HZ BOOKS

软件工程技术丛书

设计系列

软件架构

组织原则与模式

Software Architecture

Organizational Principles and Patterns

(美) David M. Dikel 等著

张 恂 等译

机械工业出版社
China Machine Press

PH PTR

软件工程技术丛书

设计系列

软件架构

组织原则与模式

Software Architecture

Organizational Principles and Patterns

(美) David M. Dikel 等著

张恂 等译

机械工业出版社
China Machine Press

本书主要描述软件架构与软件组织之间的相互关系，依次介绍了作者根据多年管理经验和研究总结出的软件架构组织的 VRAPS 5 项原则——构想（Vision）、节奏（Rhythm）、预见（Anticipation）、协作（Partnering）和简化（Simplification），并通过案例分析、模式和反模式展示了如何运用这一模型。本书的主要读者为软件企业的管理人员、开发人员和软件产品的客户等，也可作为大学计算机及相关专业的本科生、研究生和教师的参考用书。

本书版权登记号：图字：01－2002－0693

图书在版编目（CIP）数据

软件架构：组织原则与模式/（美）迪克尔（Dikel，D.M.）等著；张恂译．－北京：机械工业出版社，2002.8
（软件工程技术丛书）
书名原文：Software Architecture: Organizational Principles and Patterns
ISBN 7-111-10166-9

Ⅰ.软… Ⅱ.①迪…②张… Ⅲ.软件设计 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字（2002）第 021155 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：温丹丹
北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行
2002 年 8 月第 1 版第 1 次印刷
787mm×1092mm 1/16 · 13印张
印数：0 001－5 000 册
定价：35.00 元

译 者 序

与其他制造业一样，软件的生产也需要一流的生产设备、流水线和高素质的生产者与管理者；而作为高度智慧与知识的结晶，软件的生产又有着其独有的内在规律。

跨入21世纪，蓬勃发展的中国民族软件业正面临着前所未有的机遇和挑战。虽然在龙头企业的带领下中国软件业短短几年内获得了长足的进步，实力大增，但是国内软件产业的整体发展水平与美国、欧洲、印度等国家和地区相比仍明显落后，尤其在生产方式、过程控制、组织管理等领域差距更大。多年以来，与其他成熟行业（如建筑、制造业）相比，国内的软件生产和项目开发一直缺乏有效的工程化管理。

究其原因，译者认为根源在于不健全、不规范的市场环境造成软件产业链的资源分布不均衡。在急功近利、片面追求效益的行业风气影响下，国内软件工程的教育、技术和管理水平严重滞后并与软件生产实践相脱节，导致民族软件业长期在低水平上徘徊。与国外高度发达的软件工业化生产相比，我们的软件生产企业大部分还停留在手工作坊时代。这一瓶颈问题如不能得到根本解决，无疑将严重制约民族软件业健康和持续地发展。

可喜的是，近两年来，许多软件企业和有识之士逐步认识到掌握软件生产规律、提升软件工程管理水平的重要性与迫切性。软件架构（Architecture）和框架（Framework）、CMM、统一软件过程（Unified Software Process）、UML、软件模式（Pattern）与反模式（Antipattern）、构件式开发（Component-Based Development）等先进的软件工程理论和方法开始被接受，其普及与应用正方兴未艾。在落后了国外3～5年甚至更长时间之后，人们终于意识到原来还可以这样做软件。

译者从事面向对象技术和软件工程咨询与管理工作多年，有幸接触了各类规模的软件生产企业，也结识了不少业界朋友。译者在与他们的交流切磋中了解到，中国软件企业面临的主要内部问题是管理问题，而不是技术问题，其关键是当代先进的软件生产技术如何与现代化的企业管理方法有机结合，这已经成为很多人的共识。

Ivar Jacobson大师在其里程碑著作《软件重用》（*Software Reuse*）[Jacobson97]（中文版由机械工业出版社出版）中指出，影响软件产品或项目成败的主要因素包括架构、过程和组织三个方面。其中，软件架构是决定软件产品内在质量的核心因素，合理的开发过程与高效的组织管理则是软件架构乃至软件产品、产品线成功的重要保障。

本书中所说的软件架构及其组织原则的适用范围既包括单一产品架构，又包括产品家庭或产品线上多个产品共享的架构；这里所谓的“组织”既包括一个项目组也包括一个部门、一个

企业甚至有多个合作伙伴的联盟。本书的原则无论对于国内只做工程项目或单一产品的企业，还是拥有一个或多个产品线的组织都具有指导或借鉴意义。

同样以架构为中心，《软件重用》提出了面向重用的软件工程业务（RSEB）的概念，而本书则重点介绍与软件架构互动的组织管理原则，笔墨侧重于技术和人文管理的结合部分。译者认为这是此书的一大特色。

在国内我们一般只听说过程序员、高级程序员、系统分析员的认证考试，但很少听说软件架构师这样一个非常重要却易被人忽视的岗位。本书恰恰是专为软件架构师和软件企业的管理者（如高层经理、技术总监、产品经理、事业部经理、部门经理和项目经理等）写的，适合的读者还包括各类开发人员（实施者）、软件产品的客户等。

本书结构严谨，依次介绍了作者依据多年管理实践和研究经验总结出的软件架构组织的VRAPS 5 项原则（模型），即构想（Vision）、节奏（Rhythm）、预见（Anticipation）、协作（Partnering）与简化（Simplification）。针对每项原则，每个主要章节都讨论了度量准则（Criteria）与指导实施的模式和反模式。

本书的价值在于体现了组织管理原则与软件生产技术的有机结合，用高度概括的 VRAPS 5 项原则（含 15 个度量准则）、16 种模式和 17 种反模式的形式，把宝贵的软件架构组织管理的真知灼见和经验教训系统地奉献给读者。

软件模式即解决软件领域常见问题的最佳方法，而反模式则是针对特定问题应该避免的错误做法，相应地就有软件分析、设计、实现和管理等各个方面的（反）模式。建立并全面有效地利用好模式知识库，是借鉴国外软件业成熟经验、重用其先进思想方法的一条捷径，可以大大加快提升国内软件企业工程管理和技术水平的过程。不久之前机械工业出版社华章公司曾经推出过一本软件设计模式的经典之作（《设计模式——可复用面向对象软件的基础》，ISBN 7－111－07575－7）[Gamma94]，而本书很可能是国内第一本正式的有关软件管理模式的译著。

本书实例丰富，尤其详细分析了美国 Allaire 公司（其 ColdFusion 开发平台在国内有不少用户）、苹果公司、微软公司、著名的电信制造业巨人加拿大 Nortel公司（Northern Telecom）以及美国国防部课题研究的一些案例。本书深入阐述了构想、受益人（Stakeholder）、迭代递增式开发（Iterative Incremental Development）等许多国内读者不甚理解的重要概念。相信通过阅读此书，读者一定会加深对这些先进软件工程思想和方法的认识。

原书的三位作者并非业界权威或名流。国内的管理者对于书中所介绍的某些概念和经验教训也许会有似曾相识的感觉，甚至可能认为这“没什么了不起的，平时我们就是这么做的嘛”。但是，能够把这些实践经验系统完整地记录下来并有序地组织成模式，从而将其升华为管理原则和方法论，这种职业精神和创造是难能可贵的。

相信此书的翻译出版，对促进国内软件从业人员掌握现代化软件生产与管理的内在规律和

技术大有裨益，一定能引起广泛的共鸣。

非常感谢机械工业出版社华章公司的编辑们向我推荐了这样一本好书，同时也感谢我的家人、同事和朋友们（特别是郭永林和邵荣）给予我一贯的鼓励与支持，能使我顺利地完成这次翻译创作。

因时间仓促和水平局限，难免存在一些疏漏和差错，恳请各位读者批评指正。

E-mail：zhangxun2001©hotmail.com

张　恂

2002 年 2 月于上海

译者简介

张恂，1972 年生。东南大学计算机软件硕士，国内某自主研发的分布式对象通信中间件的主要开发成员，曾在国内一家著名移动通信设备厂负责一个大型通信系统软件研发项目的管理近两年，后任某民营软件公司的 CTO，目前主要从事软件工程和面向对象技术的咨询。

前　言

软件生产除了开发代码外还往往涉及到产品与相关组织整合的问题，而软件架构对于这两种活动都极其重要，如今更是如此。例如，一个普通的商业事务将穿越软件架构的许多层次，其中利用了各种共享平台，包括因特网、客户机浏览器、Web 服务器、业务逻辑构件、安全系统和后台数据库，等等。在该环境中，众多的协作者不仅必须对采用哪些核心接口和标准集，而且还要对如何使用这些标准，达成一致意见。协作者还必须对他们的贡献所增加的价值或获得的回报拥有共识。所有这些共识必须在技术的快速更新、合作重组、业务目标和需求变化以及经常发生合并与收够的情况下保持并继续有效。一旦这些共识不复存在，所开发的产品及其架构将会失败，并给开发人员、客户、管理者和发起人（sponsor）带来不小的麻烦。

本书重点介绍了软件架构与组织的相互关系。软件架构作为一种框架，不仅仅定义并规范了开发和实现产品所必需的技术层面互动，而且还包括团体与个人之间的互动。软件架构长期实现这一角色的能力依赖于组织因素。

长期以来，人们注意到软件架构与创建并使用它们的组织结构之间是相互影响的，并深入研究揭示出它们之间广泛而复杂的关系。现实中的架构与目标结构往往大相径庭，它包含了大量隐藏的软件块、怪异的连接、硬编码的“捷径”、遗漏的部分以及其他非正常的东西。同样令人吃惊的是一个组织的文化，包括成员、信念、能力和行为。实践中，架构与组织形成一个敏感、易变的矩阵。处理得当，组织和架构将会产生价值；而一旦处理不好，就有可能腐蚀掉企业的精华。

本书的目的是帮助那些对软件架构的成功拥有重要利益的人们理解、应对软件架构与组织所带来的双重挑战。这些受益人随着软件产品在开发和使用过程中跨越更多的组织边界而逐渐形成一个越来越大的独立团体，这一团体包括管理、开发、实现、维护、获得和使用软件架构的人们，其中的每一个人都能够通过更好地理解软件架构与组织的互动而受益。例如，协作技能可以帮助开发人员缩短找到软件架构或平台发布版本变更信息的时间，从而促进他们通过协商来恢复那些对于继续使用这一架构至关重要的软件特性。如果没有这些技能，软件架构师将很快失去其主顾；后者也不得不维护更多的代码，因而大大减少了可开发产品的数量。

本书的基础是我们在美国著名的大型软件开发机构和许多工作组中所从事的 5 年以上的研究，以及为各类大、中、小型单位设计产品线和实现软件架构的工作。此外，该书还吸收了我们在其他学科尤其是组织开发（organizational development）中的工作经验。我们的研究成果是一个影响软件架构成功的 5 项组织原则——构想（Vision）、节奏（Rhythm）、预见（Anticipation）、协作（Partnering）、简化（Simplification），我们称其为 VRAPS 模型。

VRAPS 原则提供了一种模型，可以利用它掌握该做些什么以提高你和公司从依赖于软件架构的产品和投资中获取长期价值的能力。VRAPS 模型将帮助读者组织并理解个人的观察，

并把它与其他人已经成功运用的实践方法和模式联系起来。读者还可以利用这一模型和原则识别长处与短处，交流思想，促进组织内外跨越角色、边界和层次的行动。

读者可以通过以下几条线索阅读本书：

- 首先通过第 1 章了解全书概貌，然后阅读第 8 章中 Allaire 公司的案例分析，了解该公司是如何运用 VRAPS 原则从一家小公司发展成为因特网应用开发工具领先供应商的。
- 从第 3～7 章读者可以了解每一个 VRAPS 原则的更多内容。在定义、描述了这些原则之后，这些章节还提供了衡量标准以帮助读者评估在本单位运用这些原则的效果。模式、实例和反模式提供了实用的向导，告诉读者为了从这些原则受益该做些什么，不该做些什么。
- 从第 2 章中读者可以全面理解 VRAPS 模型及其外部联系。第 9 章介绍了如何在现实环境中运用该模型，提供了 9 种专门的模板、工具和向导，可以帮助读者评估自己单位的状况并与其他组织进行比较。此外，我们还介绍了在一次商业基准测试中这些模板是如何使用的。

最后，请往下阅读，并欢迎访问我们的 Web 站点：www.VRAPS.com。

David M. Dikel
David Kane
James R. Wilson

目　录

第1章　无形的帮助

架构是凝固的音乐。

——斯塔尔夫人，引自拉尔夫·沃尔多·爱默森《文学和社会目标》(Letters and Social Aims)

1993年9月，当本书的一位作者David Dikel正在准备后来被提交给国会的一项美国国防部(DoD)重用计划的时候，他认识了惠普公司软件重用计划程序项目经理Ron Grace。Dikel的目的是想在做适当调整后把最佳行业实践经验写入他的DoD计划。于是Grace告诉他，当他刚调入这个项目时，曾以为最大的挑战和主要任务就是钻研并强化自己的软件工程知识和技能。可是后来他很惊讶地发现自己的大部分工作涉及到社会科学，而不是各种精妙的软件技术。

Grace举了个例子，有一个业务部门请求帮助要学习软件重用，“我们坚持要求他们的骨干首先参加关于协作的研讨班”。这段话在Dikel看来是必然的。Dikel曾经遇到过很多事例，其中的重用项目失败并不是因为缺少最新的信息库或分类技术（当时的潮流），而是因为缺少对重用对于一个组织意味着什么的基本思考。然后情况突然间明朗，原来一个协作学习班就可以暴露并帮助解决很多过去的问题。

1.1　本书主题

本书勾画了软件架构与软件开发组织的交集。软件架构正逐渐成为定义和实现技术与业务互动的框架。系统开发也从过去“烟囱”式的解决方案转变为建立在共享架构上的系统。同样地，开发和提供业务产品和服务的组织也越来越牵涉到更多不同的人员和团体。尽管这些转变带来的机遇不可否认，但是涉及共享软件架构的组织因素也会给受益人带来危险的陷阱。这些陷阱不仅威胁到组织中实现或使用架构的人，而且也影响供应商、合作伙伴和客户。

本书试图对现有的、出色的软件架构工作提供补充。本书是为所有软件架构成功的受益人——从开发人员到高级经理（executive）——而写的。本书的内容可以组织成一个模型，该模型围绕5项原则——构想（Vision）、节奏（Rhythm）、预见（Anticipation）、协作（Partnering）和简化（Simplification）（VRAPS）而建。VRAPS模型建立在运用基础理论概念有5年多研究之上，包括多项案例分析、一个商业化的软件架构基准、许多研讨班以及我们担任架构师的工作。我们利用模式和反模式说明如何在实践中运用这些原则。VRAPS模型还借鉴了其他学科的工作，尤其在组织开发方面。

1.1.1　软件架构愈显重要

如今趋势表明，一个组织开发和使用软件架构并且跨越组织边界进行合作的能力越来越重

要。不久以前，“软件”一词还很少被高层管理人员提及，而现在很少有高层经理（senior executive）敢忽视软件和软件架构所带来的战略性好处，软件和软件架构对于他们个人、职业和企业的生存都是非常关键的。为了保持竞争力，软件公司正在用 Web 技术、外包方式和许多现成产品来替代整块的、烟囱式的信息系统。

软件架构使得能够组合大量支撑产品和服务，以跟上这些趋势。一个共享的架构允许实现者很方便地分解问题，确定问题的哪些部分可以在内部解决，并能挑选出产品和服务的最佳组
2 合来填补空白。过去，整块的解决方案大多处在一个组织的控制下。而现在，你可以拥有很多优秀的方案，这些方案利用了公司内外许多团体的力量。然而，要获得共享架构所带来的好处，你的产品团队、架构设计组和整个公司可能都需要掌握一些新的组织技能。

在一条产品线上共享软件架构将带给开发过程一套核心的知识和资产集，它还可以显著减少开发和维护代码的成本，使生产、文档制作、培训和市场推广等工作有序化［Jacobson97］［Dikel97a］。此外，软件架构还有其他一些好处。如果来自同一厂商和产品线的软件特性表现不同，客户会不满意。如果应用程序接口（API）需要不同的技术，集成人员将会失去耐心和信心。遗留系统（昨日的一流产品）则更难以迁移或用新技术封装。采用产品线架构可以纠正或避免这些现象。

1.1.2　对于某些人，“他们是受益人”的消息来得太晚了

构造有效的软件架构要求对技术有深入的了解、出色的认知和沟通技巧，以及大量的艰苦工作。然而，从软件架构中获得价值并非只是技术和精明的工程师的事情，成功往往依赖于那些容易被高层管理人员、经理或实施者所忽视的组织因素。

举个例子，Nortel 公司[⊖]在数字交换架构领域的领导地位使得它占领了美国和国际市场。1984 年 AT&T 分解以后，只有 Nortel 公司能向各个贝尔运营公司提供客户平等接入长话的能力。Nortel 公司优异的产品质量使其成为日本第一家数字交换系统的供应商。

可是在成功使用了近 20 年后，Nortel 公司的数字交换架构开始显露出需要更新换代的迹象。它已经变得庞大而复杂。一个工程师曾经这样评论该架构，“它需要智商（IQ）超过 140 的加拿大人中的一半来维护”。在 20 世纪 80 年代末，公司考虑对整个架构做一次大规模的重构，然而当时的 CEO（首席执行官）决定还是等一等——这个架构是公司主要利润的基础，改变它必然导致大量的开销和风险。

这项等待的决定带来了严重的后果。产品质量开始下降，发布周期增加了两倍。Nortel 公
3 司最后把这些问题归咎于“架构的瓦解”［Ziegler93］，并终于采取了行动（包括更换 CEO），之后业绩才开始反弹。在 1993 年报告损失了 8.78 亿美元后，Nortel 公司宣布在 1994 年盈利 4.08 亿美元，1997 年增长到 8.29 亿美元（见图 1-1）。

当然，其他一些因素也影响了 Nortel 公司的运作。作为这一事件的结果，CEO 丢掉了职位，而其他一些人也认识到架构问题对他们产生了直接影响。

软件架构和组织有时可以看作完全不同的实体，而有时又完全不是独立的。架构是“架构师做的事情”；而组织是“高级经理和人力资源部门处理的事情”（见图 1-2）。在一些大的官僚

⊖ Nortel 公司（Northern Telecom）的前身是 Bell Northern Research，它们都是 BCE 的子公司。

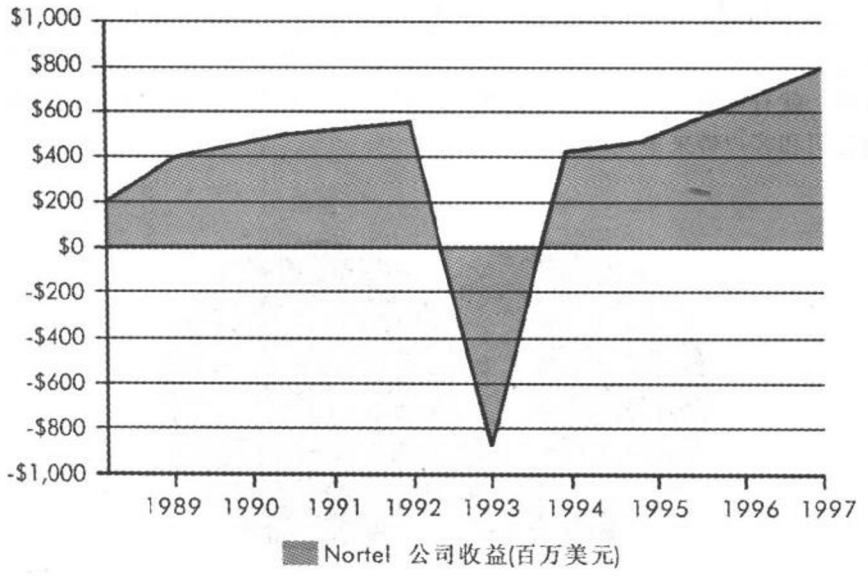

图 1-1　Nortel 公司经历了一次与架构有关的收入减少和反弹[⊖]

机构中，架构往往与组织混淆在一起。在这种情况下，架构构件本身与其功能或接口几乎没有任何关系，而与负责该构件的项目有直接的关系。

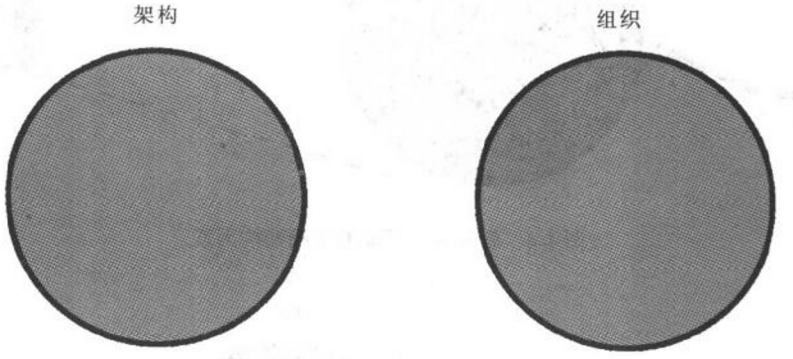

图 1-2　一些人认为架构和组织是无关的

从 1968 年起人们就开始认识到组织和架构是有关联的［Conway68］。今天，软件架构支持并且需要公司或机构内外许多不同团体的参与（见图 1-3）。为了有效地管理如今的软件架构，必须要注意观察架构是如何与所支持的组织相互影响的。

当你研究组织如何与架构互动时，就会发现组织中有多少人影响组织的生存。如果你是高级经理，你所看到的架构影响可能不是一个平面视图（见图 1-4）。结果，一个看上去无害的支

⊖ 收益数据（“扣除特别项目外的收益（损失）”）摘自 Nortel 公司 1996 和 1997 年报。

持私有 API 的承诺可能要求架构组对架构做出根本的改变，从而引起与其他厂商产品的不兼容。如果你是实施人员，组织的影响也可能对你隐藏不见（见图 1-5）。如果你不把组织的业务目标考虑在内，就有可能使用一个开放源代码构件，而它的许可证条款与公司的业务模型相抵触，这会给公司和客户带来重大风险。

5

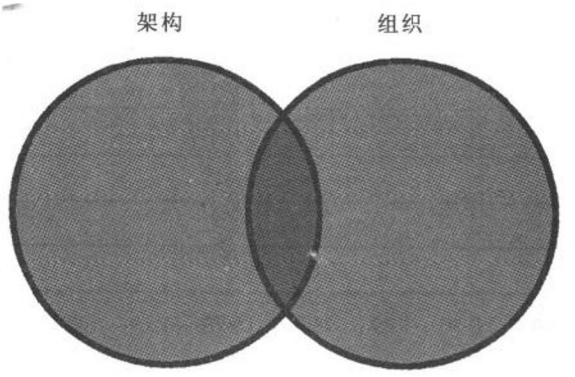

图 1-3　如今，不可否认架构与组织之间有相交部分

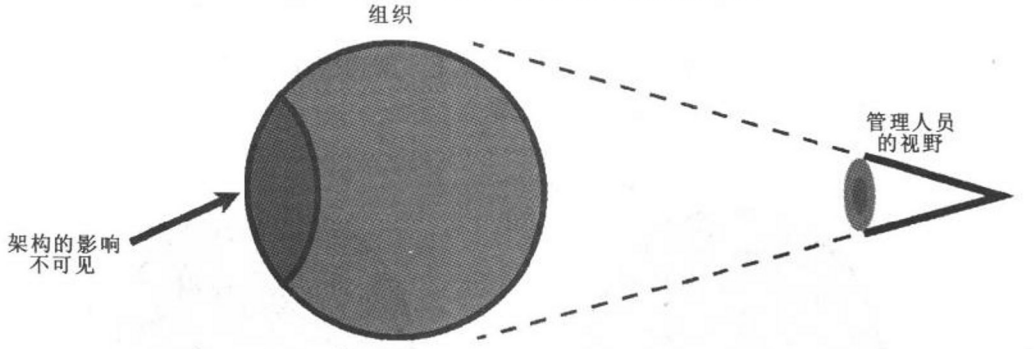

图 1-4　管理人员可能看不到架构因素

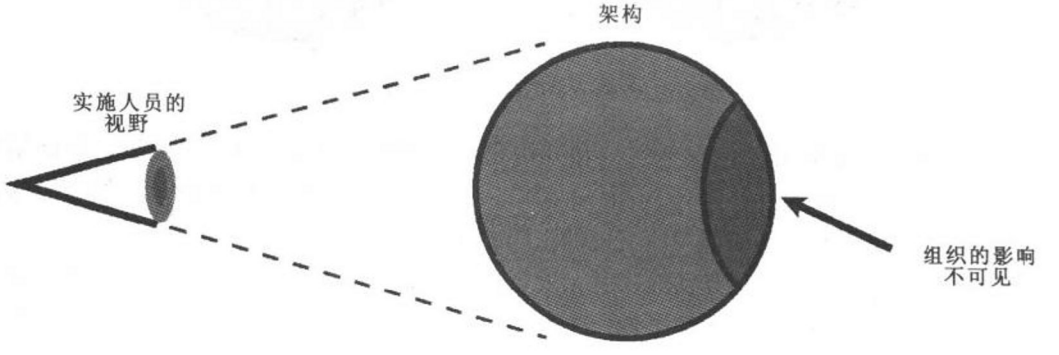

图 1-5　实施人员可能看不到组织因素

我们相信利用本书关注组织和架构的交集，可以转移实施人员、高级经理、架构师和主管们的视点，使他们不被蒙蔽，从而保证架构和组织的成功（见图 1-6）。 6

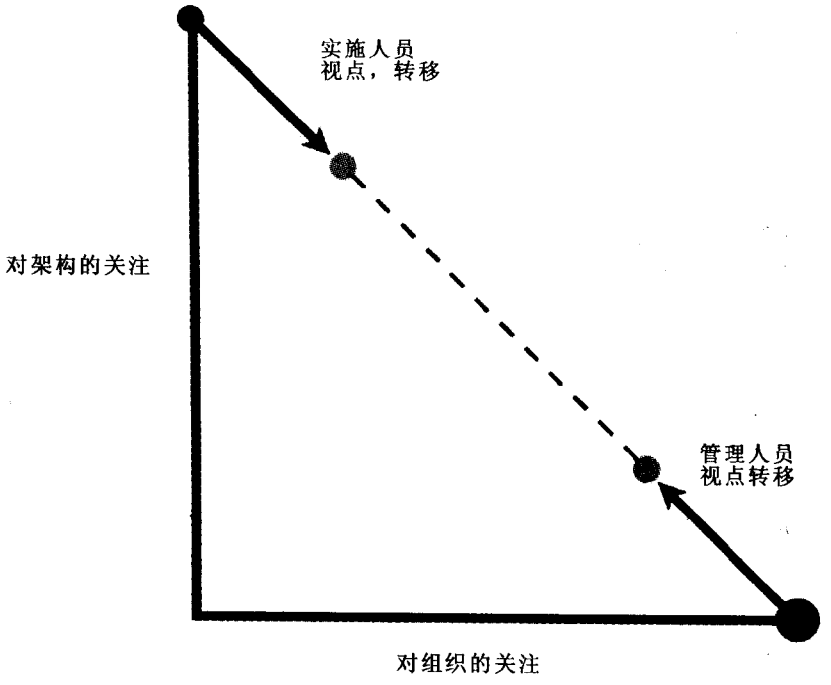

图 1-6 VRAPS能同时转移管理人员和实施人员的视点，从而提供一个更加平衡的架构与组织视图

1.2 原则揭示本质

本书考察了作为通用参考框架以应对以上挑战的 5 项原则。我们把这 5 项原则——构想、节奏、预见、协作和简化组成 VRAPS 模型。关注这 5 项原则可以帮助发现软件架构的潜在风险和机遇。下面我们简单列举了每一项原则的描述，同时给出一个关于架构经常被忽视的某方面的相关例子。后续章节将详细地介绍这些原则。

1. 构想原则

软件架构构想的来源有时不易被发现。构想原则说明了如何向架构的受益人描绘一幅一致的、有约束力的和灵活的未来图景。架构师可能认识不到他们提出的架构构想其实来自于他人的业务构想。这种情形在一个架构师与规划者（visionary）之间隔了好几层的大型组织中更加可能发生，结果并不奇怪，架构师建立的架构构想不会与公司的业务目标相吻合。 7

2. 节奏原则

延迟交付的后果有时不易被察觉。节奏原则刻画了一种在整个组织范围内的自信度，即可以定期地根据可预测的速度、内容和质量对工件进行取舍。有时管理层往往过分专注于发布一个新颖的特性而妨碍了版本的及时发布，甚至是当客户正在期待下一版提供很关键、但很平常

的修补或特性的时候。有时开发人员则会花费大量的时间完善某项技术细节，因而错过了最好的发布时机。

3. 预见原则

有时候对于未来的假设被隐藏了。预见原则要在预测未来的敏锐洞察与检测并适应现状之间做出平衡。有时首席架构师做出的技术性预测很准确，但是客户认为与即将出台的产品或标准兼容比满足首席架构师所追求的规格说明更加重要。如果不对技术和业务假设进行检测，有时结果会令架构师感到吃惊和失望。

4. 协作原则

拥有合作伙伴的必要性有时也易被忽视。协作原则建议在个人、团体之间加强合作，建立明确、公平的契约关系。在新架构计划的一阵鼓动和催促之下，有些首席架构师会在没有让对方明确理解的情况下就寻求获得周边可见的产品组长的认可。诸多细节问题（比如产品组如何参与平台的开发）被掩盖了。结果，首席架构师得到了资金投入，产品组长获得了重用。然而，当开始合并架构小组的工作时，所有人都大吃一惊——架构小组根本就不知道关键的需求，而产品小组却采用了其他方法来实现架构特性。

5. 简化原则

8

复杂性的驱动力有时也是不易被察觉的。简化原则要求澄清并最小化架构与创建、使用它的组织。试想有一位架构师，在发现两个小组开发的构件有重叠部分之后，指定了一个可以共享的构件。可是在作出建议后不久，架构师听到了为什么每个小组都能够独立开发这个构件的合理理由。不久，每个小组都开发出了所谓“简化的”设计变种。最终的结果却是更加复杂，而非简单。不仅每个小组的产品变得更加复杂，而且内含的原理也被掩盖了，因为开发小组经常连记录他们工作的基本信息也不愿做。架构师被组织问题蒙蔽了，这使得他起到了负面作用而不是作出贡献。

6. 根据原则采取行动

以上5项原则不仅能帮助你发现各种深浅的风险，而且还可以协助你评估自己的组织应用每项原则的状况，以决定下一步应该采取哪些措施。

下一节将提供一个示例，介绍一家因特网创业公司新雇用的架构师如何运用以上原则处理那些混乱却并非鲜见的情况。

1.3 实践组织原则：架构师的新任务

架构师到这家因特网创业公司上任后，遇到了一个没有架构的软件产品线。作为一项测试，他请每一位开发人员画出软件的架构图。结果这些架构图很混乱，每一张都不相同而且不很清楚。当时只有富有才华的CEO（他既是公司的创始人，也是主要的规划者和投资人）对产品线应该是什么样以及产生哪些价值有着全面而细致的构想，只差实现了。这家年轻的公司一直问题不断。大量资金投入了一项被淘汰的技术，而且损失了很多钱。两位有着优等生学历的前任产品线经理不是被解雇，就是忿忿地离开了。风险投资商对这家公司则更是喜忧交加。

为了捂住问题，成功的压力是巨大的。然而公司主要的软件代码实例化后超过数万行，犹如