上运行的商品化“业务对象”组件基础设施，既支持面向对象的语言，也支持过程语言，使大粒度业务对象也具有可接插二进制组件的能力。

1996年，“系统软件协会”（总部设在美国芝加哥的ERP软件生产商）启动了一个本书的两位作者都参与了的计划，旨在为其标志性产品BPCS提供一种真正革命化的软件组件能力。这个计划通过Riz Shakir的支持和规划，在Wojtek Kosaczynski的领导下，组成一支世界范围的团队，开始构建不亚于一个业务组件工厂的工作。在这方面得到的经验，结合这个伟大团队在这个计划上的贡献，使本书作者得以细化、提高和改进业务组件方法，最终达到了现在的程度。

1998年初，我们决定撰写一本以组件工厂及其关键概念为核心的专著。这本书就是我们的工作成果。本书不仅表达了我们对未来组件革命的信念，我们还希望它还能以某种方式对这场革命作出贡献。

致谢

在编写本书和形成本书介绍的概念的过程中，我们首先要感谢的是我们的妻子和家人。当我们在阳光明媚的周末把自己奉献给计算机键盘的时候，当我们本来应该与我们的妻子呆在一起，或与孩子们一起玩耍，或参加其他家庭活动的时候，他们一直坚定地支持我们。感谢你们，Simona和Heather，感谢你们，Solveig、Viveka、David和Richard。

我们的很多同事和朋友对于本书介绍的概念作出了重要贡献，我们非常感谢他们。他们中的一些人可能没有意识到对我们的帮助有多么大，因为本书给出的这种综合方法是通过在很多情况下的大量讨论总结出来的，我们后来对很有见地的观点进行了重新归纳，并对问题的解决方案提出了自己的观点。但是，至于本书可能存在的错误，我们自己要负全责。

对于对本书作出贡献的很多人，也许只提出一部分人的名字有失公允。但是，对于那些对实现高级软件，对本书描述的概念作出很大贡献的人，我们要特别感谢：Peter Eeles、Neil Howe、Don Kavanagh、Wojtek Kozaczynski、Phil Longden、Boris Lublinsky、Riz Shakir、Steve Terepin、Dan Tkach和Rick Williams，感谢他们对本书初稿提出的很多很有见地的意见，我们还要特别感谢Robert Mickley和Jeff Sutherland。

此外，对于进行过直接或间接贡献和指导的人，Peter要感谢Jean-Claude Bourut、Rich Burns、Ludovic Champenois、Stefano Crespi-Reghizzi、Alberto Dapra、Carlo Ghezzi、Charles Haley、Hans Gyllstrom、Francesco Maderna、Radouane Oudhriri、Paul-Eric Stern、Ed Stover和Tom Vayda。对于在OMG和其他地方提出的很多观点，Oliver要感谢Martin Anderson、Cory Casanave、Fred Cummins、Dave Frankel、Steve McConnell、Kevin Tyson和Dave Zenie。

Oliver Sims的个人说明

当我第一次遇到Peter Herzum时，我有关分布式组件的思考已经相当成熟（那时我把分布式组件叫做“协同业务组件”）。我的同事和我都清楚地理解分布式组件是实现业务概念的可插入组件，在不同的分布层次上有不同类型的分布式组件，这4个层次本身已经被定义并经过测试。那时，我们还有很多生产能够为功能程序员提供所需抽象级别的中间件经验，支持和构建过不同语言（既有面向对象的语言也有过程语言，既有编译语言也有解释语言）的分布式组件，我们也见证过我们的技术在实际系统中得到验证。

虽然我知道除了技术之外还需要其他东西——很多其他东西，不管是我还是我所遇到的其他任何人，对目标是将业务概念同构地映射到可独立部署的大粒度软件组件的可伸缩开发所需的方法论、开发过程和项目管理技术，都还没有很好的理解。Peter在这些方面作出了重要贡献，但是他还提出了另外两个方面的问题，也许是更重要的问题。首先，虽然我也知道现实世界业务概念和在运行时执行的组件之间一定有某种联系，当时的思考是不完备的，特别是在开发过程方面的思考。缺失的这种联系很重要。Peter找到了这种缺失联系的答案，即不仅涵盖物理分布、体系结构视点和开发生命周期要素，而且还提供项目管理方法的单一一致的概念，同时，还要强化需求时的单一业务概念到部署时的单一独立可交付产品之间的映射。这种概念，即业务组件，是本书的核心。业务组件方法对于开发大型和小型的分布式业务系统，都是全新的方法。第二，正是Peter运用业务组件的能力，把软件生产提高到工业化生产的水平上。

因此，不仅在设想核心业务组件概念方面，而且在扩展业务组件概念成为一种完整、一致的通过业务组件工厂在软件生产率上取得革命性提高方面，本书都对Peter突破性的工作做了显著扩展。同时，业务组件方法还指出了一条未来商品化软件组件的发展道路，使最终用户能够在感兴趣的粒度层次上开发商品化软件组件。在与Peter合作完成本书过程中，我希望能够对整体概念有所贡献。但是，当业务组件工厂变得很常见，软件开发机构成为习惯地支持快速业务演变和改革，而不是在至关重要的道路上徘徊不前时，我相信这样的未来很大一部分要归功于Peter的贡献。

补充信息

我们试图全面论述企业基于组件开发一些最重要方面的问题。虽然这种论述肯定需要进化和改进，但是我们认为业务组件方法已经相对比较成熟，我们希望业务组件方法能够帮助读者和读者所在机构快速转向更经济高效的软件制造。在准备编写本书的过程中，很多方面的问题还没有考虑，包括一些技术细节。我们已经建立了一个网站，即www.ComponentFactory.org，计划在这个网站上提供有关业务组件概念和实践方面的进一步信息，提供到其他有用网站的链接，提供相关会议和出版物的信息。

目　录

译者序
作者介绍
前言

第一部分　组件概念

第1章　基于组件的开发 ……2
1.1　组件是什么 ……2
1.2　什么是基于组件的开发 ……5
1.2.1　简要历史 ……6
1.2.2　软件制造的要求 ……10
1.2.3　收益 ……11
1.3　演化 ……13
1.3.1　今天的软件制造 ……13
1.3.2　迁移过程中的风险 ……16
1.3.3　基于组件的开发的成熟度 ……16
1.4　小结 ……19
1.5　注释 ……19
第2章　业务组件方法 ……20
2.1　组件粒度的层次 ……20
2.1.1　分布式组件 ……22
2.1.2　业务组件 ……23
2.1.3　业务组件系统 ……24
2.1.4　系统级组件联邦 ……27
2.2　体系结构视点 ……28
2.3　开发过程 ……29
2.3.1　主要特征 ……29
2.3.2　阶段 ……30
2.4　业务组件方法 ……31
2.4.1　一种统一概念 ……32
2.4.2　开发成本 ……33
2.4.3　最低依赖 ……34
2.4.4　5种要素 ……34
2.5　业务组件工厂 ……35
2.6　场景 ……37
2.7　业务组件方法的适用性 ……39
2.7.1　基于OLTP的应用程序 ……40
2.7.2　批处理应用程序 ……40
2.7.3　基于Web和电子商务应用程序 ……41
2.7.4　基于个人计算机的应用程序 ……41
2.8　小结 ……41
2.9　注释 ……43
第3章　分布式组件 ……44
3.1　概念 ……44
3.1.1　概述 ……45
3.1.2　特征 ……46
3.1.3　分类 ……52
3.2　内部要素 ……53
3.2.1　功能开发人员的程序设计模型 ……54
3.2.2　隔离分层 ……56
3.2.3　语言类分类 ……58
3.3　外部要素 ……63
3.3.1　接口 ……64
3.3.2　业务数据类型 ……64
3.4　小结 ……66
3.5　注释 ……66
第4章　业务组件 ……68
4.1　概念 ……68
4.1.1　定义概念 ……68
4.1.2　派生概念 ……70
4.1.3　实现考虑 ……71
4.2　内部要素 ……72

4.2.1 分布层 ……72
4.2.2 分布层的特性 ……75
4.2.3 分布域 ……78
4.3 外部要素 ……79
4.3.1 接口 ……79
4.3.2 依赖关系 ……84
4.3.3 插座 ……85
4.4 开发生命周期 ……86
4.5 地址簿场景 ……89
4.5.1 用户界面 ……90
4.5.2 企业分布式组件对企业分布式组件的调用 ……92
4.6 相关软件工程概念 ……93
4.6.1 业务对象 ……93
4.6.2 模块 ……94
4.6.3 UML包 ……94
4.7 小结 ……95
4.8 注释 ……96
第5章 业务组件系统 ……97
5.1 概念 ……97
5.1.1 例子 ……97
5.1.2 业务组件组装 ……98
5.1.3 产品组装 ……100
5.1.4 业务数据类型系统 ……100
5.1.5 组件模型与组件图 ……101
5.1.6 特征 ……103
5.2 内部要素 ……106
5.2.1 业务组件分类 ……107
5.2.2 实用业务组件 ……109
5.2.3 实体业务组件 ……109
5.2.4 过程业务组件 ……110
5.2.5 辅助业务组件 ……111
5.3 外部要素 ……111
5.3.1 接口 ……112
5.3.2 系统级组件 ……114
5.4 信息系统 ……115
5.4.1 电子商务 ……115
5.4.2 报表编写器 ……117
5.5 小结 ……118
5.6 注释 ……119
第6章 系统级组件联邦 ……120
6.1 业务问题 ……120
6.2 互操作性概念 ……121
6.2.1 互操作参考模型 ……121
6.2.2 与体系结构视点的关系 ……128
6.2.3 交互模式 ……129
6.3 联邦的概念 ……132
6.3.1 联邦的特征 ……132
6.3.2 标记数据与XML ……136
6.3.3 剖析协议模型 ……141
6.4 经过体系结构设计的联邦 ……144
6.4.1 特征 ……144
6.4.2 设计联邦的体系结构 ……146
6.5 小结 ……148
6.6 注释 ……148

第二部分 建立组件工厂

第7章 开发过程 ……151
7.1 概念 ……151
7.1.1 制造过程 ……152
7.1.2 10个黄金特征 ……153
7.2 构建 ……155
7.2.1 需求 ……157
7.2.2 分析 ……160
7.2.3 设计 ……163
7.2.4 实现 ……166
7.3 确认与验证 ……167
7.3.1 评审 ……168
7.3.2 测试 ……170
7.4 迭代 ……173
7.5 小结 ……175
7.6 注释 ……176
第8章 技术体系结构 ……177
8.1 概念 ……178
8.1.1 业务组件虚拟机 ……178
8.1.2 可移植性 ……180

8.1.3 实现隔离分层 ……182
8.2 技术核心 ……183
8.2.1 组件调用 ……183
8.2.2 组件生命周期 ……184
8.2.3 并发性 ……185
8.2.4 异步消息传递 ……186
8.2.5 动态继承 ……187
8.3 服务与设施 ……188
8.3.1 事务 ……189
8.3.2 错误处理 ……191
8.3.3 事件 ……192
8.3.4 持久性 ……194
8.3.5 兆数据 ……196
8.3.6 用户界面框架 ……200
8.4 扩展集成开发环境 ……202
8.4.1 组件规格说明工具 ……204
8.4.2 基于库的开发 ……204
8.5 小结 ……205
8.6 注释 ……206
第9章 应用体系结构 ……207
9.1 体系结构原则 ……209
9.1.1 非循环 ……209
9.1.2 体系结构规范化 ……211
9.1.3 其他原则 ……211
9.2 体系结构风格 ……212
9.2.1 基于类型的风格与基于实例的风格 ……213
9.2.2 基于事件 ……221
9.2.3 体系结构模式 ……222
9.3 协同模式 ……224
9.3.1 业务事务 ……225
9.3.2 默认管理 ……227
9.3.3 替代与历史数据 ……228
9.3.4 确认 ……229
9.4 从业务组件虚拟机到功能开发 ……229
9.4.1 错误处理 ……231
9.4.2 数据类型系统 ……233
9.4.3 标准与方针 ……234
9.5 小结 ……234
9.6 注释 ……235
第10章 项目管理体系结构 ……236
10.1 概念 ……237
10.2 软件配置管理与版本控制 ……239
10.2.1 术语 ……239
10.2.2 开发过程的软件配置管理视图 ……243
10.2.3 集成软件配置管理策略 ……244
10.3 依赖管理 ……246
10.3.1 依赖模型 ……246
10.3.2 输出 ……249
10.3.3 输入 ……250
10.3.4 动态依赖 ……251
10.3.5 举例 ……252
10.4 完善开发环境 ……253
10.4.1 目录树结构 ……253
10.4.2 组件依赖管理器 ……257
10.4.3 脚本 ……259
10.5 项目管理人员的模型 ……260
10.5.1 开发机构 ……260
10.5.2 项目管理视图 ……262
10.5.3 软件生态学 ……262
10.6 小结 ……262
10.7 注释 ……263

第三部分 制造基于组件的软件

第11章 基于组件的业务建模 ……266
11.1 概念 ……266
11.1.1 业务建模者与功能架构师 ……266
11.1.2 主要建模构件 ……267
11.1.3 联邦建模 ……269
11.1.4 划分业务空间 ……271
11.1.5 功能子类 ……274
11.2 标识策略 ……276
11.2.1 粒度 ……276
11.2.2 标识 ……276
11.2.3 业务组件系统模型的评价 ……278
11.3 过程建模 ……279

11.3.1 工作流管理 ……279
11.3.2 基于规则的建模 ……282
11.3.3 快速进化 ……282
11.3.4 作为有限状态机的组件 ……284
11.4 实体建模 ……286
11.4.1 业务组件的内部协同 ……286
11.4.2 贸易伙伴 ……288
11.4.3 合同 ……290
11.4.4 价格与合同项 ……291
11.5 实用业务组件 ……292
11.5.1 地址簿 ……292
11.5.2 邮政编码簿 ……293
11.6 小结 ……294
11.7 注释 ……295
第12章 基于组件的设计 ……296
12.1 大型基于组件系统的思想 ……296
12.1.1 分布式系统现实 ……296
12.1.2 组件粒度 ……297
12.1.3 自治性 ……298
12.1.4 可伸缩性优先级 ……299
12.2 接口 ……299
12.2.1 特征 ……299
12.2.2 组件标准化 ……300
12.2.3 业务数据类型 ……304
12.2.4 带标记的数据 ……306
12.3 业务组件 ……308
12.3.1 用户工作间域与企业资源域 ……308
12.3.2 用户界面 ……311
12.3.3 工作间层 ……317
12.3.4 企业层 ……319
12.4 持久性 ……320
12.4.1 数据库的组件化 ……320
12.4.2 持久性框架 ……324
12.4.3 适合关系模型的面向对象 ……326
12.4.4 管理数据完整性 ……327
12.5 小结 ……328
12.6 注释 ……329
第13章 迁移 ……330
13.1 概念 ……330
13.1.1 方法论 ……331
13.1.2 最佳实践 ……332
13.1.3 体系结构 ……333
13.1.4 软件工厂 ……333
13.1.5 重用程序 ……334
13.2 迁移程序 ……337
13.2.1 特性 ……338
13.2.2 知识转移 ……338
13.2.3 迭代 ……340
13.3 小结 ……341
13.4 注释 ……342
附录A 命名约定 ……343
附录B 术语表 ……345
参考文献 ……349

第一部分

组 件 概 念

第一部分将提出“业务组件方法”的概念框架。业务组件方法是业务组件工厂的总体构思和理念，其基础是一种特定的构件概念，即业务组件。这种概念框架由开发行业水平和企业级软件组件所需的主要概念和视点构成。这些概念并不取代当前的分布式系统思想，而是在这种思想之上补充一种增加效果的概念框架，即一种新的构思。

这一部分的前两章将介绍基础知识，后面的各章将按照组件的粒度层次和相关的软件工件进行组织。这些粒度层次合在一起构成业务组件方法5维要素中的一个。这里的“要素”是我们的一个术语，读者从本书中可以看到，“要素”特指我们认为必不可少的模式和视点的一种特定术语。

第一部分的组织如下：

第1章：基于组件的开发。这一章将给出业界的通常含义和我们提出的基于组件开发的含义。当然，不首先弄清楚软件组件的概念就不可能讨论基于组件的开发。

第2章：业务组件方法。这一章将全面介绍业务组件方法的主要概念和视点，将介绍企业基于组件开发所需的主要概念维和体系结构视点，并提供开发过程的高层描述。

第3章：分布式组件。这一章将详细解释分布式组件概念，而分布式组件概念就是今天大多数第一线工作者在讨论企业软件组件时所说的软件组件概念。

第4章：业务组件。这一章将详细解释业务组件概念。这是一种支持整个开发生命周期、覆盖所有体系结构视点的强有力的软件组件概念。业务组件概念是业务组件方法的核心概念，并因此而得名。

第5章：业务组件系统。这一章将提出要被通过业务组件建模、设计和实现的业务系统的概念、考虑和原则。这一章还将介绍系统级组件概念，即被看作为单一软件组件的整个系统。

第6章：系统级组件联邦。这一章将介绍讨论系统或系统级组件联邦所需的主要概念。这里的“联邦”是指通常由不同软件生产商开发，需要一起工作的一组信息系统。

第1部分提出的概念框架通过运用一种特定体系结构风格的例子进行说明，在这种风格中，所有组件都是对象的管理者，但是没有给出概念实例。我们把这种风格叫做“基于类型”或“基于服务”的体系结构风格。其中一个例子是管理业务系统中所有销售订单的“订单”组件。通过基于类型的风格，单个订单不能通过网络直接处理。除了使例子更简短，这种风格还可以很好地映射到当前的组件技术。不仅如此，我们认为这对于那些对面向对象没有进行过深入研究的读者会感到更熟悉一些。当然，这并不是惟一可能的风格，基于类型的风格和其他体系结构风格，包括基于实例的风格，都将在以应用系统体系结构为主题的第2部分第9章中讨论和比较。

第 1 章 基于组件的开发

本章将说明为什么向基于组件的开发发展是基本趋势。到目前为止，基于组件的开发是控制迅速增长的复杂性和业务信息系统成本的最有希望的途径。基于组件的开发还是确定体系结构、设计、实现和部署可伸缩系统的最佳途径，这样的系统提供当前业务环境所需的灵活性和敏捷性。本章分为以下三节:

- 我们所说的“组件”是什么意思——组件是什么，不是什么。
- 我们所说的“基于组件的开发”是什么意思，简单地说，就是软件行业怎样演化到组件，这种方法带来的好处。
- 为什么我们认为基于组件的开发将会发展成熟。

在描述组件和基于组件开发的含义的过程中，我们还要介绍我们的目标，即我们想要向哪里发展——我们的期望和梦想。这是对下一波计算浪潮的期望，这个浪潮带来业务软件的根本转变，具有以下两个特点:

1. 作为组件的软件世界，真正意义上的易于交换、混合和匹配、在企业层次上插入并运行，各种企业级的组件可以通过因特网或软件商店买到。

2. 推动下一次工业革命。在这场革命中，软件是一种日用品，而不是在业务发展关键道路上需要花很多钱的长期不变的东西。

我们深刻地认识到，企业级软件会成为一种货架日用品，能够很容易与其他类似日用品的东西集成与搭配，可以不断变化构建和组装、宣传和销售、部署和发展。但是对于这场就要到来的革命，在人们的面前还有很长的路要走，还有一个似乎难以企及的最终目标: 大幅度降低生产良好软件的成本。

1.1 组件是什么

在软件界，“组件”这个词有很多不同的用法。例如，这个词用于实现为ActiveX或JavaBeans的用户界面组件，用于诸如数据库管理系统这样的大型基础设施对象，用于能够被重用的任何软件工件。

本书使用的“软件组件”一词常常省略“软件”这个限定词，表示经过完备定义的一个或一组接口的自包含软件。这带有明显的运行时间和部署时间内涵，也就是说，组件具有运行时间可以被访问的接口，并且在组件开发生命周期的某个时点上，组件可以被独立地交付和安装。我们还要求组件易于与其他组件合并、组合，以提供有用的功能: 一般来说，单个组件只有通过与其他组件协同才能实现自己的有用性。

组件是可以插入系统中的对象。但是，如果要插入组件，就必须有组件能够插入的对象。软件组件的挑战不仅是要考虑好如何设计和构建具有清晰接口的自包含的有用软件，而且还要保证兼容组件在部署时能够适配的软件插座。例如可以把水壶看作是一种厨房组件，不过如果水壶使

用500伏的电压和5芯插座就不是一种有用的组件。再考虑一下插入发动机组并在压缩时堵塞的火花塞。如果我们的发动机组有带螺纹的火花塞插座，没有堵塞插座，这样的火花塞就没有什么用处。当然，我可以重新建造自己的发动机组，也可以为水壶构建自己的电压调节器和适配器，但这是很不实际的。换句话说，组件要插入的技术环境与组件所提供的接口同样重要。这种环境，即插座，既包含技术基础设施，也包含为了使组件能够正常运行所要求的其他组件。

在考虑组件时，一个常常被人们忽略但又很重要的因素是组件的使用者。例如，对计算机主板制造商很有用的计算机芯片对我们就没有多大用处，因为我们不会把芯片插入个人计算机的总线上。但是借助主板我们就可以做到这一点。对于软件也是一样。给定组件在设计时就是有针对性的（但是很少有人说明这点）。有些组件就是设计给开发人员使用的，有些则是供最终用户使用的。

总之，软件组件模型具有四个基本特征：组件、组件插座、组件与其他组件的协同能力、组件的使用者。

1. 组件是自包含的软件结构，具有确定的使用，具有运行时间接口，可以被自动地部署，并且在构建时有预先确定的具体组件插座。

2. 组件插座是提供到组件将要适配的支持基础设施的完备定义和众所周知的运行时间接口的软件。设计时间接口本身是必要的，但不是充分的，因为这种接口只有当被某种软件，即基础设施实现后才会存在。

3. 构建组件是为了与其他组件合成和协同的。

4. 组件插座和对应的组件是供拥有一套技能和工具的人使用的。

作为这最后一点的必然结果，还可以进一步得到以下两点：

- 组件使用者可能没有创建该组件所需的技能或工具。这意味着存在一种组件层次结构，使得一个人的最终产品是其他某个人的组件。
- 创建组件的人一般不创建组件要适配的插座，即基础设施。

以上描述与软件界逐渐形成的观点是一致的。例如，“面向组件程序设计研讨会”ECOOP'96把软件组件定义为“一种合成单元，具有以契约形式描述的接口，并只有明确的环境依赖关系。软件组件可以被独立部署，是第三方合成的主体”。同样，Szyperski也把组件描述为“作为系统的一个可孤立的部分进行实际部署的对象”[Szyperski 1998，第10页]。当然，微软公司的COM、Enterprise Java Beans和OMG开发[OMG 1999]，总体上也都支持这种特定的组件特性观点。

但是，今天的业界仍然把组件作为一种单一的可插入软件模块来关注。虽然很有必要，但是这种技术本身还远不足以应对企业开发的很多挑战。正是在企业层次上，才会出现最复杂的软件开发。人们所需要的是，在前面所描述的组件概念基础上构建软件组件，使这些组件有助于大幅度降低大型分布式业务系统软件开发成本。这种组件就是本书要讨论的组件。一般来说，我们把这些组件叫做“企业组件”，表示用于构建整个企业系统的组件。除了前面介绍的基本特征，企业组件还有以下四个特征：

1. 企业组件要应对企业挑战。这种组件要明确地应对企业级分布式系统所固有的挑战，包括并发、安全、事务管理、数据库访问和很多其他问题。

2. 企业组件具有我们称作“网络可寻址”的接口，即可以通过网络调用，可以在系统中或甚至世界上的任何地方在运行时间寻址。这种组件要插入到经过完备定义的插座中，本意是供其他软件工件易于使用或重用，从而创建软件系统。