高等学校计算机科学与技术教材

软件自动化测试技术

陆　璐　王柏勇　编著

清 华 大 学 出 版 社
北京交通大学出版社
·北京·

内 容 简 介

本书全面、系统地论述软件工程与软件测试自动化的理论及应用技术。全书共分 14 章。前 6 章介绍了软件测试的基本理论，包括软件测试是一个持续质量改善的过程、软件测试流程与生命周期、现代软件开发中软件测试的基本流程、软件测试工具种类的评估、软件产品维护阶段的软件测试技术等。第 7 章到第 11 章介绍软件测试的一些通用的测试工具，包含客户端、应用服务器、数据库端的性能、功能和安全性测试的工具和方法。第 12 到第 14 章介绍利用目前国际主流软件测试平台之一的 Segue 为电力部门大型应用系统进行的软件测试实例，包括软件测试用例设计、测试文档的自动生成、内存自动检查分析、故障自动跟踪分析、软件测试用例的自动回收、测试的实例结果分析及相关报告模版等。附录给出了软件测试项目的国际标准模版，供读者参考。

本书既可作为高等院校软件测试课程的教材或参考书，也可作为从事计算机软件开发的科技人员和软件项目高级管理人员学习软件工程和软件测试自动化技术的参考书。同时对于那些希望增强软件测试方面知识的程序员及软件开发团队的其他人员，本书也具有很好的参考价值。

版权所有，翻印必究。举报电话：010－62782989　13501256678　13801310933
本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
本书防伪标签采用特殊防伪技术，用户可通过在图案表面涂抹清水，图案消失，水干后图案复现；或将表面膜揭下，放在白纸上用彩笔涂抹，图案在白纸上再现的方法识别真伪。

图书在版编目（CIP）数据

软件自动化测试技术／陆璐，王柏勇编著. —北京：清华大学出版社；北京交通大学出版社，2006.9
（高等学校计算机科学与技术教材）
ISBN 7－81082－836－3

Ⅰ. 软…　Ⅱ. ①陆…　②王…　Ⅲ. 软件－测试－高等学校－教材　Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2006）第 081574 号

责任编辑：谭文芳
出版发行：清华大学出版社　邮编：100084　电话：010－62776969　http://www.tup.com.cn
　　北京交通大学出版社　邮编：100044　电话：010－51686414　http://press.bjtu.edu.cn
印 刷 者：北京东光印刷厂
经　　销：全国新华书店
开　　本：185×260　印张：18.5　字数：470 千字
版　　次：2006 年 9 月第 1 版　2006 年 9 月第 1 次印刷
书　　号：ISBN 7－81082－836－3／TP·295
印　　数：1～5 000 册　定价：32.00 元

本书如有质量问题，请向北京交通大学出版社质监组反映。对您的意见和批评，我们表示欢迎和感谢。
投诉电话：010－51686043，51686008　传真：010－62225406　E-mail：press@center.bjtu.edu.cn。

序　言

随着现代化软件开发技术的发展和软件功能需求日益复杂，软件规模越来越庞大，软件开发人员的角色分工也越来越细，相互之间协作性要求也随之提高。越来越多的人对软件测试技术的重要性有了更进一步的认识，测试已经成为软件开发的一个重要环节。IEEE 对软件测试进行了准确定义 软件测试是使用人工或者自动手段来运行或测定某个系统的过程，检验该过程是否满足规定的需求或是弄清预期结果与实际结果之间的差别。

本书全面、系统地论述软件工程与软件测试自动化的理论及应用技术。全书共分 14 章。前 6 章介绍软件测试的基本理论，包括软件测试是一个持续质量改善的过程、软件测试流程与生命周期、现代软件开发中软件测试的基本流程、软件测试工具种类的评估、软件产品维护阶段的软件测试技术等。第 7 章到第 11 章介绍软件测试的一些通用的测试工具，包含客户端、应用服务器、数据库端的性能、功能和安全性测试的工具和方法。第 12 到第 14 章介绍利用目前国际主流软件测试平台之一的 Segue 为电力部门大型应用系统进行的软件测试实例，包括软件测试用例设计、测试文档的自动生成、内存自动检查分析、故障自动跟踪分析、软件测试用例的自动回收、测试的实例结果分析及相关报告模版等。附录给出了软件测试项目的国际标准模版，供读者参考。

本书理论联系实际，反映了当今世界软件测试的最新技术成果，是一本非常实用的软件测试教材。本书总结了作者多年从事软件测试与持续质量改善课程教学的经验和在微软公司从事软件测试的工作经验。本书既可作为高等院校软件测试课程的教材或参考书，也可作为从事计算机软件开发的科技人员和软件项目高级管理人员学习软件工程和软件测试自动化技术的参考书。同时对于那些希望增强软件测试方面知识的程序员及软件开发团队的其他人员，本书也具有很好的参考价值。

本书以陆璐为主编写，王柏勇教授级高工为本书第三部分提供了实用测试案例，研究生叶瑜、邱传龙、彭涛、周维、吴雯涵等进行了整理和校稿工作。本书在编写过程中也得到了微软公司软件测试部门高级经理张帆女士，广州供电局苏凯工程师，AVON 公司大中华区沙泰国副总裁，信息技术开发部高级经理张杰先生及姚少玲女士等的大力支持，在此一并表示感谢。限于作者水平，书中难免会有不妥和错误之处，敬请广大读者批评指正。

读者在本书的学习和阅读过程中如果有什么问题，欢迎进入作者的研究主页（http：//www. xjsurc. net）进行相关讨论和技术交流。

陆　璐

2006 年 8 月

目　　录

第一部分　软件测试基本理论篇

第 1 章　软件质量定义及软件质量保障方法 …… 2
1.1　质量保证框架 …… 2
1.1.1　什么是质量 …… 2
1.1.2　改善软件质量的基本概念和要素 …… 3
1.1.3　软件质量保证体系 …… 4
1.1.4　质量保证体系的组成 …… 5
1.1.5　软件配置管理的组成 …… 7
1.2　软件质量保证计划 …… 8
1.3　测试技术常用方法介绍 …… 10
第 2 章　软件持续质量改善过程 …… 13
2.1　Edward Deming 对软件测试学的理论贡献 …… 13
2.1.1　统计方法的功能 …… 13
2.1.2　Deming 的 14 条质量改善原则 …… 14
2.1.3　通过 PDCA 来持续提高过程 …… 17
2.2　测试生命周期介绍 …… 18
2.2.1　瀑布开发技术 …… 18
2.2.2　测试圣经 软件测试计划 …… 21
第 3 章　软件测试生命周期介绍 …… 25
3.1　测试过程综述 …… 25
3.2　逻辑设计阶段 …… 28
3.3　物理设计阶段 …… 31
3.4　单元设计阶段 …… 33
3.5　编码阶段 …… 35
第 4 章　软件测试开发方法概述 …… 42
4.1　生命周期阶段的缺陷 …… 42
4.2　客户 – 服务器环境下软件测试遭遇的新挑战 …… 42
4.2.1　客户 – 服务器螺旋测试心理学 …… 43
4.2.2　项目目标 :质量保障与开发的整合 …… 44
4.2.3　原型法的类型 …… 48
4.3　基于原型的开发方法 …… 50
4.4　软件测试信息收集 …… 54
第 5 章　软件测试工具评估标准 …… 63
5.1　正确选择测试工具 …… 63

5.1.1 何时考虑使用测试工具 …… 63
5.1.2 何时不考虑使用测试工具 …… 63
5.1.3 测试工具选择审核表 …… 64
5.2 测试工具的常用类型 …… 65
5.3 评估测试工具的方法 …… 66
第 6 章 维护阶段的软件测试工作 …… 72
6.1 软件维护综述 …… 72
6.2 软件系统重新设计标准 …… 73
6.3 软件变更的类型与维护方法 …… 76
6.3.1 完善性维护 …… 76
6.3.2 适应性维护 …… 78
6.3.3 更正性维护 …… 79
6.4 基本路径测试 …… 80

第二部分 软件测试基本工具篇

第 7 章 页面层——Microsoft ACT 压力测试工具介绍 …… 84
7.1 什么是 ACT？ …… 84
7.2 ACT 的主要概念 …… 86
7.3 运行 ACT …… 91
小结 …… 103
第 8 章 应用层——应用软件执行的系统监测工具 …… 104
8.1 系统监测器的使用 …… 104
8.2 监视对象、计数器和进程性能瓶颈现象 …… 113
8.3 典型的处理器相关的问题和解决方案 …… 114
小结 …… 123
第 9 章 网络层——应用网络分析 …… 124
9.1 进行一个应用网络分析 …… 124
9.2 微软网络监视器 …… 129
9.3 使用 Compuware 公司的应用专家 …… 135
9.4 使用应用专家解释网络捕获 …… 139
小结 …… 142
第 10 章 Web 层的分析和性能优化 …… 143
10.1 准备工作 …… 143
10.1.1 理解配置和性能 …… 143
10.1.2 理解 Web 应用程序 …… 145
10.2 剖析一个 .NET Web 应用程序 …… 146
10.2.1 IIS 日志文件 …… 146
10.2.2 在代码级别跟踪问题 …… 149

10.2.3 系统监视器计数器 …… 153
10.3 性能优化技巧 …… 155
10.3.1 应用程序和会话状态 …… 155
10.3.2 ASP .NET 中的缓存 …… 156
10.3.3 禁用 ViewState …… 158
10.3.4 ADO .NET 技巧 …… 158
10.4 常见的 Web 层瓶颈 …… 162
10.5 伸缩 Web 层 …… 163
10.5.1 外扩、上扩与性能调整 …… 163
10.5.2 何时伸缩 Web 层 …… 164
10.5.3 怎样外扩 Web 层 …… 164
小结 …… 165
第 11 章 SQL 数据库层应用测试分析 …… 166
11.1 开始 …… 166
11.2 瓶颈的确认 …… 166
11.2.1 SQL Server 2000 的常用工具 …… 166
11.2.2 阻塞问题 …… 170
11.3 索引调整 …… 175
11.3.1 分析执行计划 …… 175
11.3.2 了解索引 …… 180
11.3.3 选择正确索引 …… 181
小结 …… 190

第三部分 软件测试大型应用分析篇

第 12 章 ××供电局营销系统测试计划 …… 192
12.1 ××供电局电力营销管理系统简介 …… 192
12.2 测试流程 …… 193
12.3 制订测试计划 …… 193
12.3.1 建立测试目标 …… 194
12.3.2 分析被测系统 …… 194
12.3.3 建立测试环境 …… 197
12.3.4 系统功能模块的脚本系统 …… 197
12.4 执行测试 …… 214
12.5 测试结果分析 …… 214
12.6 制订工作量估计表格 …… 214
第 13 章 营销系统测试任务书 …… 217
13.1 引言 …… 217
13.2 计划 …… 217

13.2.1 软件说明 …… 217
13.2.2 测试内容 …… 217
13.3 测试流程 …… 222
13.3.1 测试环境与测试工具 …… 222
13.3.2 测试 …… 222
13.4 评价 …… 225
13.4.1 评估依据 …… 225
13.4.2 评估尺度 …… 225
13.4.3 测试结果报告 …… 226
13.5 附录 :测试用例和测试问题描述模版 …… 226
13.5.1 测试用例表 (功能测试) …… 226
13.5.2 问题描述模版 …… 227
第 14 章 营销系统使用 Segue 软件测试流程 …… 228
14.1 Segue 负载测试流程 …… 228
14.2 制订测试计划 …… 228
14.2.1 建立测试目标 …… 228
14.2.2 分析被测系统 …… 228
14.2.3 建立测试环境 …… 229
14.3 创建测试 …… 230
14.4 执行测试 …… 232
14.5 测试结果分析 …… 234
第 15 章 ××供电局营销系统测试用例设计——计量管理子系统 …… 237
15.1 文档版本控制信息 …… 237
15.2 引言 …… 238
15.3 系统维护管理 …… 239
15.4 电能表管理 …… 241
15.5 资料查询 …… 249
附录 A 软件测试项目国标模版 (GB 8567－88) …… 252
参考文献 …… 285

第一部分

软件测试基本理论篇

第 1 章　软件质量定义及软件质量保障方法

几乎每一个软件相关人员都会对软件质量有所要求。经理们要求高质量，软件开发人员想开发出高质量的产品，而用户希望能使用可靠性高的软件。

为了能评测并且提高软件的质量，许多组织专门成立了软件质量保证组（Software Quality Assurance Groups）。然而，对于软件质量这个概念，目前并没有一个被广泛接受的说法。因此，在不同的组织中，软件质量保证组可能执行着不同的职能，他们参照各自的过程来执行计划。

在某些组织中，软件测试是交给软件质量保证组来做的。但是在其他一些组织中，则有可能让研发部门来做，或者干脆将它交给一个独立的机构。

许多软件质量保证组制定出来的软件保证计划与测试计划差不多。但是实际上，软件质量保证计划应该包括测试计划，并且涵盖其他更加丰富的内容。

1.1　质量保证框架

1.1.1　什么是质量

在韦氏字典中，质量被定义为“事物的本质的、内在的、独特的属性，用来度量或划分其优秀程度”。而在计算机术语中，质量有两个被广泛接受的定义。第一个是：质量就是要“满足需求”。就这个定义而言，需求是要能够被度量的，并且能够判断出一个产品的需求是否已经满足。从这个意义上讲，质量就有两种状态：一个产品要么是高质量的，要么就不是。一个产品可能十分复杂，也可能很简单。但是只要需求能够被度量，就能判断出这个产品是否满足了需求，是否具有高质量。其实，这是从生产者的角度来看待这个问题所得出的结论。满足需求也就成了最终目标。

质量的另一个定义是客户自己对质量的定义，这个定义也是本书使用的。在这个定义中，质量就是产品或服务是否满足了客户的需要，换一种说法就是“适用性”。对于产品目的的描述，通常存在于用户需求说明书中。需求是最重要的文档，并且整个质量系统都是围绕需求来运转的。另外，质量属性也可以被描述为用户的需求说明，比如易用性、可移植性和可复用性。

虽然大家都在关注质量，但是人们对质量的认识存在一些混淆，这些也直接阻碍了质量的改进。

- 质量需要有任务指派，这些指派通常来自高级管理层。只有管理层和普通员工紧密配合才能保证高质量。
- 许多人认为不需要质量检测的产品或服务是不存在的。所以，产品出现一定范围内的缺陷是很常见的现象，也是可以接受的。
- 质量和成本是紧密相关的，高质量也就意味者高成本。产生这种看法的原因是将设计质量（Quality of Design）和一致性质量要求（Quality of Conformance）混淆了。

- 质量需要非常详细的需求说明,这样生产出来的产品质量才能被量化。但是许多组织没有能力或者不愿意去花额外的精力来制定详细的说明。
- 技术人员认为,各种各样的标准将扼杀他们的创新能力,所以他们不愿意受束于标准。然而,要获得高质量,定义良好的标准和过程都是必不可少的。

1.1.2　改善软件质量的基本概念和要素

1. 预防与检测

对已经完成的产品进行评定是没有意义的。为了完成目标,就需要事先预防质量的缺陷和不足,从而能使用质量保证方法对产品进行评定。质量保证的方法包括:依靠软件开发标准使得开发过程结构化,使开发过程得到方法、工具及技术上的支持,等等。

除了产品评定,过程评定对于质量管理方案而言也是至关重要的。例如:编码标准文档,标准、方法和工具的指示及使用,数据备份步骤,变更管理,缺陷文档及重新合并。

质量管理能够降低产品的成本,这是因为越早发现并纠正缺陷,由长期所导致的成本也将更低。可能这会使得在初期的投资有所增加,但从长期而言,它将带来更高质量的产品及更低的维护费用。

有效的质量管理的总成本由四部分组成:预防、检测、内部故障和外部故障。预防成本来自于为了第一时间预防缺陷的发生而采取的措施。而度量、评估、审计产品与标准的符合程度则导致了检测成本。在缺陷产品被分发之前,对其缺陷进行纠正,这部分的成本称为内部故障;而当有缺陷的产品被发布之后,这些缺陷所导致的成本成为外部故障。其中,后者将导致毁灭性的后果,因为它可能破坏组织的声誉,或者影响以后的销售情况。

因此,回报率最高的方式就是预防。在预防上多花些功夫,可以减少产品缺陷,降低产品成本及维护费用。这样才可以将测试中存在的问题消灭在萌芽阶段,而不要做事后检测工作,因为此时花费的代价相对更大。

2. 校验与确认

在开发生命周期中,先前正确执行的一系列活动会将需求明确化,校验就是要证明产品是否满足了这些需求。而确认则是在生命周期的最后,检查系统是否满足客户的需要。确认能够保证系统是否达到用户预想中的要求,是否能按照预先规定的方式运行。测试产品的产生与产品的确认之间的关系与产品检验之间的关系相比更为密切。传统上,软件测试就是一个确认的过程,是生命周期的一个阶段。在开发结束之后,就需要对系统进行确认和测试,以检查其功能上及操作上的性能。

如果校验被加入到测试中,测试将贯穿开发生命周期的全过程。为了得到最好的效果,在测试过程中将查证和确认结合起来使用是不错的选择。校验包括回顾、分析和测试等一系列系统化的构成。从需求分析开始,一直到编码阶段,它们都将贯穿其中。校验可以保证软件产品的质量及可维护性。另外,校验还会使得开发过程组织化、系统化。这就能让程序更容易被理解,也使得将产品交予独立组织进行评估成为可能。

校验其实在十多年前就出现了。在当时的航天工业中,系统的可靠性非常重要。任何一个细小的错误都可能导致整个任务的失败,并造成时间上和经济上的极大损失,甚至导致危及生命的方案。校验的概念包括两个基本的衡量标准。首先,软件必须足够地且正确地执行预想的功能;其次,软件的功能不能使整个系统的性能明显下降,无论这种效果是这个功能本身

所带来或是多种功能综合作用的结果。校验的最终目的也就是要保证在生命周期中开发的软件能够满足客户的需要,并且达到软件需求文档所描述的需求目标。

校验也使软件文档的各章节及需求说明的各个部分易于把握。充分的校验能保证软件的所有性能和质量的需求都得到足够的测试,并且测试结果在变更产生之后仍能够重现。校验是一个持续的改进过程,它没有明显的结束标志,需要被贯穿到整个系统生命周期中,以确保配置和操作一致性。

校验要确保软件具有预想的功能和应有的特性,例如可移植性,并且能够尽可能地使软件包含更少的错误。它在任何时候都能提供一种方法,这种方法可以密切地监视软件开发项目,并且能为管理者提供有关项目的详细信息。在运用查证过程的时候,管理者就能确保开发人员正在遵循正规的、有序的、可跟踪的软件开发过程来保证软件质量。而在这个过程中,所牵涉到的活动将最少。

有一种反对校验的观点认为,校验在相当程度上增加了软件开发的成本。但是如果考虑到一个项目从开始到最后失败所带来的损失,校验无疑降低了软件的总体成本。有了一个有效的校验计划之后,在安装的系统中,缺陷经常会有 4∶1 的下降。因为,到了操作和维护阶段才来纠正缺陷,其代价要比在设计时高出 20～100 倍。而开始阶段的花费相对于整个过程所带来的收益,就显得微不足道了。

1.1.3 软件质量保证体系

对于软件质量保证,一个正规的定义是 软件质量保证是一系列活动,这些活动能够提供整个软件产品的适用性的证明。要实现软件质量保证,就需要使用为确保一致性和延长的软件周期而建立的质量控制规则。而质量保证、质量控制、审核功能及软件测试之间的关系经常容易使人迷惑。

为了生产出满足客户需求的产品,就必须遵循一定的过程。质量保证是一系列的支持措施,有了这些措施,这些过程的建立和改进就有了保障。在质量保证的过程中,产品质量将和可用的标准相比较,同时也要和不一致产生时的行为相比较。而审核则是一个检查/评估的活动,用以验证与计划、原则以及过程的一致性。

软件质量保证是一种计划好的行为,它可以保证软件满足评测标准,并且具有具体项目所需要的特性,例如可移植性、高效性、复用性和灵活性。它是一些活动和功能的集合,这些活动和功能用来监控软件项目,从而能够实现预计的目标。它不仅仅是软件质量保证组的责任,项目经理、项目组长、项目人员和用户都可以参与到其中。

质量保证是用来管理质量的。“保证”也就意味着,如果遵循了一定的过程,管理者就能够确保产品的质量。质量保证也是一个接触反应式的功能,它能激起管理者和工作人员对于质量的积极态度。成功的软件保证管理者懂得如何使人们关心质量,并深深懂得质量对于个人和组织具有何等重要的意义。

要实现软件质量的目标,主要是需要遵循软件质量控制计划。为了确保每个里程碑 (即过程中的关键时间点,在这个时间点项目必须处于的状态) 所提交的文档和产品都具有高质量,项目就必须引入一些方法来使之得到保证,而这些方法就在软件质量控制计划中进行声明。这一外在的方式会保证一些步骤得到实施,而这些步骤的目的也就是要获得软件质量并且能对那些行为的文档进行管理。这个计划也制定用于检测而不是仅仅设定一个不可能完成的目

标的评测标准,例如,希望生产一个零缺陷的软件或者百分之百可以信赖的软件。

软件质量保证是一种风险管理的策略。因为软件质量的成本很高,需要纳入到项目的正规的风险管理中来,所以软件质量保证的存在是非常有必要的。下面是一些较差的软件质量的例子。

- 被分发出去的软件频繁地出现故障。
- 系统失败导致不可接受的结果,这种结果可以是经济上的损失或者是危及生命的情况。
- 在需要执行预定功能时,系统不可用。
- 系统增强的成本非常的高。
- 检测和缺陷纠正的成本过高。

尽管大部分的质量风险都和缺陷有关,但相对需求而言,系统失败的地方就是一个缺陷。如果需求本身不够完整,甚至是错误的,那么缺陷的风险就更大。而这样所导致的结果就是许许多多内在的缺陷和不能被查证的产品。一些风险管理策略和技术包括软件测试、技术复查、同等复查及符合程度的查证。

1.1.4　质量保证体系的组成

软件质量保证的活动基本上可以分为三类,分别是 软件测试 (例如校验与确认),软件配置管理和质量控制。但是软件质量保证也依赖于一系列内在的标准、实践、约定和规范。三部分的关系如图 1-1 所示。

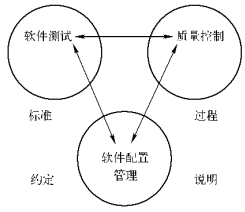


图 1-1　质量保证构成

1. 软件测试

软件测试是一种被普遍采用的风险管理策略,其作用是用来查证软件是否已经满足功能需求。这种方式的不足之处在于,当进行测试时,已经太迟了,很难将高质量融入到软件当中。这些测试虽然只是包含了测试用例,但通过各种输入和系统状态的各种可能的组合,可以保证软件确实满足了需求。然而,并不是所有的缺陷都能在测试中表现出来。软件测试包括那些列举在纸上的活动,比如校验和确认。在很多组织中,这些活动或者他们的监督者都将被囊括到软件质量保证的职能当中。而那些处于设计和编码之外的人员参与到软件质量保证中的程度和范围,就会是一个有关制度、组织和项目的问题。

校验和确认的最主要的目的就是要保证软件的设计、代码和文档都能符合需求。所谓需求包括了用户的需求,来自于用户需求及为满足用户需求而制定的规范、代码审查、检查标准、组件级测试需求、子系统、集成的软件层次,还有当代码与硬件集成之后的接受测试。

在软件的设计和实现的过程中,任何一个软件生命周期的阶段中产生的产品必须满足前一个阶段所制定的需求,而校验就能帮助人们了解到这一点。当校验工作贯穿到开发过程中时,它并不需要花费很多的时间,而且也不会很复杂。

2. 质量控制

质量控制的定义是“用于监控工作及观察需求是否被满足的过程和方法”。它的要点是在产品推出之前就发现并纠正产品的缺陷。质量控制属于生产该产品的部门的职责范围。让生

产产品的小组同时担负起质量控制的职责是完全可行的,或者也可以在该生产部门内部成立质量控制小组来专门负责这件事。

产品质量保证计划中将制定针对产品的很多定义良好的检查项,而质量控制就应该包括这些内容。就软件产品而言,质量控制就包括了规范复审、文档和代码的检查,以及对最终移交给用户的产品的检查。通常情况下,在生命周期的每个里程碑都会检查文档和产品,以确保其是否满足了软件质量保证计划所制定的要求。这些要求的形式主要是需求规格说明书、概要设计、详细设计文档及测试计划。最终交给用户的文档则包括需求规格说明书、设计文档、用户接受测试报告、源代码、用户手册,以及操作和维护手册。如果还有其他的文档,在软件质量保证计划中也要进行说明。

质量控制可以来自于多个方面。对于大多数的项目而言,项目监督小组或者部门的软件质量协调员将对文档进行检查。而在大项目中,一个配置管理团队将担负起质量控制的任务。整个团队可能包括用户或用户代表、一个软件质量保证部门的成员及项目负责人。

检查是比较传统的质量控制方式,例如,进行独立的检查以评价产品是否满足评测标准。监督人员和专家将对规范和工程产品进行审核,从而发现产品缺陷并提出改进意见。检查用于证实产品是否与书面的项目规则相符合。这些规则一般是在项目的里程碑建立,也可以是在项目负责人及软件质量保证人员认为有必要的其他时候。一个检查可能包含很详细的检查项清单,也可能只包含简要的内容以证明需要移交的文档都已经齐备。另外,一份声明了检查目的和依赖的报告将被提交给管理者、项目负责人和项目人员。

软件质量保证计划中声明了检查的职责。在小型的项目中,项目负责人或者部门的质量协调员都可以执行检查。而在大型项目中,软件质量保证小组的一个成员可以发起检查,但是检查工作需要由一个审核团队来完成。这与前面所提到的配置控制团队很类似。在检查完成之后,根据特定的日程安排,项目人员将进行纠正缺陷的工作。

质量控制用于检测和纠正缺陷,而质量保证的目标则是预防缺陷的产生。虽然那些过程原本被认为是零缺陷的,但检测意味着它们确实可能存在缺陷。质量保证是一种管理职能,它采取建议和引导的方式,目的是预防而不是碰到问题再纠正。

3. 软件配置管理

系统中的各个软件元素都存在变化,软件配置管理所关心的就是标识、跟踪并且控制这些变化。通过对软件组件版本及关系的管理,它控制着整个软件系统的演化发展过程。

软件配置管理的目的就是在生命周期的各个阶段中,识别所有互相作用的组件,并且控制它们的版本演化。软件配置管理作为一条原则能够被应用到多个活动当中。这些活动包括软件开发、文档控制、问题跟踪、变更管理及维护。因为所有的组件和组件之间的关系都已经被定义清楚,所以软件配置管理能够在复用性方面带来巨大的收益。

软件配置管理包括一些活动。当设计和代码要发生变更时,必须查看文档及这个变更将要带来的影响。否则,这些活动就不允许产生变更。软件配置管理的目标是保证代码和相关的文档的一致性,文档所描述的产品和实际被测试的产品必须一致。因此,不切实际和随意的变更都不会发生。

对于那些并发的软件开发项目,软件配置管理将会带来很大的好处。它在使软件处于开发过程的同时,能够尽量使每一次变更都能得到控制。如果软件项目中有非常多的变更发生,或者组件的选择具有很高的风险,那么软件配置管理将会对这些情况起到非常明显的稳定

作用。

1.1.5　软件配置管理的组成

软件配置管理识别了系统的配置，从而在软件生命周期中能系统化地控制变更、维护一致性、增加配置的易处理性。需要控制的部分包括计划、分析、设计文档，源代码，可执行代码，工具，任务控制语言，测试计划，测试脚本，测试用例，以及开发报告。软件配置过程通常包括四个部分 软件组件识别、软件版本控制、配置生成和软件变更控制。

1. 组件识别

一个基本的软件配置管理活动就是软件组件的识别。在开发过程的每个阶段，这些组件就组成了可以交付的部分。软件配置管理提供了一些原则，用来识别和命名软件基线、软件组件和软件配置。

软件组件要经历一系列的变更。为了管理开发过程，唯一地命名各个版本，就必须建立一套方法和命名标准。为组件版本命名的一个简单的办法就是使用一系列分隔的数字。第一个整数可能表示软件的外部发布版本，第二个整数可能表示内部发布版本。例如，从版本 2.9 转变到 3.1 就意味着有了一个新的外部发布版本 3。当软件组件被签入到软件库中时，它的版本数会自动增加。其他的限定词也可能被用上，例如新版本的日期。

一些软件的组成部分共同合作完成了某一项业务功能，这些组成部分的集合就被称为一个软件配置，例如一个订单系统的所有程序模块就是一个软件配置。识别配置与识别单个的软件组件很相似。配置也可以具有一系列的版本。每一个配置的命名不能与其他配置相同。每一个配置的版本都必须与其他版本不一样。识别一个配置，也要包括它的批准状态，以及对如何生成的描述。

识别配置的一种简单的技术就是把所有的软件组件存放在一个库中。所有组件的清单也要有相应的文档。

2. 版本控制

一个应用程序经过一段时间之后，就会有很多个不同的版本，此时就需要有一个组织化的过程来管理软件组件及它们之间的关系。另外，也需要对并行的组件开发和维护进行支持。

软件的一连串暂时的状态被称为版本。在版本演化的过程中，软件是不断变化的。软件配置管理中的版本控制工具就是一个软件配置管理库。版本控制能够记录每一次的软件变更，包括谁做的变更，变更原因和变更时间。

在软件的生命周期中，软件组件不断演化，在到达某一点时会处于一个相对稳定的状态。当缺陷被纠正，系统属性增强之后，这些变化就导致了组件的新版本的产生。保持对这些软件版本的把握就称为版本控制。

为了与其他组件相区别，一个组件会被识别并且进行标识。当一个组件被修改之后，它的旧版本和新版本应能够被单独地识别出来。所以，所有的版本，除了初始版本之外，都具有一个前辈。版本的连续性标志着组件的历史。不同的版本也可以作为备份来使用，可以将软件恢复到以前的版本。

3. 配置生成

为了生成软件配置，必须正确识别软件版本和执行组件生成过程，这经常被称为配置生成。

一个软件配置包含了一系列得到的软件组件,例如从源代码得到可执行的目标程序。得到的软件组件要和原组件正确地关联,这样才能准确地匹配。配置生成模型定义了得到的组件组合在一起的方式。

配置生成模型需要一些输入和输出。输入包括主输入,例如源组件,版本选择过程,定义组件如何关联的系统模型;而输出就是目标配置及独立获得的软件组件。

软件配置管理环境使用不同的方法来选择版本。选择版本最简单的方法就是维护一个组件版本的列表。其他的方法包括选择最近测试过的组件,或者选择那些在特定日期被修改的组件。

4. 变更管理

软件的修改需要被提交、评估、批准或者拒绝、制定进度并且跟踪,而这些都依赖于一个过程,即变更控制。变更控制的基础就是变更控制过程,这个过程报告组件状态,它是一个审核的过程。

软件变更控制用于在使软件发生变更时进行决策。一些被提交的变更将被批准,然后获得执行。其他被拒绝或者推迟的变更,就不会被执行。变更控制还为影响分析决定依赖。

对一个配置的修改至少包括四个方面:一个变更请求、一个变更影响分析、一系列修改和新加的组件,以及作为新基线可靠地安装修改的方法。

一个变更往往导致多个软件组件的修改,所以仅仅为一个文件提供多个版本的存储系统一般是不够用的。有一种技术可以把一系列的修改当成一个变更进行识别,这种技术称为 Delta 存储(Delta Storage)。

每一个软件组件都有一个开发生命周期。一个生命周期包含了许多的状态及状态之间的过渡。当一个软件组件产生变更时,它应当总是被复审,从而在下一个版本出来之前,不会产生新的变更。负责复审的人必须赞成或者拒绝被修改的组件,一旦软件组件被确定之后,它们都将被放到软件库中。软件库同时也是所有被批准的组件的仓库。

一个被提交的组件与它的源是相联系的,并且它们具有相同的状态。另外,一个配置不能拥有比其他任何一个组件更多的状态。因为当一个配置的相关组件并没有全部确定时,此时对它进行复审是没有意义的。

所有被软件配置管理所控制的组件都存放在软件配置库中,包括工作产品,例如业务数据和工作模型、架构组、设计单位、测试完毕的应用软件、可复用的软件,以及其他特别的测试软件。当一个软件组件要被修改的时候,它会从仓库中被单独签出,到一个专有的工作区当中。就在它暂时离开配置管理控制区域的时间内,它可能经历了许多状态的变化。

当一个变更完成之后,这个组件将被签入到库中,并且成为新的软件版本。但是先前的组件版本也会被保留。

1.2 软件质量保证计划

在软件开发中,为了保证一定的软件质量,需要一些质量保证措施。软件质量保证计划(Software Qualty Assurance,SQA)就是讲这些措施简要地陈列出来的方法。在将软件的实际质量与计划中预期的质量相比较时,软件质量保证计划将作为比较的基线。当软件的实际质量等级达不到计划中的质量等级时,管理层就会按照计划采取相应的措施。

这个计划为开发可理解的和可维护的代码提供了框架和准则。这些因素也将保证软件项目的质量。生产和维护高质量的软件需要在内部按照契约来进行,而软件质量保证计划能够提供一些过程来保证这一条件。这些过程会对软件产品的计划、设计、编码、测试、文档、存储及维护产生影响。它必须以这样的方式进行组织,因为计划只是保证软件的质量,而不是对开发维护软件的细节过程进行描述。

开发并执行一个软件质量保证计划的步骤一般过程如下。

步骤 1 :制订计划

软件质量保证计划需要包括以下部分。

目的 :描述软件质量保证计划的目的和范围。要列举计划中所涉及的软件项的名称及其使用目的。同时也要对每一个软件项所涵盖的生命周期部分进行声明。

引用文档 :罗列计划中所引用的所有文档。

管理 :描述项目的组织结构、任务和责任。

文件 :所有监督开发、查证和确认、使用和维护软件的文档都在这一部分进行说明。同时也会对如何检查文档的完备性进行说明,包括确定一个文档是否完备的评测标准。

标准、实践、习惯、公约 :包括需要遵守的标准、实践、习惯和公约,同时也描述了如何监控和保证符合这些项的要求。

复审和检查 :定义技术上的和管理上的复审、走查及检查,也定义如何完成复审、走查、审查、其他后续活动及批准。

软件配置管理 :这一部分的内容将在项目软件配置管理计划中进行详述。

问题报告和纠正措施 :这一部分的内容将在项目软件配置管理计划中进行详述。

工具、技术和方法 :对那些为了支持软件质量保证计划而使用到的特殊工具、技术、方法进行声明。声明使用目的,描述如何使用。

代码控制 :在开发的所有阶段会产生很多软件版本。为何维护、存储、文档化这些版本,并且保障它们的安全,就需要定义一些方法和工具。这一部分就是对这些方法和工具的定义。在实施的时候可能会加入计算机程序库,程序库可能是软件配置管理计划的一部分,也可能不是。

介质控制 :声明为了识别每一个计算机产品的介质所用到的工具和方法,也包括存储这些介质所需要的文件。另外还包括了复制和恢复过程。保护程序的物理介质在开发的各个阶段中不受到非法访问,无意损坏。这些内容可能也会由软件配置管理计划来提供。

提供者控制 :为了使提供者满足预定需求,这一部分对附加条款进行了陈述,从而保证提供者能够获得足够和完备的需求。对于以前开发的软件产品,这一部分将对保证产品稳定性的方法进行陈述。对于将要开发的软件,提供者会根据标准制定和实施一个软件质量保证计划。同时,对于标准所规定的开发者必须遵守的要求,也会进行陈述。

记录收集、维护和保持 :说明要被保留的软件质量保证计划的文件。它声明用来组装、保护、维护这些文件的方法和工具,并且会制定保持期。对于要执行的计划及计划的开发,批准对于软件质量保证计划的实施是必要的。然后,软件质量保证计划将被作为它的执行结果。

测试方法 :定义需要用到的测试方法,技术和自动化工具。

步骤 2 :获得管理层的赞同

成功地实施软件质量保证计划,管理层的参与是必不可少的。管理层负责保证软件项目

的质量,同时提供软件开发所需要的资源。

对于一个项目来说,管理层参与的程度取决于项目的范围。如果一个项目跨越了组织边界,那么批准就来自于所有相关的领域。如果获得批准,那么软件质量保证计划就已经处于配置控制之下。

在管理层批准过程中,管理层将对软件质量的控制权交给软件质量控制计划管理员,以此来提高软件质量。软件质量又常常是开发人员的责任。质量是重要的,但是管理层也不得不考虑正规的软件质量保证计划所带来的成本。员工们必须明白,管理层把这个计划看成改进质量的方法,而不是一个结束。

每一个被实施的项目,无论它是否具有软件质量管理计划,都需要将成本进行正式的估算。总而言之,实施一个正规的软件质量管理计划,对于管理层来说,在经济上是明智的。

步骤 3 :获得开发人员的赞同

因为软件开发和维护人员是软件质量管理计划的主要使用者,所以他们的合作态度是非常重要的。软件项目组的成员必须坚决支持项目的软件质量保证计划,每个人都必须接受它,照章执行。

在制订计划的过程中,如果没有软件组成员和他们的管理者参与进来,软件质量保证计划是不会很好地执行的。因为项目组通常只有少数成员,所有的团队成员都要积极地参与到计划的编写工作当中。当一个项目变得非常大的时候,项目子小组的代表就要提供输入。来自代表的及时反馈将有助于团队成员获得对计划的赞同。

步骤 4 :准备实施软件质量保证计划

计划、表达和起草软件质量保证计划的过程需要员工和文字处理资源。负责实施计划的人员应该有权获得这些资源。另外,资源的分配需要经过管理层的同意和支持。为了方便资源的分配,管理层应该意识到任何可能阻碍实施过程的项目风险 (例如,有限的员工和设备)。对于软件质量管理计划的起草,复审和批准,都需要制订计划。

步骤 5 :执行软件质量管理计划

实际的执行计划的工作是由软件开发和维护团队来完成的。其中,为了监控过程,需要选择必要的审核点。在软件产品实现的过程中就应该把审核工作计划好。这样的话,对于软件项目不正确的监控才不会伤害到软件质量保证计划。在开发中,需要每隔一段距离就设置一个审核点。另外,对于特别的里程碑,也可以设置审核点 (例如,在主复审或者项目的一部分已经被提交之后)。

1.3 测试技术常用方法介绍

错误猜测是一种不正规的测试技术,它依赖于激发、创造性思维和头脑风暴来设计测试。虽然这种测试技术非常重要也很管用,但是它还是不能取代正规的测试技术。正规测试技术为保证测试覆盖面和可靠程度提供了更大的可能性。测试用例的设计需要一个规格说明,用来描述功能输入和功能输出。测试用例也是系统文档的一部分。每一个测试用例对于测试目标必须有一个清楚的定义。风险管理将有助于确定测试目标,尤其是在高风险的部分。下面介绍主要的测试设计方法。