HZ BOOKS
软 件 工 程 技 术 丛 书
测试系列

软件测试自动化

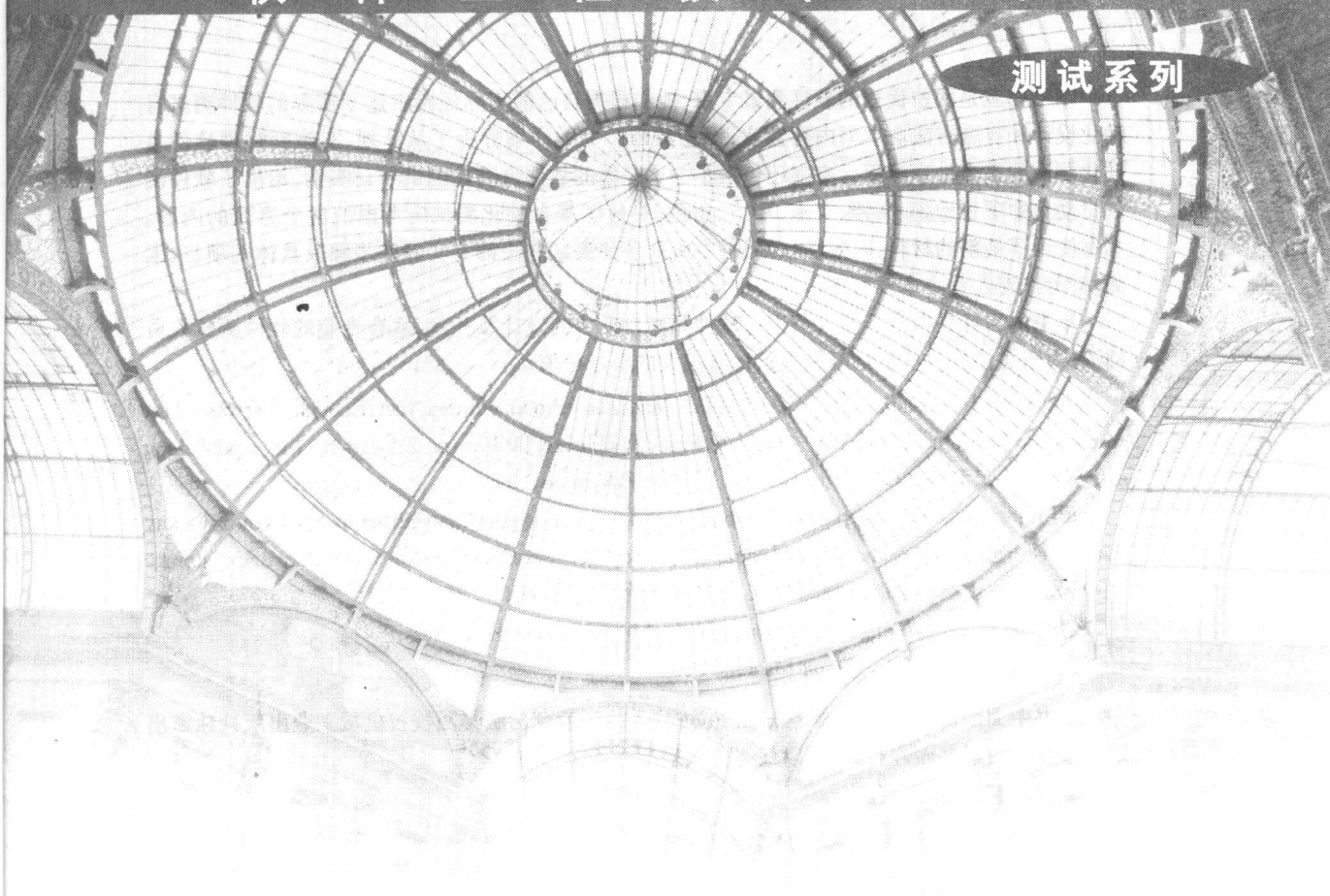
Just Enough Software Test Automation

（美） Daniel J. Mosley
Bruce A. Posey 著
邓波 黄丽娟 曹青春 等译

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

测试系列



软件测试自动化

Just Enough Software Test Automation

(美) Daniel J. Mosley
Bruce A. Posey 著

邓波 黄丽娟 曹青春 等译

机械工业出版社
China Machine Press

本书作者是两位优秀的软件测试专家，他们根据自己的实际经验讲述了实际的软件测试自动化设计和实施方法。本书内容翔实，概念清晰。书中详细分析了从计划、实施到管理软件自动化测试的整个处理过程。本书还包括了一个完整的软件自动化测试项目实例，给出了软件测试自动化中项目计划、文档、实施、环境、角色、责任等等自动化测试应考虑的各个方面的内容。本书使用了最新的材料，时效性很好，书中所列各种测试方法都有实际的步骤及具体的项目，因而操作性较强，是一本不可多得的好书。

本书的目标是指导开发人员进行软件测试自动化的设计及开发，适合专业的软件测试人员阅读，对于关注软件质量的开发人员也有很大的参考价值。

本书版权登记号：图字：01-2003-1001

图书在版编目(CIP)数据

软件测试自动化/(美)莫斯里(Mosley, D. J.)等著；邓波等译. －北京：机械工业出版社，2003.10

(软件工程技术丛书　测试系列)

书名原文：Just Enough Software Test Automation

ISBN 7-111-12818-4

Ⅰ.软…　Ⅱ.①莫…②邓…　Ⅲ.软件－测试－自动化　Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字 (2003) 第 068433 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：李　英

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2003 年 10 月第 1 版第 1 次印刷

787mm×1092mm　1/16　·15 印张

印数：0 001-5000 册

定价：29.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线电话：(010)68326294

译者序

在IT产业迅速发展的今天，硬件在质与量方面的飞速发展给人们留下了深刻的印象。相对而言，软件在量的方面同样发展迅速，上千万行的大型系统软件及百万行的应用软件已屡见不鲜，然而软件的质量却一直是令所有人头疼的问题，因为随着规模的扩大，对于质量的保证已经成为了一项异常艰苦的工作。面对这种挑战，人们提出了很多软件过程方法，力图通过严谨的开发过程保证软件的质量。这些努力当然是创造高质量软件产品的有效方法，但对软件进行测试仍然是保证软件质量最重要和最有效的方法。

软件规模的扩大同样给测试工作带来了新的问题，手工测试的速度太慢，效率太低。因此人们想到了自动化测试，想通过脚本程序的运行让测试工作自动进行。然而哪些测试内容可以自动化，什么时候进行自动化，这些都是常让测试人员困惑的问题，因为测试自动化是一种新的测试方法，没有什么经验可循。

本书的两位作者都是软件测试自动化的专家，在业界最早进行了数据驱动自动化测试的实验并取得了成功。在本书中，他们通过深入浅出的论述和实际的例子向读者介绍了软件测试自动化方法。不同于泛泛而谈，本书中介绍的方法都是以实际案例为基础进行论述和分析的，因而具有极强的操作性，相信各位读者看完全书后也会有自己的体会。对于读者而言，本书致力于讨论软件自动化测试的以下问题：

- 实施自动化测试时如何考虑测试过程？
- 什么类型的测试可以自动化实现？
- 自动化测试的时机如何确定？
- 自动化测试的内容是什么？
- 实施自动化测试时如何在时间和代价约束下考虑自动化测试的有效程度？

本书的目的是尽量提供对软件自动化测试概念和实施过程的清晰认识和理解。

本书共有10章：第1、2章介绍了软件测试自动化的概念及其应用场合；第3章讲述了自动化测试的准备工作，包括测试需求和测试数据的定义；第4章介绍了自动化测试脚本的相关知识；第5、6、7章分别详细讲述了单元测试、集成测试、系统/回归测试的具体方法和内容；第8章深入介绍了CSDDT的知识；第9章的内容为手工测试与自动化测试的结合；第10章讲述了自动化测试的管理。此外本书的附录还包括关于软件测试自动化的网上讨论、自动化实施的方案模板。这些对于读者进一步了解学习和实际应用都会有所帮助。

我们相信，本书不仅对软件测试人员提供有效帮助，对关心软件质量和软件过程控制的开发人员也有非常大的参考作用。希望本书的翻译出版能为提高国内软件自动化测试水平作出一点贡献。

全书由邓波、黄丽娟、曹青春、崔晓斐、王梅蓉、李昂、金旭、赵宏智、卢世凤、齐悦、郝敏、曹振南、韦华颖、刘学敏等进行翻译。本书最后由邓波统稿。由于时间仓促，且译者的水平有限，在翻译过程中难免会出现一些错误，请读者批评指正。

译　者

2003年5月

前　言

迄今，市面上已经有了很多关于测试自动化的书籍。它们所提倡的软件测试自动化方法各不相同，级别也有所差异。当然也有人尝试过将软件测试自动化的生命周期用最大众化的方式描述出来(参看第2章中Dustin的论述)。传统上，软件人员通常的做法是尝试用假设的模型来描述我们使用的过程。有些时候这种方法行得通，但有些时候它是行不通的。其问题在于没有经验性的证据能够证明在这些模型上使用的方法是否能够在现实世界中工作。对于大多数受到推荐的软件测试及软件开发方法来说，其基础只是轶事逸闻和项目管理的狂热，而这些是由那些所谓的信息系统专家发起并被首席信息官(Chief Information Officer，CIO)们推动发展的。

我们不相信自动化测试的生命周期。这只是一个人工构造的概念，而且我们发现它没什么用处。我们也不相信软件测试生命周期，我们所真正相信的是软件测试是属于软件开发过程中的一组紧密相关的活动。我们还相信好的软件测试需要有特定规范的项目管理以及一套它自己的操作技巧。测试也需要一套测试工程师在进行测试活动时能够依赖的工具。这些工具可以是与测试相关的产物，例如测试工程师可以依据的打印出来的测试场景，或者是在执行测试时工程师可以填写的打印出来的测试日志。

我们并非是说必须等到有了诸多经验证据之后才能接受并使用IS权威们所鼓吹的工具和技术。我们的意思是我们必须对这些工具和方法进行评估并去其糟粕。真正行之有效的经验是：当从业者(你和我们)尝试使用这些技术时，能够知道哪种技术可行而哪种不行。

本书第一作者Dan Mosley探讨了他在1985年使用的第一个自动化测试工具，那时的技术还相当简陋。他给本科生教授最早的软件测试课程时获得了该产品的评估版拷贝(圣路易斯的华盛顿大学，1985－1992)。在20世纪90年代中期，他与当时在Software Quality Automation (SQA)公司、现在在Rational Software公司(该公司后来吞并了SQA及其产品，其业务主要是提供自动化软件测试工具以及软件测试工程师所需要使用的东西，而译者现在得知Rational Software公司业已被IBM购买，但仍保持Rational品牌)的Eric Schurr通了长途电话，作了一次长谈。

他们讨论了一个好的自动化工具应有的功能和应该包括的特征。通过与Eric的联络，Dan获得并使用了SQA(现在是Rational)的自动化测试工具，那是在早期的1.0版后面出来的版本。现在最新的版本(写本书时)是“Rational Suite TestStudio 2002”。Dan对这个产品的使用经验表明，测试自动化不是对草率的测试工作的简单仓促的修正。此外，他的经验还证明了自动化测试不能代替手工测试。Glen Myers在20世纪70年代后期提出了软件测试的基本概念。直到今天，他的《Art of Software Testing(1979)》仍被认为是关于软件测试的最早著作。在我们进行手工测试以及构建自动化测试基础设施时我们仍要遵循他的一些建议。

我们对现在测试自动化实践的主要意见是其缺乏前期的测试计划和测试设计活动。我们重复地犯着那些软件开发人员早已成为痼疾的经典错误——我们在还没有对问题做合适的分析以及对测试基础设施进行设计时就开始了测试工作。这令我们想起了几年前的一个卡通片——某

位经理对程序员说:“你开始编写代码吧,我会去研究到底要做什么。”那些“测试”错误东西的自动化测试所带来的结果只有一个:不合适的测试更迅速地被执行完了。

任何自动化测试的最终目标都应当是与一套测试需求相对应的一套有计划的测试,这些测试需求反过来也能在自动化测试中体现出来。此外,测试工作的中心是测试数据,而不是测试脚本。这就是为什么有那么多人鼓吹将数据驱动自动化测试作为自动化实现的框架。其基本前提是数据应该驱动测试的进行并且应该体现被测程序(Application Under Test,AUT)的特征。测试脚本仅仅是数据传送工具。只有当设计出好的测试数据后,自动化测试才会有效。

自动化测试框架的另一个可操作目标是让测试脚本的维护量减至最少。各测试工具供货商多年主打的传统“捕获/回放”模型导致了高得异乎寻常的脚本维护量,因为测试数据在测试脚本程序中是以硬编码方式实现的。Mosley开发自动化测试脚本的最初经验明确地表明严格使用“捕获/回放”是行不通的。此外,工具内建的测试用例除了测试应用程序的图形用户界面(GUI)外,实际上是没有用处的。真正的功能测试需要测试工程师编写对AUT进行深度探查的测试数据。GUI测试能够用自动化测试做,并且用最少的工作量就能实现它的自动化。在实际测试中,我们通常用一个单独的测试脚本处理GUI对象,它会对所有GUI对象的属性进行基准和确认测试,这个测试脚本在每一个GUI显示屏都会执行。

自动化功能测试需要复杂全面的测试数据对AUT进行测试。这些测试数据必须能够重复产生演示重要系统特征的测试场景。因此,与其他测试相比,自动化功能测试更加复杂也更加困难。它要求测试工程师写出测试脚本的主要部分而不是记录测试。它还意味着要设计有效的测试数据。理解要测试什么(有一套书面的测试需求)并且设计出验证这些需求的数据是编写有价值的自动化功能测试的关键所在。

理解如何对测试结果进行验证与知道测试什么同等重要。自动化测试验证也具有数据依赖性。作为最主要的验证方法,自动化测试经常将基准数据捕获并存储起来,稍后与回归测试中捕获的相同数据进行比较。更全面的测试会在测试执行之前、之中和之后对数据库记录进行访问和操作。

功能强大的自动化测试框架必须提供测试计划、测试设计、测试构建、测试执行和测试结果验证的工具。一个有效的测试基础设施是建立在集成的中央知识库之上的,测试相关产物可以存储在这个知识库中并能够重用。这个基础设施还应提供可以定制的报告功能。

我们两人在合作此书之前就有过合作交流的经历。1996年我们碰到了一起,一同做出了到那时为止第一个真正成功的自动化测试项目。从那时起,我们在很多测试自动化项目中都相互合作。我们逐渐知道了实施成功的自动化测试项目需要什么,以及在IS研发和测试组织中推广自动化测试需要什么。我们知道什么可行,同样也知道什么不可行。

经过共同努力,我们实现了一个数据驱动自动化测试的范型,在此之前我们甚至还没有听说过“数据驱动测试”这一现在已经泛滥的业界名词(我们不知道Richard Strang在1996年STAR会议上发表的文章;参看第1章)。在他人刚开始探讨及写作相关文章时,我们已经率先对数据驱动方法进行了实现和完善。当然,对于任何新的热门技术来说,它并不是真正全新的,仅仅是被重新发现了。数据驱动测试也不例外。Berzer在《Software Testing Techniques(1983)》一书中就描述过“基于数据驱动的测试设计”。该书出版的时间是大型机时代的晚期以及PC革命的早

期，所以它的思路是关于大型机应用程序测试的。他提出的这种过程不是为了测试数据库，而是为了使用数据库结构生成测试数据。他认为该方法“最适合验证系统规约中所规定的功能需求”。通过很简单的步骤就可以将这种方法扩展，以包括那些基于被数据库表结构所支持的业务规则的测试。将测试 GUI 对象和它们行为的数据加上，你就得到了数据驱动测试。

这一时期我们还进行了结构化测试脚本的编写（也称作基于框架的测试）。这也不是什么新的技术。测试脚本就是使用修改过的普通编程语言（VB 或 C）编写的程序。它们的不同之处在于它们有一些为了软件测试任务而改造的特征。有很多文献都讲述了“结构化程序设计”的概念，例如功能分解、模块结合、模块耦合及模块（功能）重用。由于测试脚本包含的程序是要对另一个程序进行测试，所以它也要遵循被测程序所遵循的设计和构造规则。因此，结构化程序设计概念也适用于自动化测试脚本。

因为自动化测试脚本也背负着与其他软件编程相同的包袱，所以它们也有逻辑错误、死循环、硬编码变量等等问题，它们能够由过程和数据一同在测试脚本中实现。所有这些加起来就增加了相关测试套件的维护开销，就像测试脚本所测试的软件系统需要相应的维护开销一样。创建结构化的基于组件的测试脚本并让这些脚本与其所执行的数据相分离是创建有效软件测试自动化基础设施的惟一途径，这样可以达到最佳的测试精度并将测试维护减到最少。

最近还有人在开发一种高级的自动化语言，通过使用这种语言，非技术人员例如商业及产品分析人员或系统消费者也可以设计并实施自动化测试。这种尝试被认为是自动化测试发展的下一个阶段。然而迄今为止，我们还没有看到一种足够简化的测试脚本开发方法真正能够得到应用。只有在研究出一套可以支持公共的类 Java 的测试脚本语言的通用脚本库后，我们才会发现这样做的意义。然而，迄今的多数框架所支持的是精心设计的高级命令语言，而不是面向对象的语言。此外，支持库在提供给脚本编写者的功能方面还不够成熟。为了完成他们的测试需求，各机构必须在现存库的子例程和功能之上添加额外的代码。

由于我们是从业者，所以本书的目标是从开发者和用户的角度提供关于测试自动化的有用建议。这其中包括在设计和实现测试自动化基础设施时应该做什么的实际建议和不应做什么的一些告诫。此外还包括对现在流行的测试方法可以和不可以为你的测试做哪些事情的一些建议。

我们的例子是在 Rational Suite TestStudio 平台上开发的，但是我们认为它们可以很容易地被移植到其他自动化测试平台上使用。此外，本书还有一个 FTP 支持站点（www.phptr.com/mosley）。这个站点中既包括 ArcherGroup 的 CSDDT（Control Synchronized Data Driven Testing，控制同步的数据驱动测试）方法的模板文件，也包括 Carl Nagle 的 DDE（Data Driven Engine，数据驱动引擎）方法（适用于 Rational 环境）的模板文件，还有 Keith Zambelich 使用 Mercury Interactive的 WinRunner 自动化测试工具的完全数据驱动方法（Totally Data-Driven），该方法的基础是 Zambelich 的“测试计划驱动（Test Plan Driven）”框架并使用了他为WinRunner开发的工具包。通过使用这些资源，你可以轻松地立刻开始进行数据驱动自动化测试。

目　录

第 1 章
什么是测试自动化

本书并不论述怎样选择和实现自动化的测试工具包，因为目前有很多关于软件测试自动化工具选择的文章和书；本书也不是关于软件测试自动化的介绍性书籍。本书面向对测试自动化有经验的读者，面向那些在测试自动化实际应用方面遇到严重问题的读者。当然，在你所在的领域，测试自动化可能有也可能没有成功的实现，但无论何种情况，你肯定曾经遇到过测试自动化在操作上、政治上、文化上的缺陷。你需要的是一本指导性书籍，它可以提供一些实用的技巧、建议以及经过检验的方法。如果你想要的是这些，那么本书将适合你。

1.1 请拒绝新模型

“注意：不要新模型!”这句话反映的观点我们完全同意［14］。如同在前言中提到的，目前已经有太多的软件测试过程的模型［6、10、11］和自动化软件测试过程的模型［4、7、8、9、12、15］，包括软件测试自动化生命周期模型［2］。尽管这些模型是完全正确的，并且在讨论软件测试和测试自动化时，某些方面是起作用的，但它们对实际从业者作用甚微。

卡耐基梅隆（Carnege Mellon）大学的软件工程研究所（Software Engineering Institute，SEI）已经建立了软件测试管理关键过程域（Key Process Area，KPA），它对于达到软件过程能力成熟度模型（Capability Maturity Model，CMM）［11］第二级——可重复级（Repeatable）是必须的。软件成熟度模型具有普遍指导作用，但是它并不能从严格意义上定义

一个对测试工程师有用的过程。使用该模型时，会给测试管理者一种既亲切又模糊的感觉，但实质上测试过程活动并没有真正反映该模型。软件测试自动化生命周期模型也是如此。我们并不相信生命周期，相反，我们相信控制工作流程的过程。每一个测试团队都有其工作过程，有些时候它是一个混乱无序的过程，有些时候它是一个有组织的过程。

Krause为自动化的软件测试提出了四级成熟度模型［3］。在该模型中，他将软件测试成熟度模型［1］和SEI（软件工程研究所）的软件过程成熟度模型［4］（演变为CMM）联系起来了。四级成熟度模型中的四级分别为：附属级自动化（Accidental Automation），初始级自动化（Beginning Automation），主体级自动化（Intentional Automation），优化级自动化（Advanced Automation）。尽管这个模型从概念上描述了测试自动化，但实际上它并不能促进测试自动化的实现，它仅仅是描述作者在一些典型测试组织中注意到的一些问题。

Dustin、Rashka和Paul合作公布了自动化测试生命周期方法学（Automated Test Lifecycle Methodology，ATLM）——这是一种经过调整的结构化方法学，能确保自动化测试的成功实现［2］。它定义了一种四阶段方法学：自动化测试的决定；自动化测试的介绍；测试计划、设计和开发；自动化测试的执行和管理。

尽管这个模型从管理和控制角度来说是有效的，但是从测试自动化工程师的观点来看，它并不实用。对此，Powers提出了一些实际建议。这些建议将对那些负责构建和实现测试自动化框架的软件测试工程师很有帮助。他从一般意义上讨论了编程风格、命名规则以及用于编写自动化测试脚本的其他一些惯例［9］。

该书［9］还综合论述了数据抽象的原则。对自动化的软件测试来说，数据抽象是数据驱动方法的基础。在该书中，Powers还讨论了如何定义数据以及测试脚本如何使用数据的编码的可选方法。他认为“原则就是应尽量少地依赖于变量和常量的字面值，而尽量多地依赖于它们在测试中的意义、角色或用法”。他还提到“产生数据的约束”，他对此定义如下“……这种数据抽象最简单的形式就是用已命名的程序常量来代替字面值”。同时也提到“产生数据的变化”，并举例如下“……数据抽象的原则要求程序员用程序变量来代替字面值，如用‘Sfullnme’来代替‘John Q.Private’，并且在程序中只能给该变量赋值一次。字面值只出现一次意味着要修改该变量的值，只有一个地方可以对它进行编辑。”［9］

Powers 所做的论述可以立即给人这样一种印象：当讨论到自动化测试脚本的维护时，数据抽象可以带来益处。他进一步建议将这些值存放到知识库中以便于让测试脚本代码访问，“所需的只是可从中取值的知识库，以及可以实现从知识库中取值的编辑机制。”[9]

以下将讲述 Strang 的数据驱动自动化测试方法的原则。他的方法使用脚本框架从测试数据知识库中读取所需要的值，这需要用到包含了输入以及它的预期操作的数据文件。他的方法使数据抽象不仅存储字面值，还存储预期的结果值。这种方法既适合于手工数据生成，也适合于自动化的数据生成。而且测试脚本也必须以能够将错误结果和正确结果区分开的方式来编写。[12]

在本书的第 7 章和第 8 章讲述数据驱动方法时，会再一次提到 Powers 和 Strang 的工作。Archer Group 的控制同步数据驱动测试（Control Synchronized Data Driven Testing，CSDDT）是应用这里所讲述的概念的一种方法的例子。

Rational 软件公司已经颁布了 Rational 统一过程（Rational Unified Process，RUP），它包含了具体测试阶段，用于支持它的自动化测试工具包[10]。即使你不是 Rational 用户，测试过程信息也会给测试（甚至是手工测试）提供良好的基础。RUP 本身包含了软件开发所有方面（而不仅仅是测试）的过程文档。而且它的价格并不高——RUP 的光盘只卖 1 000 美元。RUP 测试方法最重要的一个方面就是它可以用来支持数据驱动的自动化测试框架。这是我们过去使用它以及本书介绍它的主要原因。

1.1.1 生命周期不是过程

到目前为止，本书中所引用的作者和其他业界权威所采用的测试方法存在着和所有生命周期模型一样的问题，那就是——它们是面向管理层的，而不是面向从业者的。而且，生命周期方法对操作过程（也可称之为自动化的测试过程）起的作用极小。其他的一些方法，如数据驱动自动化测试，正如一些作者批评的那样，虽然提出了很多方法和技巧，但却难以应用到日常的自动化测试活动中。这种直线思维所产生的模型同样给测试管理者一种又亲切又模糊的感觉，正如上面所提到的测试成熟度模型一样。

尽管生命周期模型致力于成为经验模型，但它在自动化测试上的表示并不是在演绎基础上发展起来的。它是基于归纳推理的一种理论，归纳推理又多半以多个案例的证据为基础，正如信息系统（IS）文献所倡导的大

多数模型一样。另一方面，非管理性技术是基于演绎推理的，非管理性技术是操作性的，而不是管理性的，它应用于自动化过程的具体任务。数据驱动测试是非管理性技术的一个例子。这些技术是经过从业者不断尝试和受挫逐渐形成的——这是一种沿用至今的传统工程方法。

1.1.2 工具不是过程

关于CSST技术公司的一个小规模调查结果表明：40%（258份返回问卷中的102份）的人认为软件测试方法或过程的实现最大程度地促进了他们的测试工作；24%（63份）的人认为改进的软件需求文档是最重要的促进因素；19%（50份）的人认为软件测试标准的实现是最重要的方面；10%（25份）的人认为改进的测试计划才是最重要的因素；只有7%（18份）的人认为花更多的时间来测试将会促进他们的工作。

购买软件测试工具包并不意味着实现了软件测试过程。过程是一系列的步骤，通过这些步骤得到所要达到的目标或生产出产品。这些过程实现测试活动，测试活动导致测试执行和测试产物的生成。自动化的软件工具支持现存的测试过程，并且，当过程是混乱过程时，它能强加一些必需结构到测试活动中。软件测试工具实现失败的一个主要原因就是在购买工具前几乎不存在任何测试过程。

当我们设计和创建自动化测试时，甚至不曾考虑到过程。我们考虑到的只是：任务、进度、人员的分配。对我们来说，软件测试自动化应该达到帮助我们有效工作的程度。如果我们没有商业化的软件测试工具可用，我们也会创建自己的工具或者是使用客户日常在工作站上装载的桌面工具。

图1-1描述了一个测试过程，这个测试过程松散地基于RUP测试方法[10]，它是为我们的一个客户定义的。我们的这种过程方法不同于Rational公司颁布的RUP，因为我们将测试脚本设计视为测试实现的一个部分，然而在RUP中这被认为是测试设计行为。我们之所以不同于RUP是因为我们相信测试设计所需要的技巧中并不包括以前的编程经验，但测试实现需要以前的编程经验。

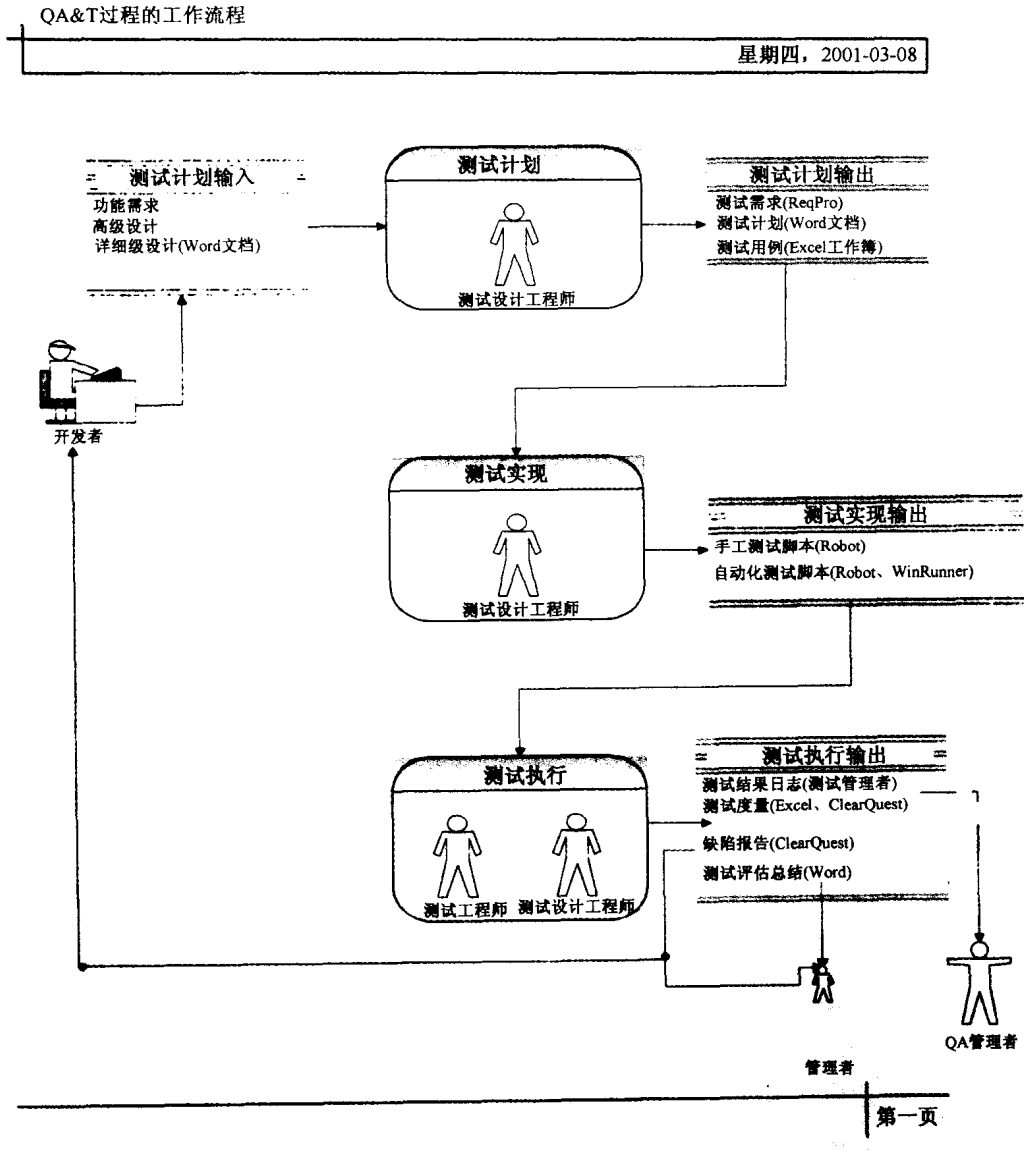


图 1-1 质量保证和测试（QA&T）过程

1.2 自动化需要达到什么程度才足够

这个问题在自动化测试工具发展的最初阶段就有人问了。工具销售商已经给我们提供了一个观点，并且业界专家也给我们提供了各种不同的观

点。最初，销售商提供基本的捕获/回放工具，这些工具已逐渐发展成了一些复杂的高度集成的测试套件。他们似乎想让从业者来决定基本的捕获/回放模型之外的一些东西。测试自动化方面的专家写过很多的文章和书籍，他们引用成功的以及失败的自动化测试来做研究，最后在我们必须做什么上稍微达成一致意见，但是就我们如何来做并没有任何进展。在这一节，我们将给出我们关于如何做测试自动化的观点。我们认为业界就该做什么已经争论太久。我们一直拥有一个能使用的自动化框架工作原型，直到工具套件达到一个新高度以及直到它们更复杂。

为了知道自动化程度需要达到什么程度才足够，我们必须了解这些领域：能自动化的软件测试过程以及应该自动化的软件测试过程。测试工具和测试过程是不相同的。工具是用于促进测试过程的。工具能被用于实现一个过程并执行测试过程的各种规范。在很多情况下，工具自带的内建程序可以被理解为过程。然而，它们往往也是不完整的，不能正确反映过程。最好的软件测试工具是你能够将它和你的测试需求达成一致。而且它们提供高度可自定义的工作流程和跟踪报告能力。

什么测试类型能够自动化呢？它们包括：单元测试、集成测试和系统测试。对自动化系统测试进一步分类如下：安全测试、配置测试和负载测试。自动化回归测试贯穿于整个开发过程的单元测试、集成测试和系统测试，并使用最大和最小发布版本的系统产品分别测试。

我们应该考虑测试过程的哪些方面呢？一般包括以下几个方面：测试计划、测试设计、测试构建、测试执行、测试结果的捕获和分析、测试结果验证和测试报告。还有一些活动是和测试活动紧密相连的，它们包括问题（缺陷）跟踪和解决、软件配置管理以及软件测试度量。总之，测试过程的这些活动是密不可分的，就好像软件开发过程一样，由好的项目管理技术粘结在一起。

所有领域的自动化水平应该达到这样一种程度，它能够根据时间和成本适应于你的组织。你实现的自动化程度越高，你的测试过程就越好越有效。这种观点总是对的，只要你的工具是适合的，并且被正确地实现。在这里，实现（implement）是指一个集成的测试自动化框架已经被创建并在使用中。

1.3 测试过程的各方面

让我们依次了解测试过程的各个组成部分。