除了本书以外，几乎还没有什么书从系统化的角度介绍发现和修正编程错误的方法。
——摘自本书序言
James Larus 微软研究院

Broadview
www.broadview.com.cn

Why Programs Fail
——系统化调试指南

ANDREAS ZELLER

Why Programs Fail
——A Guide to Systematic Debugging

[德] Andreas Zeller 著
王咏武 王咏刚 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
HTTP://WWW.PHEI.COM.CN

Why Programs Fail

——系统化调试指南

[德] Andreas Zeller 著
王咏武 王咏刚 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

调试一直是软件开发过程中“最痛苦”的环节，本书有望改变这一现状，因为它将调试的科学原理与业界的实践经验有机地融合起来，阐释了有关发现和修正程序错误的最佳方法和实践过程。

本书一共分为 15 章，以系统化的方式向读者介绍了整个调试过程，从跟踪和重现故障开始，一直到自动化和简化测试用例，寻找故障最可能的来源，分离故障的起因和结果，并最终修正程序缺陷。本书不仅涵盖了 delta 调试、程序切片、观察、监视、断言、检测反常等多种基本的静态和动态程序分析技术，还用浅显的语言说明如何使用一些调试领域最前沿的高水平调试工具。

本书适于那些希望掌握如何以系统化和自动化的方式调试程序的计算机编程人员、及相关专业的研究生以及高年级本科生。

Why Programs Fail, First Edition
Andreas Zeller
ISBN: 1558608664

ISBN: 981-259-694-1

图书在版编目（CIP）数据

Why Programs Fail：系统化调试指南 /（德）泽勒（Zeller,A.）著；王咏武，王咏刚译. —北京：电子工业出版社，2007.3
书名原文：Why Programs Fail, First Edition: A Guide to Systematic Debugging（Paperback）
ISBN 978-7-121-03686-6

I. W… II. ①泽… ②王… ③王… III. 软件—调试—指南 IV. TP311.5-62

中国版本图书馆 CIP 数据核字（2006）第 158931 号

责任编辑：顾慧芳
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：27　字数：652.5 千字
印　　次：2007 年 3 月第 1 次印刷
定　　价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系电话：（010）68279077；邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

Broadview®
www.broadview.com.cn

《Why Programs Fail——系统化调试指南》

读者调查表

尊敬的读者：

感谢您对我们的支持与爱护。为了今后为您提供更优秀的图书，请您抽出宝贵的时间将您的意见以下表的方式及时告知我们（可另附页）。我们将从中评选出热心读者若干名，免费赠阅我们以后出版的图书。

姓名：	性别：☐ 男 ☐ 女	年龄：	职业：
通信地址：			邮政编码：
电话：	传真：	E-mail：	

1. 影响您购买本书的因素（可多选）：

☐ 封面封底　☐ 价格　☐ 内容提要、前言和目录　☐ 书评广告　☐ 出版物名声
☐ 作者名声　☐ 正文内容　☐ 其他_____

2. 您对本书的满意度：

从技术角度　☐ 很满意　☐ 比较满意　☐ 一般　☐ 较不满意　☐ 不满意
☐ 改进意见_____

从文字角度　☐ 很满意　☐ 比较满意　☐ 一般　☐ 较不满意　☐ 不满意
☐ 改进意见_____

从版面、封面设计角度　☐ 很满意　☐ 比较满意　☐ 一般　☐ 较不满意
☐ 不满意　☐ 改进意见_____

3. 您最喜欢书中的哪篇（或章、节）？请说明理由。

_____

_____

4. 您最不喜欢书中的哪篇（或章、节）？请说明理由。

_____

_____

5. 您希望本书在哪些方面进行改进？

_____

_____

6. 您感兴趣或希望增加的图书选题有：

_____

_____

通信地址：北京万寿路 173 信箱　博文视点（100036）　　电话：010-51260888

如果您对我们出版的图书有任何意见和建议，也可以发邮件给我们，我们将及时回复。

E-mail：jsj@phei.com.cn，editor@broadview.com.cn

反侵权盗版声明

举报电话：（010）88254396；（010）88258888

传　　真：（010）88254397

E-mail：　dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱

　　电子工业出版社总编办公室

邮　　编：100036

原书扉页书评

著名的数据显示，调试器 DDD 的作者现在推出了这部权威的调试指南。这本书以系统化的方式向读者介绍了整个调试过程，从问题跟踪、调试测试、重现问题开始，一直到高水平的调试工具，诸如生成伪对象以重现那些难于重现的事件，自动寻找引发故障的输入数据，以及分离可能的故障起因等工具。

如果你是一位经验丰富的程序员，并自认为已经了解了调试的每个方面，那么你不妨再思考一下。Zeller 这本书里装满了可以追踪程序缺陷的建议、智慧和工具，适合于所有不同程度、使用不同编程语言的读者阅读。

本书文理清晰地阐释了每项技术的基本准则，也没有让读者受到细枝末节的困扰。最方便的是，在每章的末尾，它都会告诉你到哪里下载那些奇妙的工具。这是一本为专业软件开发者以及那些对自动化调试前沿领域感兴趣的学生准备的优秀图书。

——Walter F. Tichy，德国 Karlsruhe 大学教授

尽管许多程序员都把调试看作软件开发中“最痛苦”的部分，计算机科学专业的学生和从业者却找不到几本可用来学习科学的调试方法的图书。

在这本书里，Andreas Zeller 出色地向大家展示了有用的调试技术以及来自学术界和产业界的调试工具。本书不但容易阅读，而且十分有趣——千万别忽略书中有关 Bug 的很多故事。

我向那些对软件工程研究有兴趣的研究生和本科生全力推荐本书。它不仅能帮助你发现有关调试的新视角，还用浅显易懂的语言教会你一些基本的静态和动态程序分析技术。

——Miryung Kim，华盛顿大学（University of Washington），
计算机科学与工程系研究生

Andreas Zeller 的《Why Program Fail——系统化调试指南》一书为程序员、教育工作者和研究者打下了良好的基础。Andreas Zeller 用科学而熟练的写作技巧为我们提供了深入的分析、翔实的方法和直观的例证。

——David Notkin，华盛顿大学（University of Washington），
计算机科学与工程系教授

译者序——徜徉在经验与科学之间

今天，似乎已没有人否认调试也是一门科学。但在绝大多数工作在第一线的程序员眼中，调试还更多地停留在经验的范畴内。

程序出问题了！刚刚掌握 Java 语法的新手多半会手足无措，一个老练的 Java 程序员却总会在第一时间为程序添加上几条关键的 log 语句（今天更时髦的方法是用 ASPECTJ 构造一个专用于调试的方面），然后指着程序输出对可怜的新手说："瞧，错误是 ConcurrentMagicTimer 类的第 1043 行语句引发的。"

这个时候，也许每一个坚定的科学主义者都会进一步追问："我该怎样知道，到底要在哪些地方记录日志，到底该如何观察输出信息呢？"很遗憾，没有几个编程老手会将其中的奥秘一五一十地告诉你——因为往往连他们自己都不大明白；他们只会耸耸肩膀说："编程编多了，你自然就知道了。"

很高兴看到 Zeller 教授将与调试相关的几乎所有层面的知识、方法和经验总结成了科学的、可以重复和自动化实施的方法与原则。如果非要在软件开发领域找出一些有资格编写系统化调试指南的专家来的话，Zeller 教授的名字肯定会在名单中跻身前十名。这不仅仅是因为 Zeller 教授是大名鼎鼎的可视化调试器前端 DDD 的作者，更重要的是因为 Zeller 教授总会以一种独特的视角——即，将调试方法的科学原理与业界实践中相应的经验背景结合起来的视角——审视与调试相关的每一个重要问题。

对于一本指南性的调试技术专著来说，这种视角的最大好处就是，读者既可以从书中学到大多数调试高手最喜欢使用和炫耀的"神奇"技术，也可以豪不费力地了解隐藏在"神奇"技术背后的那些其实并不算十分"神奇"的科学思路。对调试来说，经验是好的，但科学也是好的。前者可以让你在任何程序故障面前都拥有十足的自信，而后者则可以帮助你在艰苦的调试过程中始终保持清醒的头脑和冷静的思路。

更重要的是，经验和科学的有机融合几乎可以让所有的调试步骤都变成利用电脑和工具自动完成的"自动化调试"。想像一下，面对一个由数十万行 C++ 代码构造而成的庞然大物，如果可以利用 Zeller 教授推介的方法和工具，让电脑自动将故障范围缩小到一个只有 50 行代码的算法片断之内，甚至让电脑自动找出出错的代码行，我们为什么还要在 GDB 等调试器中大海捞针般地辛苦

搜寻呢？事实上，在调试自动化方面，Zeller 教授在本书中带给我们的远比我们所能想像的要多得多。

有机会翻译并向国内读者推介本书是我们的荣幸，尽管整个翻译过程充满了挑战和艰辛。Zeller 教授为本书创建的网站（http://www.whyprogramsfail.com）是我们在翻译过程中参考得最多的电子资源，此外，Google 的网页和学术搜索为我们更快地搜寻与验证原始资料提供了无可替代的帮助。更值得一提的是，除了在翻译过程中根据 Zeller 教授发布在网站上的正式勘误表改正了原书排印时的许多错误外（因为有勘误表可查，书中这些改正一般不再另行注明），我们还在翻译过程中与 Zeller 教授进行了有效的沟通，并进一步发现了更多的排印错误——对于这些新增的勘误，Zeller 教授几乎都是在第一时间更新了他的勘误表，其工作效率和敬业精神令人钦佩。

因为经验、水平有限，再加上本书公式、术语较多，翻译中的疏漏、错误在所难免。真诚地欢迎读者批评、指教。有关本书的任何反馈信息都可以发至我们的电子邮箱：wangyw@contextfree.net

王咏武、王咏刚

2006 年年末于北京

关于作者

Andreas Zeller 是德国 Saarland 大学的计算机科学教授。他的研究方向集中在提高程序员的生产力方面：即哪些事情可以使程序员的生活和工作更轻松？在 Linux 和 UNIX 程序员中，Zeller 因 GNU DDD——一个拥有内嵌数据可视化机制的调试器前端——而享有盛名。在研究人员和高级程序员中，Zeller 因为 delta 调试——一种可以自动分离计算机程序故障起因的技术——而声名显赫。

他的工作时间被平均地分给教学、阅读、写作、编程，以及在大西洋两岸飞来飞去。他与家人一起生活在德法边界德国一侧的 Saarbrücken。

序

在《联邦党人文集》第51篇中，詹姆斯·麦迪逊（James Madison）写道："如果人人都是天使，那么根本就不需要政府。"如果麦迪逊能活到今天，他可能会说："如果软件开发者都是天使，那么根本就不需要调试。"但是，在设计和编写软件时，我们中的大多数都会出岔子，很多人还会犯错误。我们必须发现并修正差错，这是一项自打有计算机程序起就一直存在的工作，它的名字叫"调试"。今天，虽然每个编写出来的计算机程序都要经过调试这一步，但调试本身却不是一种被广为研究和传授的技能。除了本书以外，几乎没有什么书从系统化的角度介绍发现和修正编程错误的方法。

老实讲：调试原本就像编程一样重要、困难或值得研究吗？或者，它仅仅是你为了完成一个项目所必须做的一项工作？但软件开发者却在调试上花费了大量的时间——大约要占去他们工作时间的一半或更多。更快、更有效地发现和修正Bug可以直接提高生产力，并可以通过在现有资源的基础上尽量减少缺陷来改进程序质量。能预先杜绝错误当然更好，但至今还没有人找到彻底杜绝错误的技术手段，所以，有效的调试仍是不可或缺的。

改进程序设计语言和工具，通过静态识别错误以及动态检测不变量违例，可以承担调试的一部分工作，但不能完全替代它。例如，Java和C#等现代语言中的类型检查系统可以防止C程序员常犯的许多初级错误。此外，这些语言的运行时边界检查机制可以在程序访问超出边界的情况下终止程序——如果不是这样，可能要经过数十亿条指令才能将错误暴露出来。但不幸的是，可以将程序导致错误的方式数不胜数，其中的绝大多数都是语言和工具无法发现并阻止的。例如，最近几年里，人们做了相当多的工作来检验程序的执行序列是否正确。工具可以保证程序在读文件之前该文件已被打开，但是它们却无法检查打开的是不是正确的文件，或者程序是不是正确地解读了文件的内容。如果出现任何一种差错，程序员都必须调试程序，以便了解错误的起因并决定如何修正它。

另外，调试也可以成为一件快乐的工作，就像优秀的侦探小说或电子游戏中都会有捕猎和追逐的刺激一样。但另一方面，一次在代码中拖沓的、不成功的搜寻Bug的经历会迅速丧失其自身的魅力，特别是当你的老板正在反复追问

你有关进度（落后）的问题时。只有学会有效调试才会热爱软件开发。

本书可以指导你如何更有效地调试。这是一本全面的、实用的调试总论，其作者是一位天才的研究者，他创造了许多分离 Bug 的巧妙方法。本书阐释了有关发现和修正程序错误的最佳实践，涵盖了系统化地跟踪错误报告、重现故障、观察症状、分离起因以及修复缺陷等方面。除了基本技术和常用工具以外，本书还展示了作者有关分离最小输入以重现错误，以及在整个程序中追踪起因和结果的创新技术。

学习本书可以使你成为更好的程序员。你可以更快、更有效地在你（或同事）的代码中发现和修正错误，这是一种可以使你更快地完成项目并发布缺陷更少的程序的宝贵技能。另外，如果你能真正理解本书文句背后的内涵，你将学会如何编写更适于测试和调试的代码，这会进一步提高你发现和修正缺陷的能力。同时，对程序错误来源的认真思考可以帮助你预先避免差错，这样一来你需要调试的东西就少之又少了。

James Larus
微软研究院
2005 年 8 月

前　言

这是一本关于计算机程序中的 Bug 的书——如何重现 Bug？如何定位 Bug？以及如何修正 Bug，使它们不再出现？本书将教会你很多技术，使你能以系统的甚至是优雅的方式调试任何程序。而且，这些技术在绝大多数情况下都可以自动运行，这样，就可以让你的计算机来完成大部分的调试工作。本书试图解决以下问题：

- 怎样如实地再现故障？
- 怎样分离与故障相关的所有因素？
- 故障是从哪里来的？
- 怎样用最好的方式修正程序？

一旦理解了调试的工作原理，你就会以不同的方式来看待调试。你会考虑起因和结果，你会以系统化的方式构造和完善假设以跟踪故障起因，而不是只看到一大堆混乱的代码。你对调试的理解甚至能帮助你创建自己的自动化调试工具。所有这些将会使你在调试上花费更少的时间，这也就是你对自动化调试感兴趣的主要原因，不是吗？

写作本书的缘由

尽管我是一个研究人员，但是我更愿意把自己看作是一名程序员，因为我的大部分时间都是在编写程序。在编程过程中我会犯错误，因此不得不调试代码。我很希望自己是那种被称为“超级程序员（überprogrammer[①]）”的程序员——永远不会犯错——但我和你们一样，只是一个普通人。

在学习过程中，我了解到少量的预防措施要比大量的治疗措施有价值得多。我还学会了许多种预防错误的方法。今天，我又把它们传授给我的学生。但是，在努力预防的同时也不能忘了治疗。如果我们是医生，不能仅仅因为病人没有采取所有可能的预防措施，就拒绝进行治疗。

---

① 译者注：über 在德语里是“超级”的意思。

所以，与其设计一种终极的预防措施，不如寻求一个好的治疗方法。现在，遍及全球的其他研究人员已经接受了这种务实的观点。我很高兴地宣布我们已经成功了。今天，许多先进的调试技术已经被广泛地应用于自动化调试过程中。

这些技术不仅可以使调试过程自动化，而且可以把调试从一种“旁门左道”转变为一门系统的、组织严密的学科，该学科可以像任何一门软件工程科目一样被纳入教学计划。因此，我开设了一门关于自动化调试的课程，并把该课程的讲稿整理成了一本书。现在，它就呈现在大家的面前。

读者对象

本书适于那些希望掌握如何以系统化和自动化的方式调试程序的计算机专业人员、研究生以及高年级本科生。我们假定读者熟悉编程和手工测试，无论这些知识是从介绍性课程还是从工作经验中获得的。

本书的涵盖范围

本书的重点是 Bug 的*治疗*——即，在出现程序故障后进行的分离和修正程序代码中的缺陷的活动。本书只会涵盖一部分预防缺陷的话题，你可以从很多其他书籍中找到关于预防的深入探讨。实际上，可以说大部分计算机科学都是与如何预防 Bug 相关的。但是，当预防措施失效时就必须进行治疗，这就是本书的主旨。

内容提要

本书共分为 15 章和一个附录。其中第 1 章、第 6 章和第 12 章是阅读各自后续章节的先决条件。

在每一章的最后都有一个被称为“概念”的小节，它概括了这一章的关键概念。其中的一些概念前面标有“How To”，它们概括了一些很容易遵循的技

巧。（目录中有一张表列出了所有的“How To”。）此外，每一章的结尾还有一些练习题，用来检验你学到的知识，以及一个被称为“进一步阅读指南”的小节。本书的内容组织形式如下所示：

第 1 章：故障从哪里来

你的程序出故障了，为什么会这样？答案是程序员写了一段有缺陷的代码。程序运行时，有缺陷的代码造成了程序状态中的感染，随后引发了一个可感知的故障。为了寻找有缺陷的代码，你必须以这个故障作为起点回溯追踪起因。本章定义了和调试相关的基本概念，并且介绍了后续章节将会讨论的一些技术——希望能像正餐前的开胃酒，增进大家的食欲。

第 2 章：跟踪问题

本章阐述如何管理用户报告的问题：如何跟踪和管理问题报告？如何组织调试过程？如何跟踪多个版本？以上问题构成了整个调试过程的基本框架。

第 3 章：让程序出错

在调试程序前必须首先配置好程序的测试环境——即，带着使程序出错的意图运行程序。本章会回顾一些基本的测试技术，并且重点讨论自动化和分离技术。

第 4 章：重现问题

调试过程的第一步是重现当前的问题——即创建一个测试用例使程序以某种特定的方式发生故障。这么做的第一个目的是使故障处于我们的掌控之中，这样才能更好地观察它。第二个目的是验证修正是否正确。本章的主题是关于如何重现问题的操作环境、历史和症状的典型策略。

第 5 章：简化问题

重现了问题之后，下一步就必须简化它——即，找出与该问题无关的环境因素，并忽略它们，其结果就是得到一个只包含相关环境因素的测试用例。在

最好的情况下，简化的测试用例产生的报告能立刻精确地定位缺陷。本章介绍一种能自动简化测试用例的自动化调试方法，即 delta 调试技术。

第 6 章：科学调试

重现和简化问题之后，还必须理解故障的来源。我们可以通过某些方法得到能解释宇宙的某个方面的理论，这一过程被称为*科学方法*，它同样适用于获得问题的诊断结果。本章介绍了很多关于科学方法的基本技术，包括怎样构造和验证假设、怎样进行试验、怎样系统地实施整个过程，以及怎样使调试过程清晰明了。

第 7 章：推演错误

本章开始研究第 6 章介绍的构造假设的技术。首先从*推演*技术开始——这是一种从*抽象的*程序代码到*具体的*程序运行过程的推理技术。我们还要特别讨论*程序切片*技术，一种能确定变量值的潜在来源的自动化方法。我们可以使用程序切片技术有效地缩小可疑的错误状态感染点的数量。

第 8 章：观察事实

尽管使用推演技术无需考虑具体的运行过程，但观察技术则可以用来确定具体的运行过程中的*事实*，即具体运行过程中发生的情况。本章将会深入探究实际的程序运行过程，并且介绍一些被广泛应用的分析程序运行过程与状态的技术，其中包括传统的日志记录技术、交互式调试技术、事后调试技术，以及很有启发的可视化与概括技术。

第 9 章：跟踪来源

一旦在调试过程中观察到一个被感染的错误状态之后，就必须揭示出它的来源。本章将讨论*全知调试*，这是一种记录整个运行历史的技术。使用这种技术，用户不需要重新运行程序，就能探查运行过程中任意时刻的情况。除此之外，本章还会探讨*动态切片*技术，通过这种技术可以跟踪某些特定取值的来源。

第 10 章：断言预期结果

仅仅使用观察技术还不足以进行调试，我们还必须把观察到的事实与期望的程序行为进行比较。本章会讨论如何使用著名的断言技术来自动进行比较，除此之外还会展示如何确保内存等重要的系统组件的正确性。

第 11 章：检测反常

虽然我们能从一次程序运行过程中得到相当多的信息，但是比较多次运行过程能给我们提供更多定位正常和反常的机会——反常通常能帮助我们定位缺陷。本章会讨论如何检测代码覆盖和数据访问中的反常，还会展示如何从多个测试运行过程中自动推断不变量，其目的是检测随后的不变量违例。所有这些反常都是潜在的感染点。

第 12 章：起因与结果

推演、观察以及归纳技术在搜寻潜在缺陷方面都表现优异。然而，单独使用其中任何一种技术来定位故障起因都是不够的。那么我们如何识别起因？又如何做到分离出的不仅仅是某一个起因，而是与故障相关的那个真实起因？本章的基本内容是为系统地、自动地寻找故障起因打基础的技术。

第 13 章：分离故障起因

本章的主要内容是如何使大部分调试过程自动化。我们演示了如何使用 delta 调试自动分离故障起因——包括程序输入、程序的线程调度以及程序代码中的故障起因。在最好的情况下，使用 delta 调试技术得到的起因能立刻定位缺陷的位置。

第 14 章：分离因果链

本章介绍一种进一步缩小故障起因的方法。即，通过抽取和比较程序状态，应用 delta 调试技术自动分离引发故障的变量及其取值，最终可以得到关于故障的因果链，例如：“变量 x 的值为 42，因此变量 p 的值变为 null，于是程序发生了故障。”

第 15 章：修正缺陷

一旦理解了故障的因果链，我们就可以确定故障来源。但是我们还必须找出错误状态开始感染的位置——也就是缺陷的实际位置。本章会讨论如何系统地缩小缺陷的范围——以及如何修正已定位的缺陷。

附录：规范定义

为了便于阅读，附录中收录了本书所有的规范定义及其证明。

参考书目

参考书目列出了大量与本书主题相关的可供进一步阅读的资料。

索引

本书的结尾是一张关键词索引表。

补充材料、资源及官方网站

本书涉及的大部分材料以前从来没有在哪本教科书中出现过。后面一些章节讨论的内容甚至还没有被实践广泛验证。像那些涉及正在发展的领域的书籍一样，本书也有待于从更多的理论研究和实践工作中得到完善。换句话说，书中肯定会有很多错误，欢迎大家对本书提出自己的意见。你可以写信给 Morgan Kaufmann 出版公司并由他们转交给我，或者直接发 E-mail 给我：zeller@whyprogramsfail.com。本书还有一个官方网站：

```
http://www.whyprogramsfail.com
```

从那里可以找到新近发布的信息或更新（包括修正）。