

软件架构领域开创性著作，资深软件架构师数十年工作经验结晶，具有里程碑意义
深刻阐述如何用架构视点和架构视图的方法定义软件架构，如何用架构视角的方法确保软件质量，以及如何用架构视点和架构视角的方法与利益相关者合作

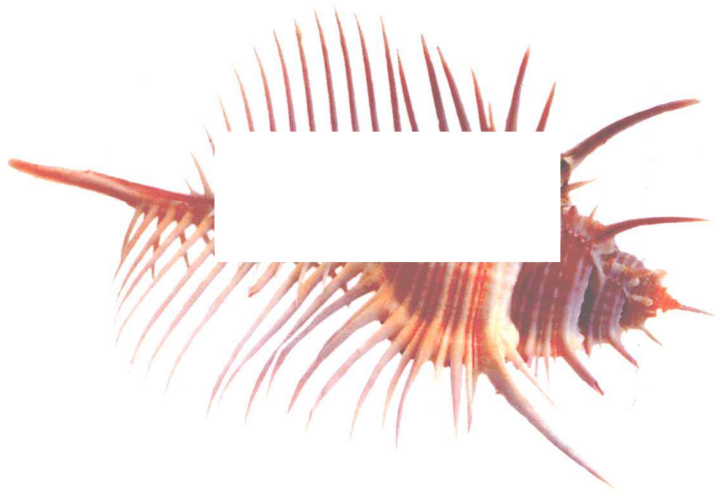
Software Systems Architecture
Working with Stakeholders Using Viewpoints and Perspectives
Second Edition

软件系统架构

使用视点和视角与利益相关者合作

(原书第2版)

(英) Nick Rozanski Eoin Woods 著
侯伯薇 译



机械工业出版社
China Machine Press

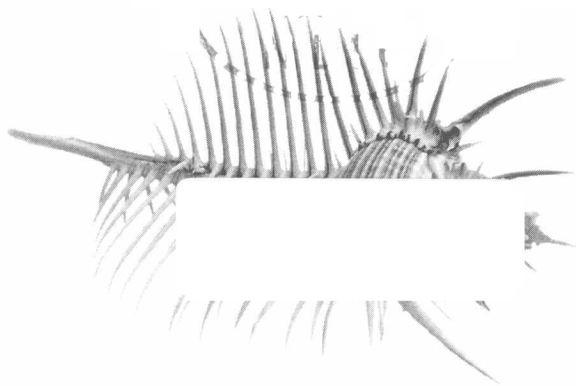
Software Systems Architecture
Working with Stakeholders Using Viewpoints and Perspectives
Second Edition

软件系统架构

使用视点和视角与利益相关者合作

(原书第2版)

(英) Nick Rozanski Eoin Woods 著
侯薇薇 译



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

软件系统架构：使用视点和视角与利益相关者合作（原书第2版） / （英）鲁然斯基（Rozanski, N.），（英）伍兹（Woods, E.）著；侯伯薇译. —北京：机械工业出版社，2013.5

（华章程序员书库）

书名原文：Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, Second Edition

ISBN 978-7-111-42186-3

I. 软… II. ①鲁… ②伍… ③侯… III. 软件开发 IV. TP311.52

中国版本图书馆CIP数据核字（2013）第077228号

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2012-1276

本书是软件系统架构领域的开创性著作，是两位拥有数十年软件行业工作经验的架构师工作经验的结晶，围绕利益相关者、视点和视角三大主题，创新性地提出了如何用架构视点和架构视图的方法来定义软件架构，如何用架构视角的方法来确保软件质量，以及如何用架构视点和架构视角的方法与利益相关者合作，具有里程碑意义。本书还展示了一种实用的、经过验证的框架，你可以应用它来处理架构定义过程，并应对创建软件架构工作所带来的挑战。

本书分为五个部分，共30章。第一部分（第1~5章）阐释利益相关者、架构描述、视点、视图和视角等基本概念，并描述软件架构师的角色；第二部分（第6~14章）描述作为架构师所要从事的重要活动，如协商项目的范围、识别并管理利益相关者、使用场景和模式、创建模型以及为架构创建文档并对其加以验证等；第三部分（第15~23章）集合了在创建架构描述时最重要的七种视点：情境、功能、信息、并发、开发、部署和运维视点；第四部分（第24~29章）集合了对于信息系统最重要的视角，包括安全性、性能和可伸缩性、可用性和适应性、演进、位置、开发资源、国际化等；第五部分（第30章）把这些概念融合在一起，并阐释了如何把这些理论应用到实践中。

Authorized translation from the English language edition, entitled SOFTWARE SYSTEMS ARCHITECTURE: WORKING WITH STAKEHOLDERS USING VIEWPOINTS AND PERSPECTIVES, 2E, 9780321718334 by ROZANSKI, NICK; WOODS, EOIN, published by Pearson Education, Inc, publishing as Addison-Wesley Professional, Copyright ©2012.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and CHINA MACHINE PRESS Copyright © 2013.

本书中文简体字版由Pearson Education（培生教育出版集团）授权机械工业出版社在中华人民共和国境内（不包括中国台湾地区和香港、澳门特别行政区）独家出版发行。未经出版者书面许可，不得以任何方式抄袭、复制或节录本书中的任何部分。

本书封底贴有Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

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：秦 健

三河市杨庄长鸣印刷装订厂印刷

2013年5月第1版第1次印刷

186mm×240mm·27.5印张

标准书号：ISBN 978-7-111-42186-3

定 价：99.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991 88361066

购书热线：（010）68326294 88379649 68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

译者序

作为一名程序员，对未来的职业规划会有很多种。有人想要走管理路线，基本目标是成为项目主管或者项目经理；有人对数据库感兴趣，期望成为 DBA；有人对测试感兴趣，期望成为测试方面的专家或者质量保证方面的高手；还有很多人希望做架构师，能够从总体上把握一个系统或者一个组织中的所有系统。

和其他角色相比，在一个系统的设计和开发过程中，架构师的地位显得尤为重要，颇有“运筹帷幄之中，决胜于千里之外”的架势。但是，很多时候，在很多组织中，架构师只不过是一个职位的概念，很多人即便在拥有这个头衔之后，还是会有很多的疑问亟待解答，比如：

- 架构师需要哪些能力？
- 架构师需要从事哪些工作？
- 想要设计出良好的架构，需要考虑哪些问题？
- 如何与系统的各种利益相关者相互协调？他们的关注点一般又会有哪些？
- 如何编写出能够让相关人员都很好了解的架构文档？
- 如何评估设计出的架构？
- 在解决某些特定系统问题时，应该采用何种合适的方法？

.....

很少有一本书能够系统地回答这么多问题，能够为迷茫的架构师做出有效的指导。我自己也是一样，作为一名程序员，对架构方面还是如履薄冰，了解了一些皮毛，绝不敢以架构师来自居。

直到偶然与这样的一本书相遇，我才发现，其实想要成为架构师，或者在成为架构师之后，想要更好地完成相应的工作，这本书都能够提供很好的指导和帮助。其中提出的各种视图、视点和视角，可以帮助我们更好地了解架构的诸多方面，从而在设计架构的时候，能够根据最重要的利益相关者的关注点，更好地进行平衡。

也正因为如此，我才争取到这样的机会，把这么好的一本书翻译成中文，呈献给国内的程序员朋友们，希望他们能够通过阅读这本书，更好地了解架构设计和架构师这种职业，也更好地为他们解决架构方面的问题。

不得不说，翻译这样一本大部头的确是很辛苦的事情，这也是我所翻译的最厚的一本书，所以首先要感谢我的家人对我的理解和支持，让我抽出晚上和周末的时间来从事该书的翻译工作。还要感谢关敏老师，容忍我一再延期交稿。还要感谢张勇等人的参与，正是大家的帮助，才让我最终完成了这本书的翻译。

愿这本书能够帮助更多有志成为架构师或者已经是架构师的朋友们！

前 言

和十年前我们开始撰写本书第 1 版时相比，当前 IT 业界的状况已经发生了天翻地覆的变化。计算机和互联网已经成为很多人日常工作和生活的重要组成部分，这使得整个世界更紧密地连接在一起。同时，这也让用户和其他利益相关者对系统寄予更大的期望，他们希望系统功能完备、易于使用、健壮稳定、方便扩展且安全可靠。我们认为，在实现这些目标的过程中，架构师扮演了非常重要的角色，而且广大专业的软件开发者以及高级业务和技术经理也深以为然。

我们非常高兴地看到，本书第 1 版得到了软件从业者、有抱负的软件架构师以及学术界的认可。读者认为它有用、可理解且内容丰富。然而，架构是一种不断变化的规则，因此本书第 2 版会反映出从第 1 版的出版到现在我们在实践中所学到和已经做出的改善。它还包括了读者发来的大量非常有用的评论和改进建议，对此我们深表感谢。

然而，本书的基础内容保持不变。本书的首要关注点依然在于：架构是为利益相关者提供的服务，并且是确保信息系统能够满足他们需求的一种方式。本书还是会强调视图（view）的极端重要性，它会以利益相关者可以理解的方式来显示架构的复杂性。我们还坚持认为，架构必须定义系统能如何提供利益相关者所要求达到的质量属性——如可扩展性、弹性和安全性——还要定义静态和动态结构，以及能够提供有效方式达到这些目的的透视图。

本书的主要受众是架构师或者期望成为架构师的人，但是，我们希望其他可能会和架构师一起工作的 IT 专业人士、某天可能会成为架构师的学生，都认为这本书值得一读。

第 2 版中做出的最重要改变如下。

- 本书引入了一种叫做情境的新视角（Context viewpoint），它会描述系统及其所处环境（人、系统以及与之交互的外部实体）之间的关系、依赖和交互。这部分内容还会将第 1 版第 8 章中对范围和情境相对简要的讨论进行扩展、格式化和标准化。
- 第二部分进一步讨论了架构角色的不同方面。
- 修订了大多视角和透视图的定义，特别是功能（Functional）和并发（Concurrency）视图以及性能（Performance）和可扩展性（Scalability）透视图。
- 修订和扩展了大多数章节的参考文献和延伸阅读部分。
- 更新了第 1 版中的部分内容，使其可以与新的国际架构标准 ISO 42010（它继承自 IEEE 标准 1471）中的概念和术语保持一致。
- 更新了 UML 建模的建议和示例，从而适应 UML 在第 2 版中所引入的变更。

我们希望你能够发现，本书第 2 版对第 1 版做出了很有用的提升和扩展，我们也邀请你访问本书的网站：www.viewpoints-and-perspectives.info，在那里你可以获得更多软件架构的资源，或者可以与我们联系，提供对本书的反馈。

致谢

除了在第 1 版中感谢的人，我们还要感谢审阅了本书第 2 版的各位朋友：Paul Clements、Tim Cull、Rich Hilliard、Philippe Kruchten 和 Tommi Mikkonen，还有勤勉负责的文字编辑 Barbara Wood。特别要感谢 Paul 彻底、具有洞察力和挑战性的意见与改进建议，这些对我们帮助很大。

第1版前言

本书的两位作者都是经验丰富的软件架构师，在信息系统开发项目中担当这个角色多年，在有些项目中是共同担任架构师，在有些项目中是分别担任的。在那期间内，我们看到软件架构师的数量得到了显著增加，而且该角色的重要性也得到同事、管理层和客户的认可。现在，所有大型的软件开发项目都需要架构师的参与，甚至在先进的开发团队中，会存在架构师小组。

尽管人们已经达成共识，认为软件架构师的角色非常重要，但是关于这一角色所包含的工作并没有一致的意见。谁是我们的客户？我们应该对谁负责？他们期望我们交付什么？一旦架构设计完成，我们又该做些什么？以及可能最重要的是，需求、架构和设计之间的边界是什么？

因为当前软件项目（特别是项目的架构师）中有很多亟待解决的问题，所以缺少对架构师角色的清晰定义，这导致了更多问题。

- 用户和其他利益相关者在功能、容量、推向市场的时间以及灵活性方面提出了更多苛求。
- 漫长的系统开发时间导致范围持续改变，进而导致系统架构和设计经常变更。
- 现在的系统要比以往功能更加丰富，结构更加复杂，经常会混用直接可用的和自定义构建的组件。
- 很少有系统能够独立存在，大多数都需要和很多其他系统交互操作并交换信息。
- 让系统的功能结构（即设计）正确只是问题的一部分。系统如何表现（例如，它的质量属性）与其对有效性同样重要。
- 技术持续改变的步伐让架构师的技术专业知识无法保持最新。

当第一次担任软件架构师的时候，我们试图寻找某种软件架构手册，希望它能够指导我们完成创建架构设计的过程。毕竟，其他架构方面的准则，在理论上和确立的最佳实践上都要领先软件架构几个世纪。

例如，在公元1世纪，罗马人 Marcus Vitruvius Pollio 就写出了第一本架构手册《建筑十书》（Ten Books on Architecture），这本书描述了建筑架构师的职责和所需要的技能，为标准的架构结构提供了丰富的资料。1670年，日记作家 Samuel Pepys 的朋友 Anthony Deane，曾是英国 Harwich 镇的镇长，后来又成为一位议员，出版了一本开创性的书《船舶工程学》（A Doctrine of Naval Architecture），其中详细描述了当时大型船舶设计方面的先进方法。Deane 的观点和原则在很多年间帮助人们将船舶工程的实践系统化。1901年，英国化学工业的顾问工程师 George E. Davis 发表了《化学工程学手册》（A Handbook of Chemical Engineering），从而创建了一种新领域的工程学。这是第一本定义很多实践性原则的书籍，这些原则支持了工业化化学过程，并在之后的很多年引领了该领域的发展。

这些最佳实践存在的重要结果就是统一了方法。如果你想要给几位架构师和工程师一项设计楼房、旅游船或者化工厂的任务，他们创建出来的设计可能会彼此不同。然而，他们使用的过程、

在纸上（或者计算机屏幕上）展现设计的方式以及用来确保设计合理的技术非常类似。

遗憾的是，我们的行业还没有创建出可流传下去的主流界的最佳实践。当我们查看的时候，发现很少会有介绍性的书籍，可以为信息系统架构师的工作提供详细指导。

我们承认，在特定的技术上已有很多书籍，不管是 J2EE、CORBA 还是 .NET，还有一些更广泛的话题像 Web 服务或者面向对象（尽管由于软件技术变化的速度，很多像这些书籍在几年内就过时了）。也有大量优秀的软件架构方面的书籍，我们会在部分章节中对其加以引用。但是因为这些书很多旨在创建在各种系统中都可以应用的原则，所以使用很通用的方式进行编写，大多数特定文字的目标都是针对实时和嵌入式系统社区的朋友。

我们觉得，如果你是一位新的信息系统软件架构师，很少有书会告诉你如何完成自己的工作，学到你所需要知道的最重要的知识，并成功地做出架构设计。我们不想取代已经存在的软件架构方面的书籍，也不敢与 Vitruvius、Deane 和 Davis 比肩，让我们决定撰写这本书的驱动力就是要解决上述需求。

具体来说，本书将会向你展示：

- 软件架构的相关内容有哪些，你的角色对于成功交付项目为什么极度重要。
- 如何决定谁对你的架构感兴趣（你的利益相关者），理解什么对他们重要（他们的关注点），并设计反映和平衡他们不同需求的架构。
- 如何以可理解的方式与利益相关者沟通你的架构，从而说明你已经满足了他们的关注点（架构描述）。
- 如何关注对架构重要的内容，安全地把其他方面的设计工作交给设计师，而不会遗漏如性能、弹性和位置等问题。
- 作为架构师，哪些重要的活动是你最需要执行的，例如识别并引入利益相关者、使用场景、创建模型以及编写架构文档并加以验证。

贯穿全书，我们首先关注的是大规模信息系统（即用来自动化大型组织的业务操作的计算机系统）的开发。然而，我们也试图以一种特别的方式来展现内容，这种方式不依赖你正在设计的信息系统的类型、开发者将要使用的技术以及项目所遵循的软件开发生命周期。我们标准化了一些事情，例如在大多数图表中使用了统一建模语言（Unified Modeling Language, UML），但是我们只做了这么多，因为 UML 是最广为理解的建模语言。想要理解这本书，你不需要是一位 UML 专家。

我们并不打算让本书成为创建你的信息系统架构的确切指导——这样一本书可能永远都不可能完成，并且需要大量跨广泛技术领域专家的合作。我们也不想编写一本规范方法的书。尽管我们展示了说明如何产出可交付物的一些活动图，但是设计这些活动图是为了与当前使用中的多种软件开发方法相兼容。

我们期望达到的目的是，创建一种可实践的、面向实践者的指导，说明如何为信息系统设计出成功的架构，以及如何保证它们可以成功实现。这是我们在开始从事软件架构师工作时想要看到的那种书，也是我们现在期望参考的书。

你可以访问我们的网站：www.viewpoints-and-perspectives.info，找到更多有用的软件架构资

源，并和我们联系，提供对本书内容的反馈。我们非常期待能够听到你的声音。

致谢

本书的出版得到了很多人的建议、帮助和支持，否则这本书永远都不会问世。

我们非常感谢多位审校者，他们在我们创作本书的不同阶段对文字给出了意见，包括 Gary Birch、Chris Britton、Kelley Butler、Sholom Cohen、Dan Haywood、Sallie Henry、Andy Longshaw、Robert Nord、Dan Paulish、Martyn Thomas 和 Hans van Vliet。

我们还要感谢 Addison-Wesley 的团队成员，正是他们的工作让这本书成为现实，包括 Kim Boedigheimer、John Fuller、Peter Gordon、Chrysta Meadowbrooke、Simon Plumtree 和 Elizabeth Ryan。

还有很多人为我们提供了建议、鼓励和启发，包括 Felix Bachmann、Dave Borthwick、David Emery、Wolfgang Emmerich、Rich Hilliard、Philippe Kruchten、Roland Leibundgut、Mike Mackay、Dave Maher、Mark Maier、Lucia Rapanotti 和 Gaynor Redvers-Mutton。

我们还要感谢我们的家人，他们始终为我们提供持续的爱、鼓励和支持。

目 录

译者序		
前言		
第1版前言		
第1章 简介	1	
1.1 利益相关者、视点和视角	1	
1.2 本书结构	4	
1.3 谁应该阅读本书	5	
1.4 本书约定	5	
第一部分 架构的基本原则		
第2章 软件架构概念	8	
2.1 软件架构	8	
2.1.1 系统元素和关系	8	
2.1.2 基本系统属性	9	
2.1.3 设计和发展的原则	10	
2.1.4 系统属性和内部组织形式	10	
2.1.5 软件架构的重要性	13	
2.2 架构元素	13	
2.3 利益相关者	14	
2.3.1 个人、团队或组织	14	
2.3.2 兴趣和关注点	15	
2.3.3 利益相关者的重要性	16	
2.4 架构描述	16	
2.5 核心概念之间的关系	17	
2.6 小结	18	
2.7 延伸阅读	19	
第3章 视点和视图	20	
3.1 架构视图	22	
3.2 视点	23	
3.3 核心概念之间的关系	24	
3.4 使用视点和视图的好处	24	
3.5 视点缺陷	25	
3.6 视点目录	25	
3.7 小结	27	
3.8 延伸阅读	28	
第4章 架构视角	29	
4.1 质量属性	29	
4.2 架构视角	30	
4.3 向视图应用视角	33	
4.4 应用视角的结果	34	
4.4.1 深入的观点	34	
4.4.2 提升	35	
4.4.3 精品内容	35	
4.5 核心概念之间的关系	35	
4.6 使用视角的好处	36	
4.7 视角的缺陷	37	
4.8 视角与视点对比	37	
4.9 视角种类	38	
4.10 小结	39	
4.11 延伸阅读	39	

第5章 软件架构师的角色	41	7.7.2 迭代方法	65
5.1 架构定义过程	41	7.7.3 敏捷方法	65
5.1.1 架构定义不仅是设计	42	7.8 小结	66
5.1.2 需求分析和架构定义之间的 区别	43	7.9 延伸阅读	67
5.1.3 架构定义和设计之间的区别	43	第8章 关注点、原则和决定	68
5.2 架构师的角色	44	8.1 专注于问题的关注点	70
5.3 核心概念之间的相互关系	46	8.1.1 业务策略	70
5.4 架构专门化	47	8.1.2 业务目标和驱动力	70
5.5 组织情境	47	8.1.3 系统范围和需求	71
5.5.1 业务分析师	47	8.1.4 业务标准和政策	72
5.5.2 项目经理	47	8.2 专注于解决方案的关注点	72
5.5.3 设计主管	48	8.2.1 IT策略	72
5.5.4 技术专家	49	8.2.2 技术目标和驱动力	72
5.5.5 开发者	49	8.2.3 技术标准和政策	73
5.6 架构师的技能	49	8.3 其他现实世界中的约束	73
5.7 架构师的责任	50	8.4 什么决定了好的关注点	75
5.8 小结	51	8.5 架构原则	75
5.9 延伸阅读	51	8.5.1 什么造就了好的原则	78
		8.5.2 定义自己的原则	78
		8.6 架构决定	79
		8.7 使用原则关联关注点和决定	81
		8.8 检查列表	82
		8.9 小结	83
		8.10 延伸阅读	83
		第9章 确定并引入利益相关者	84
		9.1 利益相关者的选择	84
		9.2 利益相关者的类别	85
		9.2.1 出资方	86
		9.2.2 评估者	86
		9.2.3 沟通者	86
		9.2.4 开发人员	87
第二部分 软件架构过程			
第6章 软件架构过程简介	54		
第7章 架构定义过程	55		
7.1 指导原则	55		
7.2 过程产出物	56		
7.3 过程情境	56		
7.4 支持活动	57		
7.5 架构定义活动	60		
7.6 过程完成标准	62		
7.7 软件开发生命周期中的架构定义	64		
7.7.1 瀑布式方法	64		

9.2.5 维护人员	87	10.7.3 尽早使用场景	101
9.2.6 生产工程师	87	10.7.4 包含对系统质量场景的 使用	101
9.2.7 供应商	87	10.7.5 包含对故障场景的使用	101
9.2.8 支持人员	87	10.7.6 让利益相关者紧密参与	101
9.2.9 系统管理员	88	10.8 检查列表	102
9.2.10 测试人员	88	10.9 小结	102
9.2.11 用户	88	10.10 延伸阅读	103
9.3 示例	88	第11章 使用样式和模式	104
9.3.1 非专门设计的部署项目	88	11.1 设计模式介绍	104
9.3.2 软件产品开发项目	89	11.2 样式、模式和惯用法	105
9.3.3 合作开发	89	11.2.1 架构样式	106
9.4 代理利益相关者	90	11.2.2 软件设计模式	106
9.5 利益相关者组	90	11.2.3 语言惯用法	106
9.6 利益相关者的责任	90	11.2.4 使用样式、模式和惯用法	107
9.7 检查列表	91	11.3 模式和架构策略	107
9.8 小结	91	11.4 架构样式的例子	108
9.9 延伸阅读	92	11.5 使用架构样式的好处	110
第10章 识别并使用场景	93	11.6 样式和架构描述	111
10.1 场景类型	93	11.7 应用设计模式和语言惯用法	111
10.2 使用场景	94	11.8 检查列表	113
10.3 识别场景并排定优先级	95	11.9 小结	113
10.4 捕获场景	96	11.10 延伸阅读	113
10.5 什么造就了好场景	98	第12章 创建架构模型	115
10.6 应用场景	98	12.1 模型为什么重要	115
10.6.1 纸质模型	98	12.2 模型的类型	117
10.6.2 走查	99	12.2.1 定性模型	117
10.6.3 模拟	100	12.2.2 定量模型	118
10.6.4 原型实现的测试	100	12.2.3 示意图	119
10.6.5 完整规模真实测试	100	12.3 建模语言	119
10.7 有效使用场景	100	12.3.1 架构描述语言	119
10.7.1 识别一系列重点场景	101		
10.7.2 使用清晰的场景	101		

12.3.2	统一建模语言	120	13.4.2	内容表	135
12.3.3	可执行的领域专用语言	121	13.4.3	介绍和管理纲要	135
12.3.4	其他建模语言	121	13.4.4	利益相关者	136
12.4	创建有效模型的准则	121	13.4.5	通用架构原则	136
12.4.1	有目的地建模	121	13.4.6	架构设计决定	136
12.4.2	应对受众	122	13.4.7	视点	136
12.4.3	仔细、准确地抽象	122	13.4.8	视图	136
12.4.4	根据风险确定工作重点	123	13.4.9	质量属性摘要	137
12.4.5	选择描述性的名称	123	13.4.10	重要的方案	137
12.4.6	定义你的术语	123	13.4.11	亟待解决的问题	137
12.4.7	以简单为目标	124	13.4.12	附录	138
12.4.8	使用已定义的标记法	124	13.5	展现架构描述	138
12.4.9	了解暗示的语义	124	13.6	检查列表	139
12.4.10	验证模型	125	13.7	小结	140
12.4.11	保持模型的活力	125	13.8	延伸阅读	140
12.5	和敏捷团队一起建模	125	第14章	评估架构	141
12.6	检查列表	126	14.1	为什么要评估架构	141
12.7	小结	127	14.2	评估技术	142
12.8	延伸阅读	127	14.2.1	演讲	142
第13章	创建架构描述	128	14.2.2	正式评审和结构化的走查	143
13.1	有效架构描述的属性	129	14.2.3	通过使用场景来评估	144
13.1.1	正确	129	14.2.4	原型和概念验证系统	145
13.1.2	充分	129	14.2.5	骨架系统	146
13.1.3	及时	130	14.3	基于场景的评估方法	146
13.1.4	简洁	131	14.3.1	以架构为中心的活动	147
13.1.5	清晰	131	14.3.2	以利益相关者为中心的 活动	149
13.1.6	最新	132	14.4	在软件生命周期内评估	150
13.1.7	精确	133	14.5	验证现存系统的架构	151
13.2	词汇表	134	14.6	记录评估结果	153
13.3	ISO标准	134	14.7	选择评估方法	154
13.4	架构描述的内容	135	14.8	检查列表	154
13.4.1	文档控制	135			

14.9 小结	155	16.5 延伸阅读	172
14.10 延伸阅读	155	第17章 功能视点	173
第三部分 视点类型			
第15章 视点类型简介	158	17.1 关注点	173
第16章 情境视点	160	17.1.1 功能能力	173
16.1 关注点	161	17.1.2 外部接口	174
16.1.1 系统范围和责任	161	17.1.3 内部结构	174
16.1.2 外部实体和服务以及所用数据 的标识	161	17.1.4 功能设计哲学	174
16.1.3 外部实体的本质和特征	162	17.1.5 利益相关者的关注点	175
16.1.4 外部接口的标识和职责	162	17.2 模型	176
16.1.5 外部接口的本质和特征	163	17.3 问题和缺陷	184
16.1.6 其他外部依赖关系	163	17.3.1 设计很差的接口	184
16.1.7 对系统环境的影响	164	17.3.2 难以理解的职责	184
16.1.8 总体完成度、一致性和连 贯性	164	17.3.3 基础架构作为功能性元素	184
16.1.9 利益相关者的关注点	165	17.3.4 过载的视图	185
16.2 模型	165	17.3.5 没有元素定义的图	186
16.2.1 情境模型	165	17.3.6 难以调节多位利益相关者的 需求	186
16.2.2 交互场景	169	17.3.7 错误的详细程度	187
16.3 问题和缺陷	169	17.3.8 “神元素”	187
16.3.1 遗漏或者错误的外部实体	169	17.3.9 过多依赖关系	188
16.3.2 遗漏隐藏的依赖关系	169	17.4 检查列表	188
16.3.3 松散或不精确的接口描述	170	17.5 延伸阅读	188
16.3.4 详细程度不合适	170	第18章 信息视点	190
16.3.5 范围蔓延	170	18.1 关注点	191
16.3.6 隐藏或假设的情境和范围	171	18.1.1 信息结构和内容	191
16.3.7 过于复杂的交互	171	18.1.2 信息目的和用途	191
16.3.8 过度使用术语	171	18.1.3 信息所有权	192
16.4 检查列表	172	18.1.4 企业拥有的信息	193
		18.1.5 标识符和映射关系	194
		18.1.6 信息语义的易变性	195
		18.1.7 信息存储模型	196

18.1.8	信息流	197	19.1.7	启动和关闭	219
18.1.9	信息一致性	198	19.1.8	任务故障	219
18.1.10	信息质量	199	19.1.9	重入	219
18.1.11	及时性、延迟和寿命	200	19.1.10	利益相关者的关注点	220
18.1.12	归档和保留信息	201	19.2	模型	220
18.1.13	利益相关者的关注点	201	19.2.1	系统级别的并发模型	220
18.2	模型	202	19.2.2	状态模型	225
18.2.1	静态信息结构模型	202	19.3	问题和缺陷	228
18.2.2	信息流模型	204	19.3.1	对错误的并发建模	228
18.2.3	信息生命周期模型	206	19.3.2	错误地对并发建模	228
18.2.4	其他类型的信息模型	207	19.3.3	过度复杂	229
18.3	问题和陷阱	209	19.3.4	资源竞争	229
18.3.1	数据展现不兼容	209	19.3.5	死锁	230
18.3.2	不可避免的多个更新器	210	19.3.6	竞争条件	230
18.3.3	键值匹配缺陷	211	19.4	检查列表	230
18.3.4	接口复杂	211	19.5	延伸阅读	231
18.3.5	过载的中心数据库	212	第20章	开发视点	232
18.3.6	不一致的分布式数据库	213	20.1	关注点	232
18.3.7	信息质量很差	213	20.1.1	模块组织	232
18.3.8	信息延迟过大	213	20.1.2	通用处理	233
18.3.9	容量不足	214	20.1.3	设计的标准化	233
18.4	检查列表	214	20.1.4	测试的标准化	233
18.5	延伸阅读	215	20.1.5	测试辅助	233
第19章	并发视点	216	20.1.6	代码行组织	233
19.1	关注点	217	20.1.7	利益相关者的关注点	233
19.1.1	任务结构	217	20.2	模型	234
19.1.2	功能元素与任务的映射 关系	218	20.2.1	模块结构模型	234
19.1.3	进程间通信	218	20.2.2	通用设计模型	235
19.1.4	状态管理	218	20.2.3	代码行模型	238
19.1.5	同步和整合	218	20.3	问题和缺陷	239
19.1.6	支持可伸缩性	219	20.3.1	过多细节	239

20.3.2	过载的架构描述	239	21.3.7	提供的净空不合适	254
20.3.3	不平均的重点	240	21.3.8	未指定灾难恢复环境	255
20.3.4	缺少开发者的关注	240	21.4	检查列表	255
20.3.5	精度不足	240	21.5	延伸阅读	256
20.3.6	特定环境的问题	240	第22章	运维视点	257
20.4	检查列表	241	22.1	关注点	257
20.5	延伸阅读	241	22.1.1	安装和升级	257
第21章	部署视点	243	22.1.2	功能迁移	258
21.1	关注点	243	22.1.3	数据迁移	258
21.1.1	所需的运行时平台	243	22.1.4	运维监控和控制	259
21.1.2	硬件或者托管平台所需的规格 和品质	244	22.1.5	警告	260
21.1.3	第三方软件需求	244	22.1.6	配置管理	260
21.1.4	技术兼容性	244	22.1.7	性能监控	260
21.1.5	网络需求	245	22.1.8	支持	261
21.1.6	所需的网络能力	245	22.1.9	备份和还原	261
21.1.7	物理约束	245	22.1.10	第三方环境中的运维	262
21.1.8	利益相关者的关注点	245	22.1.11	利益相关者的关注点	262
21.2	模型	246	22.2	模型	263
21.2.1	运行时平台模型	246	22.2.1	安装模型	263
21.2.2	网络模型	249	22.2.2	迁移模型	265
21.2.3	技术依赖关系模型	250	22.2.3	配置管理模型	265
21.2.4	模型之间的关系	251	22.2.4	管理模型	267
21.3	问题和缺陷	252	22.2.5	支持模型	270
21.3.1	不清晰或者不精确的依赖 关系	252	22.3	问题和缺陷	273
21.3.2	未经验证的技术	253	22.3.1	缺少运维人员的参与	273
21.3.3	不合适或者遗漏的服务级别 协议	253	22.3.2	缺少撤销计划	273
21.3.4	缺少专家的技术知识	253	22.3.3	缺少迁移计划	273
21.3.5	未能及时考虑部署环境	254	22.3.4	不充分的迁移窗口	274
21.3.6	忽略站点间的复杂性	254	22.3.5	遗漏管理工具	274
			22.3.6	生产环境的约束	274
			22.3.7	缺少到生产环境中的整合	275
			22.3.8	不充分的备份模型	275

22.3.9 不合适的警告	275	25.2.5 机密性	288
22.4 检查列表	276	25.2.6 完整性	288
22.5 延伸阅读	276	25.2.7 可用性	288
第23章 保持视图一致性	277	25.2.8 可说明性	289
23.1 视图之间的关系	277	25.2.9 检测和恢复	289
23.2 情境和功能视图的一致性	278	25.2.10 安全机制	289
23.3 情境和信息视图的一致性	278	25.3 活动: 应用安全视角	290
23.4 情境和部署视图的一致性	279	25.3.1 确定敏感资源	290
23.5 功能和信息视图的一致性	279	25.3.2 定义安全性策略	291
23.6 功能和并发视图的一致性	279	25.3.3 识别对系统的威胁	293
23.7 功能和开发视图的一致性	280	25.3.4 设计安全性实现	294
23.8 功能和部署视图的一致性	280	25.3.5 评估安全风险	296
23.9 功能和运维视图的一致性	280	25.4 架构策略	296
23.10 信息和并发视图的一致性	281	25.4.1 应用识别出的安全性原则	296
23.11 信息和开发视图的一致性	281	25.4.2 对用户进行身份验证	298
23.12 信息和部署视图的一致性	281	25.4.3 授权访问	298
23.13 信息和运维视图的一致性	281	25.4.4 确保信息保密性	299
23.14 并发和开发视图的一致性	282	25.4.5 确保信息的完整性	299
23.15 并发和部署视图的一致性	282	25.4.6 确保可负责性	300
23.16 部署和运维视图的一致性	282	25.4.7 保护可用性	300
		25.4.8 整合安全性技术	301
		25.4.9 提供安全性管理	301
		25.4.10 使用第三方的安全性基础 架构	301
		25.5 问题和缺陷	302
		25.5.1 复杂的安全性策略	302
		25.5.2 未经验证的安全性技术	302
		25.5.3 没有针对故障设计系统	302
		25.5.4 缺少管理工具	303
		25.5.5 技术驱动的方法	303
		25.5.6 没有考虑时间源	303
		25.5.7 过度依赖于技术	304
		25.5.8 没有清晰的需求或模型	304
第四部分 视角			
第24章 视角类型简介	284		
第25章 安全性视角	286		
25.1 对视图的适用性	287		
25.2 关注点	287		
25.2.1 资源	287		
25.2.2 当事人	287		
25.2.3 策略	287		
25.2.4 威胁	288		