JIYU PYTHON YUYAN DE
RUANJIAN CESHI JISHU

张　民／编著

基于Python语言的软件测试技术

电子科技大学出版社
University of Electronic Science and Technology of China Press

基于Python语言的软件测试技术

JIYU PYTHON YUYAN DE
RUANJIAN CESHI JISHU

张　民／编著

电子科技大学出版社
University of Electronic Science and Technology of China Press

·成都·

图书在版编目(CIP)数据

基于python语言的软件测试技术 / 张民编著. -- 成都：电子科技大学出版社，2019.5
ISBN 978-7-5647-7054-9

Ⅰ. ①基… Ⅱ. ①张… Ⅲ. ①软件工具—程序设计② 软件—测试 Ⅳ. ①TP311.561②TP311.55

中国版本图书馆CIP数据核字（2019）第101630号

基于python语言的软件测试技术

张　民　编著

策划编辑　杨仪玮
责任编辑　杨仪玮

出版发行　电子科技大学出版社
　　成都市一环路东一段159号电子信息产业大厦　邮编 610051
主　　页　www.uestcp.com.cn
服务电话　028-83203399
邮购电话　028-83201495

印　　刷　四川煤田地质制图印刷厂
成品尺寸　185mm×260mm
印　　张　20.75
字　　数　488千字
版　　次　2019年5月第一版
印　　次　2019年5月第一次印刷
书　　号　ISBN 978-7-5647-7054-9
定　　价　78.00元

目录 MULU

理 论 篇

实 战 篇

理 论 篇

第一章 绪 论

随着越来越多的人使用计算机技术，软件工程所涉及的内容也越来越多，其复杂性也不断增加，人们也逐渐认识到软件质量的重要性。因此，在软件的开发阶段，开发人员开始使用各种用于保证软件质量的测试软件，以避免开发过程中可能会出现的错误和缺陷。对于一些规模比较大的软件，一些细小的错误可能导致财产的重大损失。为了防止出现这样的情况，人们开始认真地对软件进行测试。

软件测试就是在软件开发完成后、在投入运行前，对软件的需求和其代码进行的最终审查，这样就可以确保软件能够按预期的那样运行。所以，软件测试也可以说成是根据需求设计和开发后的软件内部结构而设计出测试用例，然后通过运行这些用例，从而显示错误的过程。

在软件生存期中，软件测试能够跨越两个阶段：（1）单元测试，在软件开发过程中，每个模块开发完成后，都会对它进行测试；（2）测试阶段，所有模块都开发完成后，即软件整体都开发完后，对整个软件进行综合测试。

有关机构研究表明，国外软件开发厂商约40%的工作时间要花在测试上，对一些可靠性、安全性要求较高的软件更是不惜人力物力进行测试。以微软为例，早在1999年发布Windows 2000操作系统时，微软投入了250多个项目经理、1700多个开发人员，内部测试人员则达到3200人，比前两者之和还要多。

1.1 软件危机

1.1.1 软件危机的定义

在20世纪60年代到70年代，西方计算机科学家把软件开发和维护过程中遇到的一系列严重问题统称为“软件危机”。

1.1.2 软件危机的表现

随着计算机的不断发展，由于其硬件技术也不断增强，计算机在容量、运算速度和可靠性等质量方面有了显著提高，但是其价格却在不断降低。计算机硬件价格的降低，使得其能够更多应用于社会当中。在这种情况下，人们也开始要求计算机软件也能有所提高，希望软件能够跟上硬件的发展，也就是说，人们期望软件可以解决一些大型、复杂的问题。但是一个软件系统的开发需要很多方面的投资，比如资金和人力等，软件所占的投资比例会高于硬件。另外值得注意的一件事是，在软件开发过程中，不同开发人员有自己的编程习惯和方法，所以每个人在编程过程中，会凭自己的喜欢工作，没有统一的标准，因此，软件开发就会打上“个人单干”的标签，所研制的软件系统的质量就难以保障。20世纪60年代，由于软件系统的开发越来越复杂，“软件危机”就此出现。软件危机有以下六点表现形式：

（1）软件开发的进度计划与成本的估计很不准确，实际成本可能会比估计成本高出一个数量级，而实际进度却比计划进度延迟几个月甚至几年，这就导致了整个项目的大量延期和经费暴涨。开发商为了赶进度与节约成本会采取一些权宜之计，这往往会使软件的质量大大降低。这些现象极大地损害了软件开发商的信誉。

（2）由于一些管理不规范的问题，比如文档书写不规范、管理出现严重漏洞等，会使得软件维护成本大大上升，可能还会使得系统出现问题并无法修复。

（3）因为没有强有力的软件质量检测方法，所以用户最后得到的软件质量往往比较差，在运行过程中，会出现各种各样的问题，有可能会带来严重亏损的后果。

（4）开发的软件产品往往与用户的实际需要相差甚远，软件开发过程中不能很好地了解并理解用户的需求，也不能适应用户需求的变化。

（5）软件研制完成后，不可能一成不变，需要随着时间不断更新，但软件系统的更新难度是相当大的。

（6）软件开发的生产率远远不能满足客户需要，使得人们不能充分利用现代计算机硬件所提供的巨大潜力。

1.1.3 软件危机实例

1. 软件开发进度难以预测，以丹佛新国际机场为例

该机场规模是曼哈顿机场的2倍，宽为希思机场的10倍，可以全天候同时起降3架喷气式客机；该机场投资了1.93亿美元建立了一个地下行李传送系统，总长33.8公里，有4000台遥控车，可按不同线路在20家不同的航空公司柜台、登机门和行李领取处之间发送和传递行李；支持该系统的是5000个电子眼、400台无线电接收机、56台条形码扫描仪和100台计算机。按原定计划要在1993年万圣节前启用，但一直到1994年6月，机场的计划者还无法预测行李系统何时能达到符合机场开放要求的稳定程度。

2. 软件缺少适当的文档资料，以IBM公司的OS/360系统为例

美国的IBM公司开发的OS/360系统，共有4000多个模块，约100万条指令，每年投入5000人，耗资数亿美元，结果还是延期交付。在交付使用后的系统中仍发现不少于2000个错误。由于没有完整的开发文档，难以修改。

3. 软件不能如期完成，以美国银行信托软件系统开发案为例

美国银行1982年进入信托商业领域，并规划发展信托软件系统。项目原定预算2000万美元，开发时程9个月，预计于1984年12月31日以前完成，后来至1987年3月都未能完成该系统，期间已投入6000万美元。美国银行最终因为此系统不稳定而不得不放弃，并将340亿美元的信托账户转移出去，并失去了6亿美元的信托生意商机。

1.1.4 产生软件危机的原因

1. 客观原因

软件本身的特点会导致软件危机的产生。软件是无形的，没有物理形态，只能通过运行状况来了解功能、特性、和质量。软件渗透了大量的脑力劳动，人的逻辑思维、智能活动和技术水平是软件产品的关键。软件的逻辑部件复杂、规模庞大，

而且需要不断地进行缺陷维护和技术更新，这在无形之中也造成了软件危机。

2.主观原因

人们自身开发方法的不恰当同样会导致软件危机的产生。在开发过程中，有些开发商没有及时地收取用户的需求，最后开发出的产品可能会和用户理想中的产品相差甚远。在许多人的印象中，软件开发就是一个敲代码的过程，而实际情况中，软件开发需要多方协调工作，边开发边测试，并不停地听取客户的意见。另外，软件是一个需要不断更新的产品，为了适应时代的潮流，给用户更好的体验，找出软件中暗藏的缺陷，增加新的功能，软件需要定期地进行维护，人们如果轻视这个过程，势必也会造成软件危机。

1.1.5 软件危机的解决途径

1. 组织管理

现在，人们开始意识到，软件危机主要是由于软件缺陷的不断积累所导致的，“失之毫厘，谬以千里”，一些早期的比较小的软件缺陷很可能在后期带来巨大的麻烦。在软件研制过程中不断投入的人力和物力，使其成本越来越大，但是因为软件自身存在缺陷，最后的开发成果不尽如人意，会出现反复的修改过程，这也是开发效率低下的原因之一。软件的开发是没有标准的，不同的研制人员，就会有不同的软件成果，这是软件维护困难的重要原因。所以，创造一个比较规范化的标准来使软件开发井然有序，是现在人们迫切需要的。1986年，在北大西洋公约组织的学术会议上，人们第一次提出了“软件工程”这个词，开始提倡大家按一定规则和方法进行软件的开发。

由于软件工程一开始是为了应对软件危机而提出的，如果软件在开发过程中能较好地利用软件工程的原理对软件开发的过程进行有效的管理，就可以充分保证软件开发的质量和生产率，反之就有可能造成项目的失败。下面列举正反两个方面的实例。

【成功案例】美国联邦速递公司的管理信息系统

美国联邦速递公司是一个具有数十亿美元资产，经营速递业务的大型企业。它拥有643架飞机、43 000辆汽车、138 000名员工，每天运送超过310万个包裹，通达全世界近200个国家和地区。该公司为适应管理的需要开发了覆盖整个公司的管理信息系统，从系统的架构、分析、开发直至运行、维护，始终遵循需求至上的原则，将先进的软件工程的原理与方法贯穿整个开发过程，最终该项目取得了圆满的成功。通过管理信息系统，公司在任何时间都可以知道每件包裹在什么位置以及以后的运送路线，客户只要登录该公司的网站也可以得到同样的信息。后来，该系统又逐步扩展集成了从一个工厂的成品到送达用户之间的所有涉及分拣、运输、仓储、递送等每一步的状态数据。

【失败案例】英国伦敦的急救服务管理信息系统

伦敦急救服务中心覆盖了680多万人口，每天接送5000个病人，接听2500个电话。为提高对急救电话的响应速度，更有效及时地处理紧急情况，该中心开发了相应的急救信息管理系统，试图通过该系统对急救信息进行实时管理，最终目标是实

现平均每14分钟响应一个电话的目标。1992年10月新系统正式投入运行。由于新的系统既没有经过严格的调试，也没有经过完全的测试尤其是满负荷下测试，同时全体职员更没有经过对新系统的使用培训，使得系统在运行过程中发生了一系列致命的问题：有些紧急电话要花30分钟才能打进去，由于救护车延迟了3个小时，造成数十人死亡。伦敦急救服务中心的一位发言人说："真是一场可怕的噩梦!"

以上正反两个方面的例子充分说明了软件工程在软件开发中的重要作用。

2. 技术措施

为了有效地避免软件危机，人们通过提供符合标准的分析设计方法和一些相关的工具软件，来避免或减少在软件研制过程中可能会出现的一些错误。

这里介绍几种主流的软件开发方法。

（1）结构化方法——1978年，E. Yourdon和L. L. Constantine提出了结构化方法，即SASD方法，也可称为面向功能的软件开发方法或面向数据流的软件开发方法。1979年TomDeMarco对此方法做了进一步的完善。Yourdon方法是20世纪80年代使用最广泛的软件开发方法。它首先用结构化分析（SA）方法对软件进行需求分析，然后用结构化设计（SD）方法进行总体设计，最后是结构化编程（SP）。这一方法不仅开发步骤明确，SA、SD、SP相辅相成，一气呵成，而且给出了两类典型的软件结构（变换型和事务型），便于参照，使软件开发的成功率大大提高，从而深受软件开发人员的青睐。

（2）面向数据结构的软件开发方法——1975年，M. A. Jackson提出了一类至今仍广泛使用的软件开发方法。这一方法从目标系统的输入、输出数据结构入手，导出程序框架结构，再补充其他细节，就可得到完整的程序结构图。这一方法对输入、输出数据结构明确的中小型系统特别有效，如商业应用中的文件表格处理。该方法也可与其他方法结合，用于模块的详细设计。

（3）问题分析法——PAM（Problem Analysis Method）是20世纪80年代末由日立公司提出的一种软件开发方法。PAM方法希望能兼顾Yourdon方法、Jackson方法和自底向上的软件开发方法的优点，而避免它们的缺陷。它的基本思想是：考虑到输入、输出数据结构，指导系统的分解，在系统分析指导下逐步综合。这一方法的具体步骤是，从输入、输出数据结构导出基本处理框，分析这些处理框之间的先后关系，按先后关系逐步综合处理框，直到画出整个系统的PAD图。从上述步骤中可以看出，这一方法本质上是综合的自底向上的方法，但在逐步综合之前已进行了有目的的分解，这个目的就是充分考虑系统的输入、输出数据结构。PAM方法的另一个优点是使用PAD图。这是一种二维树形结构图，是到目前为止最好的详细设计表示方法之一，远远优于NS图和PDL语言。这一方法在日本较为流行，软件开发的成功率也很高。由于在输入、输出数据结构与整个系统之间同样存在着鸿沟，这一方法仍只适用于中小型问题。

（4）面向对象的软件开发方法——面向对象技术是软件技术的一次革命，在软件开发史上具有里程碑的意义。随着OOP（面向对象编程）向OOD（面向对象设计）和OOA（面向对象分析）的发展，最终形成面向对象的软件开发方法OMT（Object Modelling Technique）。这是一种自底向上和自顶向下相结合的方法，而且它

以对象建模为基础，从而不仅考虑了输入、输出数据结构，实际上也包含了所有对象的数据结构，所以OMT彻底实现了PAM没有完全实现的目标。不仅如此，OO技术在需求分析、可维护性和可靠性这三个软件开发的关键环节和质量指标上有了实质性的突破，彻底地解决了在这些方面存在的严重问题，从而宣告了软件危机末日的来临。

为了应对软件危机，除了熟悉一些软件开发方法，了解相关的软件工具也是有必要的。软件工具是指为支持计算机软件的开发、维护、模拟、移植或管理而研制的程序系统。它是为专门目的而开发的，在软件工程范围内也就是为实现软件生存期中的各种处理活动（包括管理、开发和维护）的自动化和半自动化而开发的程序系统。软件工具的种类繁多，从软件过程的角度通常将其分为：项目管理工具、配置管理工具、分析和设计工具、程序设计工具、测试工具以及维护工具等。具体介绍如下。

（1）项目管理工具：支持项目管理活动的工具。通常，这类工具把重点放在特定的管理环节上，例如工作量、成本和工期估算以及项目调度计划等。

（2）配置管理工具：支持完成配置项标识、版本控制、变化控制、审计和状态统计等任务的工具。

（3）分析和设计工具：辅助建立软件的系统模型和设计的工具。分析和设计引擎工具将成为新一代分析设计工具，该工具可以对任何分析和设计方法进行定制，根据需要，支持特定的分析和设计方法。

（4）程序设计工具：包括常规的编码工具——编译程序、编辑程序、排错程序及第四代语言、应用程序生成器、数据库查询语言和面向对象程序设计环境等。

（5）测试工具：可以分为数据获取工具、静态分析工具、动态分析工具、模拟工具以及测试管理工具等。其中，静态分析工具通过对源程序的程序结构、数据流和控制流进行分析，得出程序中函数（过程）的调用与被调用关系、分支和路径、变量定义和引用等情况，发现语义错误。动态分析工具通过执行程序，检查语句、分支和路径覆盖，测试有关变量值的断点，即对程序的执行流行探测。另一类动态分析工具称为截获/播放工具。测试管理工具用以控制并协调软件测试的每一个主要步骤，进行回归测试，比较运行结果和期望输出之间的差异，并可实施程序的成批测试。

（6）维护工具：支持软件维护的工具。大致可分为逆向工程工具和再生工程工具。逆向工程工具对已经开发完成的源程序进行分析，抽取程序的系统结构、控制结构、逻辑流程、数据结构和数据流等信息，并生成分析和设计模型以及其他设计信息。再生工程工具用来支持重构一个功能和性能更为完善的、改进的软件系统。

1.2 软件生命周期

软件生命周期是指一个软件从定义开始，经过开发、运行、维护，直到最后被废弃的全过程，是软件工程中的基本概念之一。软件生命周期可以划分为若干阶段，每个阶段都有明确的任务，从而使规模大、结构和管理较为复杂的软件开发变得容易控制和管理。通常，软件的生命周期包括以下几个阶段。

（1）问题的定义及规划：需要系统分析员同用户进行交流，弄清楚“用户需要计算机解决什么问题”，然后提出关于“系统目标与范围的说明”，并从经济、技术、法律等多方面进行可行性研究分析，提交给用户审查确认。

（2）需求分析：弄清用户对软件系统的全部需求，编写需求规格说明书和初步的用户手册，提供给用户评审。

（3）软件设计：根据需求分析的结果，对整个软件系统进行设计，如系统框架设计、数据库设计等。

（4）程序编码：利用选定的程序语言，如Python，完成源程序的编码。在程序编码中必须要制定统一、符合标准的编写规范，以保证程序的运行效率。

（5）软件测试：主要目的是发现整个设计过程中存在的问题并加以纠正。测试的主要方法为白盒测试和黑盒测试两种。在测试过程中需要建立详细的测试计划并严格按照测试计划进行测试，以减少测试的随意性。

（6）运行维护：软件维护是软件生命周期中持续时间最长的阶段。在软件开发完成并投入使用后，因多方面的原因和用户要求需要对软件的使用寿命进行延长。软件维护通常包括纠错性维护和改进性维护两方面。

从软件生命周期概念提出以后，软件产品就开始进入软件生命周期。在经历需求、分析、设计、实现、部署后，软件将被使用并进入维护阶段，直到最后逐渐消亡。这样一个过程被称为“生命周期模型”。典型的几种生命周期模型包括：瀑布模型、原型模型、增量模型、螺旋模型、构建组装模型。

1.2.1 瀑布模型

瀑布模型是最传统的开发模型，又称为线性顺序模型。它把软件开发分为三个部分：定义阶段、开发阶段、运行维护。它的基本思想是：把软件周期细分为软件计划、需求分析、软件设计、程序编码、软件测试、运行维护等阶段，把每个阶段看作是瀑布中的一个个台阶，各个台阶自上而下、相互衔接、次序固定。它把软件的开发过程比喻成瀑布流水在这些台阶上奔流而下。瀑布模型的示意图如图1-1所示。

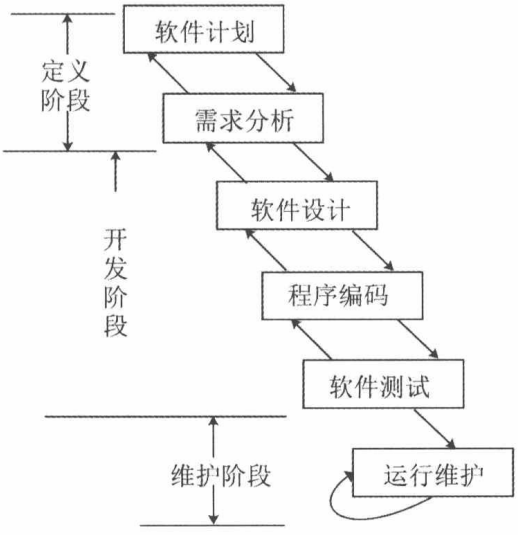

图1-1　瀑布模型示意图

结合图1-1分析，瀑布模型有以下特性：下一项工作的进行建立在前一项工作完成的基础上，并且在每个阶段完成后需要提交相应的文档资料，经过评审和复查，审查通过后才能进入下一阶段，否则返回前项任务，甚至向更前项任务返工。

瀑布模型有几个比较明显的缺点如下：

（1）模型缺乏灵活性，无法解决软件需求不明确或者不准确的问题；

（2）整个模型几乎都是以文档驱动的，这对于非专业用户而言难以理解；

（3）前阶段所出现的误差到后阶段所导致的后果往往比较严重，纠正所需要的代价更高。

由于瀑布模型不够灵活、纠正错误代价过高等缺点引发的问题显得尤为严重，但彻底抛弃瀑布模型的思想也是不正确的。为了弥补瀑布模型的不足，使它更符合人们生产需要，近年来提出了多种其他模型。

1.2.2 原型模型

原型模型又称为样品模型，即借用已有系统作为原型模型，通过对“样品”的不断改进，使得最后的产品符合用户需求。原型模型从需求收集开始，开发者和客户在一起定义软件的总目标，快速设计实现那些对用户和客户可见的部分，再通过用户的评估进一步精化待开发软件，使其逐步满足客户需求。这个过程往往是迭代的。原型模型采用逐步求精的方法完善原型，使得原型能够快速开发，避免了像瀑布模型一样在冗长的开发过程中难以对用户的反馈做出快速的响应。相对瀑布模型而言，原型模型更符合人们开发软件的习惯，是比较流行的一种实用型生命周期模型，示意图如图1-2所示。

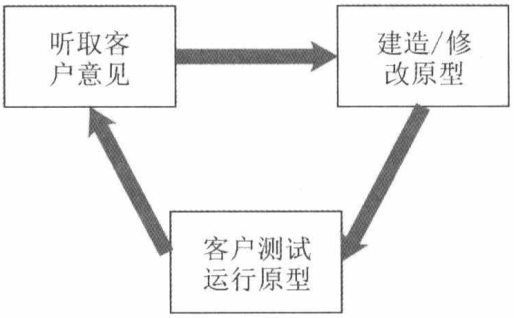

图1-2 原型模型示意图

结合图1-2分析，原型模型具有以下显著特点：该模型是基于原型驱动的；开发者和用户沟通和交流更全面，能更好地制定需求定义，适应用户的需求变化，保证产品的实用性；原型系统开发周期短、开发费用低、维护容易并且对用户更友好。

尽管开发者和用户都喜欢使用原型模型，但原型模型也有其固有缺点：

（1）在对原型的理解上用户和开发者有很大差异，用户以为原型就是软件的最终版本，而开发者只将原型当成一个漂亮的外壳，没有考虑软件的整体质量和长期的可维护性，当客户被告知该产品还需不断修改才能应用时，很容易引起客户不满。

（2）原型是开发者快速开发出来的，而开发者对所开发领域的陌生容易将次要部分当成主要框架，做出不切题的原型。

（3）软件的整个开发都是围绕原型展开的，在一定程度上不利于开发人员创新。

为了扬长避短，原型模型通常适用于以下情况：用户定义了软件的一组一般性目标，但不能详细指定输入、处理以及输出要求；开发者不确定算法的有效性、操作系统的适应性或人机交互的形式等，需要在实施过程中做大量用户化开发工作的项目。

1.2.3 迭代增量模型

增量模型将软件看成是一系列相互联系的增量，每次迭代完成一个增量，增量以迭代的方式运行瀑布模型。每个迭代过程都包含软件生命周期的全部，即包含了分析、设计和实现等各个阶段。该模型是一种渐进开发、逐步完善的软件版本模型。随着时间的推移，增量模型在每个阶段运行线性序列，每个线性序列生产出一个软件的可交付增量，如图1-3所示。

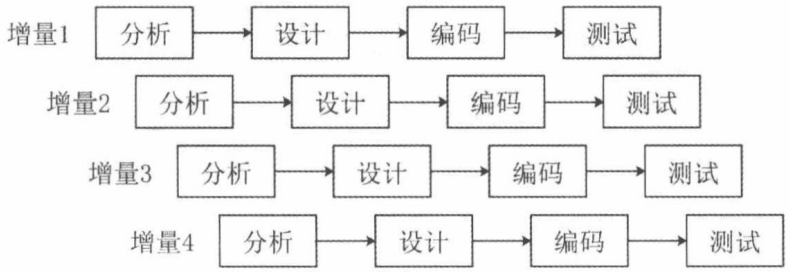

图1-3 迭代增量模型示意图

迭代增量模型最大的特点是将待开发的软件系统模型模块化和组件化，使得该模型具有以下优点：可以分批次提交软件产品，使用户能及时了解项目的进程；以组件为单位开发降低了软件开发风险，一个子模块开发周期出现差错不会影响到整个软件系统；开发顺序较为灵活，开发人员可以先完成需求稳定的核心组件，当组件优先级发生变化时，还能及时地进行调整。但缺点是如果待开发的系统难以被模块化，采用增量模型将会变得十分复杂，故增量模型适用于具有以下特征的开发项目：

（1）软件产品可以分批次进行交付；

（2）待开发的软件系统能够被模块化；

（3）软件开发人员对应用领域不熟悉，难以对系统进行一次性开发；

（4）项目管理人员把握全局的水平较高。

1.2.4 螺旋模型

螺旋模型采用一种周期性的方法进行系统开发。它兼顾了增量原型的迭代特征以及瀑布模型的系统化与严格监控。螺旋模型的最大特点在于引入了其他模型不具备的风险分析，使软件在无法排除重大风险时有机会停止，以减少损失，因此特别适用庞大、复杂并具有高风险的系统。采用螺旋模型的软件过程如图1-4所示。