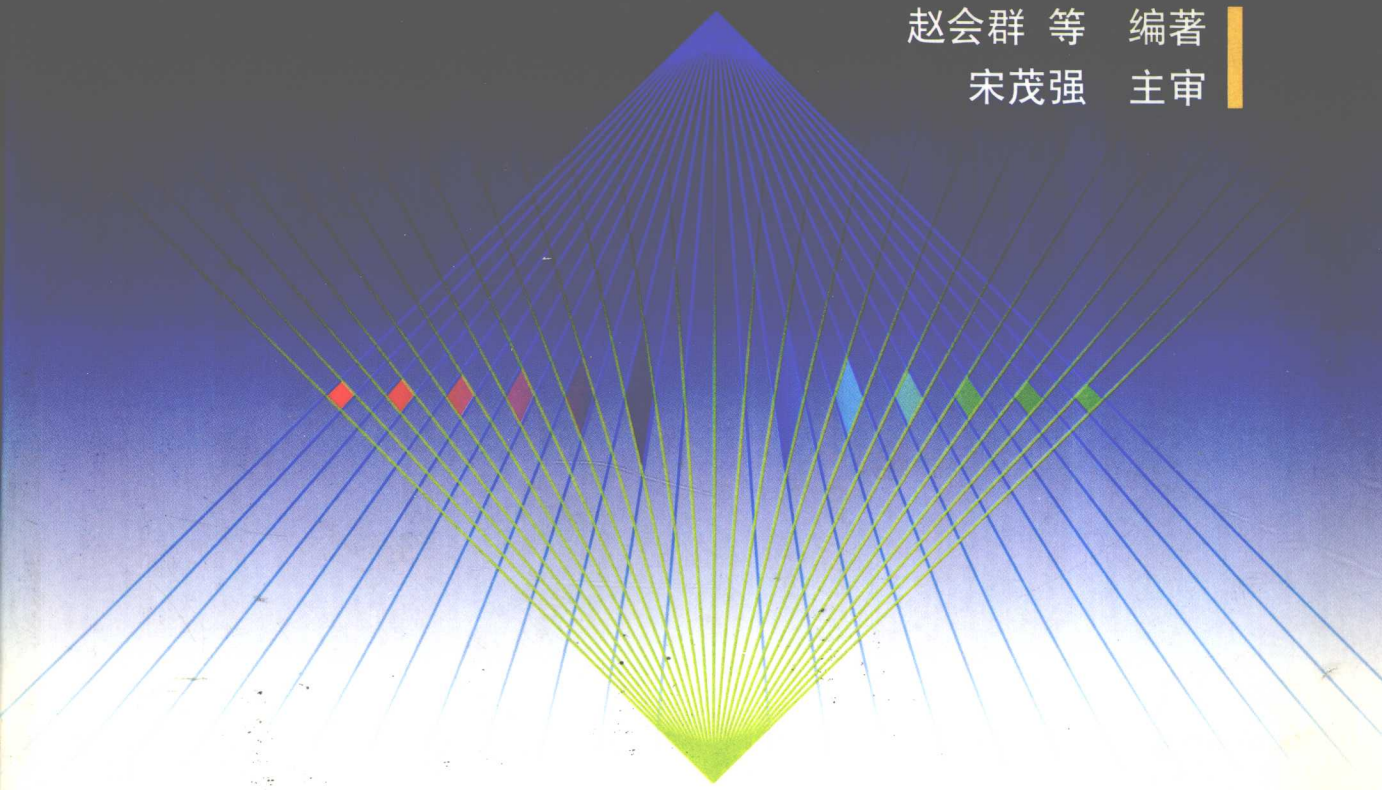
通信软件测试技术基础

赵会群 等 编著
宋茂强 主审



人民邮电出版社
POSTS & TELECOM PRESS

通信软件测试技术基础

赵会群　等编著

宋茂强　主审

人 民 邮 电 出 版 社

图书在版编目（CIP）数据

通信软件测试技术基础/赵会群等编著. —北京：人民邮电出版社，2004.2
ISBN 7-115-11860-4

Ⅰ.通… Ⅱ.赵… Ⅲ.通信软件—测试 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字（2003）第 115694 号

内 容 提 要

本书首先介绍软件测试的基本概念、基本内容和测试方法的分类，作为软件测试的基础，重点介绍了软件测试的白箱法和黑箱法。在此基础上深入讨论了通信协议软件测试的基本方法，全面介绍协议软件测试的建模工具树表描述语言 TTCN 和时序说明语言 LOTOS/E-LOTOS。TTCN 和 LOTOS/E-LOTOS 是国际标准化组织颁布的协议软件测试的形式化建模技术标准，在协议软件测试领域中占有主导地位，大多数协议软件测试工具都是基于上述两大标准设计开发的，所以 TTCN 和 LOTOS/E-LOTOS 也是本书的重点。为了使读者能够更好地理解 TTCN 和 LOTOS/E-LOTOS，本书给出了一些典型的应用实例。

本书内容新颖，参考性强，可以作为大专院校计算机专业、通信专业高年级学生、研究生和软件学院学生的教材，也可以作为通信软件开发人员、软件测试技术的科研人员的参考书。

通信软件测试技术基础

◆ 编　　著　赵会群　等
主　　审　宋茂强
责任编辑　杨　凌

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www. ptpress.com.cn
读者热线　010-67129258
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：14.25
字数：339　　2004 年 2 月第 1 版
印数：1-4 000 册　　2004 年 2 月北京第 1 次印刷

ISBN 7-115-11860-4/TN・2206

定价：23.00 元

前　言

计算机技术已经越来越广泛地应用于国民经济和国防建设的各个领域。然而，在实际应用中，由于计算机软件缺陷而造成计算机系统故障并导致严重后果的事例屡见不鲜。因此，如何保证软件产品的质量就成了必须解决的一个问题，而对软件进行有效的测试就是解决软件质量问题的方法之一。

目前介绍通信软件测试的书籍并不多见，虽然偶尔可以见到一两本新书，但这些书有相当多的内容是重复的。随着软件技术的发展，软件测试技术和方法也在不断地更新。尤其是随着网络技术的发展，以实现网络协议为内容的软件大量出现，这样就需要一本能够全面、深入介绍软件测试技术的参考书来满足广大软件测试技术人员和软件专业学生的需求。正是基于这种考虑，作者在多年从事软件测试工作和软件测试技术学习的基础上编写了本书。

本书从软件测试技术的角度出发，讨论了软件测试中所采用的基本方法，以及这些技术和方法在实际测试工作中的应用。全书共分 6 章。第 1 章为软件测试概述，全面介绍软件测试的基本概念、基本方法和软件测试所涉及的主要内容，本章内容是对软件测试基础知识的浓缩。第 2 章为软件测试基础，介绍了几种典型的软件测试的方法，主要介绍软件测试白箱法和黑箱法，而且还介绍根据上述两种方法提出的软件测试灰箱法。第 3 章为树表描述语言（TTCN，Tree Tabular Combine Notation）。TTCN 是 ISO/IEC 颁布的协议一致性测试基本框架和方法标准（ISO/IEC 9646）中的第三部分。本章介绍 TTCN 的基本语法和基于 TTCN 的协议软件测试方法。第 4 章为时序说明语言 LOTOS/E-LOTOS。这一章首先介绍一种基于进程代数的软件描述形式化技术——LOTOS/E-LOTOS，它是 ISO 颁布的形式化技术标准，是一种新的协议软件描述和测试技术，接着再在 LOTOS/E-LOTOS 的基础上讨论了基于 LOTOS/E-LOTOS 的软件测试技术。第 5 章为 TTCN 应用举例，介绍 TTCN-3 在 SIP 和 OSP 以及 IPv6 测试中的应用。第 6 章为 LOTOS/E-LOTOS 应用举例，介绍 LOTOS/E-LOTOS 在安全协议测试和病态路由检测中的应用。第 5、6 章是应用篇，其内容有相当一部分是作者自己的研究成果。

本书内容新颖，可参考性强。书中虽然有很大篇幅介绍了 ISO 的标准，但注意抽象和具体相结合，理论与实际相结合，使读者既能在理论上有所提高，也能在实践方面得到收获。

本书由赵会群博士编写，宋茂强教授主审。在本书的编写过程中，北方工业大学信息学院的孙晶副教授参加了第 1 章和第 2 章的编写工作，研究生吕春和刘娟参加了本书的文字整理工作，北京邮电大学软件学院雷友励博士为本书提供了大量的参考资料。在此向他们表示衷心的感谢。

作　者

2003 年 10 月于北京

前　言

计算机技术已经越来越广泛地应用于国民经济和国防建设的各个领域。然而，在实际应用中，由于计算机软件缺陷而造成计算机系统故障并导致严重后果的事例屡见不鲜。因此，如何保证软件产品的质量就成了必须解决的一个问题，而对软件进行有效的测试就是解决软件质量问题的方法之一。

目前介绍通信软件测试的书籍并不多见，虽然偶尔可以见到一两本新书，但这些书有相当多的内容是重复的。随着软件技术的发展，软件测试技术和方法也在不断地更新。尤其是随着网络技术的发展，以实现网络协议为内容的软件大量出现，这样就需要一本能够全面、深入介绍软件测试技术的参考书来满足广大软件测试技术人员和软件专业学生的需求。正是基于这种考虑，作者在多年从事软件测试工作和软件测试技术学习的基础上编写了本书。

本书从软件测试技术的角度出发，讨论了软件测试中所采用的基本方法，以及这些技术和方法在实际测试工作中的应用。全书共分 6 章。第 1 章为软件测试概述，全面介绍软件测试的基本概念、基本方法和软件测试所涉及的主要内容，本章内容是对软件测试基础知识的浓缩。第 2 章为软件测试基础，介绍了几种典型的软件测试的方法，主要介绍软件测试白箱法和黑箱法，而且还介绍根据上述两种方法提出的软件测试灰箱法。第 3 章为树表描述语言（TTCN，Tree Tabular Combine Notation）。TTCN 是 ISO/IEC 颁布的协议一致性测试基本框架和方法标准（ISO/IEC 9646）中的第三部分。本章介绍 TTCN 的基本语法和基于 TTCN 的协议软件测试方法。第 4 章为时序说明语言 LOTOS/E-LOTOS。这一章首先介绍一种基于进程代数的软件描述形式化技术——LOTOS/E-LOTOS，它是 ISO 颁布的形式化技术标准，是一种新的协议软件描述和测试技术，接着再在 LOTOS/E-LOTOS 的基础上讨论了基于 LOTOS/E-LOTOS 的软件测试技术。第 5 章为 TTCN 应用举例，介绍 TTCN-3 在 SIP 和 OSP 以及 IPv6 测试中的应用。第 6 章为 LOTOS/E-LOTOS 应用举例，介绍 LOTOS/E-LOTOS 在安全协议测试和病态路由检测中的应用。第 5、6 章是应用篇，其内容有相当一部分是作者自己的研究成果。

本书内容新颖，可参考性强。书中虽然有很大篇幅介绍了 ISO 的标准，但注意抽象和具体相结合，理论与实际相结合，使读者既能在理论上有所提高，也能在实践方面得到收获。

本书由赵会群博士编写，宋茂强教授主审。在本书的编写过程中，北方工业大学信息学院的孙晶副教授参加了第 1 章和第 2 章的编写工作，研究生吕春和刘娟参加了本书的文字整理工作，北京邮电大学软件学院雷友勋博士为本书提供了大量的参考资料。在此向他们表示衷心的感谢。

作　者

2003 年 10 月于北京

目　录

第1章　软件测试概述

计算机技术已经越来越广泛地应用于国民经济和国防建设的各个部门，并以不可阻挡之势渗透到人们工作和生活的各个领域，尤其是在航天、航空、核能、通信、交通、金融等一些关键领域中，计算机的作用更加至关重要。同时，这些领域对计算机软件的可靠性和安全性也有严格的要求。近年来，由于软件错误而造成经济损失、导致严重后果的事例屡见不鲜，因此，如何保证软件产品的质量和可靠性就成为人们必须解决的一个重要问题，而软件测试便是保证软件质量的一个重要手段。据统计，国外在软件开发中开发费用的近一半甚至更多是用于软件测试，由此也可以看出软件测试在软件开发中的重要地位。

本章简单地讲述了有关软件测试的概念、方法和过程等方面的基础知识，以使读者对软件测试有一个比较全面的了解，并为进一步讨论软件测试技术奠定基础。

1.1　软件故障与软件测试

在计算机故障中，有相当一部分是软件故障。下面让我们来看两个例子。

例 1：英特尔奔腾浮点除法软件故障

在计算机的“计算器”程序中输入以下算式：

（4195835/3145727）×3145727−4195835

如果答案是 0，说明计算机没有问题；如果得出的结果不是 0，则说明计算机的工作不正常。看起来这不应该是个问题，可实际上它就发生了。

1994 年 12 月 30 日，美国 Lynchburg 大学的 Thomas R.Nicely 博士在一台奔腾 PC 上做除法运算时发现，上面的算式不等于 0。后来他把这一个惊人的发现在 Internet 上发布出去，引起了一场风暴，成千上万的人都发现了同样的问题。那么，是什么原因造成这样的算式计算错误呢？这是由固化在奔腾 CPU 上的运算器芯片中的软件故障所致。

例 2：千年虫（Y2K）问题

所谓的千年虫问题，是指由于计算机用于表示年份的位数不足而带来的计算机系统缺陷。下面通过一个有关千年虫的传说来说明千年虫的危害。

20 世纪 70 年代一个叫 Dave 的程序员，他负责本公司的工资系统。他使用的计算机存储空间很小，这就迫使他尽量节省每一个字节。Dave 自豪地将自己的程序压缩得比其他人小。他使用的一个方法是把 4 位数日期缩减为 2 位（例如 1973 年为 73）。因为工资系统极度依赖数据处理，Dava 节省了可观的存储空间。但 Dava 并没有想到这是个很大的问题，他认为只有在 2000 年时程序计算 00 或 01 这样的年份时才会出现错误。他知道那时会出问题，但是在 25 年之内程序肯定会更改或升级，而且眼前的任务比未来更加重要。1995 年，Dava 的程序仍然在使用，而 Dava 退休了，谁也不会想到进入程序去检查 2000 年的兼容性问题，更不用说去修改了。

关于 Y2K 问题的说法不一，但根本的问题是用 2 位表示年份的问题。这是一个十分典型的软件设计缺陷。Y2K 问题涉及 4 个方面：硬件、操作系统、应用软件及数据。

有关 Y2K 的例子有很多，这给计算机产业带来了一次震惊和恐慌。许多国家和大的计算机公司都动用了大量的人力和物力来解决千年虫问题，尤其是解决关系到国家安全、国家支柱产业正常运转和与百姓生活息息相关领域的计算机系统的千年虫问题。

微软作为全球最大的软件供应商，其产品涵盖了操作系统、应用软件及数据等领域，而在 PC 平台上形成最为广泛的应用。关于这一问题，微软对其产品进行了全面的兼容性测试。

微软自 1996 年起开始涉及有关 Y2K 问题的研究，对此，微软采取的是完全对外公开的策略，其中包括产品及其他任何有关 Y2K 的信息。作为一家既面向企业用户也针对广大个人用户的软件产品供应商，微软认为解决 Y2K 的首要问题是对产品进行全面深入的 2000 年兼容性测试。

首先，需要找到一种统一的方法来对不同的产品进行 2000 年兼容性测试；同时，由于微软的产品在全球得到了极为广泛的应用，因此要对产品进行不同语言的测试。截止到 1999 年底，微软共测试了 4052 种产品，这可谓迄今为止历史上最大的软件测试工程之一。其中，97%达到了兼容性要求，3%不兼容。而不兼容的产品基本上都是老产品，像 DOS 版本的 WORD5.0。同时，在整个测试过程中并未做任何推断性测试，即不能在未对某种语言进行实际性测试的前提下，从其他语言的同种产品的测试结果进行推断（如假设简体中文的测试结果没问题，则简单推断韩文也不存在任何问题）。在计算机系统采用的语言中，英文及其他欧洲语言只占 5%，而 20%以上是采用 UNICODE 编码的双字节语言（如大多数亚洲国家的语言）；同时，世界各地不同的民族采用不同的历法，这些都是测试需要考虑的问题。测试时应尽力确保最新产品和最大量使用的产品以及采用关键技术，如 ODBC 的产品的 2000 年兼容性测试，然后再测试客户有特别需求的，采用某项专有技术的产品。

从上面的两个例子中可以看出，软件缺陷是造成软件故障的主要问题，也是软件测试的主要对象。那么，什么是软件缺陷故障？什么是软件故障？软件测试定义为何？它们之间的关系为何？下面给出一组有关软件测试的相关术语，先明确它们之间的关系，进而给出软件测试的概念。

缺陷（bug）	偏差（variance）
缺点（defect）	失败（failure）
问题（problem）	矛盾（inconsistency）
错误（error）	事故（incident）
异常（anormaly）	谬误（fault）

在上述名词中，有一些含义相近，属于一个范畴。第一类是缺陷、缺点和偏差，它们是一类意义相近的概念，我们不妨把它们统称为缺陷。它们都是软件开发过程中潜在的缺陷，这些缺陷可能在软件投入运行后出现，因而使得软件的性能和可靠性等方面与系统的设计需求不符。有时这些问题可能不出现，软件的性能和可靠性并不会因为它们的存在而受到影响。第二类是错误、谬误、问题、异常和矛盾等，我们把这一类问题统称为错误。这类错误与软件运行状态有关，它们是在软件运行过程中可观测到的软件错误。这些问题出现的原因是由软件缺陷所致。最后一类是失败、事故或灾难等，我们把这类统称为失败。这是软件运行给

用户造成的损失的一类软件故障，它强调软件失败的结果。失败的直接原因是软件系统存在软件错误。并不是所有的软件错误都会导致软件失败，如果对软件错误加以适当的控制，软件错误可以导致安全。

综上所述，软件故障大体上可分为三种类型，每一类对应软件生命周期的不同阶段，贯穿整个软件开发和使用的全部过程。其中，第一类缺陷是软件故障的根源，后两类故障是软件缺陷的直接后果。所以，在软件开发过程中，发现和排除软件故障是一项长期艰苦的工作，而这一项工作的基础是加强软件设计时设计缺陷的检测。

那么，什么是软件测试呢？所谓软件测试，是指为了评价一个软件系统的质量和发现错误而从事的一种工作过程。从软件测试作为软件的执行过程来看，可分为局部软件的局部运行和全部运行；从运行的环境来看，可分为仿真运行和实际运行。这就存在一个软件测试中的方式和方法的问题。而方法又与采用的技术相关，技术不同，方法也不同。所以，软件测试技术是测试的关键。

从软件故障的分类中可以看到，软件的故障分布在软件开发的全过程，所以软件测试也就伴随软件开发和使用的整个过程，在下一节中我们将分析软件测试与软件生命周期的关系。把握软件测试与开发过程的阶段关系，为有针对性地开展软件测试奠定基础。

1.2　软件测试与软件开发过程

软件开发过程中的各种活动构成了软件开发的生命周期，而随着这些活动的组织方式和方法的不同，就构成了不同的软件开发生命周期模型。然而，无论是什么样的生命周期模型，软件开发无一例外地要经历从软件需求分析到软件测试这样一个过程。也就是说，虽然软件开发的生命周期模型有所不同，但软件开发的阶段性始点和终点是相同的，而且软件测试是不可缺少的一项工作。

软件开发的生命周期并不是独立存在的，它是整个计算机系统产品生命周期的一部分。在产品的生命周期内，软件被维护和纠错。当产品就是软件本身时，软件的维护和测试也是相当复杂的一项工作。不同的软件模块被组合成一个大的软件系统，这给软件测试工作带来了一定的难度。

有许多不同的软件生命周期模型，都需要对它们进行测试。本节讨论各种软件开发生命周期模型与软件测试的关系，从而进一步明确软件测试在软件开发中的重要作用，为在不同软件开发方法下灵活运用软件测试的方法和技术奠定基础。

1.2.1　顺序生命周期模型（Sequential Lifecycle Models）

所谓的顺序生命周期模型，是把软件开发的整个过程定义为有序的开发活动序列，随着开发工作的进展，软件生命周期的状态随之迁移，如图 1-1 所示。顺序模型也称为 V 模型或瀑布模型。

对于瀑布模型（如图1-2所示），也有许多变形。这些模型可能随着软件开发需求的不同增加新的状态，这些状态有着不同的边界。下面给出一组典型的开发状态描述。

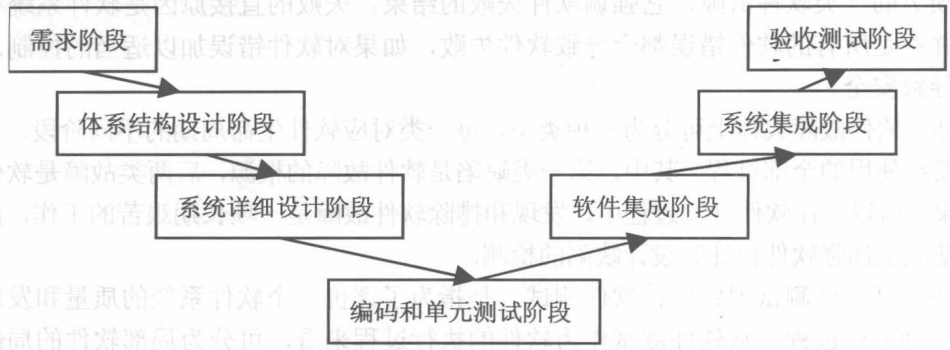


图 1-1　V 生命周期模型

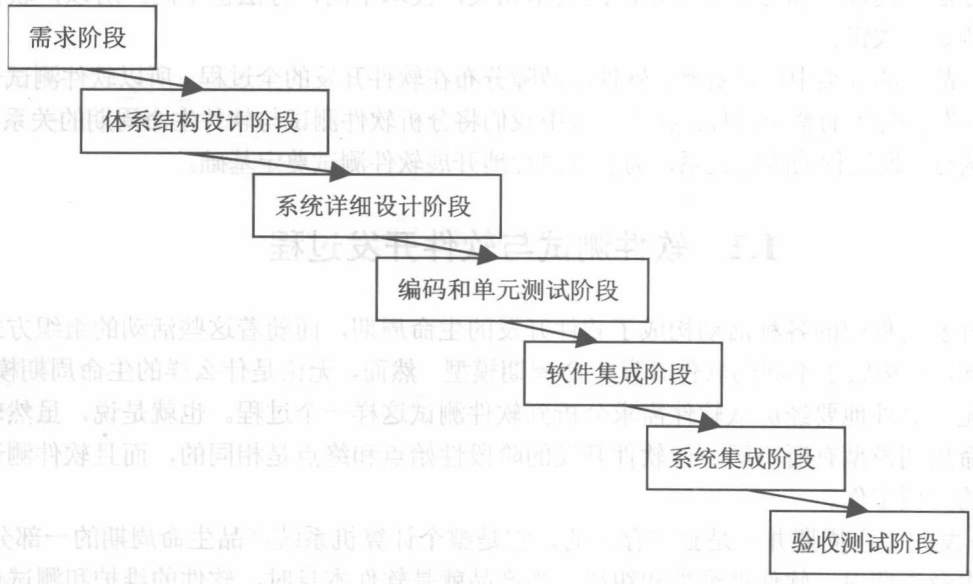


图 1-2　瀑布生命周期模型

- **需求阶段（Requirements phase）**。在需求阶段需要对用户的需求进行分析，将要开发的软件进行严格的定义，这种定义应该是非二义性的。
- **体系结构设计阶段（Architectural Design phase）**。在该阶段对软件系统的体系结构进行分析、设计和定义，进而说明体系结构中组件和组件之间的联系。
- **系统详细设计阶段（Detailed Design phase）**。在该阶段对构成系统的各组件给出详细的设计和说明。
- **编码和单元测试阶段（Code and Unit Test phase）**。在该阶段对设计的组件进行编码，并验证组件代码与详细设计阶段定义的组件细节的正确性。
- **软件集成阶段（Software Integration phase）**。在该阶段把已经测试过的组件组装到一起，并对集成系统进行测试，直到由组件构成的软件满足设计要求为止。
- **系统集成阶段（System Integration phase）**。在该阶段对软件及其他系统部件进行集成，并进行系统测试，直到系统正常工作为止。
- **验收测试阶段（Acceptance Test phase）**。在该阶段将按照系统分析、设计定义的各

个方法对系统进行测试，从而检验系统开发的正确性。

在上述系统生命周期中，前三个阶段是系统的定义阶段，后面四个阶段都需要对阶段成果进行测试。测试工作同样需要设计和说明，在生命周期的各个层面中，均需要定义该层面的测试点。

1.2.2　渐进式（Progressive Development）开发生命周期模型

顺序生命周期模型是一个理性化的软件开发生命周期模型。在实际开发过程中，可能情况要复杂得多。这样根据特殊情况设计具体的软件开发生命周期模型是常见的。比如软件需求往往随着开发工作的进行而不断地扩充，或者为了避免开发周期过长，而先开发一个中间系统投入运行，等等，这些特殊情况改变了原有生命周期的各个阶段的分配。

下面我们来介绍另一个软件开发生命周期模型，这个模型称为渐进生命周期模型或阶段生命周期模型，如图 1-3 所示。

软件开发过程中的一个矛盾是，开发时间过长，而工期往往要求很短。解决此问题的一个方法是在两者之间折衷处理，首先用较短的时间开发一个中间系统，而功能相对比设计要求少一些。这个中间系统还需要进一步扩展，直到满足所有的功能需求。中间软件的开发可以有效地减少软件开发风险。我们把通过这种开发方法所建立的软件开发生命周期称为渐进式（Progressive Development）开发生命周期模型或阶段实现（Phased Implementation）生命周期模型。

在渐进式模型中，每一个开发阶段仍然服从顺序模型中的规范，而整个生命周期的阶段可能有所增减，它的实际阶段取决于实际开发过程。

每一个提交的软件都要经过测试，以检验软件是否满足设计要求。测试工作分布在软件开发的各个阶段，不但要对各个阶段的成果进行测试，而且还要对各个阶段的集成成果进行检验。如果这种测试工作能够在控制范围内完成，该工作将不会影响整个生命周期。然而，测试中有时发现的问题是严重的，软件需要重新开发。这是最坏的情况，在这种情况下，整个软件的生命周期将被打乱。无论是哪种情况，测试都需要花费很多时间和精力。所以，我们应该尽可能地及早发现问题，减少不必要的开发代价。一个办法就是利用原型法构造系统的原型。系统的原型是提供快速实现系统设计功能的方法，开发人员和用户可以在系统原型上测试所开发的系统是否满足设计要求。一个原型系统最终要进一步扩展成一个实际的系统，但从原型系统到实际系统的过渡过程中仍然需要测试。

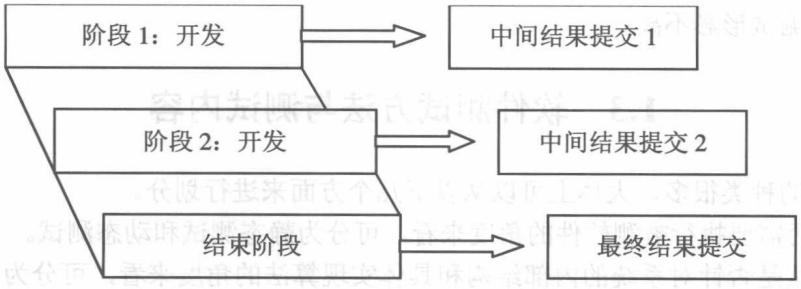


图 1-3　渐进式开发生命周期模型

1.2.3 迭代生命周期模型（Iterative Lifecycle Model）

在迭代生命周期模型中，开发工作的最初并不要求对软件需求进行详细的说明，而是随着软件开发工作的进行，逐渐辨别所开发软件的需求，并加以说明。上述过程可能要重复多次，产生新的软件版本。迭代模型就是这样一种开发模型，其生命周期表现为 4 个不断迭代的阶段，如图 1-4 所示。

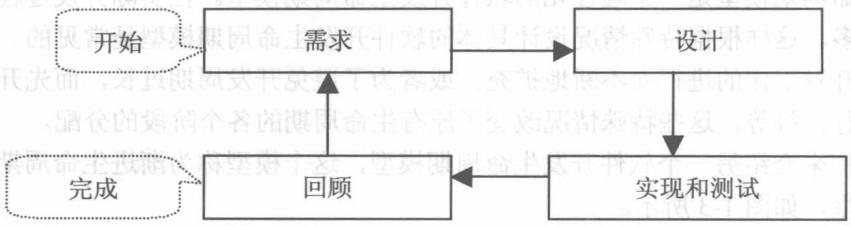


图 1-4 迭代生命周期模型

- **需求分析阶段**。该阶段对软件的需求进行收集并分析。在此基础上，不断的迭代将会得到最终的需求定义。
- **设计阶段**。该阶段根据需求进行设计，设计可能是新的设计，也可能是对早期设计的扩展。
- **实现与测试阶段**。该阶段的中心任务是对软件进行编码、集成和测试。
- **回顾阶段**。在该阶段要对软件进行评估，回顾软件的需求，改变或增加新的需求。

对于迭代中的每一个周期循环，必须决定软件中哪些部分或全部组件需要在下一次周期循环中被保留，或是被摒弃掉。最终将达到一个目标，就是软件需求被满足，这时软件被提交。或者该软件不可能达到设计要求，而不得不从头开始。

迭代模型可以形象地比喻为通过连续的逼近方法来开发软件。这有一些像数学中的逼近法，它通过不断的逼近最终使问题求解。然而，在数学中，逼近有时没有解，每一次迭代都在可行解的左右摆动，甚至是发散的；迭代的次数可能会很多，甚至是不可能的。那么在软件开发中，是不是也这样呢？情况十分相似。软件中的错误会使得软件生命周期没有尽头，这也是一种发散。

在迭代模型中，成功的关键是在周期的每一个阶段和循环中都要对结果进行严格的测试。模型的前 3 个阶段是顺序模型的浓缩，每一个循环都要对软件进行单元测试，随着生命周期的延续，测试形影不离。

1.3 软件测试方法与测试内容

软件测试的种类很多，大体上可以从以下几个方面来进行划分。

- 从是否需要执行被测软件的角度来看，可分为静态测试和动态测试。
- 从测试是否针对系统的内部结构和具体实现算法的角度来看，可分为白盒测试和黑盒测试。
- 从测试范围角度来看，可分为单元测试、系统测试、集成测试等等。