软 件 工 程 技 术 丛 书

分析系列

实用软件需求

Practical Software Requirements

A Manual of Content & Style

（美） Benjamin L. Kovitz 著

胡辉良 张 罡 等译

方贵宾 审校

MANNING

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

分析系列

实用软件需求

Practical Software Requirements

A Manual of Content & Style

（美） Benjamin L. Kovitz 著

胡辉良 张 罡 等译

方贵宾 审校

机械工业出版社
China Machine Press

本书从实用的角度出发，通过全新的视角介绍了书写良好需求的格式和指导原则，以及分析需求问题的框架模型。本书讲述了问题域概念，列举了需求文档和规格说明书所包含的具体内容，重点引入了Michael Jackson的问题框架概念，通过非层次化的方法，举例演示了如何使用问题框架把巨大、复杂的问题分解成简单的问题。

本书内容丰富、编排合理，为程序员、测试人员、用户接口设计人员以及技术写作人员等提供了所需的全部信息。

Benjamin L. Kovitz: Practical Software Requirements, A Manual of Content & Style (ISBN 1-884777-59-7).

Original English language edition published by Manning Publications Co., 209 Bruce Park Avenue, Greenwich, Connecticut 06830.

本书版权登记号：图字：01-2004-4839

图书在版编目（CIP）数据

实用软件需求 /（美）科维茨（Kovitz, B. L.）著；胡辉良等译. -北京：机械工业出版社，2005.1

（软件工程技术丛书 分析系列）

书名原文：Practical Software Requirements, A Manual of Content & Style

ISBN 7-111-13106-1

Ⅰ. 实… Ⅱ. ① 科… ② 胡… Ⅲ. 软件开发 Ⅳ. TP311.52

中国版本图书馆CIP数据核字（2004）第112257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：王高翔 迟振春

北京中兴印刷有限公司印刷·新华书店北京发行所发行

2005年1月第1版第1次印刷

787mm × 1092mm 1/16 · 19.25印张

印数：0 001-5 000册

定价：39.00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

有关如何书写软件需求和程序规格说明书（程序接口设计）的书籍通常是很呆板的、程序化的、枯燥无味的，而本书则立足于全新视角，从实际出发，从实用的角度出发，通过合理的组织和非教条化的方法来讲述需求，这使本书富有极强的可读性。

本书讲述了问题域概念，列举了需求文档和规格说明书所包含的具体内容，重点引入了Michael Jackson的问题框架概念，通过非层次化的方法，举例演示了如何使用问题框架把巨大、复杂的问题分解成简单的问题。本书内容丰富，编排合理，为程序员、测试人员、用户接口设计人员以及技术写作人员提供了所需的全部信息。

这本书确实是比较实用的软件需求类的书籍，提供了书写良好需求的格式和指导原则，并提供了分析需求问题的框架模型，读来颇有启发，对国人软件工程水平的提高有帮助，对我国软件产业中外包业务的拓展提供了思考的工具。

本书旁征博引，涉及的领域非常宽泛，包括：社会经济领域中公司债券的生命周期、美国公民申请税款返还以及美国联邦平衡预算；法律上不动产交易及法律、美国法律的案例引用规则；文学上《悲惨世界》的情节描写、《1984》的鸭叫来源；通信业中电话网络、美国电话线路；医药卫生领域中医药处方；技术领域中卫星勘察技术、复印机技术、条形码技术及其格式；印刷领域中大量英文文法和排版惯例；语言学领域中介词结尾的争论，等等。本书实例丰富，引用恰当，没有丝毫的堆砌感，极具吸引力。另外，在本书的最后两章，列举了作者在Information + Graphics System公司时书写的真实需求文档，这是极其罕见、难能可贵的。本书引领读者进入真实项目的无规律和不可预知的复杂性环境中，去了解其他软件人员如何同真实问题艰苦奋战，感触真实项目的困难之处。

总之，本书素材的选择、内容的安排、表达的准确、文字的流畅无不是上乘的。不管你是第一次书写需求的编程者或项目经理，还是有经验的系统分析员，本书都将帮助你熟练和有效地进行工作。

由于时间仓促，译者水平有限，难免存在错误或疏漏，希望读者批评、指正，也希望与同行一道共同学习、探讨。我们的电子邮箱为 requirements@126.com。

译 者

2004年7月于上海

前　言

本书介绍如何书写许多软件项目开发早期所需要的两种文档：*需求文档*，它描述软件所产生的期望效果；*程序规格说明书或者接口设计*，它描述产生这些效果的计算机的外部行为——输入/输出设备的行为。

同时，这些文档向编程人员提供必需的信息以创建程序，产生规格说明书中所描述的行为，并最终满足需求。这一信息也为测试人员设计测试计划提供了基线，为技术写作人员生成用户手册或在线帮助提供了操作规程和背景信息。

以下人员有必要书写这些文档：

- 系统分析员——客户和开发团队其他人员之间的联络员。在大型组织中，书写需求文档通常是系统分析员的职责。本书为系统分析员提供了许多写作技术，能使读者更加清楚和更加方便地使用他们的文档，并且相对于其他书籍，本书提供了软件的其他方面与需求之间更加清晰的界线。
- 用户接口设计人员——根据最终用户操作软件的想法，绘制屏幕和创作概念世界的人。对于用户接口设计人员，本书介绍了程序员、测试人员和技术写作人员根据用户接口设计完成他们工作所需的信息，以及如何简洁和清晰地表达它们。
- 程序员——特别是正在对开发过程进行规范化的小公司中的那些人。许多这样的公司没有系统分析员，并首次招募自己的程序员来书写需求文档。实际上，许多程序员已经做了大量的需求工程和接口设计，只不过没有这样称呼他们的工作。对于许多程序员来说，定义问题和挑选接口行为以解决问题是他们感兴趣的工作之一；许多人认为这些工作是编程的最本质的部分。本书以更加狭隘的概念来解读*编程*这个词，以便把精力集中在需求和接口设计上。
- 需要书写需求文档的项目经理。特别是在小公司中，人们以多种角色工作。项目经理经常与客户有更多的联系，因此，他就是书写需求文档最合理的选择。这将是特别令人头痛的任务，特别是第一次，即公司中没有人有书写需求的经验。本书提供了许多具体实例以及高层和详细指南去帮助新手从头开始。

第一部分，*基本原理*，提供了区分需求、接口设计和程序设计的现行原则。这些原则主要依赖于Michael Jackson划时代的工作，更多的是以理论的形式呈现在他的书中：*Software Requirements & Specificatons: A Lexicon of Practice, Principles and Prejudices*。

第1章和第2章涵盖了程序员如何真正书写程序以及如何裁剪需求以使其可用。第2章定义了软件问题的概念。第3章把软件问题同其解决方案的主要元素相关联，以奠定本书剩余章节的

基础。

第4章到第6章介绍Jackson问题框架概念：把软件问题的基本元素转换为适合于详细文档化的形式、适合于用已知（或未知）技术创建解决方案的形式。第4章引入域、共享现象和需求等基本元素。第5章描述了软件问题的五种常见类型以及如何框定它们。第6章举例演示了如何使用问题框架把巨大、复杂的问题分解成简单的问题。不像大多数其他技术，该技术并不采用层次分解。相反，该技术更加灵活，允许一个子问题的元素和另外子问题的元素之间存在任何类型的重叠。

第二部分，内容，给出每种类型文档所包含内容的详细信息，以及如何简洁、清晰地描述这些内容的技巧。

第7章描述开发团队中每个人阅读需求和规格说明书的工作，以及他们所需要的信息。第8章是两份内容检查清单的集合，每类文档一份清单。清单列出灵活的指导原则，能帮助你发现可能忽略的必要信息，但清单不是强加于任何文档上的通用内容表。软件问题和它们的解决方案远不止如此简单。

第9章介绍如何文档化出现在大多数软件问题中的两种集合：类以及类间的关系。显示这两类集合的主要技术是标准的实体关系图或类图，并且补充了如何采用文本形式描述在图中略述的所有信息的指导原则。第9章也显示了如何描述软件所要回答的查询。

第10章是有关描述序列的技术，特别是事件序列。这包括需求文档中所需的事件本身的描述技术，以及程序规格说明书中所需的计算机和用户对这些事件响应的描述技术。

第11章解决如何描述因果关系和规则这个特别棘手的问题。因果关系可以存在于与软件交互的世界中，在需求文档中描述这些因果规则；或者它们是计算机行为所遵循的因果规则，在程序规格说明书中描述这些因果规则。第10章为读者提供了各种技术以便解决特定问题的特殊之处。

第12章介绍了大量与内容相关的其他主题，包括抽取、把面向对象设计强加到问题域描述的诱惑以及用例灾难。

第三部分，风格，关于如何选择合适的形式以展现相同的内容。大多数这些信息是良好技术写作的标准元素，尽管有关需求和规格说明书的标准元素侧重点不同。

第13章列举了大量的技术写作原理，举出了一些需求文档中的通病，并就如何纠正它们提出了建议。大多数这些错误产生于书写需求时呆板的概念或规程——强迫内容符合描述而不是形式服从内容。

第14章列举了文档组织技术：如何进行信息分组，如何选择展现这些信息的顺序。策略总是根据第二部分识别的内容来创建内容表，但不要从内容表开始，也不要在填写内容表时不考虑项目的特殊之处。

第15章是有关微小细节的指南：如何进行版面布局使之更具可读性，如何改写模糊语句，标题页面如何放置信息，如何表达定义等等。

第四部分，实例，包含真实项目中的两个示例文档。

第16章是位于美国科罗拉多州Boulder的Information + Graphics System公司在软件开发过程中，跟踪发现的bug的小软件的需求文档。

第17章是来自相同项目的用户接口设计文档的一组摘录。这些摘录包括很多示例信息，这些信息的简洁文档化非常需要技巧。

致　谢

我特别感谢Michael Jackson在写作本书时给予我的耐心帮助。除了仔细地审阅手稿外，他还多次回答我漫无边际的、稍有些缺乏组织的有关需求和软件工程的问题，他的见识为我消除了迷惑。能向世界最著名的软件工程导师学习、请教是一种莫大的荣耀。

我非常感谢Ray Agostinelli、Paul Agostinelli、Heidi Anderson、Harold Cheney和Asim Jalis许多令人鼓舞的彻夜交谈——或者更恰当地说，容忍我激昂的发泄和大声地喧叫，甚至是在星期一晚上的足球之夜。由于Ray多次仔细审阅草稿，发现了细小的错误和许多不清楚的章节，所以手稿才得以特别的改进。

Sonia Kovitz，我的母亲，对于大多数困难章节，给了我一些最苛刻的批评。她经常发现逻辑主线丢失的确切位置，帮助我改进文本，她的作用不可估量。

还有很多人不辞劳苦地抽出时间去评审各种不同手稿，他们是：William Bail，Steve Colwell，Bruce Ediger，Anne Frank，Richard Gabriel，Earl Glynn，Lon Gowen，Haim Kilow，Peter Luise，Charlie Whiting。他们的领域经验远超过我本人，反馈过滤掉了书中的许多错误，并且给本书添加了大量有价值的内容。

同时也感谢给我有关软件问题域详细信息的人们，作为程序员、需求写作人员或用户，他们非常理解软件问题域，他们是：Michael Button，Joanne Curme，Ben Eng，Lisa Higgins，Brent Jones，James Jones，Charles Libicki，Craig Lloyd，Scott Meyers，Scott Peter，Darrin Smith，Kim Swaney，Rachel Tillman，Gina Vick，Don Whiteside，Bob "Zimbob" Zimering。一些人允许我接连几个小时对他们的领域知识的细枝末节进行盘问。本书所使用的例子取自于非常广泛的软件应用范围。如果没有这些人的帮助，本书不可能包含这么大量的例子。James Jones几乎在电话中给我上了一堂债券扫盲课。Rachel Tillman也审阅了手稿，她的洞察力和丰富的经验反映在整本书中。

要特别感谢Information + Graphics System公司的Craig Bachmann和Steven Bruny，他们慷慨地允许我出版我在IGS时书写的一些需求文档。这在我们的软件业界是非常罕见的现象：由于真实项目的文档是有专利的，所以大多数有关需求的书籍仅包含假想项目的专门准备的案例。但是只有真实文档才具有真实项目的无规律性和不可预知的复杂性。人们需要了解在这些真实实例中其他人是如何同真实问题艰苦奋战的，书写需求的简单方案永远不能适应真实的问题。

同样感谢IGS中的Bug Log团队：Judy Bennett，Rae Bernhardson，Stacey Glazer，Jennifer Key，Bjorn Laukli，Randy Law，Tim Phillips，Ivan Storck。一组良好需求所体现的知识比系统分析员所赋予的更多。这些实例文档体现出他们的重要思想。

当然，这些帮助过我的人不是都赞同本书所陈述的每个思想。我已很努力地去排除错误思

想和不确切的方法，但我不能获悉本书的所有缺陷。上面所列的许多人可能能指出很多。

最后，感谢Manning Publications的杰出员工。我不可能找到一个更加投入、更加专业、更加令人愉快的团队。我特别感谢Marjan Bace，他确切地知道什么时候让创造性过程自由进行，以及什么时候给创造性过程施加压力，如此巧妙但又严格地去满足最后期限，或者激进地花费几个月时间去修订章节组织。Marjan几乎无形的冷静，以很多微妙的方式改进了本书，并可能以我们无法了解的方式改进着本书。

作者在线

读者可以免费访问私人因特网论坛，在那里你可以评论本书、询问技术问题，并可以从作者本人或其他读者那里获取帮助。该论坛的网址为：

http://www.manning.com/Kovitz

在那里你可以订阅论坛。该站点也提供如下信息：如何访问论坛，注册后可获取什么帮助，以及操作论坛的规则等。

我已努力完善和校正本书，然而错误和遗漏是在所难免的。如果你发现了本书中的错误或者遗漏，请告诉我。本书可能会有另外一个版本，如果那样，我希望尽可能地做出校正和完善。请通过作者在线论坛来指出它们。

目　录

第二部分 内 容

第三部分 风 格

第四部分 实 例

第一部分

基本原理

第1章

问题解决

本书不涉及编写程序，而是介绍如何定义“人们在编写程序时要解决的”问题。定义问题就意味着要提供程序员和接口设计人员所需的、能使计算机产生机外应有效果的所有信息。对于软件来说问题的完整描述就叫做*需求*。

编程是一种配置机器使其按特定方式运转的活动，而编写软件需求是人们之间进行交流的一种形式。期望软件实际效果的人，如想打印报表，控制生产过程，生成3D图像，或想使用软件，无论哪种都需要与设计机器行为而产生此类效果的人（即接口设计人员）进行交流。设计机器行为的人需要把他们的思想传达给那些真正配置机器的人——程序员。在软件领域工作的其他人（测试人员和编写用户手册的人）也需要这些相同的信息去完成他们的工作。这本书就是为所有这些人提供他们所需要的信息。

因而本书是一本关于技术写作特定类型的专著：如何书写软件需求文档，见图1-1。有时技术写作被狭隘地解释为只涉及语法和格式化技术。本书采用一种更广的视角：包括文档内的所有内容的选择以及如何描述各种选择，小到单词大到所有组织结构的决定[⊖]。

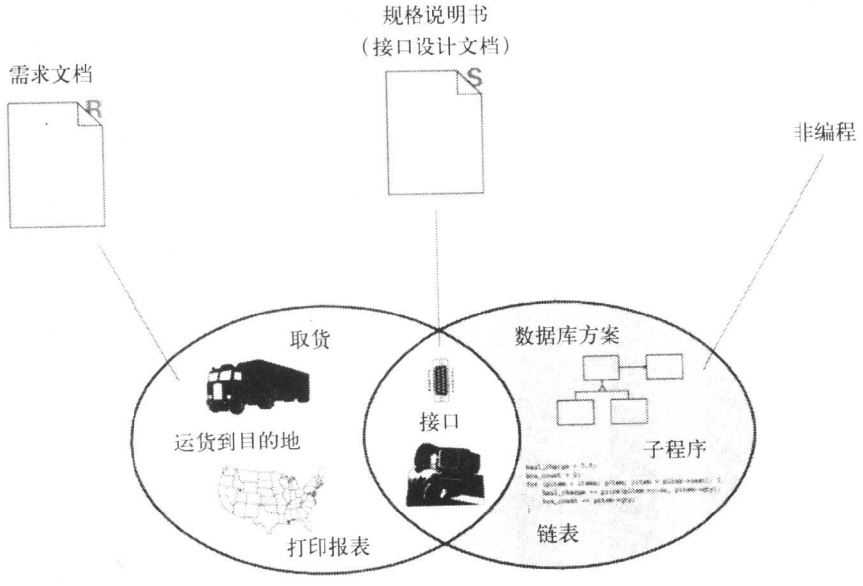

图1-1　本书的内容

⊖ [Schriver 1997] 是对技术写作广义概念的极好的、广泛的介绍。

其他成百上千有关写作的书籍都涵盖了语法和标点的细节，因此在这里就不涉及它们了。本书还不包括为所解决的问题做出好的选择的原则，或如何想出好的解决问题办法，或由什么组成一个好的用户接口等等的一些原则。本书只涉及如何书写对开发团队有用的文档，包括他们所需的所有信息，以及如何呈现这些信息的技术，使现实中的人能理解和使用它们。

第1章讲述需求、接口、编程的基本原则——它们是什么，以及相互的区别。但第一步，在我们能很好地定义软件问题之前，需要了解程序员在现实生活中是怎样着手解决它们的。首先看看过去几十年里软件业的一些经验教训——那是一个还未完成从软件作为纯研究领域到软件作为工程学科的转变阶段。

1.1 功能分解的神话

各种工程，不只是软件工程，其目的是赋予人们一些他们现在不能做到的能力——例如：在24小时内从洛杉矶赶往悉尼，获取过去5秒钟内纽约证券交易所的股票交易信息，洗衣服只花费很少的体力劳动。工程师的任务就是去设计一个能够赋予人们所期望能力的设备——飞机、信息系统或洗衣机。

宽泛地定义，*需求*就是成功地设计物品而必须满足的任何准则——粗略地说，在物品未被创造之前人们所不能做的事情，以及制造该产品的原因。我们将在第3章为需求提供一个更精确的定义，但目前仍使用这个宽泛的定义。为了书写一份有用的需求文档，我们需要了解工程师为了产生满足需求的设计而做了些什么。

工程为需求和可用材料之间的沟壑架设起基本的桥梁。而为各种不同需求和不同材料之间搭起桥梁的技术就组成了各种不同的工程领域。航空学工程师知道如何把金属和其他材料聚合在一起做成一架飞机，而满足适合的飞行需求；化学工程师知道如何设计仪器去驱动化学反应，而满足适合产生物质的需求；等等。

软件工程师知道如何配置计算机去完成各种与信息有关的任务，例如为人们提供信息，传输信息，使对象按照具体规则活动等等。软件工程师使用的材料与其他工程师的材料不同，因为它们是无形的。它们是计算机能够执行的指令，或者是能够被操作系统使用的子例程和指令块，子例程库和高级编程语言。

要为需求和材料之间的沟壑搭建桥梁通常不是件容易的工作，特别是当沟壑又宽又深时。假定有一块纱布，很容易找到洗净碗碟的方法；但手上有原料和制造洗碗机之间却存在很大的沟壑。工程师们费了几个世纪去探索制造飞机之路，其中经历了无数的死胡同。一旦有人想出如何跨越沟壑，就像莱特兄弟在飞行中所做的那样，那个设计就可重复使用，而且微小改变就可解决新的问题，但当沟壑又宽又深时，如何为它搭建此桥梁呢？例如，当你的资源只是计算

机语言中的一部分小语句，它只能加减数字、从磁盘中读写数据、执行某一指令块要依赖于特定内存位置等等，你如何着手编写一个程序去管理全国范围内的商业行为呢？

1.1.1 功能分解

已经提出了很多如何为大型工程沟壑搭设桥梁的方法。在20世纪70年代，工程师们如何在材料和需求这两个巨大沟壑之间建立可靠联系的理论开始流行起来。*功能分解*，有时也叫*自顶向下设计*或*逐步求精法*，主导着软件业大约20年的历史，并对书写软件需求的系统分析员有着特殊的影响。

根据功能分解理论，工程师能够创建满足他所能遇到的各种需求的设计。下面是功能分解理论的步骤：

1) 工程师确定所构造系统的功能。功能就是系统要做什么，而不是系统如何实现此功能或系统是什么样的。例如，我们不说用户想要洗衣机，而说用户想在具体的时间内，花费比某一指定数量更少的体力劳动能够洗完有一定大小的一堆衣物。

2) 如果功能直接映射到可利用单元（螺母、螺钉、计算机指令），工程师就可以把功能分配到这些单元，设计就实现了。

3) 否则，工程师把功能分解成子功能并重复步骤2和步骤3，直到每个子功能小到能映射到设计的最小单元。工程师要细心把一些设计决策从这些子功能的规格说明中排除出去。而且，每一子功能说明子系统必须做什么，而不是怎么做。

例如，洗衣服功能可能包括这些子功能：从用户那里接受衣物、把衣物返还给用户，从衣物上除去污垢。前两项功能可以分配给洗衣机门；最后一项功能将再细分。注意到最后一项功能还没具体涉及到肥皂或漂洗循环或马达。这些可以描述污垢是如何被祛除的。因此，这是个设计决策。

乍一看，这对工程来说好像是一个相当完美的、系统的方法。如果一个功能对人脑来说太大，以至于不能立即勾画出如何实现它，就需要把它重复分解成小到你足以能处理的子功能。用户所需求的每个主功能可以精确地分配给一个设计元素，保证每个功能都被实现。每个设计元素都可追溯到需求的功能，保证设计不包括多余元素。

意外的收获是：不同的子功能可以分配给不同的工程师。对于一个大型工程，如飞机或操作系统，这种分工是必需的。在大型工程中，分析员或系统工程师的一个主要任务就是区分不同的系统主要子功能以使它们能被分配到不同的设计单元，这样人们就可以在集成之前独立地实现和测试这些设计单元。

一切听起来很好，但直到你试过之后才知道如何。

1.1.2 测试一下

虽然这本书不是关于如何编写程序的，但为了给程序员写出有用的需求文档，我们需要了