HZ BOOKS
软 件 工 程 技 术 丛 书
测试系列

软件测试经验与教训

Lessons Learned in Software Testing

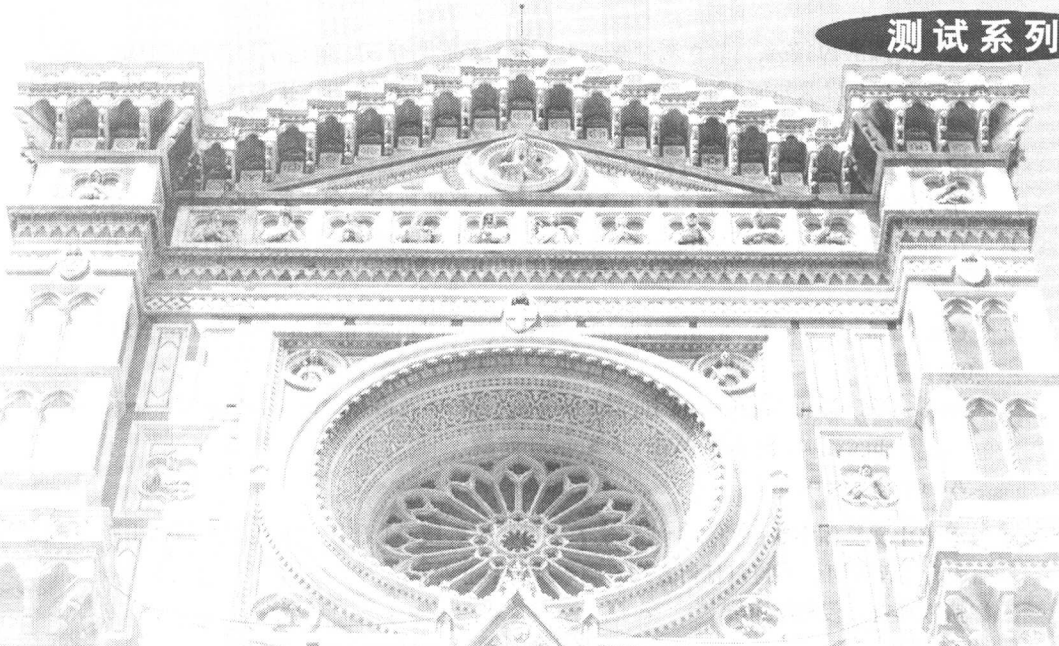
（美） Cem Kaner
James Bach
Bret Pettichord 著

韩柯 等译

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

测试系列



（美） Cem Kaner
James Bach
Bret Pettichord 著
韩柯 等译

软件测试经验与教训

Lessons Learned in Software Testing

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

本书汇总了293条来自软件测试界顶尖专家的经验与建议，阐述了如何做好测试工作、如何管理测试，以及如何澄清有关软件测试的常见误解，读者可直接将这些建议用于自己的测试项目工作中。这些经验中的每一条都是与软件测试有关的一个观点，观点后面是针对运用该测试经验的方法、时机和原因的解释或例子。

本书还提供了有关如何将本书提供的经验有选择性地运用到读者实际项目环境中的建议，在所有关键问题上所积累的经验，以及基于多年的测试经验总结出的有用实践和问题评估方法。

Cem Kaner, James Bach, Bret Pettichord: Lessons Learned in Software Testing (ISBN: 0-471-08112-4).

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

本书版权登记号：图字：01-2002-1057

图书在版编目（CIP）数据

软件测试经验与教训/（美）凯纳（Kaner, C.）等著；韩柯等译. -北京：机械工业出版社，2004.1

（软件工程技术丛书　测试系列）

书名原文：Lessons Learned in Software Testing

ISBN 7-111-12975-X

Ⅰ.软…　Ⅱ.①凯…②韩…　Ⅲ.软件-测试　Ⅳ.TP311.5

中国版本图书馆CIP数据核字（2003）第080767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：杨　文

北京中加印刷有限公司印刷·新华书店北京发行所发行

2004年1月第1版第1次印刷

787mm×1092mm 1/16·16.25印张

印　数：0 001 - 5 000册

定　价：35.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话：(010) 68326294

译者序

应该承认，这是一本很吸引人的书。它的精彩之处在于它使各类软件测试人员，甚至是与测试人员打交道的人，都能得到很好的启发；在于它能够引导读者对自己在实际工作中的得与失做比较全面的思考，重新审视过去很多深信不疑的软件测试规则；在于它直接面对令大家感到困惑的问题，从更实际的角度进行诠释。

作者结合自己丰富的软件测试实践经验，大胆地对软件测试界很多人多年来鼓吹的所谓最佳实践、关键活动、甚至国际标准进行了深刻的反思，令人信服地提出了自己的观点，对一些关键问题做了哲学思考，内容涉及与软件测试有关的各个方面，很适合有一定实际经验的软件测试、项目管理、软件开发、软件工程等方面的工程技术人员阅读。

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、审校和其他辅助工作的还有：原小铃、李津津、王威、屈健、黄惠菊、韩文臣、朱军、杜蔚轩、解冀海、付程、孟海军、耿民、王强等。

译 者

序

请想像自己拿着一瓶50年的陈年波尔多红葡萄酒。这里有一种品味的方法，虽然这并不是惟一的方法，但是有多年饮用波尔多经验的人大都会发现，有些经验有助于更好地享用波尔多。以下就给出几条经验：

经验1：不要直接用瓶子喝。如果没有玻璃杯，也没有其他可用的容器，可把少量波尔多倒在自己的手掌中吸吮。在吸吮的同时，应该嗅一嗅波尔多的芳香，让波尔多在舌头上来回流动，不要一口咽下。

经验2：不要整瓶喝下去。如果喝波尔多是为了解渴，那么还是请把波尔多放下，而喝一大杯水吧。每次饮用少量的波尔多，会使整瓶波尔多带来的享受达到极致。

经验3：不要污染波尔多。如果有人建议尝试一下用橙汁、海水和波尔多勾兑的鸡尾酒，那么请礼貌地拒绝他，带着绚烂的微笑说："可是我想享用的是一杯波尔多。"

经验4：不要独贪波尔多。把波尔多藏起来，就意味着再也不能享受一边与朋友轻松地闲谈，一边品味波尔多的乐趣。波尔多最好还是与喜欢来一杯的朋友共享。请记住，他们可能也会有一瓶波尔多。

读者手中拿的并不是一瓶波尔多，而是本书，这是一本有关软件测试的非常有价值的专著。它通过作者多年经验的酿造，已经至善至纯。波尔多是你味觉感官的朋友，而本书则是你头脑的朋友。我想读者还会发现其他值得注意的差别。我已经体验了本书，并总结出以下经验，希望这些经验能够帮助读者通过本书得到最大收益。

经验1：不要直接用瓶子喝。阅读本书时要带上自己的容器。也就是说，带上自己所有的软件开发和测试经验。如果读者从来没有参加过正式的软件开发工作，则本书对你可能度数太高了。它会使你头晕，在一段时间内浑身无力。如果读者有一定的经验，则可以结合自己项目背景，享用本书的内容。

经验2：不要整瓶喝下去。不要一口气把它读完。读上一两条经验，合上书，想想你对Kaner、Bach和Pettichord的话有何反应。读者会发现，他们把自己的方法叫作"语境驱动"的测试。只有读者才会知道自己工作的背景，必须靠自己确定本书所给出的经验在哪些方面适合自己的具体工作。

经验3：不要污染波尔多。有人要为本书的293条经验加上标题，并制成表。但愿读者没有这种想法。本书的核心是每条经验的解释。还要当心有人会立即尝试把本书内容ISO化或CMM化。我现在可以看到"使用这本书将达到CMM第293级"这样的文章标题。哈！正如作者所说："……我们不相信'最佳实践'，我们相信在一定条件下，一些实践比另一些更有用。"这些话代表了达到软件测试大师级专家境界的思想精华。

经验4：不要独贪波尔多。如果有什么书要和同事一起读的话，本书就是。买上一箱，送给搞测试或管理测试员的人每人一本。一次有选择性地读上几条经验，然后聚在一起边讨论，边喝咖啡、吃午餐，甚至享用波尔多！阅读、反思、享受。来，干杯！

Tim Lister
美国纽约市
大西洋系统协会
2001年8月17日
lister@acm.org

前　言

提出软件工程知识体系（Software Engineering Body of Knowledge，SWEBOK）的目标是为颁发政府执照、规范软件工程师以及开设软件工程大学课程提供一种恰当的基础。SWEBOK文件声称以大部分人的意见为基础，希望这样的文件能够包容这个领域已经积累的知识和智慧（已经积累的经验教训）。

有关探索式测试，SWEBOK只有如下的叙述：

也许，最广泛采用的手段还仍然是专门定制的测试，即依靠测试员的技能和直觉（“探索”测试），依靠测试员在类似工程上积累的经验所导出的测试。虽然人们建议采用更系统化的方法，但是专门定制测试方法对于确定形式化手段不能轻易“捕获”的特殊测试用例还是很有用的（但是只有当这些测试员都是真正专家的时候）。此外还必须指出，这种手段的有效性会有大幅度的变动（SWEBOK 0.95，2001，第5~9页）。

SWEBOK怎样看待这种在测试领域中被最广泛地采用的手段呢？它丝毫没有涉及这种手段的使用方法，只是说应该由真正专家进行探索，建议使用其他方法，认为其他形式化的手段会使有效性更加稳定。

哈！

我们并不打算向读者正式描述反映测试领域大多数人观点的所谓**知识体系**，但是我们确实要大量谈论这个领域中最常见的实践。本书不是要排斥探索式测试，而是要站在使用探索法（以及很多其他方法）在现实条件下达到最佳测试效果的人员立场上，来描述测试应该是怎样的。

欢迎阅读本书

本书是我们实际软件开发经验的总结。我们几个人加起来已经有了50~60年的开发经验（当然这要看怎样计算了），其间，我们既看到了大量出色的工作，也看到了大量不那么出色的工作。

本书并不打算讨论软件工程在更有规则、更受控的环境中怎样运用，而是要讨论我们所要面对的现实世界。

在我们的世界中，软件开发团队常常要在时间很紧张的情况下工作，要在探索该做什么的同时，探索该怎样做。所使用的方法，有时比较正式，有时就不那么正式。这取决于千差万别的实际环境。

我们采用的是软件测试中的**语境驱动法**（context-driven approach）。我们认为在某些环境中很有效的方法，在另外一些环境中就没有效果。我们不谈论最佳实践，而是谈论最适合当前特定环境的实践。本书的最后要讨论语境驱动法，但是从本质上看，语境驱动的

测试要从“谁”、“什么时候”、“哪里”、“为什么”和“如果这样会出现什么”五个方面，来研究测试的“内涵”（手段、工具、策略等）。

我们的目标是将所选择的实践与当前具体环境匹配起来，以取得理想的测试效果。我们不认为通过接管项目，或通过跺着脚告诉项目经理（或执行经理）“真正的专业人士”该怎样管理项目，就能够取得理想的测试效果；不认为通过强迫程序员，或通过奉承他们，就能够取得理想的测试效果；不认为通过填写数以千计的小卡片（或对应的电子记录），或通过在没有必要的过程上浪费其他所有人的时间，就能够干好测试工作。

我们并没有严格的条条框框，也没有理想化的有关测试的灵丹妙药。

这种灵丹妙药根本就不存在！

我们认为好的测试包括要求具有很高技能的技术工作（搜索缺陷）和准确、有说服力的沟通。

技艺高超的搜索毕竟也是一种探索。有无限多的测试工作要做，但是只有少量时间，只能完成其中很小一部分测试。要完成的每个测试，要写的每份文档，要参加的每个会议，都要占用可能会发现关键缺陷的测试时间。面对这种限制，我们要优化测试过程，以便充分利用不断积累的产品及产品市场、产品应用和产品弱点等方面的知识。我们今天所学的知识，会在明天更好的测试中体现出来。即使：

- 产品被相当完备地描述。
- 规格说明准确地反映了需求文档。
- 需求文档准确地表达了产品项目相关人员的实际需要。

（读者是否确实见过具备所有这些条件的项目），我们在测试这样的产品时，仍然会学到很多这种产品的测试方法。具体地说，当发现错误时，我们知道这组程序员会犯怎样的错误。规格说明告诉我们，程序如果编写正确，应该怎样完成功能，但是不会告诉我们预期会出现的错误，也不会说明应该如何设计测试以发现这些错误。对于这件关键工作，我们要从头至尾地在一个项目中逐渐改进，一个项目一个项目地逐渐改进。

不管外界怎么看，只要我们在用我们的头脑进行测试，我们的工作就是探索。

本书的读者对象

本书针对测试软件和管理测试员的所有读者，针对在其软件开发项目中要与测试员打交道的所有读者。

我们主要针对的读者应该已经从事了几年的测试工作，可能最近被提升到管理岗位。我们希望这样的读者能够从本书找出大量与自己的经历一致的内容，从而对我们的经验能够有新的理解，能够找出可以向自己的经理转述的经验，能够很喜欢我们的一些阐述，以至于将我们所说的话放大贴在办公室中的显著位置。读者也许会对我们的至少一句话反应如此强烈，以至于将它贴在自己的飞镖靶盘上。（我们要启发读者的思想，而不只是想让读者

赞同我们的观点。）

刚开始从事测试工作的读者（以及正在申请从事测试工作的读者）不会有很多机会感觉到已经经历过我们所谈论的问题。对于这样的读者，本书可以提供一些早期认识和忠告，对测试员要面临的问题有一个很好的体验。

提示 如果读者对测试绝对是新手，正在找一本书来研究，以便为一次工作面试做准备，那么本书是不合适的。如果这是你能够找到的惟一一本书，那么请仔细研究第3章和第10章中的内容。如果有多本书可以挑选，我们建议阅读《测试计算机软件》（Testing Computer Software）（Kaner等，1993）的前5章。

对于必须与测试员打交道的程序员、项目经理和执行经理，本书会有助于这些人正确期望测试小组的工作。我们希望本书有助于这些读者在不赞同测试小组的政策，或感到他们没有明智地利用自己的时间时，能够评估和讨论对测试小组的意见。

本书包含的内容

多年以来，我们学会了很多有用的实践和很有帮助的问题评估方法。我们得出结论的基础是经验。本书归纳了我们的很多经验，将这些经验组织为接近300个短小、可读性好的经验系列描述。

选择这些经验时，我们遵循了以下一些准则：

- 经验应该是有用的，或有自己的观点。
- 经验应该通过90分钟独立思考的考验。如果是什么人漫不经心地花90分钟就可以提出观点的话，那么这样的经验不值得收入本书。
- 经验必须基于我们的实际经历。我们中至少有一人（最好是三人）曾经成功地运用过我们给出的建议；我们中至少有两人在尝试我们所批判的实践时，遭受过挫折。（请注意：有时我们会根据自己不同的经历，分别得出不同的结论。偶尔读者会发现我们决定提供两种观点，而不是一种。即使只提供一种观点，读者也不能就认为我们三人都完全赞同这种观点。当出现不同意见时，我们三人很可能会服从对该问题有最多经验的一两个人。）
- 经验应该根据我们同事的经历进行调整。通过“测试自动化Los Altos研讨会”、“软件测试经理圆桌协会”、“启发式与探索测试技术研讨会”、“测试自动化Austin研讨会”、“软件测试模式研讨会”、“基于模型的自动化测试研讨会”、“系统有效性管理集团”等几十个软件测试会议，以及不太正式的同级相互培训（例如在Crested Butte举行的一年一度的顾问营），我们收集了详细的经验报告。
- 经验应该简短，切中有害，但又易于理解。
- 经验可以长一些，但是只限于解释如何做，或提供有用的工具。很长的解释和详细背

景信息不适合本书。

- 经验应该是自完备的，读者可以从书中任何地方开始阅读。
- 经验合在一起，应该使读者对我们怎样做测试以及考虑测试，能有一定的认识。

本书不包含的内容

本书并不是软件测试的综合指南。

本书并不是永远都正确的经验汇集。这些经验是我们自己的经验，依据的是我们的经历。我们相信这些经验可以广泛运用，但是有些在我们的经历中被证明是很有用、很重要的经验，可能并不适用于读者。读者要自己进行判断。应该指出，本书适用性的一个具体限制因素是，我们更多从事的是面向大众市场的软件开发，以及以具体合同为基础的软件开发，自用软件开发的经验较少。我们在开发生命关键软件和嵌入式软件方面的经验也不多。

本书并不是最佳实践的汇集。事实上，我们并不相信“最佳实践”。我们相信在一定环境中，有些实践比另外一些更有用。我们注意到有很多东西被作为最佳实践，不加批判地向并不适合的场合推广（并应用）。

如何使用本书

本书的结构设计旨在便于读者浏览或有选择性地阅读，而不是从头读到尾。有时（我们希望）读者会发现“金块”，发现对自己非常有吸引力的思想。我们建议读者不要不加批判地运用这些思想。（我们的经验并不是最佳实践。）相反，我们强烈希望读者能够评估这些经验对于自己实际环境的适合性。

以下是一些有助于读者进行评估的问题：

- 在什么条件下，自己的公司运用这个经验会收到效果？
- 在什么条件下，运用这个经验不会收到效果？
- 你认识什么人以前尝试过类似的经验吗？效果怎样？为什么会有这种结果？你当前的项目与那个人的项目有哪些不同？这些差别重要吗？
- 通过尝试运用这个经验，谁最有可能受益？
- 通过尝试运用这个经验，谁的利益最有可能受到影响？
- 通过尝试运用这个经验，你可以学到什么？
- 尝试任何新东西都会增加风险。通过试点，即不是投入太多地尝试该经验，或在低风险环境中研究该经验运用于自己的公司有多大作用，这有意义吗？在公司中引入试点，对当前项目（或对其服务）尝试该经验是否可行？
- 你怎么知道运用该经验是否有作用？如果有作用，你怎么知道之所以取得成功，更多的是因为该经验的内在价值，而不是靠尝试该思想时的最初热情？如果继续运用该经验，这种成功会持续下去吗？

- 作为尝试运用该经验的结果，对你会出现什么最好和最坏的事情？
- 会对其他项目相关人员，例如一个用户或开发团队的其他成员，出现什么最好和最坏的事情？
- 如果公司中有一位关键人物不赞同你要尝试的经验会出现什么情况？你怎样克服他们的反对，并向他们推销要尝试的经验？

我们希望本书能够引出不同意见和争论

本书显示出我们观点和其他观点之间的鲜明对比。我们认为这种对比有助于引起在测试领域展开本来就应该展开的论战。我们认为在软件测试或作为整体的软件工程中，还没有一种已经被普遍接受的范型，这是为什么我们不赞同通过标准化一种知识体来推动政府颁发执照、规范软件工程师的原因。根据我们的经验，对于要推行的最佳方法，还有一些有显著不同的有说服力的观点。

我们要非常明确地提出这些观点。我们经常对我们非常尊重的人们的工作提出批评。在很多情况下，我们所批评的工作是由我们很好的朋友完成的。不要把对某种思想的批判，错误地当作对该思想的支持者或提出者个人的攻击。

我们认为通过直接比较、对比这些观点，会使整个测试领域受益。对于测试领域来说，重要的是对我们的方法展开讨论。我们倡导进一步发展每一个主要方法范畴内的实践，最终我们都会认识到不同方法最适用的实际环境，到那时才会出现百花齐放的局面。

一些词汇的解释

以下是本书用到的一些关键词及其在本书中的含义：

我们。作者。

你们。读者。

缺陷（fault）是程序实现或设计中的错误、失误。

错误（error）在本书中就是缺陷。

失效（failure）是程序的错误行为，由包含缺陷的程序产生。

失效在一定条件（condition）下发生。例如，当程序试图进行除数是0的计算时，就会出现崩溃。在这种情况下，缺陷是允许被0除的代码，失效是崩溃。但是只有在除法中的一个关键变量取值为0时，才会看到失效。那个变量取值为0是失效发生必须满足的关键条件。

征兆（symptom）与失效类似，但是不那么严重。例如，如果程序存在内存泄漏，则程序在给出内存耗尽错误消息之前很久就开始逐渐慢下来。这种降速就是不能直接看得到的内部问题（内存短缺）的征兆。

程序错误（bug）是含义很广的词，可以指任何软件问题。报告程序错误的人描述的可能是缺陷、失效或使程序对项目相关人员的价值减弱的局限性。

如果把质量（quality）定义为对某人的价值（Weinberg，1998，2.i.），那么**错误报告**（bug report）就是一种陈述，说明有人认为由于被描述为程序错误的东西使该产品的价值降低。

过失（defect）有一种法律指责含义，表示“产品肯定出现了问题”。有些公司不允许这样的词出现在错误报告或与程序错误有关的备忘录中。

有些公司喜欢用**异常**（anomaly）、**问题**（problem、issue）替代程序错误。

当报告一个程序错误之后，程序员（或“变更控制委员会”）会将其更正，或决定不更正。他们会把这个程序错误标识为**已解决**（resolved）（已更正、推迟、不可重现、设计如此等）。

黑盒测试（black box testing）。通过向程序送入输入，并检查其输出，来测试程序的外部行为。在一般的黑盒测试中，测试员没有代码内部知识，但是更熟悉显式和隐式产品需求，例如使用程序的方式，有可能输入程序的数据类型，与软件试图解决或帮助解决的问题有关的任何管理问题，以及软件将使用的硬件和软件环境。

行为测试（behavioral testing）。测试程序的外部行为，类似黑盒测试，但是尽可能利用测试员能够得到并与测试有关的程序内部知识。

功能测试（functional testing）。黑盒或行为测试。

白盒或玻璃盒测试（white box或glass box testing）。利用程序内部知识的测试。

结构测试（structural testing）。关注程序内部结构的白盒测试，例如从一个决策或行动到下一个决策或行动的控制流。

冒烟测试或版本确认测试（smoke testing或build verification testing）。用于软件新版本的标准测试包。这种测试检查版本基本功能是否稳定，或检查是否遗漏关键功能，是否满足基本要求。如果版本没有通过这些测试，则不再进行进一步的测试，而是继续测试老版本，或等待新版本。

项目经理（project manager）。负责按时在预算限度内交付恰当产品的人。有些公司将上述工作划分给程序经理和开发经理完成。

最大整数（MaxInt）。用户平台或程序员开发语言可以使用的最大整数。大于最大整数的数不能作为整数存储。

客户（client）。公司要满足其利益要求的人。在一定程度上可能包括与项目有关的所有人，以及产品的最终用户。

致　谢

如果没有很多人的支持和帮助，本书是不可能出版的。我们要感谢Lenore Bach、Jon Bach、Becky Fiedler、Leslie Smart和Zach Pettichord，感谢他们在此书出版工作中对我们的支持、理解和帮助。我们要感谢Pat McGee在关键时刻给予我们的帮助。

评阅人对初稿提出了仔细、深刻的反馈意见，我们受益不小。我们在书中补充了评阅人给出的一些例子和观点。我们要感谢Stale Amland、Rex Black、Jeffrey Bleiberg、Hans Buwalda、Ross Collard、Lisa Crispin、Chris DeNardis、Marge Farrell、Dorothy Graham、Erick Griffin、Rocky Grober、Sam Guckenheimer、George Hamblen、Elisabeth Hendrickson、Doug Hoffman、Kathy Iberle、Bob Johnson、Karen Johnson、Ginny Kaner、Barton Layne、Pat McGee、Fran McKain、Pat McQuaid、Brian Marick、Alan Myrvold、Hung Nguyen、Noel Nyman、Erik Petersen、Johanna Rothman、Jane Stepak、Melora Svoboda、Mary Romero Sweeney、Paul Szymkowiak、Andy Tinkham、Steve Tolman和Tamar Yaron。

在“测试自动化Los Altos研讨会”、“启发式与探索测试技术研讨会”、“软件测试经理圆桌协会”、“测试自动化Austin研讨会”、“软件测试模式研讨会”，以及很多其他研讨会、会议、课堂上和工作中，本书得益于与很多热心探索更好软件测试方法的人的讨论。我们在这里一并向他们致谢。

软件工程技术丛书书目

丛书编号	英文书名	中文书名	作　者
1	Object-Oriented and Classical Software Engineering, 5E	面向对象与传统软件工程(原书第5版)	Stephen R. Schach
1	Object-Oriented Software Engineering	面向对象的软件工程	Ian Sommerville
1	Software Engineering: A Practitioner's Approach, 5E	软件工程:实践者的研究方法(原书第5版)	Roger S. Pressman
1	Software Engineering, 6E	软件工程(原书第6版)	Ian Sommerville
1	Software Engineering with Java	软件工程:Java语言实现	Stephen R. Schach
1	Project-Based Software Engineering:An Object-Oriented Approach	基于项目的软件工程:面向对象研究方法	Evelyn Stiller
1.1.1	Software Process Improvement: Practical Guidelines for Business Success	软件过程改进	Sami Zahran
1.1.1	Making Process Improvement Work	软件过程改进简明实践	Neil S. Potter
1.1.2	The Road to the Unified Software Development Process	统一软件开发过程之路	Ivar Jacobson
1.1.2	The Unified Software Development Process	统一软件开发过程	Jacobson/Booch/Rumbaugh
1.1.2	The Rational Unified Process: An Introduction , 2E	Rational统一过程引论(原书第2版)	Philippe Kruchten
1.1.2	UML and The Unified Process: Practical Object-Oriented Analysis & Design	UML和统一过程: 实用面向对象的分析和设计	Jim Arlow
1.1.3	Managing Global Software Projects: How toLead Geographically Distributed Teams,Manage Processes and Use Quality Models	全球化软件项目管理	Gopalaswamy Ramesh
1.1.3	Software Project Management: A Unified Framework	软件项目管理:一个统一的框架	Walker Royce
1.1.3	How to Run Successful Projects III: The Silver Bullet	成功的软件项目管理:银弹方案(原书第3版)	Fergus O'Connell
1.1.3	Successful Software Development, 2E	成功的软件开发(原书第2版)	Scott E. Donaldson
1.1.3	Six Sigma Software Development	6σ软件开发	Christine B. Tayntor
1.1.3	IT Project Management: On Track from Start to Finish	IT项目管理: 从开始到结束的历程	Joseph Phillips
1.1.3	Successful IT Project Delivery: Learning the lessons of Project Failure	IT项目成功交付的秘诀	David Yardley
1.1.3	Software Project Management, 3E	软件项目管理(原书第3版)	Bob Hughes
1.1.3	Architecture-Centric Software Project Management	软件项目管理实用指南: 以体系结构为中心	Daniel J. Paulish
1.1.4	Handbook of Software Quality Assurance, 3E	软件质量保证(原书第3版)	Gordon G. Schulmeyer
1.1.4	Software Reliability Engineering	软件可靠性工程	John Musa
1.1.4	Implementing ISO 9001:2000 The Journey from Conformance to Performance	ISO 9001:2000 实施指南	Tom Tairnubg
1.1.4	CMMI Distilled: A Practical Introduction to Integrated Process Improvement	CMMI精粹:集成化过程改进实用导论	Dennis M. Ahern
1.1.4	CMM Implementation Guide	CMM实施与软件过程改进	Kim Caputo
1.1.4	Implementing the Capability Maturity Model	CMM实施指南	James R.Persse
1.1.4	Object-Oriented Defect Management of Software	面向对象的软件缺限管理	Houman Younessi
1.1.4	Metrics and Models in Software Quality Engineering	软件质量工程:度量与模型	Stephen H. Kan
1.1.4	Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software	软件性能工程	Connie U. Smith

丛书编号	英文书名	中文书名	作　者
1.1.4	Solid Software	Solid Software	Shari Pfleeger
1.1.4	Peer Reviews in Software: A Practical Guide	同级评审	Karl E. Wiegers
1.1.5	Practical Software Measurement	实用软件度量	John McGarry
1.1.5	Software Metrics: A Rigorous and Practical Approach, 2E	软件度量(原书第2版)	Norman E. Fenton
1.1.5	Software Assessments, Benchmarks, and Best Practices	软件评估、基准测试与最佳实践	Capers Jones
1.2.1	The Object Primer: The Application Developer's Guide to Object Orientation and the UML, 2E	面向对象软件开发教程(原书第2版)	Scott W. Ambler
1.2.1	UML and C++: A Practical Guide to Object-Oriented Development, 2E	C++面向对象开发(原书第2版)	Richard C.Lee
1.2.1	Object-Oriented Methods: Principles & Practices, 3E	面向对象方法:原理与实践(原书第3版)	Ian Graham
1.2.1	Principles of Object-Oriented Software Development, 2E	面向对象软件开发原理(原书第2版)	Anton Eliëns
1.2.1	Object Solutions: Managing the Object-Oriented Project	面向对象项目的解决方案	Grady Booch
1.2.1	An Introduction To Object-Oriented Programming, 3E	面向对象程序设计导引(原书第3版)	Timothy Budd
1.2.1	The Object Advantage: Business Process Reengineering with Object Technology	对象优势:采用对象技术的业务过程再工程	Ivar Jacobson
1.2.1	The Unified Modeling Language User Guide	UML用户指南	Booch/Rumbaugh/Jacobson
1.2.1	The Unified Modeling Language Reference Manual	UML参考手册	Rumbaugh/Jacobson/Booch
1.2.1	Applying UML and patterns: An Introduction to Object-Oriented Analysis and Design, 1E	UML和模式应用(原书第1版)	Craig Larman
1.2.1	Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process , 2E	UML与模式应用(原书第2版)	Craig Larman
1.2.1	Object-Oriented Analysis and Design with Applications, 2E	面向对象分析与设计(原书第2版)	Grady Booch
1.2.2	Software Reuse: Architecture, Process and Organization for Business Success	软件复用: 结构、过程和组成	Ivar Jacobson
1.2.2	Software Reuse Techniques: Adding Reuse to the Systems Development Process	软件复用技术: 在系统开发过程中考虑复用	Carma McClure
1.2.2	Practical Software Reuse: Strategies for Introducing Reuse Concepts in Your Organization	实用软件复用方法	Donald J. Reifer
1.2.2	Large-Scale Component-Based Development	大规模基于构件的软件开发	Alan W. Brown
1.2.2	Component-based Product Line Engineering with UML	基于构件的产品线工程:UML方法	Colin Atkinson
1.4.1	Object-Oriented Analysis & Design	面向对象的分析与设计	Andrew Haigh
1.4.1	Analysis Patterns: Reusable Object Models	分析模式: 可复用的对象模型	Martin Fowler
1.4.1	Requirements Analysis and System Design: Developing Information Systems with UML	需求分析与系统设计	Leeszek A. Maciaszek
1.4.1	Systems Analysis and Design in a Changing World	系统分析与设计	John W. Satzinger
1.4.1	Advanced Use Case Modeling, Vol I: Software Systems	高级用例建模 卷1: 软件系统	Frank Armour
1.4.1	Requirements Engineering: A Good Practice Guide	需求工程	Ian Sommerville
1.4.1	Software Requirements and Estimation	软件需求与估算	Swapna Kishore
1.4.1	Effective Requirements Practices	有效需求实践	Ralph R.Young
1.4.1	Applying Use Cases: A Practical Guide, 2E	用例分析技术(原书第2版)	Geri Schneider
1.4.1	Managing Software Requirements	软件需求管理:统一方法	Dean Leffingwell
1.4.1	Writing Effective Use Cases	编写有效用例	Alistair Cockburn
1.4.1	Problem Frames Analyzing and Structuring Software Development Problems	Problem Frames Analyzing and Structuring Software Development Problems	Michael Jackson

丛书编号	英文书名	中文书名	作　者
1.4.2	Object-Oriented Software Construction, 2E	面向对象的软件结构(原书第2版)	Bertrand Meyer
1.4.2	Pattern-Oriented Software Architecture, Vol I: A System of Patterns	面向模式的软件体系结构 卷1:模式系统	Frank Buschmann
1.4.2	Pattern-Oriented Software Architecture, Vol II: Patterns for Concurrent and Networked Objects	面向模式的软件体系结构 卷2:用于并发和网络化对象的模式	Douglas Schmidt
1.4.2	Server Component Patterns	面向模式的软件体系结构 卷3:服务器组件模式	Markus Volter
1.4.2	DesignPatterns: Elements of Reusable Object-Oriented Software	设计模式:可复用面向对象软件的基础	Gamma/Helm/Johnson /Vlissides
1.4.2	Patterns of Enterprise Application Architecture	企业级应用体系结构模式	Martin Fowler
1.4.2	AntiPatterns and Patterns in Software Configuration Management	软件配置管理中的模式与反模式	William J. Brown
1.4.2	Software for Use: A Practical Guide to The Models and Methods of Usage-Centered Design	面向使用的软件设计	Larry L. Constantine
1.4.2	Software Architecture: Organizational Principles and Patterns	软件架构: 组织原则与模式	David Dikel
1.4.2	The UML Profile for Framework Architectures	框架体系结构的UML档案	Marcus Fontoura
1.4.2	Software Architect's Profession: An Introduction	软件架构师入门必读	Marc Sewell
1.4.2	Business Modeling with UML: Business Patterns at work	UML业务建模: 实用业务模式	Hans-Erik Eriksson
1.4.3	Software Fabrication:Automating Application Development	软件构造:自动化软件开发	Jack Greenfield
1.4.3	Building J2EE Applications With The Rational Unified Process	用RUP构建J2EE 应用程序	Peter Eeles
1.4.3	Programming from Specifications	从规范出发的程序设计	Carroll Morgan
1.4.4	Testing IT: An Off-the-Shelf SoftwareTesting Process Handbook	实用软件测试过程之路	John Watkins
1.4.4	Lessons Learned in Software Testing	软件测试经验与教训	Cem Kaner
1.4.4	Testing Computer Software: The Bestselling Software Testing Book Of All Time, 2E	计算机软件测试(原书第2版)	Cem Kaner
1.4.4	Software Testing in the Real World: Improving the Process	软件测试过程改进	Edward Kit
1.4.4	Effective Methods for Software Testing, 2E	有效的软件测试方法(原书第2版)	William E. Perry
1.4.4	Beta Testing for Better Software	软件Beta测试	Michael R. Fine
1.4.4	A Practical Guide to Testing Object-Oriented Software	面向对象的软件测试	John D. McGregor
1.4.4	Managing the Testing Process, 2E	测试过程管理(原书第2版)	Rex Black
1.4.4	Software Testing: A Craftsman's Approach, 2E	软件测试(原书第2版)	Paul C. Jorgensen
1.4.4	Just Enough Software Test Automation	软件测试自动化	Daniel J. Mosley
1.4.4	The Craft of Software Testing: Subsystem Testing Including Object-Based and Object-Oriented Testing	软件测试实用技术	Brian Marick
1.5	JAVA Tools for Extreme Programming: Mastering Open Source Tools, Including Ant,JUnit, and Cactus	极限编程的JAVA工具	Richard Hightower
1.5	Agile Software Development Ecosystems	敏捷软件开发生态系统	Tom DeMarco
1.5	Agile Modeling: Effective Practices For eXtreme Programming and The Unified Process	敏捷建模:极限编程和统一过程的有效实践	Scott W. Ambler
1.5	Model-Driven Development: Automating Component Design, Implementation, and Assembly	模式驱动的软件开发	David Frankel
1.5	A Practical Guide to Feature-Driven Development	特征驱动开发方法:原理与实践	Steve R: Palmer

软件工程技术丛书结构图