软 件 工 程 技 术 丛 书

HZ BOOKS

质量管理系列

Mc Graw Hill Education

更可靠的软件，更快速的开发与测试

软件可靠性工程

Software Reliability Engineering

（美） John D. Musa 著

韩 柯 译

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

TP311.5
46

质量管理系列

软件可靠性工程

Software Reliability Engineering

（美） John D. Musa 著
韩 柯 译

北方工业大学图书馆
00532766

机械工业出版社
China Machine Press

本书是软件可靠性工程领域的权威著作，全面论述了该领域的相关内容，目标是使读者快速理解什么是软件可靠性工程，如何将软件可靠性工程应用于软件开发和测试中，并帮助读者处理紧张的测试时间与软件可靠性之间的矛盾。本书注重实践，提供已经在很多应用系统中成功使用过的方法，并避免使用在实际使用中尚未证实的想法。本书还提供了相关的辅助材料和网站资源，便于读者的学习。

本书适合计算机软件工程及相关专业的学生使用，也可作为软件工程开发人员和管理人员的参考用书。

John D. Musa: Software Reliability Engineering (ISBN: 0-07-913271-5).

Copyright © 1999 by The McGraw-Hill Companies, Inc.

Original language published by The McGraw-Hill Companies, Inc. All rights reserved. No part of this publication may be reproduced or distributed in any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Simplified Chinese translation edition jointly published by McGraw-Hill Education (Asia) Co. and China Machine Press.

本书中文简体字翻译版由机械工业出版社和美国麦格劳－希尔教育（亚洲）出版公司合作出版。未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有McGraw-Hill公司防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书版权登记号：图字：01-2002-2044

图书在版编目（CIP）数据

软件可靠性工程/（美）玛萨（Musa, J. D.）著；韩柯译. －北京：机械工业出版社，2003.5

（软件工程技术丛书. 质量管理系列）

书名原文：Software Reliability Engineering

ISBN 7-111-11905-3

I. 软… II. ①玛… ②韩… III. 软件可靠性－软件工程 IV. TP311.5

中国版本图书馆CIP数据核字（2003）第023802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

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2003年5月第1版第1次印刷

787mm × 1092mm 1/16 · 17.25 印张

印数：0 001-5000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

译者序

软件可靠性工程从使用的观点验证和提高软件可靠性，而不考虑软件系统的内部结构和实现过程，力求以最小的代价，在尽可能短的时间内，使软件系统达到实际使用所需要的可靠水平。因而软件可靠性工程解决的是软件系统使用中的可靠性问题，具有很强的实践性和很高的实用价值。

软件可靠性工程借用了大量硬件可靠性工程的基本概念，而软件都是在硬件平台支撑下工作的，与硬件联系十分紧密。因而软件可靠性工程特别适合软件密集的大型复杂系统和嵌入式系统，能够很好地与硬件可靠性工程结合在一起实施，从而节省测试成本。

本书是关于软件可靠性工程的权威性著作，作者John D. Musa是世界公认的软件可靠性工程创始人之一。本书内容组织由浅入深，使用大量篇幅解答常见问题，列举大量实例，没有更多地陷入理论论述，而是尽可能突出软件可靠性工程的实用性，特别适合软件工程实际工作者，尤其是工程师和项目经理阅读。本书也是计算机应用等相关专业在校本科生、研究生很好的参考用书。

在翻译过程中，除了对原文个别明显错误进行了相应更正外，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

2003年初

前　言

本书目标是最有效地使读者理解什么是软件可靠性工程，以及如何将软件可靠性工程应用于软件开发和测试。作者的目标是帮助读者处理紧张的测试时间与软件可靠性之间的矛盾（如果测试时间不紧张，还可利用节省下的时间从事其他工作）。软件可靠性工程是一种可以提高个人竞争力的技能，不管读者是开发还是使用基于软件的系统，抑或是正在学习软件使用和开发的大学学生。本书强调的是实践，提供已经在很多应用系统中成功使用的方法，并避免使用尚未在实际使用中经充分证实的思想。

读者会发现作者特别关注测试，但这种关注的角度很广。例如，作者期望测试人员能够参加系统工程团队，并直接与基于软件产品的用户交流。此外，作者设想很多其他软件开发人员会参与到测试工作中来，因此必须更广泛地理解测试。

如果读者是软件测试人员、软件开发人员、系统工程师、系统构架师、质量保证工程师、可靠性工程师或包含软件的项目开发经理，当然还包括正在准备担任上述一种或多种角色的学生，则本书对于这样的读者特别有用。编写本书，就是要使它成为一种课程的学习教材，开始实行软件可靠性工程的案头必备读物，以及解决实际工作中具体问题的参考指南。因此，本书特别强调简化材料，并以易于学习的方式进行组织。本书材料的组织和表述基于作者7年来在这个主题上的教学经验。作者曾经向不同机构中测试各种软件应用系统开发的几千名参与者进行演讲，指导他们学习软件可靠性工程。作者特别注意步步深入地描述软件可靠性工程，并将有关内容以表格的形式组织，并强调这种层次性，便于读者查找每一步的细节。最后，作者还提供与主题有关的术语索引。

第1~6章的核心内容（即除了常见问题、特殊情况和背景以外的内容）只包括在实践中会遇到的一般情况所需的材料。第1章介绍实践中要用到的软件可靠性工程过程。本书的章节组织结构反映了这种过程，第2~6章都涉及一种主要活动，通过统一的例子Fone Follower全面地解释整个过程。这个例子取材于实际项目，但是为了便于学习，作者删除了专门的数据，简化了材料。

每一章都补充提供几节内容，即特殊情况、常见问题和背景。特殊情况部分提供通常只适用于一定项目和系统的技术。常见问题部分是作者根据教授几千名参与者的教学经验和各种机构的咨询经验归纳的大约350个问题和解答。这反映了参与不同项目开发的背景、学习方法和观点。读者会发现其中有些观点也所映了自己的观点。因此，这些常见问题有助于更好地理解本书主题。教师可以将其中的很多问题，以及很多章节末尾给出的思考题作为学生练习。背景部分给出的是能够丰富读者理解该章主题，但又不是实践所必需的补充信息。例如，它提供了证明或支持各种实践活动的理论或说明。补充材料的组织顺序采用自然形式，从在实践中有时会遇到的问题，到对实践的说明，到实践背后的原理，但这些材料并不是必须理解的。

常见问题和背景部分都可能涉及核心部分已经讨论过的问题，但以不同的视点和深度讨论。作者曾经考虑将这些材料熔入相应核心部分，但经过仔细考虑还是决定放弃。大多数实际工作的经验表明，核心部分要尽可能简短，以便迅速掌握基本内容，所有补充材料都

试读结束：需要全本请在线购买：　www.ertongbook.com

应该与核心部分分开。

第7章讨论读者如何在自己的机构中部署软件可靠性工程。丰富读者理解软件可靠性模型的背景材料相当多，因此在第8章中单独介绍。

附录A归纳了软件可靠性工程过程的步骤，如果读者是第一次利用软件可靠性工程，建议读者放在案头经常翻翻，对照检查。作者进行教学时，常常采用专题小组研讨会的方式，每个专题小组的参加人员按照产品线进行组织。讲授完每一章后，每个小组都要讨论与这一章内容有关的材料，并讨论将所学的知识用于具体项目。大学学生可将之作为整体课程的一部分来研究样本软件工程项目，而专题小组可以在实践中很好地与之结合。附录B提供作者用来指导专题小组的模板。这种模板对于项目组部署软件可靠性工程很有帮助，尤其他们采用的是自学方式。

附录C是术语表，附录D归纳的是读者可能需要的公式。附录E列出软件可靠性工程和测试功能，有助于读者使用软件工具。

附录F介绍如何使用CASRE（计算机辅助软件可靠性评估）工具（Nikora 1994）的使用方法。虽然并不是惟一的软件可靠性估计工具，但本书选择这个工具是因为它方便的图形用户界面，而且通过《软件可靠性工程手册》(Handbook of Software Reliability Engineering)（Lyu 1996）所附带的CD ROM很容易得到。CASRE由图形界面和SMERFS（软件统计建模与可靠性功能评估）软件可靠性估计程序组成（Farr and Smith 1992）。CASRE开发人员计划推出经过改进的用户界面。还可以预期推出称为SMERFS Cubed的新版本（SMERFS 3），这个新版本将具有自己的图形界面。这两个新程序都有望在因特网上发布（请参阅下面提及的软件可靠性工程网站）。

有些章节包含思考题，帮助理解所提供材料，附录G给出了这些问题的答案。附录H给出了一些参考论文和软件可靠性工程用户出版的文章。这些论文作为应用系统模型可能会对读者有帮助，特别是那些与读者应用系统类似的系统。

大学学生、研究人员和其他希望深入探索软件可靠性工程理论的人士会发现Musa, Iannino and Okumoto（1987）是很好的参考文献。《IEEE软件工程》会刊和《IEEE可靠性》会刊常常发表这个领域的论文。

IEEE软件可靠性工程技术委员会作为IEEE计算机协会软件工程技术委员会的分支机构，被公认为是这个领域的专业领导机构（网站是http://www.tcse.org）。除了其他活动之外，这个委员会还出版通讯，并资助一年一度的软件可靠性工程国际研讨会。其他有关专业组织还包括IEEE可靠性学会、美国质量控制协会和欧洲的组织ENCRESS（Web站点是http://www.csr.ncl.ac.uk/clubs/encress.html）。因特网上还有电子公告板（可通过vishwa@hac2arpa.hac.com订阅，通过sw-rel@igate1.hac.com发布消息）。

作者个人维护一个定期更新的软件可靠性工程网站（http://members.aol.com/JohnDMusa/），提供作者根据这本书讲授的课程信息。在这门课程中，读者可以把软件可靠性工程应用于自己的工作中，并得到指导和反馈。这个网站还包括一般概述、针对管理人员的专门内容、进行软件可靠性工程实践的作者所发表的文章、每月一题（带答案）、有关软件可靠性估计程序的信息、与其他网站的链接，以及其他资源。

John D. Musa

目　录

第1章
软件可靠性工程概论

在学习本章之前一定要先阅读本书的前言，了解本书的所有章节是如何组织的。

软件开发具有一种或多种风险：

1. 发布产品的不可靠性。
2. 开发进度落后。
3. 开发费用超出预算。

这些情况可能造成市场份额减小或收益减少，因此软件开发人员和测试人员所承受的压力经常是非常巨大的。

为了解决这个问题，人们将注意力主要放在开发和测试的技术工作方面，并构建了很多支持过程的工具。研究人员也重点研究软件开发和测试理论，以及它所涉及的许多难题。然而，在如何对基于软件的产品可靠性的工程方面，所投入的注意力太少了。工程软件可靠性是指以这样一种方法来开发一种产品，使得这种产品能够在合适的时间，以可以接受的成本和令人满意的可靠性投放“市场”。请注意，“市场”两个字是加了引号的，这是为了表示一种比商业产品市场更为广泛的概念。即使是为军事目的或为政府开发的产品也有市场问题，如果产品推出时间太迟、价格太高或者很不可靠，那么用户完全可以选择其他的产品来替代。

开发和测试的传统观点，没有提供足够的能力达到这个工程目标。软件可靠性工程要求一个更为广泛、更具活力的观点。例如，软件可靠性工程已经表明，最有效率的测试包括所有那些出现在整个产品生存周期和与系统工程和系统设计任务交互的工作。因此，在面对用户的需要时，软件可靠性工程使测试人员能够处于领导地位。这包括系统工程师、系统构架师、潜在用户、经理们（Musa，1996c）以及作为合作者的开发人员（Musa，1996a，1996b，1997c，1997d，1997g；Musa and Widmaier 1996）。

软件可靠性的标准定义（Musa，Iannino and Okumoto，1987），是在一段特定的自然单元或时间间隔内，无失效运行的概率。尽管失效的机制可能是不同的，不过这个定义与硬件可靠性的定义是相容的。这种相容性，使这种可靠性定义能够用于既包括软件又包括硬件的系统。

前面提到的产品特性，即可靠性、开发时间和开发费用，它们都是一个更为一般的特性的属性，即产品质量。产品质量是这些特性的一个合适的平衡点。得到一个好的平衡点，意味着必须为这三个特性找到定量的目标，并且在开发过程中进行度量。

当然，各个项目组已经能够为产品的交付时间和开发费用给出定量目标并对其进行度量。但是直到现在，仍然没有为基于软件的系统可靠性做同样的工作。从19世纪40年代开始，对硬件系统的可靠性可以给出定量目标并度量。但是，这样的纯硬件系统几乎已经消失殆尽，因此需要发展软件可靠性工程。

1.1 什么是软件可靠性工程，它是如何帮助开发和测试的

软件可靠性工程是惟一一种标准化的，被证明是最好的一种实践方法，使得测试者和开发者可以同时实现：

1. 确保产品的可靠性达到用户要求。
2. 加快产品上市的速度。
3. 降低产品的成本。
4. 提高用户满意度，降低用户不满意的风险。
5. 提高生产率。

软件可靠性工程可以应用于任何基于软件的产品的任一版本，可以开始于任一版本周期的开始。因此，老产品也可以很容易地处理。使用基于软件这样的描述是为了强调纯软件系统是不存在的，因此在分析中必须说明硬件部分的问题。在对任何产品的测试应用软件可靠性工程之前，必须首先对单元和模块进行测试（或用其他方法验证），然后把它们集成为可以执行的完整功能。

软件可靠性工程之所以有效，在于它运用了两个基本的思想：第一，通过定量描述产品的使用方式，可以更有效地开发产品的功能并且使用这些信息，以便：

1. 将资源精确地集中到最常用和最关键的功能上。
2. 使测试工作真实地反映实际条件。

关键是指它的成功执行会具有很大的附加价值，而失败会产生很大的负面影响。这种价值和影响可能关系到人的生命、财产或系统的工作能力。

第二，软件可靠性工程平衡用户对可靠性、开发时间和开发费用的需求，从而更加有效。为此，软件可靠性工程要像对开发时间和开发费用设置定量目标那样，对可靠性也设置定量目标，要制定策略来达到这些目标。最后，软件可靠性工程在测试过程中跟踪产品的可靠性，并用来作为产品是否可以发布的标准。通过软件可靠性工程，你可以交付“正好合适”的可靠性的产品，并且既避免了不必要的资金和时间成本，又避免了发生由不够可靠的产品导致的用户不满和问题。

软件可靠性工程建立在牢固的理论基础之上（Musa，Iannino and Okumoto，1987），这些理论包括操作剖面、随机过程软件可靠性模型、统计估计和顺序采样理论。从1973年起，软件小组就一直在实践着软件可靠性工程（Musa and Iannino，1991b）。在AT&T公司，从1991年5月以来，软件可靠性工程就一直是一项最佳当前实践（BCP）（Donnelly，Everett，Musa and Wilson，1996）。被选为AT&T的最佳当前实践是相当重要的，这是因为它使用了非常严格的标准。首先，必须在几个（通常是至少8到10个）项目中使用所建议的实践方法，得到以财务术语表示的效益与费用的比率。然后详细描述对这种实践方法以及在项目中如何使用，同时给出采用这种实践的商业案例。由有经验的第三或第四级的经理组成的委员会，将对此描述及案例进行长时间的仔细审查。审查一般要持续几个月，这期间由第一级的软件经理和高级软件开发人员进行详细的评审。对软件可靠性工程的最佳当前实践建议的审查，要由70个以上这样的人员参加，在最终审查之前的意见收集阶段，参与人员会超过100人。即使到那时，这个软件可靠性的最佳当前

试读结束：需要全本请在线购买： www.ertongbook.com

实践还只是1991年30个建议中获得通过的仅有的5个中的1个。

此外，美国航天航空协会1993年通过将软件可靠性工程作为一项标准。这对航空业产生了重大影响（AIAA，1992）。1996年，《软件可靠性工程手册》出版，进一步证实了这个领域的重要性（Lyu，1996）。IEEE在开发软件可靠性工程的标准方面也一直非常活跃。

曾经使用过软件可靠性工程的机构，包括阿尔卡特、AT&T、Bellcore、CNES（法国）、ENEA（意大利）、爱立信、法国电信、惠普、日立、IBM、NASA的发动机推进实验室、NASA的航天飞机计划、洛克希德·马丁、朗讯、微软、Mitre、摩托罗拉、北电、北卡罗来州立大学、Raytheon、绅宝军用飞机（瑞典）、Tandem 计算机、美国空军、美国海军陆战队等。本书附录H中给出了一些由软件可靠性工程的使用者撰写的描述其使用经验的论文。

Tierney（1997）于1997年下半年发表了一份调查报告，报告表明微软公司在其50%的软件开发小组中使用软件可靠性工程，其中包括像Windows NT和Word这样的项目。得到的益处包括测试覆盖率的提高、所需测试量的压缩、有助于建立定版评判准则的有用指标和规格说明评审的改进。

很多年来，AT&T公司网络和计算服务部的运行技术中心一直将软件可靠性工程作为标准软件开发过程的一部分。这个过程现在正在进行ISO认证。运行技术中心是AT&T公司业务单元的一个主要软件开发组织，它在1994年获得了Malcolm Baldrige国家质量奖。当时，它是AT&T中使用软件可靠性工程比例最高的单位。另一个有趣的现象是，前5个获得AT&T贝尔实验室主席质量奖的单位中，有4个使用了软件可靠性工程。

International Definity项目是对软件可靠性工程的一次应用。在这个项目中，软件可靠性工程和其他相关软件技术一起使用。与前一个没有使用这些技术的版本相比，由用户发现的错误减少了90%，这使得客户的满意度大幅度上升。结果，销售量也上升了10倍，系统测试时间和费用减少了一半，整个项目的开发时间减少了30%，程序维护费用降低了90%（Abramson等，1992）。

软件可靠性工程领域的快速发展体现了它的价值。IEEE计算机协会软件可靠性工程委员会从1990年创建时的大约40人，到1996年中期超过了1000人，每年增长速度超过70%。由向一年一度的国际软件可靠性研讨会（ISSRE）投稿的数量判断，软件可靠性工程研究的同期年增长率为35%。

应用软件可靠性工程的经验表明，对大多数项目来说，应用软件可靠性工程将占用全部开发费用的0.1%到0.2%。对大型项目，这个比例会减少；对小型项目，这个比例会增加。对小型项目来说，全部工作量将小于1个人月，包括培训、开发操作剖面、收集数据和处理的成本。

1.2 软件可靠性工程过程

应用软件可靠性工程的过程，首先要求确定哪些系统与要测试的产品相关联。在这项工作中，必须了解软件可靠性工程测试的类型。因此，首先来看一下后面这个问题，然后再讨论如何确定要测试哪些关联系统。

软件可靠性工程的过程包括5个步骤，即定义“必要”的可靠性、开发操作剖面、准备测试、

执行测试和应用失效数据得到结论。图1-1对此进行了说明，图的底部还标出了典型的项目阶段。请注意，“执行测试”和“应用失效数据得到结论”这两个步骤是同时进行的，并且联系的很紧密，随着可用的失效数据的增加，重点相对向应用数据倾斜。这5个步骤的每一步都分别在本书的一章中详细讨论（第2章到第6章）。

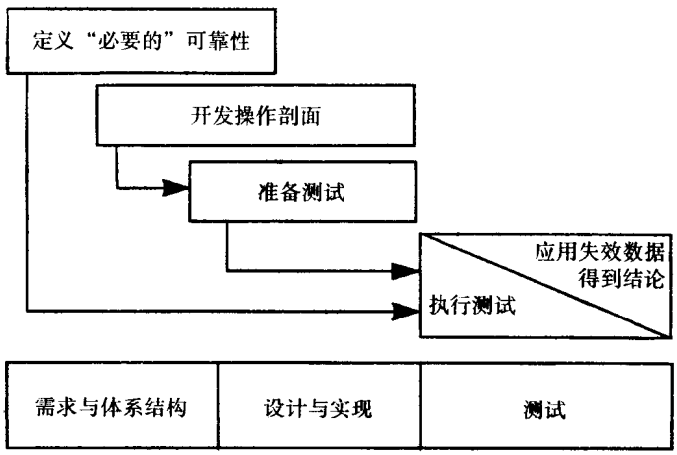

图1-1 软件可靠性工程过程框图

为了简洁起见，这个过程框图只给出了以箭头表示的软件可靠性工程的主要工作流。实际上，任务经常会有循环和反馈，这与软件开发过程的螺旋模型（与瀑布模型相反）类似。就像在软件开发过程中，一些需求和体系结构在测试之后可能会发生变化一样，在软件可靠性工程的过程中，对“必要”的可靠性定义在“执行测试”和“应用失效数据得到结论”两个步骤之后，也可能会发生变化。

软件生命周期中，测试之后的发行与维护阶段并没有在图1-1中表示出来。在这个阶段中，可以确定事实上得到的可靠性以及实际的操作剖面。这些信息会对下一个版本“必要”可靠性的定义和操作剖面的开发产生影响。它可能会产生产品和开发过程的再工程。

测试人员与软件工程师一起合作完成前两个活动，即“定义必要的可靠性”和“开发操作剖面”。我们原来曾经认为这两个活动应该完全交给系统工程师和系统构架师来完成，但是，在实践中效果并不好。测试人员需要依靠这两项工作，因而比系统工程师和构架师更有动力去成功完成。让测试人员参加系统工程和系统构架小组就可以解决这个问题。

这个解决方法还会带来意料之外的益处。测试人员与产品用户有了更多的接触，这是非常有益的，这有助于了解什么样的系统行为是不可接受的，它们在多大程度上是不可接受的，以及用户将会怎样使用这个产品。系统工程师和构架师对于测试，以及在什么地方需求和设计应该更清楚、更精确有了充分的了解，这使得测试计划、测试案例和测试过程的设计能够顺利进行。系统测试人员对系统结构的审查做出重要贡献，经常会指出一些被遗忘的重要功能。

系统体系结构阶段包括软件可靠性工程中的对失效预防、失效清除和容错的策略混合的选择活动。因此会影响产品和过程的设计。测试人员可以不做这项工作，但是必须了解它，因为它会影响他们。

1.3 Fone Follower

考虑一个贯穿全书的例子，以便使应用软件可靠性工程的过程说明得更加具体。这个例子取材于一个实际的项目，为了简洁和对专有信息的保护，其中的细节做了修改。选择这个例子是因为它是关于电话服务的，对大多数人来说很容易理解。同时，它还很小（全部人员少于10人，软件可靠性工程要求的额外工作量大概为1人月）。这绝不是说软件可靠性工程只能应用于通信系统或小的项目。实际上，软件可靠性工程已经被应用于源代码量从5 000行到10 000 000行的各类系统。除了通信系统之外还（至少）应用于医用成像系统、资产管理和财会系统、编译器、终端固件、仪器固件、军事系统和空间系统。

Fone Follower系统可以处理从世界各地打入的电话呼叫（传真或语音），包括蜂窝电话和其他移动电话。作为Fone Follower的用户，可以拨打Fone Follower，并输入想要拨打的电话号码。

从电话网来的对某一号码的呼叫都被传送到Fone Follower，它根据输入程序对传真和语音呼叫进行处理。如果某一号码对语音呼叫没有应答，并且此号码有寻呼服务，则对其进行寻呼。如果仍然没有应答或此号码没有寻呼服务，那么语音呼叫将被传送到语音信箱。

Fone Follower使用的操作系统是由销售商提供的，它的可靠性目前是未知的。在使用者看来，服务包括标准的电话服务和Fone Follower提供的呼叫传送服务。

1.4 测试的类型

软件可靠性工程测试包括两种类型：可靠性增长测试和确认测试。这两种类型与不同测试阶段无关，例如单元测试、子系统测试、系统测试或 β 测试，而是与测试的目标相关。可靠性增长测试的目标是找到并清除错误。在可靠性增长测试过程中，使用软件可靠性工程来估计并跟踪可靠性。测试人员和开发经理应用可靠性信息来指导开发和发布。可靠性增长测试一般在自己机构内开发软件的系统测试阶段。如果一边测试一边解决失效问题（清除导致它们的错误），那么可靠性增长测试也可以应用于 β 测试阶段。为了得到一个“好的”（有一定的不确定性）对失效强度的估计，在失效数据样本中对失效数据的数量有一个最低要求，通常是10到20。

可靠性增长测试包括特性测试、负载测试和回归测试。在特性测试中，操作都是独立运行的，运行场地环境的影响和交互作用被减小到最低程度。有时通过在操作之间重新初始化系统来减小交互作用。负载测试是指同时运行很多操作，并且是以相同的频率，在其他现场将会出现的相同环境条件下。这样就可以产生与在现场中可能出现的情况相同的交互作用和环境条件的影响。验收测试和性能测试都属于负载测试。回归测试是在系统发生重要改变之后进行的，包括一些（通常是随机选取的）或全部特性测试。在回归测试中应该包括所有关键操作。

负载测试通常包括对系统资源的竞争，这可能会产生排序或计时问题，同时还经常出现数据随时间的退化问题。前述的因素可能会揭示出一些潜在的特性测试和回归测试中没有被激活的由交互作用导致的失效。尽管交互作用对于多用户系统更为重要，它对于像在个人电脑上运行的软件这样的单用户系统也是重要的，因为可能会有由于操作之间的顺序而产生的不同的交互作用。

确认测试不包括调试过程，不会试图通过引起定位错误后再清除错误来解决所发现的失效。被测系统必须是稳定的，不能出现任何改变，不管是由于增加了新特性还是由于错误的清除。通过确认测试，得到一个二选一的结论：或者接受这个软件，或者拒绝它并把它退回给提供商。在确认测试中，所需要的失效数据样本的数量要少得多。事实上，如果无失效运行的时间足够长，那么可以在出现任何失效之前就做出结论。通常只在负载测试（不是特性或回归测试）中使用确认测试。

1.5 待测系统

为了学习或分析的方便，可以用各种方法对“系统”进行定义。它可以包含硬件、软件和人员要素的各种组合。一般地，可以将“系统”定义为要单独测试的每一个实体。

当然，一般情况下希望将正在开发的真实产品作为一个系统。只要是正在开发，即使是产品的一部分，我们也将使用可靠性增长测试。接下来可能是确认测试，如果用户将要进行验收测试，那么这就是一次彩排。如果只是将各个组件集成起来形成产品，那么就只对集成后的产品做确认测试。对产品的主要变体也要识别为待测系统。产品的一个不同的硬件配置（完成同样的功能）是一个不同的系统。因为必须要能够在不同的平台或不同的操作系统上运行，所以会有多个有很大差别的版本。国际性的产品在不同的国家还需要有不同的接口。

如果有*超系统*或者说是此产品在其中作为一个组件运行的系统，那么应该将它们看作潜在的要单独测试的系统。如果用户通过超系统的可靠性来判断一个产品，或者是产品和组成超系统的其他系统之间的交互非常复杂，以至于难以定义的话，那么很可能要对超系统进行单独测试。这是由于使用接口驱动来独立测试产品是有风险的，因为对接口的刻画可能是不正确的。在封闭式软件包的情况下，经常对超系统进行测试。其他的例子包括打印机和个人电脑。如果对一个或多个包含此产品的超系统进行测试，那么对此产品的变体所对应的超系统可能也需要测试。

有些超系统进行系统测试是不实际的，或不经济的，但是可以进行 β 测试，因此要为此阶段做必要的进度安排。当然，β 测试涉及现场环境中的直接执行，标识失效，并应用失效数据。β 测试不要求开发操作剖面，准备测试用例和过程，也不需要通过测试过程调用测试案例。

可能还要将*采办的软件*（不是由自己机构开发的软件）组件作为自己的系统测试，如果它们的可靠性是未知的，或是有疑问的，并且尽早发现不可靠性可以避免进度拖延或费用超支的话。通常，这种测试仅对于产品的第一版是必要的，尽管对于后续包含新的或较大变化的采办组件的版本可能也是需要的。采办软件包括下架软件，或包装的软件包，或由其他产品或组件或对象库重用得到的软件或子结构软件。如果根据以前经验得知一个组件对于此产品来说具有足够的可靠性，那么就没有必要单独对它进行测试。如果并不能肯定组件，但是与其供货商有着良好的关系，可以由他们在监督下进行可靠性增长测试，跳过验收测试并节约时间和金钱。请注意，在确定是否测试组件时，尺寸是一个重要考虑因素。如果组件是一个主要部分，则对它进行单独测试是经济的。如果都是些小的采办软件组件（包括对象），那么只有当很多系统都

用这些组件时，对其进行单独测试才是经济的。

将软件可靠性工程应用于对象库的测试上似乎很有潜力，因为这些对象库是面向对象开发中需要的，而在很多系统中都希望使用这些对象。事实上，面向对象开发的进一步发展和使用可能要依赖于这种技术融合：面向对象的概念使更好的模块化成为可能，但是通常并没有了解重用的前提和益处，因为开发者（很可能是正确的）拒绝使用那些可靠性不明确的对象。

当然，要选择多少个系统进行单独测试是有限制的。每一个单独测试都包含额外开销。尽管可以并行执行多个测试，但在某些时候，有限的人力和计算资源还是会造成进度拖延。费用包括开发一个或多个附加的操作剖面、开发测试案例和测试过程，以及执行测试。只有当从单独测试得到的益处大于其所要求的额外时间和费用时，才会对额外系统进行单独测试。益处包括客户不满意程度的降低或日程拖延（两者都将影响市场份额）的减少和额外开发费用的节省。

对一个额外系统进行单独测试的费用可以降低，如果这个系统拥有操作体系结构（它的组件对应于操作或操作组）。这是因为对它的组件开发操作剖面相对简单，使得对组件进行单独测试相对容易。

尽管对多于几个的系统进行单独测试可能是不值得的，但是确定对在一起测试的大量独立组件的失效强度并不那么昂贵，因此也很现实。这是因为，只需要将组件的失效进行分类并度量组件所做的处理（通过自然单元或者时间单元计算）。然而，由于从小的失效样本来估计失效强度时会引起误差，因此估计多个失效强度是不实际的。

在Fone Follower中，要对操作系统获得的组件、Fone Follower本身以及包含电话网和Fone Follower的超系统进行测试。可以对产品使用确认测试，但是由于这会延长交付时间，所以没有这么做。

1.6 常见问题

在本节中，首先研究的问题为是否需要应用软件可靠性工程及其带来的益处。接下来的问题集中于澄清软件可靠性工程的概念，它与其他实践方法的联系，以及何时、何处应用软件可靠性工程。

1.6.1 有效性和益处

1. 为什么软件可靠性在软件质量领域如此重要？

答：在软件质量这个较大并且可能有些模糊的概念中，软件可靠性是一个很专门的可度量的属性。它可能是最具包含性的最重要的属性，因为它处理运行中与用户描述的行为的偏离自由度。换句话说，软件以用户所希望的方式执行。严格说来，软件质量中的没有包含在软件可靠性中的其他方面（例如可维护性），在程序没有让人满意地运行的时候都是第二位的。

2. 软件可靠性与其他量化质量的方法相比，在有效性上有什么差别吗？

答：对用户来说，软件可靠性是质量最重要的方面，因为它定量地指出一个软件产品对于用户的需求的工作好坏程度。质量的其他度量可能与软件可靠性有些关系，但是相比起来它们

都很不直接。例如，剩余错误数量与软件可靠性有关系，但它是面向开发者的，并且它没有指出对操作的影响。不仅如此，剩余错误不能度量而只能是推测，这意味着精确性很低。已发现的错误数量与可靠性没有关系。如果只发现了几个错误，这或者可以表明软件是可靠的，或者表明是很差的测试和不可靠的软件。在设计或代码检查中没有发现的问题数量作为一种质量度量来说，与已发现的错误有相同的属性。其他度量，例如程序复杂性，离用户的质量概念就更远了。

3. 从硬件可靠性照搬的软件可靠性是否会导致一种不合适的技术？

答：软件可靠性不是从硬件可靠性照搬过来的。在开发软件可靠性的过程中，充分意识到了两者的区别。然而，我们故意使其定义在数学上与硬件可靠性相容，这样就可以计算那些同时包含硬件和软件的系统的可靠性。我们研究了诸如错误计数之类的其他方法，但是由于不相容等原因又否决了它们。一个不相容的定义是没有实际意义的，因为真实系统都是既包含硬件又包含软件的。

4. 软件可靠性工程对于获得Baldrige奖而知名的全面质量管理有帮助吗？

答：软件可靠性工程与全面质量管理紧密联系，并且实际上是它的主旨。它提供了一个与用户满意度紧密相连的面向用户的指标。说它是主旨是因为，没有一个面向用户的系统可靠性指标，就无法管理质量，并且没有软件可靠性度量，就无法确定基于软件的系统可靠性。

软件可靠性工程几乎与Baldrige奖打分系统中所有主要指标有关联。

5. 有人说软件可靠性工程是不充分的，因为它没有充分强调防止灾难性失效。这种看法对吗？

答：不对。这种看法没有正确理解软件可靠性工程。灾难性失效是所有失效中的一部分。跟踪和减少所有失效，也会按比例减少灾难性失效。

6. 在测试中，遇到失效我们就解决它，为什么还需要估计软件可靠性呢？

答：因为如果不这样的话，就不会知道还要执行多少测试。

7. 软件可靠性概念是否有益于根原因分析（找到造成错误的最终原因，目的是提高软件工程过程的质量）？

答：是的。操作剖面（在第3章中定义）加强了这些经常执行的功能。与这些功能有关或与会产生严重影响的失效有关的失效报告，在根原因分析的研究中应该得到优先考虑。在根原因分析基础上进行的提高质量的活动，应该首先强调那些将使客户对质量的感觉产生最大影响的因素。

8. 如何使用软件可靠性工程来改善设计和测试过程呢？

答：使用软件可靠性工程可以在以下几个方面改善设计和测试过程：

a. 使用操作剖面（在第3章中定义）的知识，将设计工作重点放在最常用的功能上。

b. 应用操作剖面来驱动测试，将首先激活最常用的功能，有助于首先找出并清除这些功能的错误，从而快速降低失效强度。

c. 开发机构和客户一起建立的失效强度目标，决定了在设计和测试中需要多大的工作量，以及要使用的技术。

d. 失效强度目标给出了一个发布标准，即在何时可以停止测试。

试读结束：需要全本请在线购买： www.ertongbook.com