密级：　　保密期限：

北京郵電大學

工程硕士研究生学位论文

题目：敏捷开发过程中软件测试技术的分析与应用

学　　号：03R0057

姓　　名：董　博

专　　业：软件工程

导　　师：杨文川

学　　院：软件学院

2010年　3月

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：黄撑　　日期：2010，6. 19

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。

本学位论文不属于保密范围，适用本授权书。

本人签名：黄撑　　日期：2010，6.19

导师签名：[illegible]　　日期：2010.6.19

敏捷开发过程中软件测试技术的分析与应用

摘 要

近年来随着软件产品应用到社会的各个领域，软件产品的质量自然成为人们关注的焦点。软件危机曾经是软件界甚至整个计算机界最热门的话题，为了解决这场危机，软件从业人员、专家和学者做出了大量的努力。现在人们已经逐步认识到所谓的软件危机实际上仅是一种状况，那就是软件中有错误，正是这些错误导致了软件开发在成本、进度和质量上的失控。Agile(敏捷)是近年来新兴的一种开发模式，SCRUM是敏捷开发方式中的一种，它是一种基于团队的敏捷软件过程。它承认软件开发过程是不可预期的，也强调软件过程是可控的，因此它把软件开发过程看作是一个可控的黑盒，是对迭代式面向对象方法的改进。采用SCRUM的开发团队具有高度自主权，紧密地沟通合作，以高度弹性解决各种挑战，确保每天、每个阶段都明确地朝目标推进，因此SCRUM非常适用于软件开发项目。目前，随着SCRUM在软件开发实践中表现出很好的应用价值，极大地提高了软件团队的生产效率，目前它已经受到IBM、微软等大公司的关注，并且开始积极实践探索。

本文结合IBM WebSphere Commerce Server（WCS）项目对SCRUM开发过程的具体实践，对软件测试在SCRUM开发过程中的应用经验进行了较为深入的研究，主要做了以下几点工作：

1、介绍和分析了软件测试的发展过程

2、介绍和分析了敏捷开发过程，以及目前基于敏捷开发的测试方案

3、简单介绍了IBM WebSphere Commerce Server产品，以及该产品的开发模式的变化，并分析了敏捷开发过程在WCS项目中应用的情况

4、结合WCS项目对敏捷开发的具体实践，设计了基于敏捷开发的测试模型，同时详细的介绍和分析了软件测试在敏捷开发过程中应用时遇到的问题，并提出改进意见和解决方法。

关键词：敏捷开发过程　软件测试　SCRUM　SPRINT

The Application and Analysis of Software Testing in Agile

Abstract

Nowadays, along with the software being used in more and more fields of our lives, the quality of software is becoming the focus of people concerns. The crisis of software was one of the hottest topics in software field even computer science. To find a way out, people working in the field, professionals and researchers devoted a large number of efforts. People have figured out that the so called software crisis is just a situation which means that there are errors in the software. The errors not only make the progress and qualify out of control, but also result in the increasing of cost.

Agile is a new development method which is being widely used. SCRUM is a variation of Agile. It is a process of Agile which is based on team work. Agile agrees that software development is unpredictable. In the hand, it also emphasizes that it software development is controllable. Therefore, Agile treats software development like a controllable black box, which is a improvement of iterative object-oriented. The team who is using SCRUM has high autonomy, seamless communication, cooperation and resolving all kinds of challenges with elastic ways. They can ensure that they are moving to the correct target in everyday and every stage. That is why SCRUM is perfectly suitable for software development projects.

Along with the values have been figured out during the practice of software development and the improvement for the efficiency of software team, SCRUM is highly concerned by IBM, Microsoft and other large companies. They are interested in practicing and researching SCRUM in their software development process.

The paper is combined with the using of SCRUM in IBM WebSphere Commerce Server (WCS) and researches deeply for the application of software testing in SCRUM. The paper has done the following researches:

1. Introduce and analyze the growth of software testing.
2. Introduce and analyze the process of SCRUM and the testing solution based on SCRUM.
3. A brief for IBM WebSphere Commerce Server and the changes of the method for its development. Analyze the application of the SCRUM being used in WCS.
4. According to the application of SCRUM in WCS, design a testing model for SCRUM. Introduce and analyze the issues found in software testing of SCRUM in detail and provide a suggestions and solutions.

Key words: Process of Agile Software Testing SCRUM SPRINT

目录

第一章 绪论

1.1 研究背景

信息技术的飞速发展，使软件产品应用到社会的各个领域，软件产品的质量自然成为人们共同关注的焦点。不论软件的生产者还是软件的使用者，均生存在竞争的环境中，软件开发商为了占有市场，必须把产品质量作为企业的重要目标之一，以免在激烈的竞争中被淘汰出局。用户为了保证自己业务的顺利完成，当然希望选用优质的软件。质量不佳的软件产品不仅会使开发商的维护费用和用户的使用成本大幅增加，还可能产生其他的责任风险，造成公司信誉下降，继而冲击股票市场。在一些关键应用(如民航订票系统、银行结算系统、证券交易系统等)中使用质量有问题的软件，还可能造成灾难性的后果。

软件危机曾经是软件界甚至整个计算机界最热门的话题，在这30多年的发展过程中，尽管并没有找到真正的“银弹 (Silver bullet)”[1]来彻底解决“软件危机”问题。但是，为了解决这场危机，软件从业人员、专家和学者做出了大量的努力。现在人们已经逐步认识到所谓的软件危机实际上仅是一种状况，那就是软件中有错误，正是这些错误导致了软件开发在成本、进度和质量上的失控。有错是软件的属性，而且是无法改变的，因为软件是由人来完成的，所有由人做的工作都不会是完美无缺的。问题在于如何去避免错误的产生和消除已经产生的错误，使程序中的错误密度达到尽可能低的程度。

因此本文认为应从以下两个方面来考虑如何解决软件开发过程中所遇到的问题:

1)重视软件测试

测试是所有工程学科的基本组成单元，是软件开发的重要部分。自有程序设计的那天起测试就一直伴随着。统计表明，在典型的软件开发项目中，软件测试工作量往往占软件开发总工作量的40%以上。而在软件开发的总成本中，用在测试上的开销要占30%到50%[2]。如果把维护阶段也考虑在内，讨论整个软件生存期时，测试的成本比例也许会有所降低，但实际上维护工作相当于二次开发，乃至多次开发，其中必定还包含有许多测试工作。因此，测试对于软件生产来说是必需的，问题是应该思考“采用什么方法、如何安排测试?”

2)运用正确的方法论来指导软件开发

软件危机产生的重要原因之一就是开发过程没有统一、规范的方法论，造成忽视软件开发前期的需求分析，文档资料不齐全，忽视人与人之间的沟通和

交流，忽视测试阶段的工作，提交用户的软件产品质量差等等。软件工程诞生之后，确实缓解了一些软件危机，特别是20世纪90年代由卡耐基梅隆大学软件工程研究所发布的软件过程能力成熟度模型(Capacity Maturity Model，简称CMM)很大程度上解决了软件危机。但是CMM的缺点使得它越来越不适应当前软件市场需求的快速发展变化[3]。敏捷软件开发是不同于以往传统软件工程的开发方法，它更注重从实践中获取和总结对软件开发有益的经验和原则。SCRUM是敏捷软件开发方法中的一种，目前已经被很多软件开发团队关注并采用，取得了很好的效果。

1.2 论文研究的现状

1.2.1 软件测试发展现状

国外软件测试起步较早，尤其是在软件规模越来越大，复杂程度不断变高的情况下，软件测试得到了很大发展。目前不论是管理方式，还是技术水平都已经发展到了一个相对成熟的阶段。在测试的投入方面，国外软件企业从来不吝惜人力和物力。把尽可能多的问题消灭在软件交付之前，始终是他们工作的重中之重。

现如今，国外软件测试研究的重点之一是测试自动化。测试自动化可以将测试人员从简单低效的工作中解脱出来，从而有更多的时间和精力去关注和分析造成缺陷的原因，目前国外许多公司已经开发出相当多的自动化测试工具[4]。国内的软件测试起步较晚，主要是由于国内软件生产在相当长的一段时间内都处于小规模的个人英雄主义状态。在这种状态下，软件过程的管理难度并不大，因此缺陷也不多，或者说不够致命。所以软件测试并没有得到足够的重视。不过近些年，随着软件测试在国内外包领域的蓬勃发展，软件测试也受到了国内软件企业的关注。尤其是国外的经验教训已经被越来多的人所重视。并且客观上，随着国内软件生产规模不断扩大，以及国外资源的向东(中国和印度)移动，对软件测试的需求也越来越多，要求也越来越高。目前，国内许多重点大学都开有软件测试课程，为我国软件测试行业的发展储备人才。另外，国内越来越多的软件外包公司，把目标瞄准了软件测试这个巨大的市场。很多国内IT企业都在从事或准备从事为国外IT企业提供软件测试的服务。并且这种软件测试的第三方服务也是软件测试本身的要求。因此，国内的软件测试虽然起步晚，但必定有很大的发展空间。

软件测试领域的改革已经改进了编写测试脚本和生成测试用例的技术，在

多年的进化过程中，软件测试希望可以合并各种测试技术，成为自动化测试。

1.2.2 SCRUM发展现状

Agile(敏捷)一词来源于2001年初美国犹他州雪鸟滑雪胜地的一次敏捷方法发起者和实践者的聚会，他们发起组成了敏捷联盟[5]。在随后的几个月中，他们创建出了一份价值观声明，也就是敏捷联盟宣言。它的内容如下：

“我们正在通过亲身实践以及帮助他人实践，揭示更好的软件开发方法。通过这项工作，我们认为：

1)个体和交互胜过过程和工具；

2)可以工作的软件胜过面面俱到的文档；

3)客户合作胜过合同谈判；

4)响应变化胜过遵循计划。

虽然右项也有价值，但是我们认为左项具有更大的价值。”

SCRUM是敏捷软件开发方式中的一种，它是一种基于团队的敏捷软件过程。它首先承认软件开发过程是不可预期的，但也强调软件过程是可控的，因此它把软件开发过程看作是一个可控的黑盒。SCRUM最初由Ken Schwaber和Jeff Sutherland提出[6]，是一种旨在寻求充分发挥面向对象和构件技术的开发方法，是对迭代式面向对象方法的改进。SCRUM这个单词来源于英式橄榄球比赛中争球队形的名字，用来比喻整个软件开发团队要像比赛中的队员一样时刻保持对场上局势的清醒判断，然后通过集体行动，努力实现大家共同的胜利。SCRUM将软件开发团队比拟成橄榄球队，有明确的最高目标，熟悉开发流程中所需具备的最佳典范与技术，具有高度自主权，紧密地沟通合作，以高度弹性解决各种挑战，确保每天、每个阶段都明确地朝目标推进，因此SCRUM非常适用于软件开发项目。SCRUM方法最初在1993年被Easel公司用于实践[7]，后来被很多公司应用，适用于需求难以预测的复杂商业应用产品的开发。SCRUM提出的SCRUM Meeting、Sprint、Backlog、SCRUM Master、SCRUM Team、Demo等模式已被规范编程语言标准组织(Pattern Languages of Programming，简称PLOP)作为组织和过程模式(Organizational and Process Pattern)的标准。而且，随着SCRUM在软件开发实践中表现出很好的应用价值，极大地提高了软件团队的生产效率，目前它已经受到IBM、微软等大公司的关注，并且开始积极实践探索。

1.3 研究内容及主要工作

IBM 公司的 WebSphere Commerce Server（WCS）产品是 IBM WebSphere 软件品牌下专业致力于电子商务解决方案的软件产品。在经历了数十年的不断发展，已经形成以商品管理、销售管理、订单管理、用户管理、营销管理等为核心模块，以电子商务行业最佳实践为支撑的一整套解决方案。该产品最新的版本的开发模式由传统的瀑布模型转向了敏捷开发模式。近几年来，作者在 WCS 开发项目的测试团队中从事着功能测试（FVT）的工作，通过研究和分析 WCS 项目对 SCRUM 开发过程的实践情况，设计了基于 SCRUM 的测试方案，并对该方案的执行过程进行了介绍，并重点介绍和分析了 FVT 测试和 SVT 测试的实际执行情况，以及作者本人在 FVT 测试中的具体的工作内容，最后根据实际的应用情况，研究分析出可能遇到的问题，并提出相应的改进方法，并对 WCS 项目对 SCRUM 测试方案的应用情况进行总结。

1.4 论文组织

第一章 绪论

论述了的研究背景及意义，分析国内外最先进技术的发展与现状，介绍论文的主要工作和内容安排。

第二章 软件测试的理论和方法

主要介绍软件测试的理论和方法。

第三章 当前 SCRUM 测试方案的介绍与分析

本章中主要介绍了 SCRUM 开发过程的理论，并对目前一般采用的 SCRUM 测试方法进行了介绍和分析。

第四章 WCS 项目 SCRUM 测试方案的设计与实现

本章首先简单的介绍了 WCS 产品和项目的发展情况，然后介绍了 WCS 产品开发模式的变化。随后设计了基于 SCRUM 的测试方案，并对其各个方面进行了介绍。然后介绍了该方案在 WCS 产品测试中的过程管理。

第五章 SCRUM 测试方案在 WCS 项目中的实施与分析

介绍了 WCS 项目对 SCRUM 测试方案的具体实施情况，以及作者在 FVT 测试中的工作，并分析测试结果论证 SCRUM 测试方案的可行性、有效性。最后总结研究了在实施过程中 FVT 和 SVT 遇到的问题，并提出解决方案。

第六章 总结与展望

总结本文的研究成果，并且指出了进一步的研究方向。

第二章软件测试的理论和方法

2.1 软件测试概述

2.1.1 软件测试的定义

什么是软件测试，在表2－1中记录了过去若干年对测试的定义。

表2-1 过去若干年对测试的定义

年份	定义
1979	测试是为发现错误而执行一个程序或者系统的过程[8]。
1983	测试是以评价一个程序或者系统的属性为目标的任何一种活动，测试是对软件质量的度量[9]。
2002	测试是指为了度量和提高被测试软件的质量，对测试件进行工程设计，使用和维护的并发生命周期过程[10]。

第一个定义是1979年 Glenford J. Myers 在自己的经典著作《软件测试艺术》一书中解释的。在Myers 撰写该著作的时期，他的定义可能是人们所能发现的最好的定义，并反映了人们在当时对测试所持的观点。简而言之，测试发生在软件开发周期的末期，其主要目的是发现错误。

如果再跳到1983年就会发现，测试的定义已经发生了改变，测试已经包含了对软件质量进行评估的内容，而不仅仅是一个发现缺陷的过程。第二个定义是在《Complete Guide to Software Testing 2nd Ed》一书中，由 Bill Hetzel 指出的。Myers 和 Hetzel 的定义在当今仍然有效，因为他们都阐明了软件测试的一个特定方面。但是，这些定义所存在的问题在于其范围。

而第三个定义正是为了解决这个问题，第三个定义是 Rick D. Craig 和 Stefan P. Jaskiel 在《系统的软件测试》中提出的。在该定义中，并没有“发现缺陷”的直接提法，尽管发现缺陷肯定仍然是测试的一个有效目标。而且该定义不仅包括度量，而且包括提高软件的质量。这种测试被人们称作“预防性测试”。

2.1.2 软件测试的目的

软件测试的目的决定了如何去组织测试。如果测试的目的是为了尽可能地

找出错误，那么测试就应该直接针对软件比较复杂的部分或是以前出错比较多的位置。如果测试目的是为了给最终用户提供一定可信度的质量评价，那么测试就应该直接针对在实际应用中会经常用到的商业假设[11]。对于软件测试目的的理解，论文在这里引用 Glenford J. Myers 在《软件测试艺术》一书中的观点：

1)软件测试是为了发现错误而执行的过程；

2)测试是为了证明程序有错，而不是证明程序无错误；

3)一个好的测试用例是在于它能发现至今未发现的错误；

4)一个成功的测试是发现了至今未发现的错误的测试。

这种观点可以提醒人们测试要以查找错误为中心，而不是为了演示软件的正确功能。但是仅凭字面意思理解这一观点可能会产生误导，以为发现错误是软件测试的唯一目的，查找不出错误的测试就是没有价值的，事实并非如此。

首先，测试并不仅仅是为了要找出错误。通过分析错误产生的原因和错误的分布特性，可以帮助项目管理者发现当前所采用的软件过程的缺陷，以便改进。同时，这种分析也能帮助软件测试人员设计出有针对性地检测方法，改善测试的有效性。其次，没有发现错误的测试也是有价值的，完整的测试是评定测试质量的一种方法，详细而严谨的可靠性增长模型可以证明这一点。

2.1.3 软件测试的阶段

在软件测试开始前，必须首先对软件测试的各阶段有非常清楚的了解，才能明确在软件测试中需要做哪些工作。软件测试工作一般要通过计划测试、设计测试、实现测试、执行测试、评估测试几个阶段来完成。其中计划测试阶段需要制定测试计划、整理测试需求；设计测试阶段要设计测试用例和测试过程，并保证测试用例完全覆盖测试需求；实现测试阶段要根据测试用例实现具体的自动化脚本或者手工的操作步骤；执行测试阶段则通过自动化测试工具或人手工来执行那些自动化或手工脚本；最后的评估阶段则要对软件的质量和测试工作自身的质量做出一个客观的评价。

2.1.4 软件测试的周期

软件的整个测试生命周期是与软件的开发生命周期基本同步的过程[12]，即当需求分析基本明确后，测试人员就应该基于需求分析的结果和整个项目计

划来进行软件的测试计划;伴随着分析设计过程同时应该完成测试用例的设计;当软件的第一个发布出来后，测试人员要马上基于它进行测试脚本的实现，并基于测试计划中的测试目的执行测试用例，对测试结果作出评估报告。这样，就可以通过各种测试指标实时监控项目质量状况，提高对整个项目的控制和管理能力。

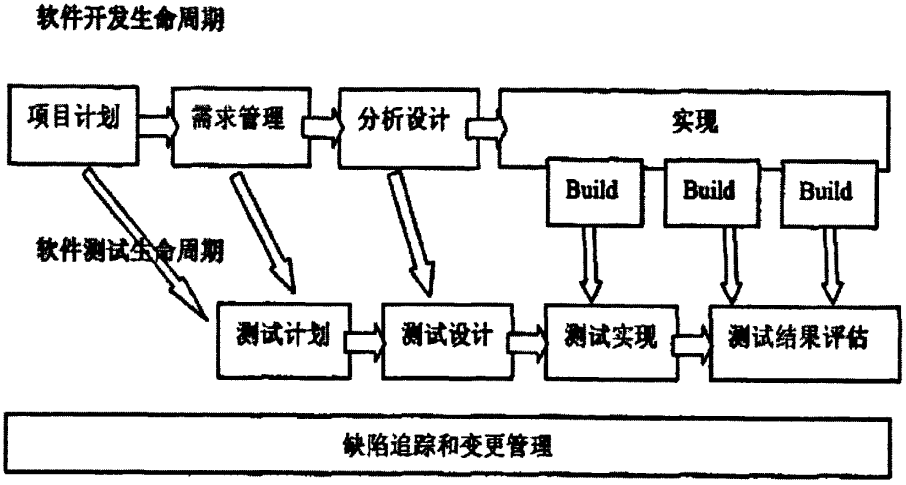


图 2-1 软件测试生命周期

2.1.5 软件测试的方法

从代码和用户使用的角度可以将软件测试方法分为如下几种[13]，如图 2-2 所示

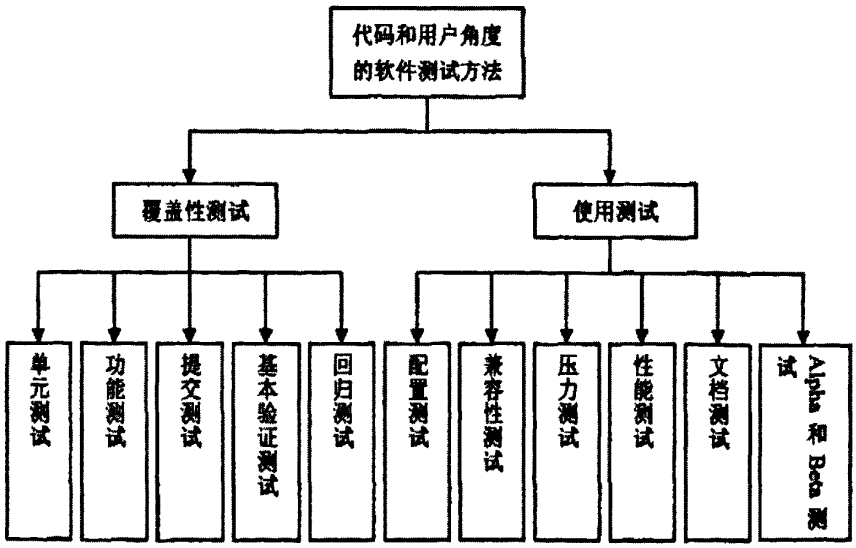


图 2-2 从代码和用户使用的角度的软件测试方法划分

其中：

覆盖性测试(Coverage Testing)是从代码的特性角度(即内部)出发的测试方法，包括以下方式。

1)单元测试(Unit Testing)

按照代码的单元组成逐个进行测试。

2)功能测试(Function Testing)或特性测试(Feature Testing)

按照软件的功能或特性逐个进行测试。例如，在Windows Live Message中，发送和接收即时信息就是两个不同的功能或特性，在测试时就分别对它们进行检查，看是否工作正常。

3)提交测试(Check In Testing)

在开发人员对代码做了任何修改，或者修复了某个Bug(软件缺陷)时，需要重新提交代码(即将修改后的代码放大到整个大的系统中)。这时，开发人员往往也要进行测试，看代码是否工作正常。为了保险起见，开发人员往往要找测试人员帮着一起进行测试(这种方式称做Bug Testing)。测试人员和开发人员之间注意协调关系是非常重要的。

4)基本验证测试(Build Verification Testing，简称BVT)

对完成的代码进行编译和连接，产生一个构造，以检查程序的主要功能是否会像预期一样进行工作。这是最简单而又最省时的一种测试方法。每产生一个新的构造时都要进行测试。如果连BVT都通不过，表明问题很严重，开发人员需要尽快修复出现的问题，测试人员也就不用浪费时间做其他测试了。

5)回归测试(Regression Testing)

过一段时间以后，再回过头来对以前修复过的Bug重新进行测试，看该Bug是否会重新出现。

使用测试(Usage Testing)是从用户的角度(即外部)出发的测试方法，它也包括许多类型：

1)配置测试(Configuration Testing)

从用户的使用出发进行多方面的测试。例如，保证软件不仅能够在Windows XP下运行也能够在Windows ME下运行，还能够在Windows NT/2000/2008下运行;或者软件不仅能够在配置高的计算机上运行，也能够在配置很低的计算机上正确地运行。总之，要考虑到用户的多种情况，用多种配置对软件进行测试。

2)兼容性测试(Capability Testing)

主要考虑兼容性问题，比如同一个产品的不同版本(如 Offiee2000 和 Office XP)之间的兼容问题，不同厂家的同一个产品(如 IE 和 Netscape)之间的兼容问题，不同类型软件(如 IE 和 Office)之间的兼容问题等。最难测试的往往就是软件的兼容性问题，往往要投入巨大的人力和物力。一些厂商开发出来的产品在兼容性上做得很不好，就是因为没有足够的人力和物力进行测试。

3)压力测试(Stress Testing)

在各种极限情况下对产品进行测试(如很多人同时使用该软件，或者反复运行该软件)，以检查产品的长期稳定性。本项测试可以帮助找到一些大型的问题，如死机、崩损、内存泄漏等，因为有些存在内存泄漏问题的程序，在运行一两次时可能不会出现问题，但是如果运行了成千上万次，内存泄漏得越来越多，就会导致系统崩溃。

4)性能测试(Performance Testing)

本项测试是保证程序具有良好的性能。如果同类的产品只需 5 秒钟就能得出结果，而自己的产品需要 10 秒钟才能得出结果，就说明自己的产品性能不好。如果在测试过程中发现性能问题，修复起来是非常艰难的，因为这常常意味着程序的算法不好，结构不好，或者设计有问题。因此在产品开发的开始阶段，就要考虑到软件的性能问题。

5)文档和帮助文件测试(Documentation and Help file Testing)

因为用户通常是通过文档和帮助文件来学习使用产品的，如果文档和帮助文件存在错误，就可能会导致用户无法正常使用产品。这项工作通常在产品即将包装发布时进行，以避免在修复 Bug 的过程中需要反复修改文档，或者忘记修改文档，导致文档与产品的特性不相符。

6)Alpha 和 Beta 测试(Alpha and Beta Testing)

在正式发布产品之前往往要先发布一些测试版，让用户能够反馈出相关信息，或者找到存在的 Bug，以便在正式版中得到解决。

另外一种分类方法就比较好理解了，主要将软件测试方法分为如下几种：

1)手工测试(Manual Testing)

即依靠软件测试人员手工来查找 Bug。

2)自动测试(Automation Testing)

即编写一些测试工具，让他们自动运行来查找 Bug。自动测试的优点是能够很快、很广泛地查找 Bug，缺点是它们只能检查一些最主要的问题，如崩溃、死机，但是却无法发现一些一般的日常错误，这些错误通过人眼很容易找到，但机器却往往找不到。另外，在自动测试中编写测试工作量也很大，因此在实

际测试中通常是手工测试和自动测试相结合，而且手工测试往往是主要的，占了1/2－2/3，而自动测试只占1/3－1/2。在不同的开发队伍中，这个比例会有所不同，但总体趋势是这样的。

2.1.6 软件自动化测试

软件自动化测试就是执行用某种程序设计语言编制的自动测试程序，控制被测软件的执行，模拟手动测试步骤完成全自动或半自动测试[14]。

全自动测试就是指在自动测试过程中，完全不需要手工干预，由程序自动完成测试的全过程。半自动测试就是指在自动测试过程中，需要手工输入测试用例或选择测试路径，再由自动测试程序按照指定的要求完成自动测试。

软件测试自动化工具的开发对任何一家软件公司而言都是一笔不小的投资，需要想办法从这项投资中获得相应的回报。运用软件测试自动化工具进行回归测试以及重复使用软件测试自动化工具是从这项投资中获得相应回报的有效手段。

因此软件测试自动化的目的可以概括为:被开发出来的软件测试自动化工具应当能够运用于公司以后开发的测试版软件和发布版软件的自动测试[15]。

使用自动化测试的过程中会遇到许多问题，意外出现的问题尤其难以处理。下面总结一些自动化测试中普遍存在的问题。

1)不现实的期望测试

业内一般对于任何新技术的解决方案都深信不疑，认为可以解决面临的所有问题。测试工具也不例外，对新工具持乐观态度已成趋势。人们都期望这种解决方案可以解决目前遇到的所有问题，厂商也自然会强调好的和成功的一面。因此如果使用者对自动化测试工具期望过高，那么无论工具从技术角度实现的多么好，都满足不了期望。

2)缺乏测试实践经验

如果缺乏测试实践经验，测试组织差，文档较少或不一致，测试发现缺陷的能力较差，在这种情况下采用自动测试并不是好办法。改进测试的有效性比改进差劲测试的效率要好得多。

3)期望自动化测试发现大量新Bug

自动化测试在首次运行时最有可能发现Bug，但是如果测试已经运行并通过，再运行相同的自动化测试发现新Bug的可能性要小得多。除非自动化测试正执行一段已修改过的代码或由于软件其他部分的修改影响到该代码，或者在不同的环境中运行。自动化测试工具是“回放”工具，即回归测试工具，用于

重复已经运行过的测试，但并不是用来发现大量新的 Bug，特别是运行在与以前相同的硬件和软件环境中。

4)安全性错觉

测试软件中没有发现任何 Bug 并不意味着软件没有 Bug，测试可能不全面或测试本身就有 Bug。比如期望的输出不正确，那么自动化测试只是简单地保留这些有 Bug 的结果。

5)自动测试的维护性

当软件修改后，相对应的自动化测试部分也需要修改(如自动化测试脚本的修改)，自动化测试才能正确执行。但是自动化测试的维护开销相当大，因此当维护自动化测试所需花费大于手工重新测试花费时，测试自动化将被丢弃。

6)技术问题

自动化测试工具与以前其他软件的兼容性是目前软件测试自动化实施过程中遇到的一个非常严重的问题。技术环境变化之快，使得自动化测试工具生产工具厂商很难跟上，许多自动化测试工具在理论上可行，但在具体环境中执行时确是不可行的。商用自动化测试工具是较庞大且复杂的产品，并要求测试人员具有较好的技术知识，才能将工具充分利用。因此除了自动化测试工具本身的技术问题，测试人员还要了解被测软件的技术问题。如果软件在设计和实现时没有考虑可测试性，那么使用自动化测试工具测试这样的软件，无疑更增加测试的难度。

7)组织问题

自动测试实施起来并不简单，必须有管理支持及组织艺术。必须对测试人员进行自动化测试工具培训、实践并了解哪种工作方式最好，并在组织内普遍使用该自动化测试工具。

2.1.7 软件测试的工具

有了好的测试方法，还需要有高效好用的辅助工具，做软件测试通常需要以下一些基本工具。

1)计算机(硬件设备)；

2)优秀的办公处理软件(如字处理软件和表单软件，用于编写测试计划和规范)；

3)视频设备；

4)秒表(程序运算时间，测试产品性能)；

5)测试用例(Test case)及Bug的跟踪系统(提供Test case的Bug记录模板，能够存储记录Test case的Bug，实现记录的跟踪)；

6)自动测试工具(产生自动化测试一Automation脚本，解析并运行Automation脚本，拥有记录运行结果的日志系统)；

7)软件分析工具；

8)好的操作系统(如Windows NT/2000，其中有很多有用的工具，如文件比较器、文件查看器、文件转换器、内存监视器等)；

9)多样化平台等

第三章 当前 SCRUM 测试方案的介绍与分析

3.1 SCRUM 概述

3.1.1 SCRUM 方法的原理

SCRUM 将工业过程控制中的概念应用到软件开发中来，认为软件开发过程更多的是经验性过程(Empirical Process)，而不是确定性过程(Defined Process)。确定性过程是可明确描述的、可预测的过程，因而可重复(Repeatable)执行并能产生预期的结果，并能通过科学理论对其优化。经验性过程与之相反，应作为一个黑箱(Black Box)来处理，通过对黑箱的输入输出不断进行度量，在此基础上，结合经验判断对黑箱进行调控，使其始终处于可以控制的范围内，从而产生满意的输出。如将经验性过程按确定性过程来处理(如瀑布模型)，必将使过程缺乏适应力。SCRUM 是一个快捷轻便的过程，它通过适应和经验型的过程管理来实现迭代递增的开发过程，同时它根据需求的改变来快速调整检验和规划，使功能不断更新和加强，确保交付给用户的软件质量。

3.1.2 SCRUM 方法的过程

SCRUM 方法包括下面三个阶段，如图 3-1 所示

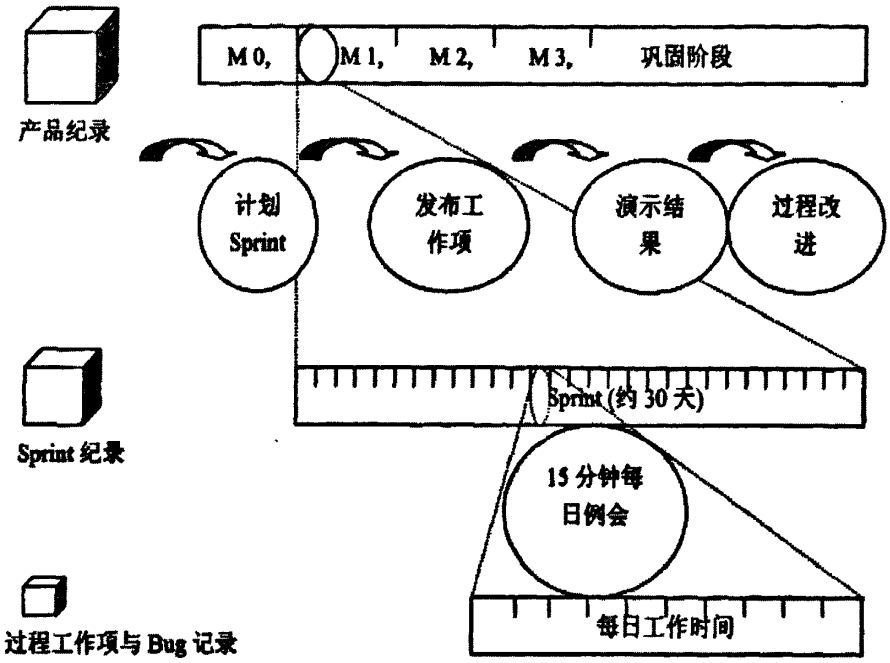


图 3-1 SCRUM 开发流程

1)体系结构设计阶段（确定性过程）

在 SCRUM 中，一个项目对应于一个 Product Backlog(产品记录)，在 Product Backlog 中列出项目开发过程中所有需要完成的任务，包括:未细化的产品功能需求、Bugs、用户提出的改进、具有竞争力的功能以及技术升级等。将这些任务按优先级排序形成 Backlog 列表，根据该表和风险评估制定产品交付基线，建立系统体系结构，如为已有系统改进或只作有限分析、调整，将 Product Backlog 项按高内聚低耦合的原则分解为一系列问题包(Packet)，每个问题包是一组对象或构件的集合，为定义 sprint 作准备，文章稍后将详细介绍 Backlog 的作用

2)Sprint 阶段[16][17]（经验性过程）

SCRUM 开发过程由一系列迭代的 Sprint 过程组成，一个 Sprint 过程就是一个冲刺过程，多个 Sprint 过程顺序进行，直至风险评估认为产品可交付为止。一个 Sprint 是在限定时间段内(Sprint 周期，通常为 2 一 4 周，可在前一个 Sprint 结束时调整)的一系列开发活动，包括分析、设计、编码、测试等。一个 Sprint 有自己特定的 Sprint Backlog，在 Sprint Backlog 中列出了开发团队在这个 Sprint 中要完成的所有任务。这些任务都是从 Product Backlog 中挑选出来的、一般属于同一问题包、并按照优先级排序的任务。

3)交付和巩固阶段(确定性过程)

一旦根据风险评估结果认为可交付产品时，即进入该阶段。该阶段的活动包括:组装，系统测试和回归测试(Regression)，准备培训材料，完成最终文档。SCRUM过程认为一个产品的开发将一直持续下去，除非经风险评估后认为应停止。产品交付后的巩固活动类似于传统方法中的维护和改善，目的在于整理Sprint阶段压力下忽略的工作，为下一个阶段的开发做准备，以便轻装上阵。简而言之，SCRUM就是一个使用迭代递增模型的快捷轻便的实践过程，它结合了极限编程和RUP的优点，是一种非常有助于提高软件生产效率的敏捷开发方法。

3.1.3 SCRUM方法的角色

从SCRUM团队的组织来看，SCRUM有两种重要的管理角色[18][19]:

1)产品经理

Product Owner代表整个产品线的利益，与SCRUM Master和SCRUM Team合作，他的主要任务就是负责管理和确定 Product Backlog中的任务列表的优先级次序，并且按照商业价值规划新产品的功能，侧重于考虑投资回报，相当于传软件测试的理论和方法统项目管理中的Product Manager，一般由比较熟悉市场的人员来担任

2)冲刺阶段负责人（Sprint Master）

SCRUM Master是一个SCRUM团队的管理者，他的任务就是领导每一个SCRUM成员执行SCRUM的规则，组织 Daily SCRUM会议。他也具有一定的决策权和消除Sprint进度中障碍的义务，并且在 Sprint Backlog的制订过程中起辅助作用。就目前微软公司的情况来看，采用SCRUM方法的团队并没有一位明确的 SCRUM Master，而是项目经理承担了 SCRUM Master的所有任务。

从SCRUM团队的分工来看，SCRUM由以下几种角色组成，如图3-2所示:

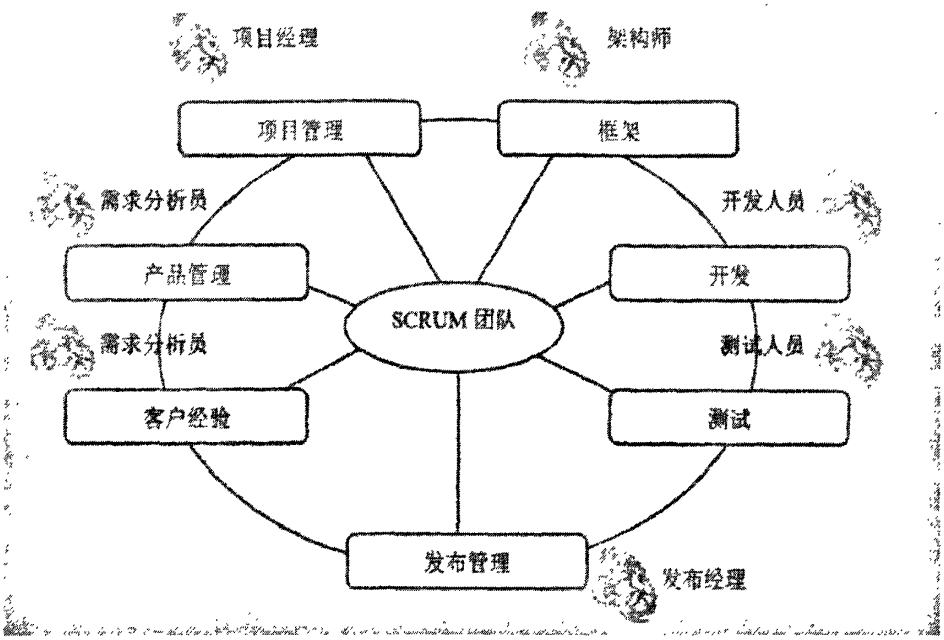


图 3-2 SCRUM 团队角色示例图

1)需求分析员(Business Analyst)

需求分析员的主要任务是分析产品需求和提出满足需求的应用软件。需求分析员和客户以及其他相关人员一起工作，理解他们的需求和目标，把它们翻译成角色定义、用例和性能指标等等，开发团队依次来构建应用软件。需求分析员是代表用户体验和产品管理的团队，这意味着他们应该频繁地了解用户和产品雇主对产品的兴趣。

2)项目经理 (Project Manager)

项目经理的主要任务是在达成的时间表和预算之内交付商业产品。项目经理负责在开发项目和迭代过程中计划和安排任务、追踪和汇报状态、鉴别和降低风险。项目经理还应该在每个迭代过程中与需求分析员策划用例和性能指标，与架构师和开发人员一起估算任务，与测试人员一起拟定测试计划，协调项目组内的交流。

3)架构师(Architect)

架构师的主要任务是从设计上保证项目的成功和应用程序的基础。这包括定义应用程序的组织架构和部署时的物理结构。要达到这样的效果架构师需要通过把系统划分成多个简洁的模块来降低复杂性。最终的组织架构是非常重要的，因为它不仅规划了系统开发的方向，而且决定了应用程序的很多性能是否能够达到一个成功的项目的基本要求。这些包括它的可用性、可靠性、可维护性是否达到性能和安全标准，是否易于升级来面对需求的改变。

4)开发人员(Developer)

开发人员的主要任务是在预先计划的时间表内实现应用程序。开发人员同样被期望帮助细化功能模块的设计、估算时间和完成每个功能模块、准备产品的部署，还有给团队提供技术方面的咨询意见。

5)测试人员(Tester)

测试人员的主要任务是发现和追踪可能对产品产生负面影响的问题。测试人员必须理解产品的背景并且帮助其他成员理解基于产品背景的决策。测试人员的最重要的目标之一就是在测试过程中发现和报告对产品有重要影响的Bug。一旦找到一个Bug，测试人员有责任准确地描述它对产品的影响和应该采用何种解决方案来消除或减小它对产品的影响。测试人员应该准确地描述Bug和重现Bug的步骤以易于理解和跟踪。测试人员参与整个项目组关于产品质量的标准定义。测试的目的是证明已知的功能模块能够正常工作并且发现产品存在的其他问题。

6)发布经理 (Release Manager)

发布经理的主要任务是管理产品的发布。发布经理与部署人员一起协调产品的发布。他们创建一个发布计划 (Release Plan)，决定发布的候选版本，管理发布和部署工作。一个开发团队在一起代表了与产品的开发、应用和维护相关的整个集合。每个团队成员，或者每个角色，有责任为所代表的小组表述详细而准确的需求，这无疑比任何其他方面都重要。这些集合在一起的观点、意见提供必要的相互制衡和取舍，以此来保证软件开发团队能够制定出正确的解决方案。

3.1.4 SCRUM方法的特点

SCRUM方法具有以下的特点[20]，使得SCRUM在敏捷软件开发方法中更具有代表意义。

1)Backlog

Backlog是一种任务列表，包括Product Backlog和Sprint Backlog两种，是指导SCRUM开发方向的指针。Product Backlog是整个产品或软件项目的关键功能列表，它由Product Owner负责。当一个项目处在初始阶段，产品经理/项目经理不可能比较全面地去预测所有的任务并准确地估计完成这个任务所需要的时间，这时候Product Backlog就规划了一个纲要性的功能列表，它完全根据商业价值确定所有的产品功能和开发任务，同时它也已经详细地涵盖了第一个Sprint需要的所有细节。在之后的开发过程中，Product Backlog

随着项目的发展不停地根据客户需求和产品自身来制定更详细的计划，同时也随时调整列表中的优先次序。Sprint Backlog 是一个 SCRUM 团队计划将要在当前 Sprint 中完成的所有功能列表。Sprint Backlog 实际上是 Product Backlog 的一个子集，在 Product Backlog 的纲要性指导下，Sprint Backlog 不断发展并且充实整个项目的 Product Backlog 使之趋于完善。SCRUM Team 维护和更新 Sprint Backlog，计算每天完成的和剩余的工作量。对于一个团队来说，选择一个 Sprint Backlog 的内容大小标准是非常严格的。因为在当前的 Sprint 中，只有严格按照当前 Sprint Backlog 才会使得整个项目按计划有序地进行。Product Owner 可以根据需求的改变决定从 Sprint Backlog 中删除某些工作，以保证当前 Sprint 对项目整体是有意义的。

2)SCRUM Meeting

SCRUM Meeting 是 SCRUM 中项目管理的有效手段，也分为两种:Sprint Meeting 和 Daily Meeting。Sprint Meeting 是在下一个 Sprint 开始之前，即在当前 Sprint 即将结束之时举行的，Sprint Meeting 讨论并决定下一个 Sprint 的 Sprint Backlog，会议举行的时间周期随 Sprint 的周期而定。Daily meeting 也称为 Daily SCRUM,顾名思义就是 SCRUM 期间每天举行的例会,Daily Meeting 一般比较简短和随意,每个团队成员都要向整个团队汇报自己的情况，内容主要包括三个方面:第一是昨天的工作进展，当前任务完成的百分比;第二是遇到的问题和困难，需要其他团队成员提供什么样的帮助;第三是今天的工作计划。和传统软件开发方法中依靠每周数小时的例会来监督项目的进度不同，通过 SCRUM Meeting 来管理项目，这种方法更加简单和直观，更加人性化，容易及时发现和纠正问题。从而有利于在宏观上控制项目，保证项目朝健康、成功的方向发展。

3.1.5 SCRUM 方法的简要分析

许多开发过程是不受控制的，输入输出都是未知的或仅仅只是粗略定义的，过程转换缺少必要的精确性，并且质量控制也是未被定义。用于确保包含足够内容的逻辑模型过渡到成功的物理模型的子过程就是这样一种过程。尽管开发过程是未完全定义的动态过程，众多的机构也已经制定出详细的开发方法，包括一些流行的开发方法(结构化的方法，面向对象的方法，等等)。瀑布式方法是其中第一个这样被定义的系统开发过程。虽然瀑布式方法管理了未定义的过程，但是它的线性特点成为它的最大问题。这种过程没有定义如何响应任何中间过程的不可预知的输出。而 SCRUM 则避免了这个问题。它将瀑布模型

缩小，尽量减少过长的线性过程带来的风险。取而代之的是在一个相对短的时间内充分发挥瀑布模型的简单线性优势，对某一小部分工作项进行规范的工作流程，然后及时的反馈结果(输出)，从而在下一阶段应对相应的变更。在引入了螺旋型方法后，瀑布式过程的每个阶段都用一个风险评估和原型活动来结束。螺旋型方法就像剥洋葱一样，使开发过程中阶梯式前进。原型让用户决定项目是否继续进行下去，或者是需要送回到前一个阶段，甚至是应该结束。然而，阶段和阶段过程仍然是线性的。需求分析仍然在需求分析阶段处理，设计工作仍然在设计阶段进行，如此类推，每个阶段仍然包含线性的、定义明确的过程。而在 SCRUM 中，需求、设计、开发和测试不是在所谓各自的阶段中进行的，他们包含在每一个 Sprint 中，而且是在不断完善和丰富的，这样也就避免了螺旋形方法看似非线性但仍属线性过程的致命伤。迭代方法是在螺旋型方法之上发展而来的。每个迭代过程包含所有的标准瀑布式的阶段，但每个迭代过程只处理解析过的一个功能子集。整个可交付的项目被细分为优先级不同的子系统，每个子系统都有清楚的接口。使用这种方法，可以在初始迭代过程中测试一个子系统，而进一步的迭代能给项目添加新的资源，同时保证交付的速度。

3.2 SCRUM 测试模型中各阶段的分析

3.2.1 体系结构设计阶段的测试分析

“软件=程序+文档!”[21]，如图 3-3 所示，对于软件测试来说，测试的对象不应当只面向程序，文档同样是重要的测试对象。在 SCRUM 体系结构设计阶段，Product Owner、需求分析人员与项目经理需要定义出项目对应的 Product Backlog 列表、项目概要设计以及其他项目相关的初期文档。因此在这一阶段，软件测试的一项重要任务就是基于文档的测试，换句话而言就是找出文档中存在的 Bug。

基于文档的测试主要是对文档内容的测试。本文认为软件测试人员应当参与到需求分析人员/项目经理与客户确定需求的过程中，或者通过良好及时的与需求分析人员/项目经理沟通，详细了解软件需求内容。如此以来，当软件相关文档定义出来以后，测试人员就可以通过阅读文档来发现文档中出现的问题(定义模糊、定义错误、图文存在差异等)，从而避免后期开发过程中因为使用有误的软件文档而引起的风险

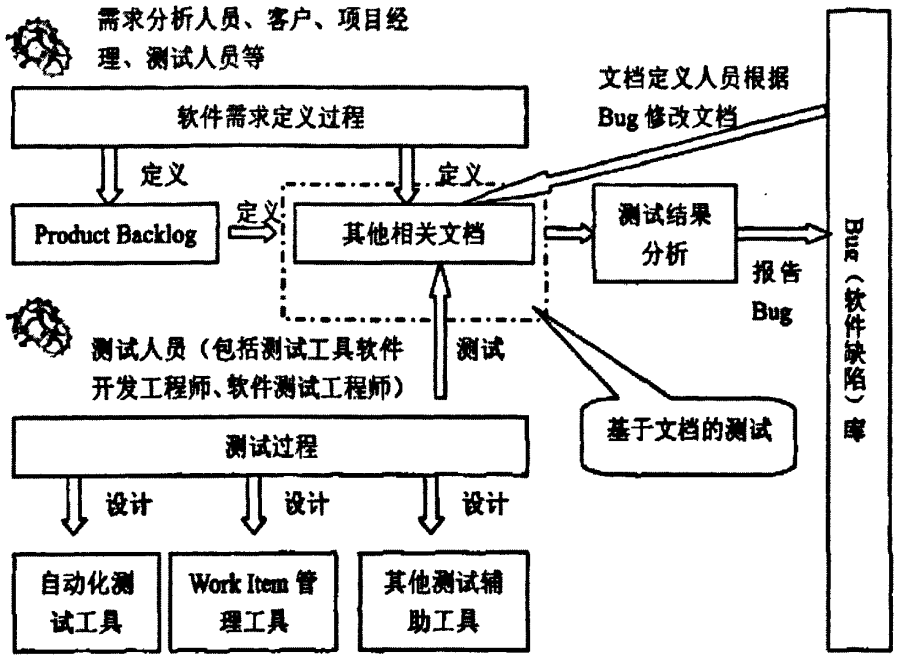


图3-3 体系结构设计阶段的测试过程

当然，在这一阶段，软件测试人员还有一项重要的工作。那就是根据软件需求设计定义软件测试工具，其中包括Work Item(本文将Test case以及Bug均定义为一种工作项—Work Item)管理工具、软件自动化测试引擎/框架等。相对于小公司或者小规模软件开发来说，建议使用第三方软件。如Work Item管理工具，可以使用微软公司的Product Studio或Visual Studio 2005 Team System等软件;软件自动化测试工具可以使用IBM公司的IBM Rational，或Mercury Interactive公司的Load Runner等自动化测试产品。对于大公司以及大规模软件开发来说，建议投入适当的人力、物力开发能够适合公司工作流程定义或具体软件项目的测试工具。

3.2.2 Sprint阶段的测试分析

Sprint阶段是SCRUM方法的核心阶段，也是软件开发测试过程中最关键的阶段。对于每一个Sprint来说，都包括分析、设计、编码、测试等一系列活动。因此，当Sprint计划推出后，开发人员根据详细设计说明书进行软件编码工作，而与此同时测试人员需要根据详细设计定义测试计划与测试用例。同时在Sprint阶段不论是测试人员还是开发人员，都会参与软件测试。

图 3-4 显示了一个目前普遍采用的 Sprint 阶段的测试过程，从图中可以看出，一般的 Sprint 阶段应当包括以下几个方面：

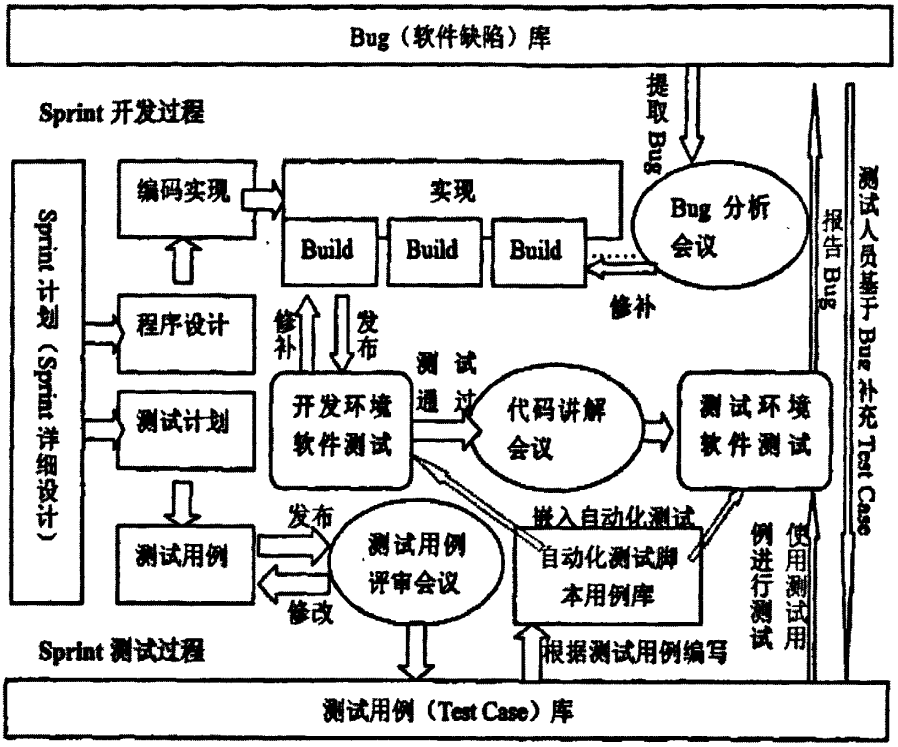


图 3-4 Sprint 阶段的测试过程

1)测试计划(Test Plan)

即测试计划书，其中详细记录整个测试过程所涉及到的各种信息以及过程定义等。

2)测试用例(Test Case)

由测试人员根据 Sprint 计划中所包含的内容并结合测试计划，来定制的为测试项目规定输入、预期结果和一组执行条件的文档。测试用例实际上是测试的核心。其中，测试用例的设计、实现和执行组成了测试的主要内容[23]。有时候根据情况需要，测试人员也会依据 Bug 库中的 Bug 来定制测试测试用例。

3)自动化测试脚本(代码)测试工具软件开发工程师根据测试用例库中的测试用例结合测试计划，并基于自动化测试引擎(工具)编码完成相应的自动化测试脚本。在这里还应当指出，由于测试用例库随着测试人员的维护可能增加/减少测试用例，因此自动化测试脚本用例库也应当由专职测试工具软件开发工程师进行实时维护。

4)测试用例评审会议(Test Case Review Meeting)

测试用例是测试工作的基础，正确、高效、针对性强的测试用例是发现 Bug

的有力工具。因此测试用例的最终确认，在每个测试阶段之前都是至关重要的。一般情况下测试用例确认会议，要有全体测试人员和项目经理参加。它的目的是根据软件需求规格说明书和设计文档，确认测试人员创建的测试用例的正确性。

在这里需要说明的是，测试用例是可以迭代重复使用的，但是每一个测试用例对于不同的迭代过程它的可用性是不同的。因此在测试用例确认会议上，第一个任务就是将不再适合的测试用例关掉，而把合适的测试用例打开。第二个任务是，把某些依然有效的测试用例加以修改，使其更适合新的迭代过程。另外，在测试用例确认会议上，测试人员之间也要进行沟通。因为创建测试用例的人不一定是最后执行测试的人，所以对测试用例的理解是否准确是非常重要的。那么在测试用例确认会议上，大家可以通过沟通知道自己对他人创建的测试用例理解是否准确，以及他人对自己创建的测试用例是否理解准确。总之，测试用例确认会议是测试过程交互中最基础的一环。

5)开发环境软件测试(Development Environment Testing)

当软件开发工程师完成编码后，由软件开发工程师在其开发环境下进行的软件测试。包括单元测试、集成测试、提交测试(软件开发工程师与测试工具软件开发工程师共同参与)等。

6)测试环境软件测试(Test Environment Testing)

这里有两种测试概念：冒烟测试(Smoke Testing)、全部路径测试(Full Path Testing)，其中冒烟测试是指只进行软件主要功能的测试，而全路径覆盖测试则要求测试人员严格按照测试用例库中测试用例对软件进行完整的测试。因此，在测试环境中软件测试包括了软件工程师进行手工测试、测试工具软件开发工程师进行自动化测试(都包括冒烟测试和全路径覆盖测试)。当然不论是手工测试还是自动化测试，此时属于集成测试方法。

7)Bug 分析会议(Triage Meeting)

Bug 分析会议是测试过程中不可或缺的一环，会议应由项目经理、开发人员、测试人员、需求分析人员组成，一般一周举行一次，有时也会根据需要调整频率。Bug 分析会议的主要目标有确定 Bug 的性质、复查 Bug 的严重程度和优先级、分配 Bug 的责任人。确定 Bug 的性质是 Bug 分析会议中对一个 Bug 进行处理的第一步。由于测试人员上报的 Bug 形形色色，不是所有上报的 Bug 都是必须要修复的，甚至有些 Bug 是不正确的。因此测试人员上报的 Bug 只能称之为备选 Bug 或潜在 Bug。所以确定 Bug 的性质便很自然的成为了处理 Bug 的第一步，简单来说就是确定一个 Bug 是不是一个真正的 Bug。当然，在实际情况中，Bug 分析会议上大家做的要比这个多的多。比如，一个 Bug 不是真正的

Bug，设计的初衷就是那样，称其为 By Design(默认设计)。或者一个 Bug 的存在是目前阶段可接受的，不需要马上修复，称其为 Postponed(推迟修复)。又或者一个 Bug 根据目前信息还不能判断其性质，需要进一步的分析，称其为 Investigate。确定了 Bug 的性质以后，紧接着就要复查 Bug 的严重程度和优先级严重的 Bug，优先级高的 Bug，就要优先去修复。所谓严重的 Bug，就是那些会造成系统崩溃或其他恶劣影响的 Bug，当然这些 Bug 通常是优先级比较高的。但是，不是所有高优先级的 Bug 的都是很严重的 Bug。所谓高优先级的 Bug，是指某些出现频率比较高，影响范围比较广的 Bug。确定了 Bug 的性质，复查了 Bu 的严重程度和优先级后，就得分配 Bug 的责任人了。需要修复的 Bug 当然是分配给开发人员，错报的 Bug 和需要进一步的研究的 Bug 一般是分配给测试人员，而推迟修复的 Bug 责会分配给项目经理。Bug 分析会议是一个项目组全体人员参加的会议，它不仅是分析 Bug 的会议，也是项目组成员相互沟通一周以来的工作状况的一个场合。因此 Bug 分析会议在测试交互中占有重要的地位。

8)代码讲解会议

代码讲解会议是开发人员给项目其他成员，尤其是测试人员讲解代码的会议。它目的是帮助测试人员更快更准的理解代码，便于测试人员生成测试用例并展开测试工作。产品的源代码是每个开发人员的主观思想的体现，虽然有注释可以参考，但是要在短时间内准确的理解源代码是很难的，即使可以也是很耗费精力的。因此最直接的办法就是让开发人员给大家讲解他写的代码。首先这样做对于开发人员来说很容易;其次这也是开发人员检查自己代码的一个机会;最后在代码讲解会议上，开发人员也可以借此跟项目组其他成员充分的沟通。

3.2.3 交付和巩固阶段的测试分析

一般的 SCRUM 方法中交付和巩固阶段的测试分为交付阶段的测试与巩固阶段的测试。因为交付阶段与巩固阶段所涉及到的测试方法和测试人员角色都有异同，所以将其分为两个阶段来阐述，有利于细化测试过程中涉及到的具体问题，从而解决问题以便提出合理、可行、有效、具体的测试解决方案交付阶段的测试(如图 3-5 所示)包括:

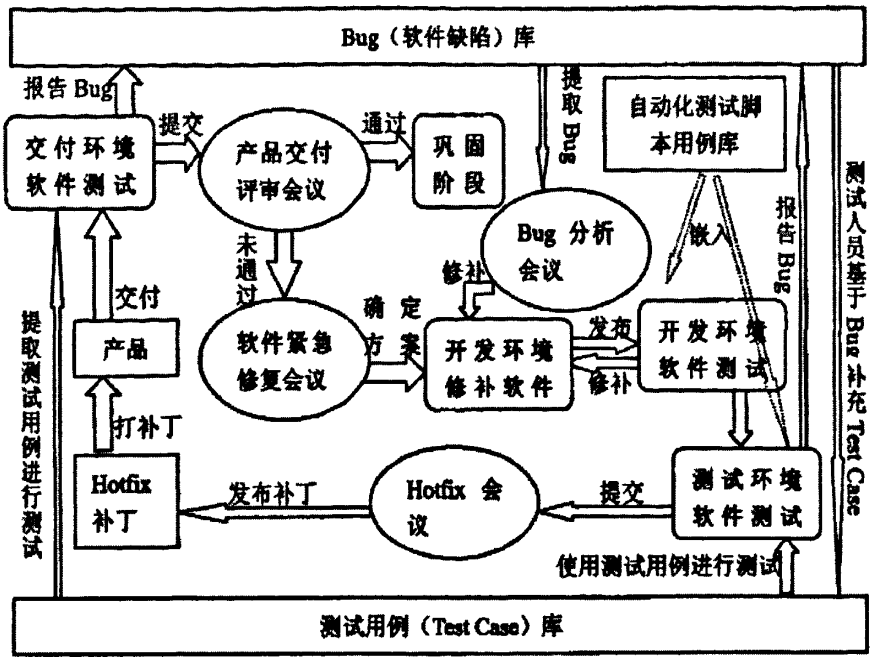


图 3-5 交付阶段的测试过程

1)交付环境测试

测试人员、开发人员、项目经理现场演示如何使用软件，并向参与交付测试参与者(一般为用户代表与专家等组成)讲解软件具体功能，然后由参与者根软件使用说明书、结合用户经验等方面对软件进行测试和评估。最终将使用体、Bug 报告等内容形成文档，提交产品交付评审会议。

2)产品交付评审会议

由评审专家(第三方)、用户(甲方)、软件开发方(乙方)组成，并根据交环境软件测试所形成的文档以及软件其他相关文档，对软件进行评议。如果评审通过，则软件产品交付给用户使用，进入软件巩固升级阶段。如果评议未通过，需要软件开发方对软件进行修复。

3)软件紧急修复会议

软件产品为通过产品交付评审后，由项目经理、需求分析人员、开发组长以及测试组长召开软件紧急修复会议，讨论如何快速、有效、同时避免引入更多风的修复 Bug，以满足客户的需求。

4)开发环境修补软件

软件开发工程师根据 Bug 分析会议结果(需要修复的 Bug 列表)，对软件进行修复。

5)Hotfix 会议

当产品通过测试环境软件测试后，提交 Hotfix(热补丁，通常指比较重要

的关系到软件产品是否成功的补丁)会议，讨论是否能够推出 Hotfix 补丁从而修复 Bug，达到满足客户最终需求。

6)Hotfix 补丁

从软件开发方的角度可以理解为，当打上 Hotfix 补丁的软件产品可以通过软件评审会议。

巩固阶段的测试(如图 3-6 所示)包括:

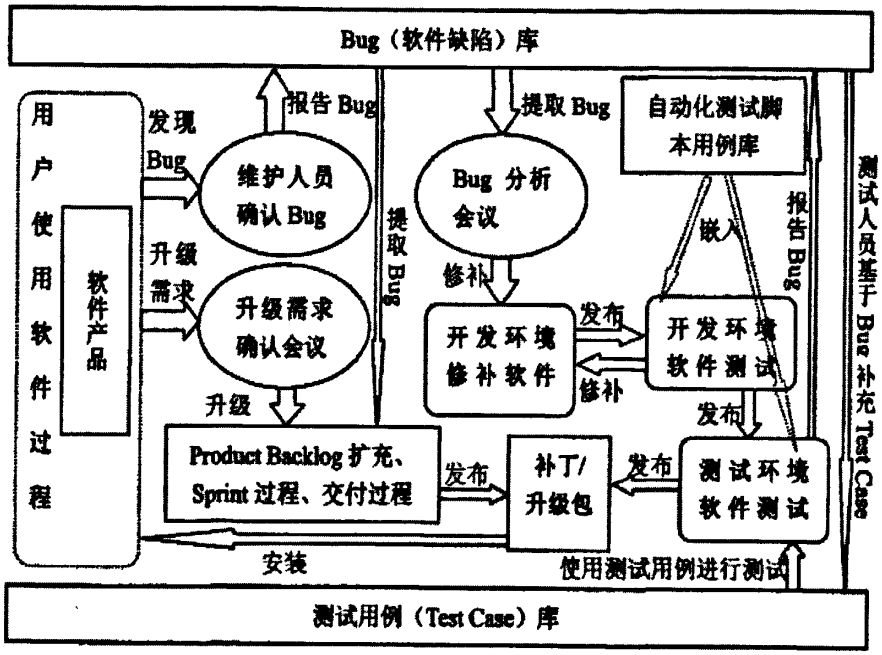


图 3-6 巩固阶段的测试过程

1)用户使用软件过程

在软件产品交付后，用户日常使用软件的过程。一般在这个过程中，用户都会发现软件存在的各种 Bug 和对软件提出新的使用需求。

2)维护人员确认 Bug

当用户发现 Bug 后，向软件开发方提交 Bug，然后由软件维护人员(从对软件整体结构和软件测试工作了解方面考虑，论文建议由软件测试工程师兼任这一角色)对用户提交 Bug 的进行确认。如果确认为 Bug，则记录入 Bug 库，同时对较为严重的一些 Bug(一般为用户要求修改)推出修改补丁。

3)升级需求会议

确认用户提出的新的需求是否可行、合理等方面问题。

4)Product Backlog 扩充、Sprint 过程、交付过程

当确定需要进行软件升级以后，Product Owner 将新的需求与目前存在的一 Bug 统计分析整理后，添加记录在 Product Backlog 中。接着对软件产品进

行加工直到产品(升级包)交付(可以看为重新经过 Sprint 阶段与交付阶段)。

第四章 WCS 项目 SCRUM 测试方案的设计与实现

4.1 项目介绍

IBM WebSphere Commerce 是 IBM WebSphere 软件品牌下专业致力于电子商务解决方案的软件产品。在经历了数十年的不断发展，以及在全球数以千记的客户成功应用之后，已经形成以商品管理、销售管理、订单管理、用户管理、营销管理等为核心模块，以电子商务行业最佳实践为支撑的一整套解决方案。

从基础架构上来看，WebSphere Commerce 是构建在 WebSphere 应用服务器平台的极其庞大而复杂的 J2EE 应用。因其模块繁多且复杂，所依赖的软件产品众多，对不同客户所要进行的定制工作量巨大，因此 WebSphere Commerce 项目的开发难度超过了一般 IT 软件项目实施。

WCS 项目组遍布世界各地，各种团队的成员一共有数百人。作者所工作的功能测试组（FVT）只是 WCS 产品测试大组中的一个小组，在测试方面，除了 FVT 外，还有 BVT、IVT、MVT、SVT、GVT 等多个测试组，分别测试着 WCS 产品的各个方面。

4.2 WCS 产品总体框架结构介绍

图 4-1 显示了 WCS 产品的总体框架结构：

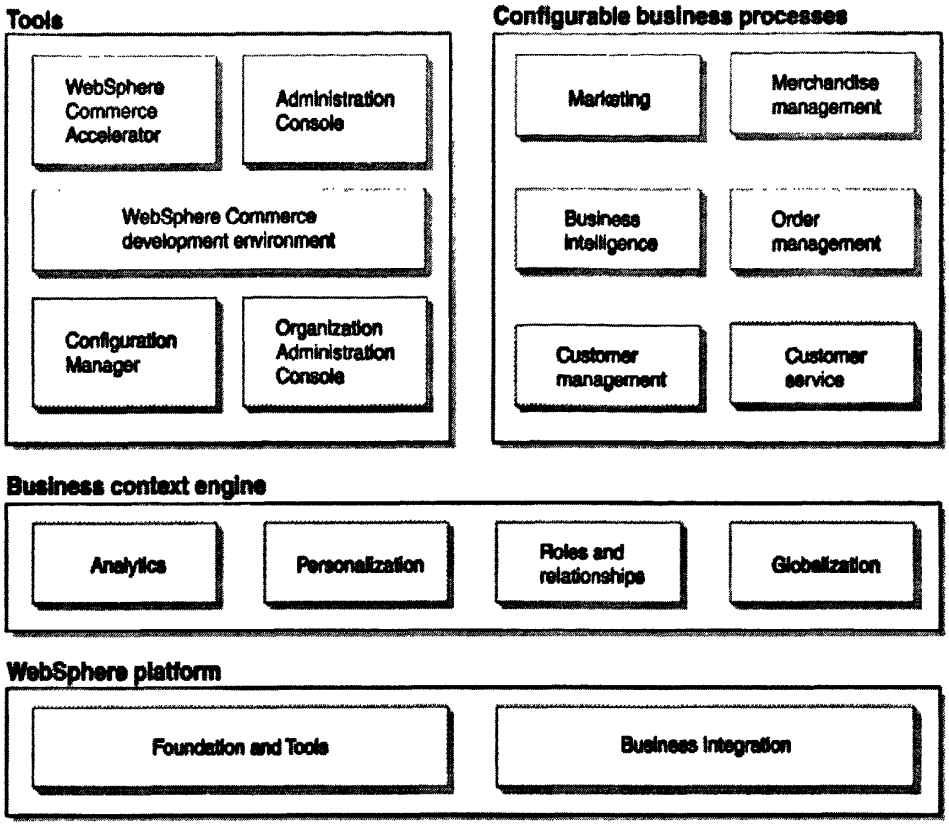


图 4-1 WCS 产品总体构架

其中：

1）Tools 模块：为客户提供了相应的管理工具和开发环境，包括客户对自己产品的管理，对电子商务流的设置，对用户和组织的管理，对系统的设置等多个工具

2）Configurable business process 模块：主要为客户提供了进行电子商务活动的相关属性的配置功能。用户可以通过这个模块设置自己的商店模式，广告位置和内容，下订单和付款模式的设置等。

3）Business context engine 模块：这里主要是用来存储各种逻辑关系的模块

4）WebSphere platform 模块：这个是底层的功能，产品的核心逻辑处理都在这个模块中实现。

4.3 WCS 项目开发过程的演变

WCS 产品经过了几十年的发展，已经经历了很多的版本。在这些版本的变更升级中，其所采用的开发模式也随着计算机软件开发技术的发展而发生着变

化。我们从最近的 10 年中 WCS 的发展来看 WCS 项目的开发模式的变化。

在 21 世纪初，WCS 的 5.6.1(Tierra)和 5.7(Terrance)版本，WCS 所采用的开发模式是老的基于瀑布的开发模式。这种模式具有框架化和模式化的开发模型，如下图：

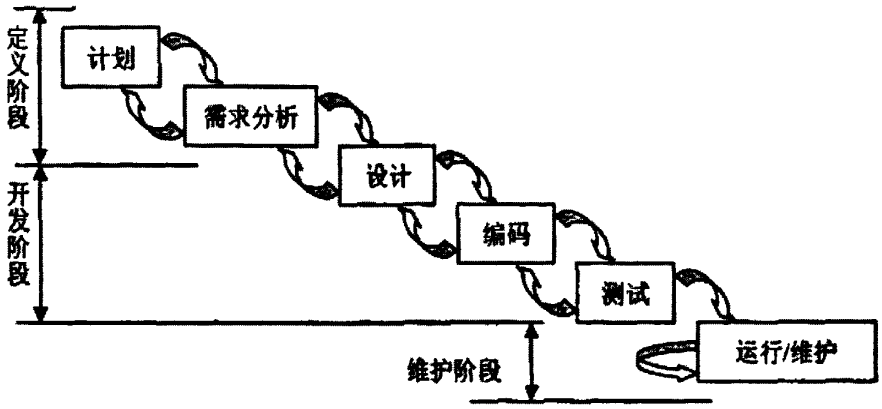


图 4-2 瀑布模型

基于瀑布模型的开发模式，WCS 的各个团队按照上图所示的瀑布开发过程进行着 WCS 的开发：

1）首先是进行产品的计划，包括产品版本，功能，市场的功能概要的计划。

2）其次根据客户的需求分析来制定开发计划。

3）随后将进入到设计开发阶段，依次进行设计-〉编码的功能的开发。

4）在开发工作完成后，测试团队才开始进行相应的测试工作。

在近几年，随着电子商务的发展，客户的需求越来越多，越来越复杂，为了更有效，方便和及时的适应客户的需求，WCS 从 6.0 版本（Darwyn）开始，逐渐地将一些新开发的功能组件（Components）构建于基于 SOA 的架构上。并且从 WCS 的 6.0.0.4 版本（Trident）开始，WCS 项目组对新增加的组件的开发开始采用 SCRUM 开发模式。相对于以往的产品版本的测试，在最新的版本中的测试工作发生了很大的变化，测试工作不再相对的滞后和封闭，而是从产品的设计阶段就开始进入，并与产品的其他团队进行着密集的交流。每次的测试计划的设计，编写，以及最后的执行都是在多次的迭代中进行着。这些特点都体现出了 SCRUM 模式的特性。

图 4-3 显示了新的开发模式结构：

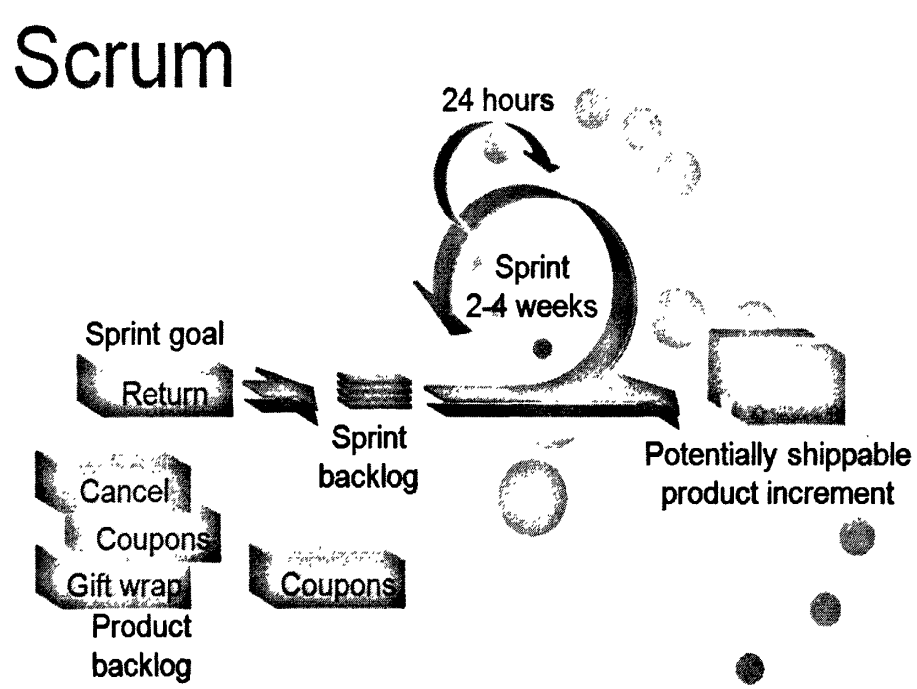


图 4-3 WCS 采用的 SCRUM 开发模式图示

4.4 WCS 项目的 SCRUM 测试方案设计

4.4.1 概述基于 SCRUM 的测试方案

根据 SCRUM 的软件开发过程原理，以及通过 WCS 产品的开发测试经验，我们可以了解到，基于 SCRUM 的软件开发过程中，产品的主要设计，生产和测试工作都将在迭代递增的 Sprint 过程中产生，但是在进入迭代递增的 Sprint 过程前，以及退出迭代递增的 Sprint 过程后，也都应该有相应的过程来控制和管理 Sprint 的进入和退出，同时产品开发组和产品测试组应该是有一个统一的开发模型来控制各组的工作，所以我们将基于 SCRUM 的开发/测试模型设计为：如图 4-4

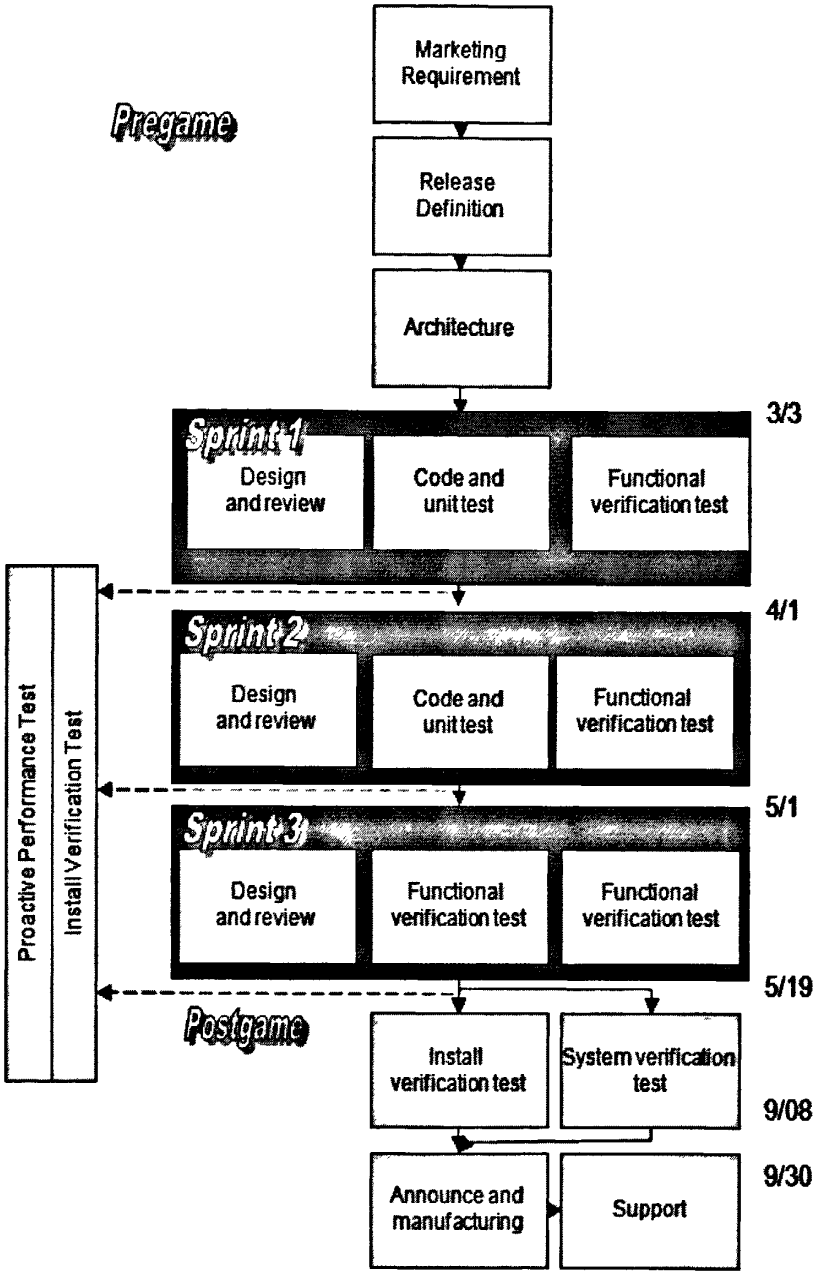


图 4-4 WCS 项目的基于 SCRUM 的测试模型图

*注：图中右侧的日期是一个示例，显示了一个 SCRUM 测试过程大概需要的时间跨度，当然这个只是一个例子，具体的项目进行时，会根据实际的情况发生着改变。

在上图中可以看到，在 WCS 产品开发过程中，所采用的基于 SCRUM 的测试方案，主要可以分为 3 个阶段：

1）前期准备（Pregame），即体系结构设计阶段：

在体系结构设计阶段中，测试人员唯一的测试对象是这一阶段定义产生的各类文档，因此这一阶段主要是文档测试。

2）迭代递增过程/冲刺过程（Sprints）阶段

结合 Sprint 阶段特点，WCS 的测试项目的执行中，将 FVT, SVT 测试、冒烟测试、安全性测试、自动化测试、代码检查、Debug 测试引入测试方案。

3）总结退出（Postgame），即交付和巩固阶段

交付和巩固阶段的测试主要包括全球化与本地化测试、文档测试、维护过程测试等。

因此，WCS 产品所采用的 SCRUM 测试方案结构，包括各个阶段应当采取的测试方法如下，如图 4-5 所示：

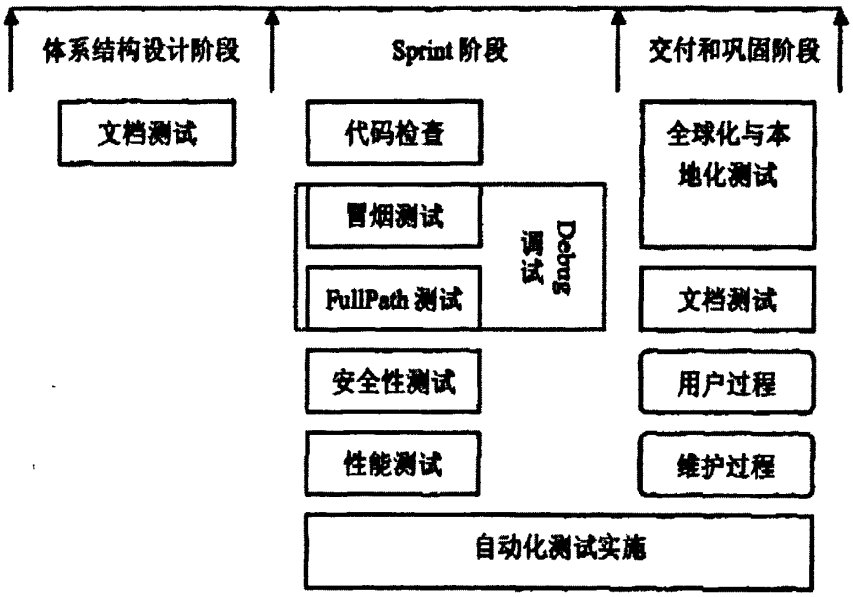


图 4-5 WCS 的 SCRUM 测试中测试方法说明图

4.4.2 WCS 的 SCRUM 测试方案的特点

综上所述，可以得出 WCS 的 SCRUM 测试方案有如下特点：

1)测试介入整个软件过程

传统上认为，只有软件的质量控制依赖于测试，但是现代软件开发的实践证明，不仅软件的质量控制依赖于测试，开发本身离开测试也将无法推进，项目管理离开了测试也从根本上失去了依据。同时根据 SCRUM 中测试环节特点，测试遍布整个软件过程，对 SCRUM 方法来说，测试不再是存在于开发之后的一个特定阶段。因此，WCS 的 SCRUM 测试方案从一开始就将危机扼杀在摇篮里，从而最大程度上降低软件开发的风险。

2)时间限制要求高效测试

根据 SCRUM 的过程特点，可以知道每一个 Sprint 的周期较短，所以测试效率就显得尤为重要。要提高测试效率，当然要借助于自动化测试手段。

3)测试过程必须可以重现整个 SCRUM 软件过程的迭代的特点，因此测试用例必须可重用，不然将无法适应 SCRUM 软件过程。比如在前一个 Sprint 中对某一个模块进行单独测试，而下一个 Sprint 中要对该模块和其它几个模块进行集中测试，这时这个模块的测试用例就会用到多次，重用就显得尤其重要。

4)测试对象类型比较丰富

由于测试从一开始便介入项目，那么最开始测试的是各类文档(如需求规格说明书于 Product Backlog)，再之后才是程序本身，因此测试对象比较丰富。

5)参与测试的角色较多

根据前面章节的分析，参与到整个测试过程中的人员，不仅有专职的软件测试工程师(包括测试工具软件开发工程师)、软件开发工程师和用户，还包括了软件团队中其他所有的角色等等。这样就扩大了传统对软件测试人员定义，从而使得软件测试贯穿于整个软件生命周期成为可能。

4.5 WCS 的 SCRUM 测试方案中的过程管理

4.5.1 测试计划

详细、可行的计划是成功实施软件测试的坚实依据，测试计划的文档形式就是测试计划任务书，整个软件测试过程的管理都应当是依据测试计划任务书来执行的。测试计划任务书应当依据产品功能性文档(详细设计)以及开发人员的建议，定义了如何能够全面的、可靠的对软件产品进行测试。结合 SCRUM 测试特点，WCS 项目测试组认为基于 SCRUM 测试的测试计划应当基于每一个 Sprint 阶段功能性文档、以及每个 Sprint 阶段对软件的修改来确定。也就是说，SCRUM 测试中的测试计划任务书不是一个，而是由不断迭代的一组任务书组成。因此测试计划任务书应当包含以下内容:

1)产品基本信息

产品基本信息应当包含三个部分。第一部分，简单介绍整个产品进行目前的 Sprint 阶段的信息;第二部分，记录前面 Sprint 各阶段的测试情况;第三部分，整个软件开发测试中所涉及到的角色的联系人列表。

2)测试范围

定义当前 Sprint 阶段所新开发的功能(项目经理编写的每个 Sprint 功能

性文档中定义的功能)，以及与新功能关系密切的原有功能。由于 Sprint 周期短，所 38 以需要快速有效的实施测试，测试范围就定义了需要测试人员重点测试的软件功能，而没有被写入测试范围的产品功能只需要测试人员进行轻量级测试。

3)开发过程中的测试要求

为了测试能够很好的实施，论文认为在产品开发阶段应当提出以下要求:开发人员在 Sprint 阶段推出的每一个 Build 都应当是一个或几个功能完成时推出的，不应当在某功能没有完成的时候推出 Build;当每一 Build 都应当包含产品的安装文档以及源代码路径;在每个 Build 被推出前，开发人员需要先对产品进行单元测试和集成测试;每个 Build 都必须通过基本验证性测试(BVT 测试)后，才能够进入测试环境测试;开发人员必须向测试人员提供正确的 DLL 文件版本编号;同时，论文认为每一个 Sprint 阶段最后推出的 Build，必须在测试计划完结日前 5 天推出，也就是说论文认为测试团队应当能够在 5 天内完成一个 Build 的测试。

4)项目管理过程中的测试要求

从整体项目管理的角度来看，对测试有如下要求:测试应当面向目前所有的软件功能，以及当前 Sprint 阶段解决的前期 Bug;项目经理需要在测试完结日前一周，确保整理和协调“沙盒”部署的完成，“沙盒”指一组集成测试环境，其中包括性能、功能、压力、单元、集成等测试环境;要确保每个 Sprint 阶段的 Bug 都被 triage(诊断)过，并分配给适当的开发人员。

5)测试过程的的要求

WCS 项目认为 SCRUM 测试计划书对具体测试过程的要求如下:在每一个 Sprint 阶段完结的时候，测试团队应当提供产品质量保证;测试团队需要编写测试计划任务书;编写 Test Case 并确保能够覆盖所有的软件功能;提交 Bug，并将 Bug 记录在项目指定的 Bug 管理记录工具中;原则上只有 Bug 的提交者(测试人员)有权更改 Bug 的优先级和严重级，但是如果 Bug 分析会议中，开发人员对 Bug 优先级和严重级提出严重质疑时(开发人员与测试人员争执不下时)，则由项目经理作为仲裁人来决定 Bug 的优先级和严重级;只有测试人员有权关闭/激活 Bug;开发人员推出修好 Bug 的新 Build 时，需要将存储在 Bug 管理工具内的 Bug 记录中的 Bug 状态，由 Active(活动)状态更改为 Resolved(已解决)状态，然后提交 Bug 所有者(提交 Bug 的测试人员)，由于 Sprint 阶段周期短，要求 Bug 处于 Resolved 状态的时间不能超过 48 小时，也就是说测试人员必须在 48 小时内验证此 Bug 会成员由测试人员、是否解决并给出回馈;测试人员需要组织 Bug 分析会议，与析会议至少每周举行开发人员、项目经理组成，在

SCRUM 测试方案中，Bug 分 2 次(星期 2、4 各一次);Bug 分析会议不仅应当包括诊断 Bug 过程，同时测试人员需要详细说明每个会议中讨论的 Bug，使开发人员对每个 Bug 有全面清晰的认识;测试中如果发现未定义在功能文档中的功能时，需要提交一个 Bug。

6)整个测试过程中使用的测试方法

列出测试具体方法、以及预期结果。

7)测试环境

软运行环境的复杂程度对软件评估具有重要作用，所以应产生尽量逼真的运行环境以便于研究。

因此定义如何部署测试环境，应当包含以下内容:整体测试环境的硬件设备以及软件部署结构层次图表;测试服务器硬件配置信息列表;各服务器上安装的软件信息列表。

8)测试发布的标准

软件在任何时候都是存在 Bug 的，也就是说没有任何方法能够确认软件中不存在 Bug，只有证明软件中存在 Bug 方法。因此，就需要定义软件测试进行到何种程度后，软件就可以进去交付巩固阶段。所以，SCRUM 测试方案中定义测试发布标准包括:Bug 库中没有激活状态的 Bug;所有 Severity(严重级)1 的 Bug 必须被解决并关闭;所有 Severity1 且 Priority(优先级)1 的 Bug 必须被解决并关闭;每个 Sprint 阶段中的主要 Build 必须实施 Full Path 测试。

9)测试过程中的风险

测试任务书的制订者，根据与项目经理沟通后定义测试存在的风险列表，如表 4-1 所示:

表 4-1 测试存在的风险列表（示例）

风险	等级	解决方案
整个项目只有一名测试人员	高	扩充测试人员
测试环境硬件配置不高	低	升级硬件

10)测试时间安排

定义测试各阶段实施开始时间、周期。如需要定义何时开始制定测试计划，编写测试计划人数书;何时进行冒烟测试;什么时候要实施 FullPath 测试等。

11)测试人员

测试人员信息表(如表 4-2 所示)。

表 4-2 测试人员信息表（示例）

测试人员姓名/角色	具体责任
Tester A/测试组长	定义维护整个软件测试过程
Tester B/软件测试工程师	组件 1 的功能性测试
Tester C/测试工具软件开发工程师	性能测试、压力测试、自动化测试

4.5.2 WCS 基于 SCRUM 的各测试方法介绍

下面对 WCS 的 SCRUM 测试方案中的各测试方法作一个简要的介绍:

1.文档测试

文档测试的主要工作是，测试人员通过与需求分析人员、项目经理、开发人员的沟通，在对整个软件项目有相当了解后，结合自身经验对项目相关的主要文档进行测试，测试通过测试人员仔细阅读文档来发现文档中 Bug。文档中的 Bug 类型主要是:定义错误、不符合文档规范、错别字等。

2.代码检查

所谓的代码检查，其实就是以组为单位阅读代码，是一系列规程和错误检查技术的集合。该过程通常将注意力集中在发现错误上，而不是纠正错误。一个代码检查小组通常是由项目经理、开发人员、测试人员、技术专家 4 方组成。对于 Sprint 阶段的代码检查而言，主要工作是通过对代码的检查发现以下问题:

1 代码规范

2)代码错误

结合 Sprint 阶段特点，代码检查有以下几个方面的特点:

i)周期短

一方面 Sprint 阶段的开发测试周期是限定的(WCS 项目中一般设定为 4 周时间)，因此作为整体测试的一部分，代码检查周期也比较短。另一方面，每个 Sprint 阶段生产的新代码比较少，代码检查的工作量相对较小。

ii)交互性强

由于周期较短，因此如何能在短时间内高效率的发现 Bug，就需要检查小组具有良好的交互，也就是说小组成员之间的沟通交流要求比较强。

iii)采用会议形式

当 SCRUM 方法进入 Sprint 阶段以后，整个 WCS 测试团队(包括所有成员)除了当前 Sprint 阶段的工作外，还需着手下一个 Sprint 阶段的准备工作。因

此，一般会采用会议形式(消耗时间短、沟通效果好)进行代码检查。

3.冒烟测试

在开发人员开发完新功能后,WCS 项目中的测试人员会马上进行冒烟测试。在检查了代码后，冒烟测试是确定和修复 Bug 的最经济有效的方法。冒烟测试设计用于确认代码中的更改是否会按预期运行，且不会破坏整个版本的稳定性，同时冒烟测试只是对软件主要功能、较大改动的部分进行测试，而不会去测试软件中优先级最低的功能和未改动部分。

通过 WCS 项目中的实际应用，作者认为冒烟测试的最佳做法是：

1)与开发人员协同工作

Sprint 阶段是由一个又一个 Sprint 模块迭代组成的，因此每一个 Sprint 模块都是在前一个 Sprint 模块的基础上进行开发的，这样有时就不可避免的需要修改前期模块的代码，以适应新模块。而冒烟测试的一个重要特点是，特别关注更改过的代码，因此就需要测试人员与编写代码的开发人员协同工作。必须了解以下内容:

.代码中进行了什么更改。若要理解该更改，必须理解使用的技术;

开发人员可以提供相关说明。

.更改对功能有何影响。

.更改对各组件的依存关系有何影响。

2)在进行冒烟测试前检查代码

在运行冒烟测试前，进行侧重于代码中的所有更改的代码检查。代码检查是验证代码质量并确保代码无 Bug 和错误的最有效、最经济的方法。冒烟测试确保通过代码检查或风险评估标识的主要的关键区域或薄弱区域已通过验证，因为如果失败，测试就无法继续。

3)创建每日构建(Daily Build)

基于“开发一点点,测试一点点”的原则,WCS 项目提出 SCRUM 测试在 Sprint 阶段应当每日构建一个 Build。每日构建要求团队成员协同工作，并鼓励开发人员彼此保持同步。如果新版本的迭代被延迟，则该延迟很容易导致具有多个依赖项的产品不同步。遵循每日构建和冒烟测试的过程，任何更改过的或新的二进制文件都可确保实现高质量。不需要执行穷举测试。冒烟测试的目的不是确保二进制文件绝对没有错误。这样需要花费太多的时间。执行冒烟测试是为了在高级别验证版本。要确保二进制文件中的更改不会破坏常规版本的稳定性，也不会导致功能中出现严重错误。

4.Full Path 测试

除了在 Sprint 阶段引入冒烟测试和每日构建外，WCS 项目进行中还提出

在每一个 Sprint 过程中，应当阶段性的整合整个软件进行 Full Path 测试。基于 SCRUM 方的特性，可以知道每一个 Sprint 模块与前面的 Sprint 模块组装后能够形成一个成运行的软件(即实现了部分功能却能够独立运行)。

一般来说 Full Path 测试就是依据目前所有的 Test Case(包括软件的所有功能义等)，对软件进行一次全面测试。这个时候，测试人员的工作量加大，尤其入软件开发的后期，Test Case 数量非常巨大，因此引入自动化测试就成为必须，就是说采用自动化测试的方法进行 Full Path 测试。

因此，Sprint 阶段的测试应当是冒烟测试与 Full Path 测试相结合的测试。二的机结合，不仅仅能够快速找出 Bug，而且也能保证测试覆盖面全。

5. Debug 调试

由于每一个 Sprint 阶段都是一个冲刺阶段，要求高效的测试。因此，对于 Sprint 阶段的测试而言，软件的源程序(源代码)对于测试人员(测试软件工具开发工程师/软件测试工程师)是可见的。也就是说，此时 SCRUM 的测试方法中主要采用白盒测试，这样不仅开发人员可以对程序进行 Debug 调试，测试人员也可以将软件源代码部署在测试环境进行 Debug 调试。测试人员可以通过 Debug 调试冒烟测试、Full Path 测试中发现的 Bug，直接找到程序内部的错误，然后详细报告 Bug。这样一来，开发人员可以节省出大量时间用于提高代码质量、优化代码的工作。

6. 安全性测试

安全性测试是有关验证应用程序的安全服务和识别潜在安全性缺陷的过程。由于攻击者没有闯入的标准方法，因而也没有实施安全性测试的标准方法。

7. 性能测试

在 SCRUM 测试中，性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。性能测试在软件的质量保证中起着重要的作用，它包括的测试内容丰富多样。性能测试主要可以概括为三个方面:应用在客户端性能的测试、应用在网络上性能的测试和应用在服务器端性能的测试。

8. 全球化和本地化测试

全球化测试[24]的目的是检测应用程序设计中可能阻碍全球化的潜在问题。它确保代码可以处理所有国际支持而不会破坏功能，导致数据丢失或显示问题。全球化测试使用每种可能的国际输入类型，针对任何区域性或区域设置检查产品的功能是否正常。

4.5.3 Bug 规范与管理

软件测试的目的是找出 Bug，并提交给开发人员，然后解决 Bug。因此，测试人员提交的 Bug 报告(记录)，是否能够准确清晰的描述问题，以及是否有一套行之有效的 Bug 追踪机制，直接影响到软件测试是否能够很好的实施。

1)Bug 报告属性定义

WCS 的 SCRUM 测试方案认为，一个完整、详细、便于追踪管理的 Bug 应当包括以下内容，如 41 页表 4-3 所示：

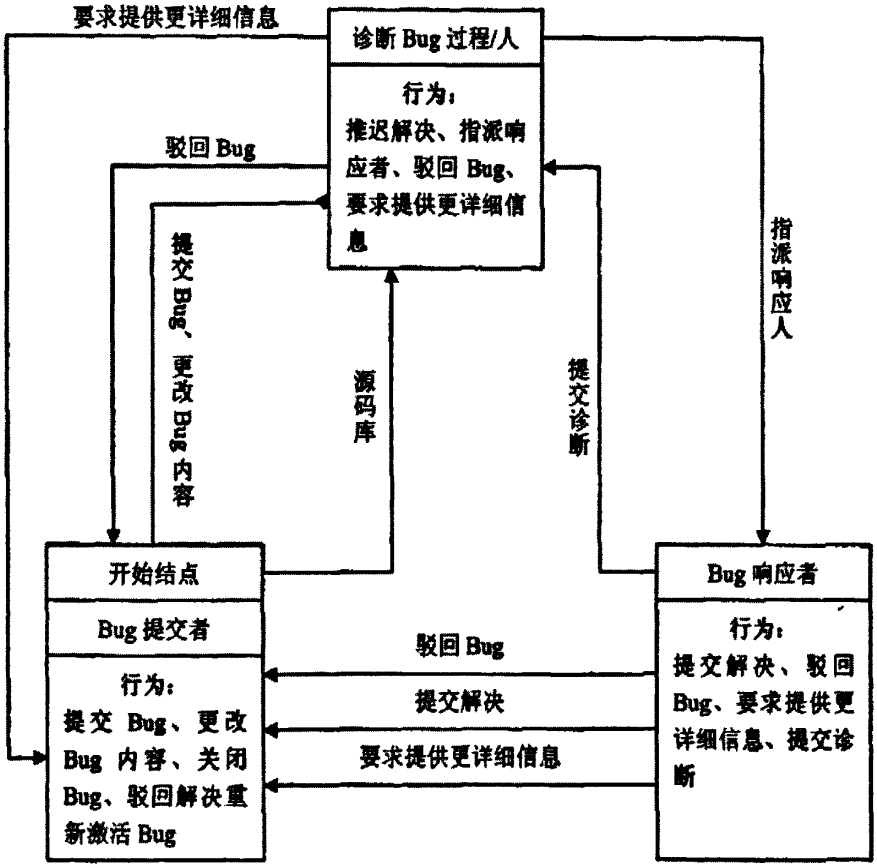


图 4-6 WCS 的 SCRUM 测试方案中 Bug 追踪机制

表 4-3 WCS 的 SCRUM 测试方案中 Bug 的属性列表

属性名	内容
Title（标题）	用最简单的话描述 Bug。
State（状态）	描述 Bug 当前状态，定义每个 Bug 都应该有三种状态：Active 激活、Resolved 解决、Closed 关闭。
Area Path（模块路径）	记录 Bug 所属的模块。
Sprint Path（Sprint 路径）	定义 Bug 所属的 Sprint 阶段。
Issue Type（Bug 类型）	记录 Bug 的问题类型，如 Code（代码问题）、Doc（文档中的 Bug）。
Severity（严重级）	记录 Bug 的严重级别（1 十分严重、2 较严重、3 严重、4 一般）。
Priority（优先级）	记录 Bug 应当被解决的优先顺序(1 马上、2 尽快、3 一般)。
Triage（被诊断）	记录 Bug 是否在 Bug 分析会议上讨论过。
How Found（如何被发现）	记录 Bug 是怎么样被发现的，如 Manual（手工测试）、Automation（自动化测试）、Review（读代码）。
Test Case ID（相关测试用例 ID）	如果 Bug 是依据 Test Case 发现的，那么这里记录此 Test Case 的 ID。
Find In（在那个 Build 中发现的）	记录发现 Bug 的 Build 编号。
Fix Build（Bug 被解决的 Build）	记录解决 Bug 的 Build 编号。
Created By（被谁提交）	记录是谁提交的这个 Bug。
Changed By（被谁改动）	记录是谁最近一个修改过 Bug 报告（记录）内容。
Closed By（被谁关闭）	记录是谁关闭这个 Bug。
Assigned To（分配给谁）	记录这个 Bug 目前需要谁来负责（修复、验证）。
Created Date（创建日期）	记录 Bug 的创建时间。
Resolved Date(解决日期)	记录 Bug 何时被解决的。
Closed Date（关闭日期）	记录 Bug 何时被关闭的。
Repro Step（重现步骤）	记录怎样操作能够使 Bug 重现。
Description（描述）	记录每个人对 Bug 的意见注释。

2) Bug 追踪管理机制

在 WCS 的 SCRUM 测试方案中，还有一套具体的 Bug 追踪制度，也就是 Bug 的生命周期，如 40 页图 4-6 所示。诊断过程在一定程度上就是 Bug 分析会议，但是有时候由于时间地点的限制不可能召开会议，此时诊断就由项目经理一个人来负责。在 SCRUM 测试方案中，Bug 提交者一般指测试人员，Bug 响应者则指的是软件开发人员。

表 4-4 中定义了图 4-6 中各种行为的具体含义：

表 4-4 WCS 的 SCRUM 测试方案的 Bug 追踪机制的各行为定义

行为	定义
提交 Bug	测试人员使用统一的 Bug 报告模板，编写 Bug 报告，并将记录在专门的 Bug 存储、管理工具中，初次提交后 Bug 状态为 Active。
更改 Bug 内容	测试人员使用 Bug 管理工具，对 Bug 内容进行修改，一般是为了提供更详细的 Bug 信息。
关闭 Bug	测试人员通过 Verify（验证）后，确定 Bug 已经被解决，即不在重现时，在 Bug 管理工具中关闭 Bug（将 Bug 状态改为 Closed）。
驳回解决重新激活 Bug	测试人员 Verify 状态为 Resolved 的 Bug 后，发现 Bug 依然存在，则将 Bug 状态由 Resolved 重置为 Active，并提交诊断。
推迟解决	经过 Bug 诊断后，决定 Bug 应当在以后的 Sprint 阶段解决。将 Sprint Path 由当前 Spint 编号改为下一个 Sprint 的编号。
指派响应人	Bug 分析会议/项目经理决定由那个开发人员具体负责此 Bug 的修复工作。
驳回 Bug	Bug 分析会议/项目经理/开发人员认为不是 Bug，将 Bug 重新 Assigned To 提交者。
要求提供更详细信息	诊断过程发现 Bug 信息不够，要求 Bug 提交者补充信息。
提交解决	开发人员修复 Bug 后，将 Bug 状态由 Active 状态置为 Resolved 状态，然后 Assigned To Bug 提交者等待 Bug 提交者 Verify。
提交诊断	开发人员对 Assigned To 自己的 Bug 存在意义，则可将 Bug 重新提交 Bug 分析会议/项目经理重新诊断。

3)Bug 的严重级和优先级

Bug 的严重级和优先级应当如何定义，什么样的 Bug 是 Severity1 或 Priority1 呢?在 SCRUM 测试方案中 WCS 给出以下定义(如表 4-5、表 4-6 所示)：

表 4-5 WCS 的 SCRUM 测试方案中的 Bug 严重级定义

严重级别	定义	分类
Severity 1（严重问题）	这类 Bug 引起系统崩溃、挂起或破坏系统文件、造成大量"脏数据"或数据丢失。	程序错误： 系统崩溃，挂起或破坏系统文件； 造成大量"脏数据"、数据丢失； 造成致命异常等； 文档错误： 文档中存在逻辑错误； 文档中定义的功能之间有显著冲突； 文档携带病毒等。
Severity 2（主要问题）	产品功能缺陷、功能实现与文档定义完全不符、抛出非致命异常。	程序错误： 非致命异常； 主要功能错误； 产品界面上出现为本地化字母； Web 页面上出现坏链接等； 文档错误： 客户反馈的帮助文档错误； 文档章节丢失标题； 文档中出现的主要网页链接有错等。
Severity 3（一般问题）	存在较小风险的问题、用户在使用中发现帮助文档与实际操作不符。	程序错误： 功能与文档定义不符，但问题不大； 文档错误： 印刷错误； 文档中出现的一般网页链接有错等。
Severity 4（可忽略问题）	产品界面上的拼写错误、用户认为界面不美观等。	程序错误： 产品界面拼写错误； 产品界面字体规格错误； 文档错误： 文档样式不够规范等。

表 4-6 WCS 的 SCRUM 测试方案的 Bug 优先级定义

优先级别	分类与定义
Priority 1（必须解决）	该 Bug 易于重现，比如经过一些较简单的步骤就可以重现；该 Bug 也比较严重，阻碍了一些软件的功能，或会阻碍后续的测试/开发，或阻碍软件的进一步运行（如产生系统错误，非友好的错误页面等）。
Priority 2（应当解决）	该 Bug 通常需要一些较复杂的步骤才能重现，或比较容易修复，修复的风险低； 另一种情况，即便修复该 Bug 较重要，但不修复也不会影响软件后续的运行。
Priority 3（有时间解决）	该 Bug 一般间断的出现，或没有重现的清晰思路，修复该 Bug 并不直观；该 Bug 等到所有高优先级的 Bug 都修复了之后再修复；该 Bug 一般都是地严重性的或只是软件人机界面相关的 Bug。
Priority 4（最低优先级）	一般用于表示修饰问题，或该 Bug 很难重现或无法规律性的重现；该 Bug 在开发人员有低风险的解决方案时才去修复；该 Bug 不会影响产品发布。

第五章 SCRUM 测试方案在 WCS 项目中的实施与分析

5.1 SCRUM 测试方案在 WCS 项目中实施情况简介

WCS 产品从 6.0 版本（Darwyn）开始，逐渐地将一些新开发的功能组件（Components）构建于基于 SOA 的架构上。并且从 WCS 的 6.0.0.4 版本（Trident）开始，WCS 项目组对新增加的组件的开发过程开始采用 SCRUM 开发模式。伴随着开发模式的变化，相对于以往的产品版本的测试，在最新的版本中的测试工作也发生了很大的变化，测试工作不再相对的滞后和封闭，而是从产品的设计阶段就开始进入，并与产品的其他团队进行着密集的交流。每次的测试计划的设计，编写，以及最后的执行都是在多次的迭代递增中进行着。这些特点都体现出了 SCRUM 模式的特性。

作者在最近几年主要负责 WCS 产品的 FVT 测试。在 WCS 最新版本的 SCRUM 测试过程中，对于新增加的基于 SOA 架构的组件，作者主要负责对这些组件的 API 的功能测试。在新组件的 SCRUM 的迭代递增开发过程中，作者在各个阶段的测试工作也是迭代递增的，根据不同阶段的设计文档的内容，对测试计划和测试用例进行着相应的修改和添加，然后实现自动化测试。下面将对作者所做的 FVT 测试工作的内容作详细的介绍和分析，并对另一个很重要的测试过程-压力测试（SVT）也做深入的分析和研究，以此来分析 SCRUM 测试方案在 WCS 项目中的实施情况。

5.2 WCS 项目的测试环境

由于 WCS 产品是一个运行在 WebSphere 服务器上的一个应用服务，因此 WCS 产品的测试环境一般包括以下部分:

1)测试工作机

测试人员人员用来测试软件的计算机，在 WCS 项目中有时还分为手工测试工作机和自动化测试工作机。在测试工作机上进行自动化测试和 UI 的界面测试。

2)产品部署服务器

用来部署被测试产品的服务器。在 WCS 项目中，每当一个新的测试版本推出，测试人员就需要将这个版本的产品部署在产品部署服务器上，然后通过测试工作机访问部署在产品部署服务器上的产品进行软件测试。

5.3 SCRUM过程中各测试阶段工作内容的详细介绍

作者在WCS项目的FVT测试组中，在SCRUM开发测试过程中，主要负责产品功能组件（Catalog）的测试。针对产品功能组件中新增加的基于SOA的功能，作者主要的工作是根据开发设计文档，设计测试计划和测试用例，然后实现对Catalog 组件的API的自动化测试。下面对各测试阶段的工作内容做一个详细的介绍。

5.3.1 前期准备阶段(Pregame)的实施与工作介绍

根据WCS产品的开发/测试经验,在产品进入主要的开发/测试过程Sprint阶段时，要有一些基本的资料来为后面的工作做准备，比如：产品记录，市场/客户调查结果，概要的的产品功能需求，概要的测试场景等，这些都应该在前期准备阶段来完成。在WCS项目中，前期准备阶段包括如下的具体工作：

- 收集需求并对解决方案进行优先级排序
- 设计概要的上层的商业流程场景
- 制作功能模块的演示

下面我们就对上面的工作进行详细的说明。

1.收集需求并对解决方案进行优先级排序

在项目开始的设计阶段，测试小组的人员就根据项目的最早的设计方案，制定相应的测试方案，并根据项目的具体要求进行需求收集，比如需要的测试环境，文档，相关的工具等。并对测试过程中可以发生的变化和遇到的困难进行集体的分析，预先设计一些可能的解决方案。

在SCRUM中,一个项目对应于一个Product Backlog(产品记录),在Product Backlog中列出项目开发过程中所有需要完成的任务，包括:概要的产品功能需求、Bugs、用户提出的改进、具有竞争力的功能以及技术升级等。将这些任务按优先级排序形成Backlog列表，根据该表和风险评估制定产品交付基线，建立系统体系结构，如为已有系统改进或只作有限分析、调整，将Product Backlog项按高内聚低耦合的原则分解为一系列问题包(Packet)，每个问题包是一组对象或构件的集合，为定义sprint作准备。

测试小组在制定解决方案优先顺序时，一般是根据以下几点为依据：

- 研究从产品组反馈回来的基于市场调研的结果
- 根据调研结果定义概要的上层的商业流程场景（比如设计一个用户登录商店的一个简单的场景）。这些场景将为后面的测试计划和测试用例的编写提

供基础。

- 为之前设计的上层商业场景制定优先级顺序，并将结果与产品组进行讨论修改，并最终达成一致。
- 从开发组获得他们的概要的开发计划：根据开发组的开发计划，可以知道开发组的开发顺序与日程，人员安排，因此可以更有效，准确地制定测试计划。
- 将安排好优先级顺序的场景设计记录，与产品组，开发组等再次共同讨论，完善场景记录，并确认出其中的一些牵制问题（比如某个场景设计的实现必须要依托于其他某个或某几个场景）。

表 5-1 是一个制定好的概要的场景优先级列表的示例：

表 5-1 概要的场景优先级列表示例

Scenario	Priority (H/M/L)	Target Release	Notes
Enhance order/payment component to support punch-out style payment.	H	Emerald	This is also important to Indian and Europe. Paypal express check out is also same flow.
Build starter store for China market with comprehensive home page, product rating/remarks, browse history and China Style Payment.	H	Emerald	The new start store will built upon the new web 2.0 store to shorten deployment time and reduce TCOI.
Integration with local delivery system e.g. China Post	H	FOAK	
China Loyalty point requirements	H	TBD	Nearly every customer in China need this feature.
Enhance promotion component to support complex e-Coupon rules in China	H	TBD	Nearly every customer in China need this feature.
High scalability hosting solution to support thousands of stores	H	Emerald	We may need some testing to find out the bottle neck of the system, and document the limitation.
Componentization – customer can buy/subscribe the required component/function	M	Future	This complies with Commerce SOA strategy.
Support different service level	M	FOAK	Customer can subscribe commerce service with different service level. E.g. max concurrent users.
Catalog/price/promotion based on Jurisdiction	M	FEP 2	Promotion is covered in Promotion solution.
Collaborative purchase	M	FOAK	Several customers showed interest on this feature.

·Priorities ·H- Solution is unusable with out this scenario ·M- Solution is usable but incomplete ·L – Nice to have	·Target Release ·Emerald – June/09 ·FEP 1 – Mar/10

2. 设计概要的上层的商业场景

在 WCS 项目中，一般目前项目刚开始，设计文档也是概要的，所以测试组的人员只能先根据这些概要的文档，先设计一些概要的设计方案，比如先设计一些简单的测试场景，为后面设计具体的测试计划和测试用例打好基础。

图 5-1 是一个为 Catalog 组件设计的一个商业场景描述的例子：

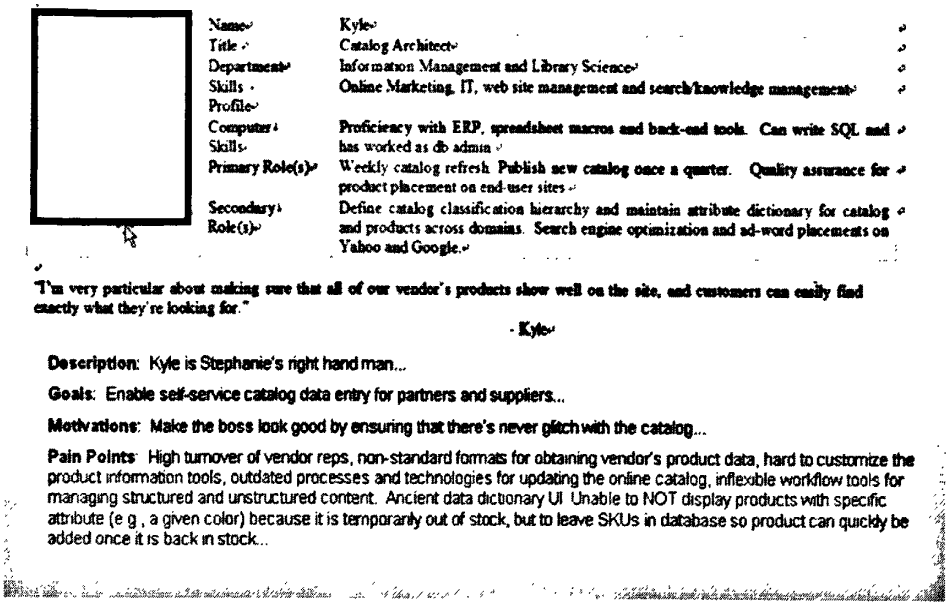


图 5-1 商业场景描述示例

3. 进行模块的演示制作

上面的一些产品记录，场景设计和商业流程分析等都是纸上东西，开发人员与测试人员通过他们都只能获得理性认识，所以为了更好的开发和测试产品，我们还需要进行模块的演示制作。

UI 模块的演示制作的相关活动：

- 用 flash 制作 UI 的仿真演示：按照设计文档，客户的需求设计，我们一般采用 flash 制作 UI 的仿真演示，这样可以更直观的看到产品最终的样子，并直观的看到产品所提供的功能和流程。
- 各个开发组，测试组，产品经理等对演示进行讨论交流：不同的人对于同一个文档可能有不同的理解，而且不同的人对于不同的技术有不同的掌握，比如客户经理设计好的场景，在开发人员和测试人员来看，会觉得很难实现，需要修改成另一种方式去实现该功能。因此在演示做完后，要进行讨论和修改，以达成一致。
- 从客户获得最早的反馈：在 flash 演示做好后，应该第一时间交到客户那里进行确认，并及时地从客户那里得到反馈，以便进行及时地修改。

图 5-2 是一个 flash 演示的示例，它显示了在 WCS 产品的 CMC 管理界面中，Catalog 组件中显示产品的功能（DisplayProducts）的最后预期效果。

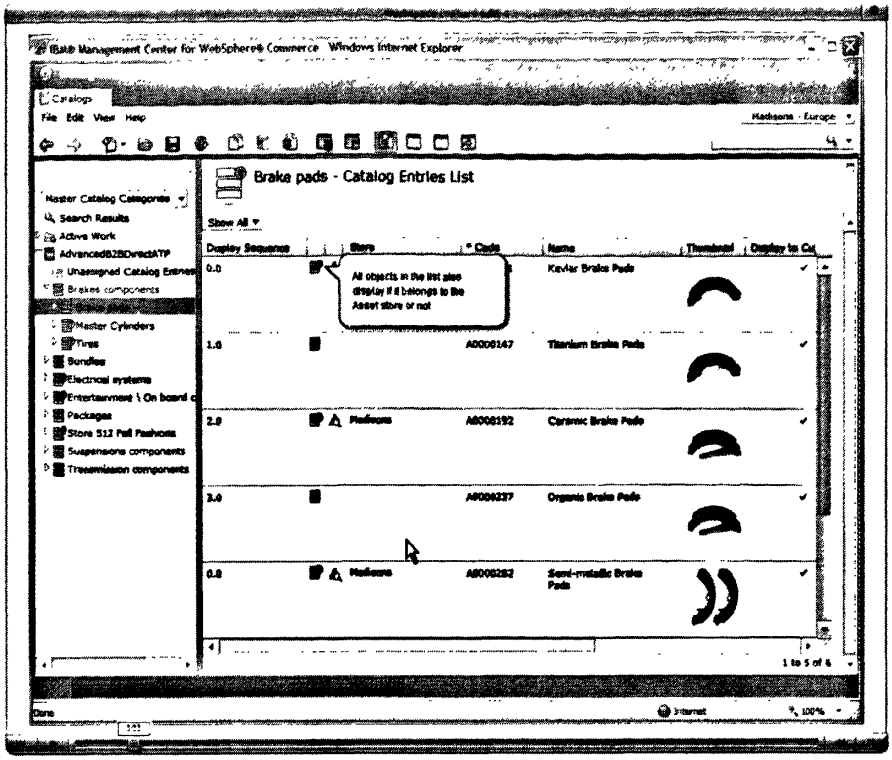


图 5-2 flash 演示的示例

5.3.2 Sprint 阶段的实施与工作介绍

Sprint 阶段作为 SCRUM 过程中最重要的一环,对于软件产品能否在 SCRUM 开发过程中顺利并成功的完成,起着至关重要的作用。WCS 产品新组件功能的分阶段代码开发和测试都在这个阶段完成,并且每个阶段都会最终生成一个较完整和成熟的经过测试的阶段产品。下面就介绍作者在 WCS 测试项目中的 Sprint 阶段的测试工作。

1. Sprint 阶段迭代过程方案

在 WCS 的 SCRUM 开发/测试过程中,Sprint 阶段是一个迭代的递增过程,在这个重复过程中,各个开发组和测试组将会一步一步地完善功能并进行相应的测试,作者根据实际应用经验,将每一个迭代过程设计为下面的图形,如图 5-3 所示:

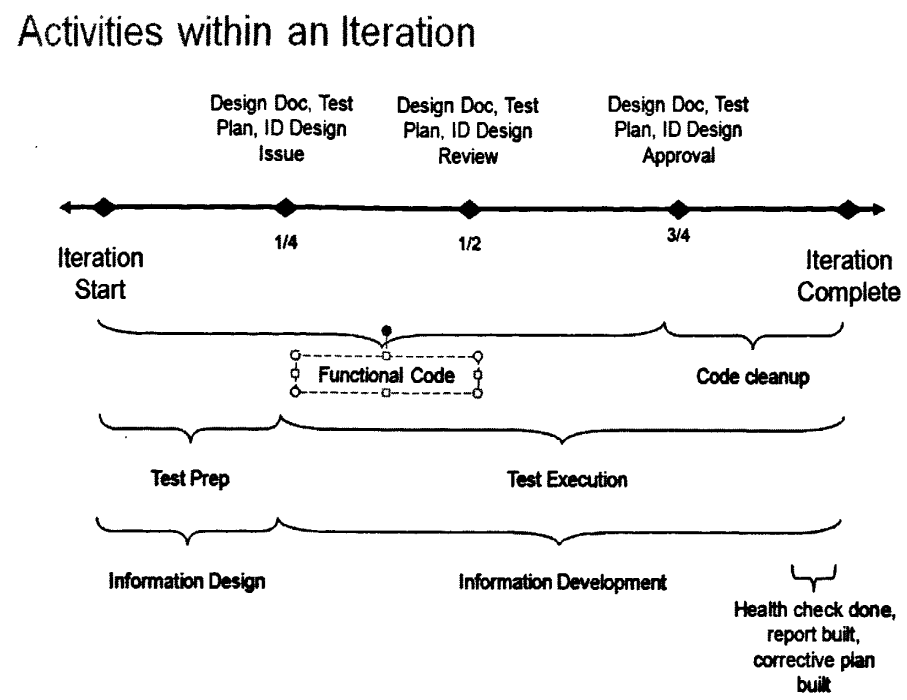
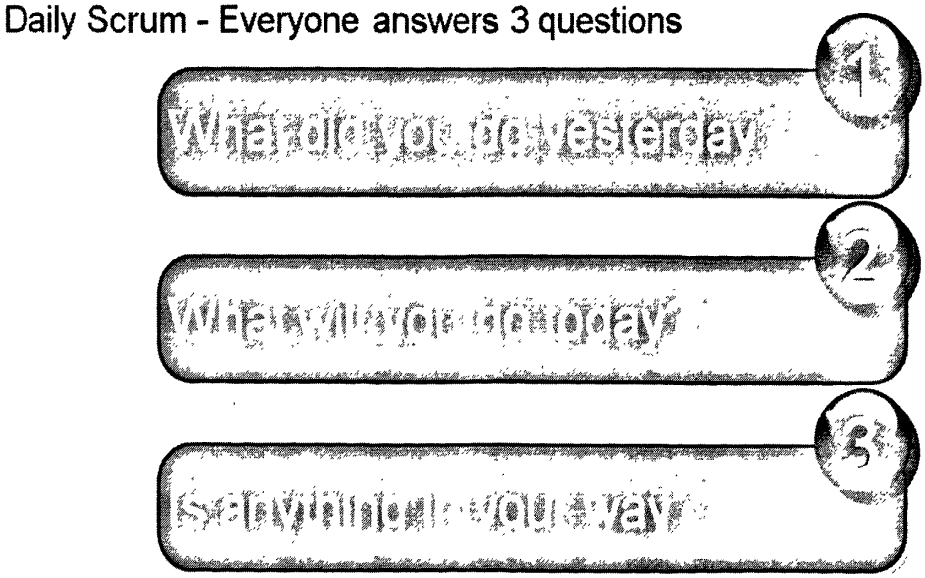


图 5-3 WCS 项目中一个 Sprint 阶段的开发/测试活动图

2. Sprint 阶段的测试工作的详细介绍

SCRUM 的开发方法与以往的开发方法最大的不同就在于它的敏捷，快速和迭代。这种特点在 Sprint 阶段体现的尤为突出。开发人员和测试人员一起，快速的开发和测试着产品。在这个阶段，测试人员几乎每天都会应对新的任务，面对着可能的新变动。在 WCS 项目中，测试人员每天都会问自己 3 个问题：



根据图 5-3 可以知道，WCS 项目中，为每一个 Sprint 阶段设计的时间一般为 4 周，在这 4 周中，开发组和测试组并行的进行着各自的工作。

作者在 Sprint 阶段测试 Catalog 组件过程中，所做的具体执行工作有：

1)开发测试计划和测试用例

WCS 项目中，由测试组长开发每个 Sprint 阶段的测试计划(功能文档预审会议后的两天之内完成)，然后测试人员根据功能文档与测试计划开发测试用例。在这段时期会遇到以下问题:功能文档不够完善或需要紧急更改，这样已经开发了的测试用例就有可能作废或者需要更改。在作者设计的测试计划中，一般包括以下的内容：

.测试文档说明

.相关人员说明

.测试范围

.测试场景

.测试用例的详细说明，包括测试用例编号，执行的具体步骤说明，每一步的输入值和输出值，用到的 API 以及最后希望的结果。

2）预审和发布测试计划

在设计完测试计划后，测试人员要发布本次测试计划给相关的人员或小组进行预审，相关人员会提出注释和修改意见。随后会进行测试计划的讨论会，在讨论会上，各方要对测试计划的注释和修改意见达成一致，并记录好会议记录。随后在测试人员修改好测试计划后，通过大家的一致认可后，由产品经理确认，发布本次测试计划。流程一般如图 5-4 所示：

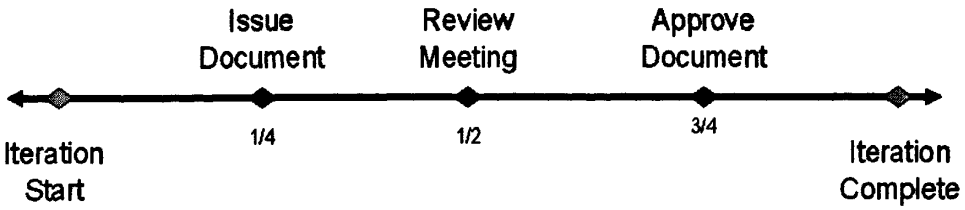


图 5-4 WCS 的测试计划（Test Plan）在每个 Sprint 阶段的流程示例图

3)开发自动化测试脚本

对于 API 的测试，主要以自动化测试为主，在 WCS 项目中，对 API 的测试是采用 Junit 测试工具来开发的，运行的测试环境是 Eclipse 软件。当测试用例完成并通过测试计划预审会议后，测试人员开始开发自动化测试脚本。

i）对自动测试脚本的设计

由于是针对 API 的自动测试，而且是在 Sprint 阶段执行，根据 SCRUM 过程的特点，我们知道每一个 Sprint 阶段都会有新的功能加入进来，并且原有

的功能也可能根据设计的变化而发生改变，因此如何设计好自动测试脚本的结构，尽量保证每次运行都进行少量的代码修改即可，就显得尤为重要。针对这个特性，作者为自动测试脚本设计了代码和测试数据分离的方法，这样如果代码有变化，或者数据有变化，进行修改时都会比较方便，这在时间很紧张的Sprint 阶段很重要。

下面是一个 Catalog 组件 API 测试的自动脚本代码示例：

测试代码：

```
//TestCase FCAT_ATLAS101 的测试代码
public class FCAT_ATLAS101 extends junit.framework.TestCase{
    String FILENAME = CatalogConstants.SCRIPTS_ROOT +
"data/FCAT_ATLAS101.xml";

    protected void setUp() throws Exception{
    }

    public void testFCAT_ATLAS101() throws
DatablockMissingException, ParameterMissingException,
TestCaseFailException, TestDataMissingException{
        //测试数据管理类
        DataSetManager dsm = null;
        //Catalog 组件相关任务类
        CatalogTaskLib tasks = null;

        dsm = new DataSetManager(FILENAME);
        tasks = new CatalogTaskLib();

        //从数据文件中提取本测试用例的数据
        dsm.setTestName("FCAT_ATLAS101");

        //执行 API "getCatalogGroup" 测试
        ShowCatalogGroupType catalogGroup =
tasks.getCatalogGroup(dsm, "getCatalogGroupA", null);

        //Retrieve the catalogGroupId from the previous task results
```

```
            Map catalogGroupMap = tasks.getCatalogGroupId(dsm,
catalogGroup);

            //执行 API updateCatalogGroup 测试
            RespondCatalogGroupType respondCatalogGroup =
tasks.updateCatalogGroup(dsm, "updateCatalogGroupDescription",
catalogGroupMap);

            tasks.getCatalogGroup(dsm, "getCatalogGroupB", null);
        }
    }
```

对应的测试数据：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Scenario name="Temp Scenario">
    <Env>
        <BusinessContext>
            <!—登陆商店的数据—>
            <Parameter name="langId" value="-1"/>
            <Parameter name="storeId" value="storeId"/>
            <Parameter name="catalogId" value="catalogId"/>
        </BusinessContext>
        <CallbackHandler>
            <Parameter name="userId" value="username"/>
            <Parameter name="userPwd" value="password"/>
        </CallbackHandler>
        <SDOPackage>
            <Parameter name="package"
value="com.ibm.commerce.catalog.facade.datatypes.impl.CatalogPackageImpl"/>
        </SDOPackage>
    </Env>



    <!—  ##################### —>
    <!—  Test for BOD as Input —>
    <!—  ##################### —>

    <!—测试用例FCAT_ATLAS101的测试数据—>
    <Test name="FCAT_ATLAS101">
        <Env>
```

```
            <BusinessContext>
                <Parameter name="langId" value="-1"/>
            </BusinessContext>
        </Env>

        <Datablock name="getCatalogGroupA">
            <Input>
                <Parameter name="getCatalogGroupA">

                </Parameter>
                <BOD name="getCatalogGroupA">
                    <_cat:GetCatalogGroup
xmlns:_cat="http://www.ibm.com/xmlns/prod/commerce/9/catalog"
xmlns:_wcf="http://www.ibm.com/xmlns/prod/commerce/9/foundation"
xmlns:oa="http://www.openapplications.org/oagis/9"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

                        <_cat:DataArea>
                          <oa:Get>
                            <oa:Expression
expressionLanguage="_wcf:XPath">{_wcf.ap=IBM_Details}/CatalogGroup[CatalogG
roupIdentifier[(UniqueID='51100000001')]]</oa:Expression>
                          </oa:Get>
                        </_cat:DataArea>
                    </_cat:GetCatalogGroup>
                </BOD>
            </Input>

            <Output/>
        </Datablock>

        <Datablock name="updateCatalogGroupDescription">
            <Input>
                <Parameter name="updateCatalogGroupDescription">

                </Parameter>
                <BOD name="updateCatalogGroupDescription">
                    <_cat:ChangeCatalogGroup
xmlns:_cat="http://www.ibm.com/xmlns/prod/commerce/9/catalog"
xmlns:_wcf="http://www.ibm.com/xmlns/prod/commerce/9/foundation"
xmlns:oa="http://www.openapplications.org/oagis/9"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                        <_cat:DataArea>
                          <oa:Change>
```

```
                        <oa:ActionCriteria>
                          <oa:ActionExpression actionCode="Change"
expressionLanguage="_wcf:XPath">/CatalogGroup[1]/Description[1]</oa:ActionE
xpression>
                        </oa:ActionCriteria>
                      </oa:Change>
                      <_cat:CatalogGroup>
                        <_cat:CatalogGroupIdentifier>
                          <_wcf:UniqueID></_wcf:UniqueID>
                        </_cat:CatalogGroupIdentifier>
                        <_cat:Description language="-1">
                          <_cat:Name>updated101 name</_cat:Name>
                          <_cat:Thumbnail>updated101
thumbnail</_cat:Thumbnail>
                          <_cat:FullImage>updated101
fullImage</_cat:FullImage>
                          <_cat:ShortDescription>updated101
shortDesc</_cat:ShortDescription>
                          <_cat:LongDescription>updated101
longDesc</_cat:LongDescription>
                          <_cat:Keyword>updated101 keyword</_cat:Keyword>
                          <_cat:Attributes
name="published">1</_cat:Attributes>
                          <_cat:Attributes name="note">updated101
note</_cat:Attributes>

                        </_cat:Description>
                      </_cat:CatalogGroup>
                    </_cat:DataArea>
                  </_cat:ChangeCatalogGroup>
                </BOD>
              </Input>

              <Output/>
            </Datablock>

            <Datablock name="getCatalogGroupB">
              <Input>
                <Parameter name="getCatalogGroupB">

                </Parameter>
                <BOD name="getCatalogGroupB">
                  <_cat:GetCatalogGroup
xmlns:_cat="http://www.ibm.com/xmlns/prod/commerce/9/catalog"
```

```
xmlns:_wcf="http://www.ibm.com/xmlns/prod/commerce/9/foundation"
xmlns:oa="http://www.openapplications.org/oagis/9"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                    <_cat:DataArea>
                      <oa:Get>
                        <oa:Expression
expressionLanguage="_wcf:XPath">{_wcf.ap=IBM_Details}/CatalogGroup[CatalogG
roupIdentifier[(UniqueID='51100000001')]]</oa:Expression>
                      </oa:Get>
                    </_cat:DataArea>
                  </_cat:GetCatalogGroup>
                </BOD>
            </Input>

            <Output>
                <xPath>
                    <Parameter name="//_cat:Description/_cat:Name"
value="updated101 name"/>
                    <Parameter name="//_cat:Description/_cat:Thumbnail"
value="updated101 thumbnail"/>
                    <Parameter name="//_cat:Description/_cat:FullImage"
value="updated101 fullImage"/>
                    <Parameter
name="//_cat:Description/_cat:ShortDescription" value="updated101
shortDesc"/>
                    <Parameter
name="//_cat:Description/_cat:LongDescription" value="updated101 longDesc"/>
                    <Parameter name="//_cat:Description/@language"
value="-1"/>
                    <Parameter name="//_cat:Attributes[@name='published']"
value="1"/>
                    <Parameter name="//_cat:Attributes[@name='note']"
value="updated101 note"/>
                </xPath>
            </Output>
        </Datablock>

    </Test>
```

ii）同时这一阶段的问题是，自动化测试脚本的开发过程也是一次软件开发测试过程，需要消耗大量人力和时间，同时在以后的阶段需要不断的更新维护。如表 5-2 所示，两个 Sprint 阶段的新增自动化脚本的开发时间都是 1 人/

天，而第二个 Sprint 阶段的维护时间是前一个 Sprint 阶段的 2 倍。可见，随着自动化测试脚本的增多，需要花费在脚本维护的时间大大增加。因此，就需要改进自动化测试工具，从而缩短自动化测试脚本的维护时间。

表 5-2 自动化测试脚本开发维护时间

Sprint	开发时间	维护时间
Sprint 1	1 人/天	2 人/天
Sprint 2	1 人/天	4 人/天

4)实施手工与自动化测试并发

当 Sprint 阶段的第一个 Build 推出后，测试需要将产品部署在产品部署服务器上，然后开始进行手工或自动化测试。在 WCS 项目中每个测试人员负责若干功能模块，每个功能模块的手工测试和自动化测试是并行的同时展开的。表 5-3 分别记录了 WCS 项目某个 Sprint 阶段手工和自动化测试发现 Bug 数量(手工测试发现的 Bug 与自动化测试的 Bug 有可能重复)。

表 5-3 某 sprint 中手工与自动化测试发现的 Bug

测试类型	新增特性直接相关的 Bug	新增特性引起的，存在于原有特性的 Bug
手工测试	6 个	1
自动化测试	4 个	3

通过分析多个 Sprint 阶段手工与自动化测试发现的 Bug 数量，论文得出这样一个推论:手工测试比较容易发现产品新增特性中存在的 Bug，而自动化测试比较容易发现由新增特性引起的存在于原有特性中的 Bug。论文认为造成这一结果的原因有:由于新增特性对应的自动化测试脚本开发不久，其中存在一定缺陷，而原有特性对应的自动化测试脚本经过测试人员长期的更新维护，具有良好的测试性能，因此自动化测试能够很好的发现由新增特性引起的存在于原有特性中的 Bug;而对于测试人员来说，对原有特性已经存在了一定的固化认识，因此不容易发现原有特性中存在的问题。由于手工测试与自动化测试这种相辅相成的关系，在 WCS 项目中一直坚持两种测试并行进行。但是存在以下问题:对硬件需求多(资金需求)、对测试人员效率要求较高(每位测试人员需要同时进行手工测试和自动化测试)。

5)对测试进行 Debug 调试

这里论文统计了 WCS 项目中两个相连 Sprint 阶段的各项测试数据结果，用来分析 SCRUM 测试方案的优缺点。

第一个 Sprint 阶段采用了 SCRUM 测试模型，引入了 Debug 调试，这个阶段的 Bug 变化率如图 5-5 所示：

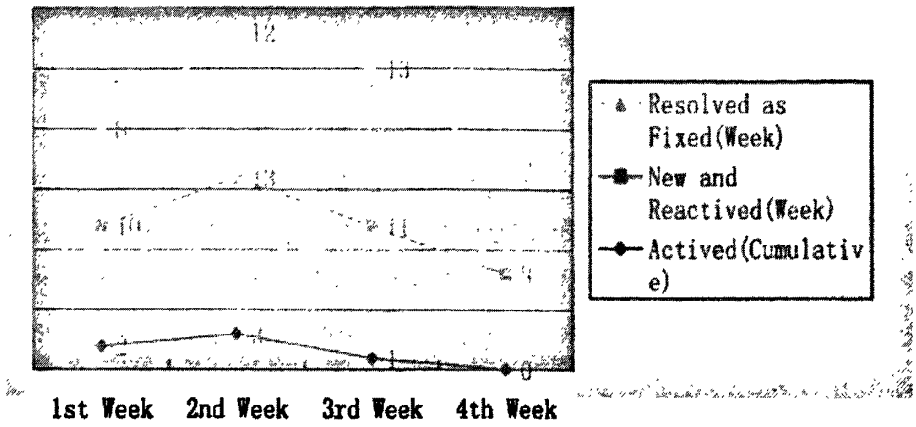


图 5-5 Sprint 1 阶段的 Bug 变化率

第二个 Sprint 阶段，采用传统的 V 模型，这个阶段的 Bug 变化率如图 5-6 所示：

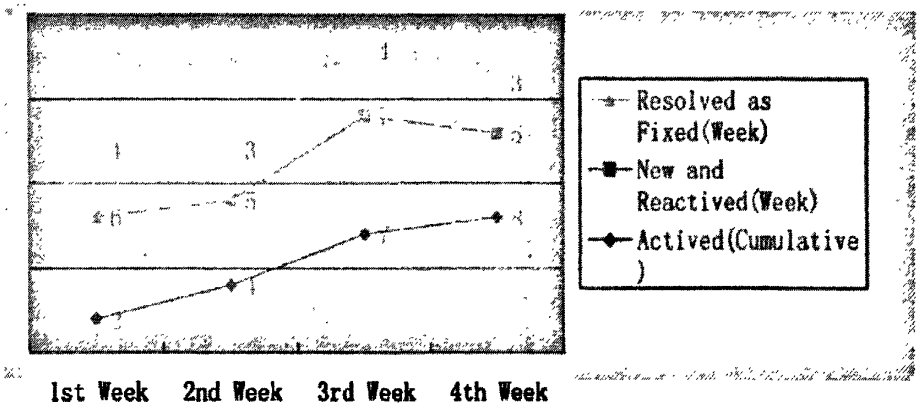


图 5-6 Sprint 2 阶段的 Bug 变化率

图 5-5 和图 5-6 中的 Actived(Cumulative)数据线中的数据，分别表示在每周内尚未解决的 Bug 总量；New and Reactived(Week)数据线中的数据，表示每周新增或重新被激活提交的 Bug 量；Resolved as Fixed(Week)数据线中的数据，表示每周修复的 Bug 量。

通过分析两个 Sprint 阶段的 Bug 变化率，可以看出采用 SCRUM 测试方案能够很快找出 Bug，达到尽早发现尽早解决。同时引入 Debug 调试后，节省了开发人员修复 Bug 的时间，提高了 Bug 修复的效率。

最后，在交付和巩固阶段，测试人员的具体工作，一般是进行软件维护性测试。通过分析实施 SCRUM 测试方法的 WCS 项目所得结果，论文认为 SCRUM 方案中存在的一个比较明显的问题是，需要测试人员投入大量的时间和精力从事

测试辅助工具的开发工作。

6）每日 Sprint 例会

从之前的 Sprint 阶段的特点来看，在这个阶段中，各个团队之间的紧密地联系和信息的及时地沟通尤为重要。它能够促进产品的顺利开发与测试。而保持这种联系和沟通的一个很有效的方法就是开会-每日 Sprint 例会。

在 WCS 项目的 Sprint 阶段中，每天都要进行 Sprint 会议，在会议上要进行各个团队的沟通，并对一些问题进行讨论。

根据 WCS 项目的实践经验，作者发现每日的 Sprint 例会的一个重要的特点就是随着 Sprint 阶段的进行，在会上所要讨论和汇报的内容会越来越少。这是因为在每一个 Sprint 阶段刚开始的时候，由于各个团队刚刚进入，对各种文档和计划都是一个渐进熟悉的过程，这势必会在一开始产生很多疑惑和问题，需要各方面的讨论和解释。同时一开始的测试过程中，由于产品功能的不完善，也会产生比较多的 bugs，这样在会上对 bug 和问题的讨论也会很多。但是这些情况都会随着 Sprint 的进行，随着各团队的共同努力，变得越来越良好，因此在 Sprint 每日例会上所耗费的时间也越来越少。图 5-7 是对这个情况的一个图形化的说明：

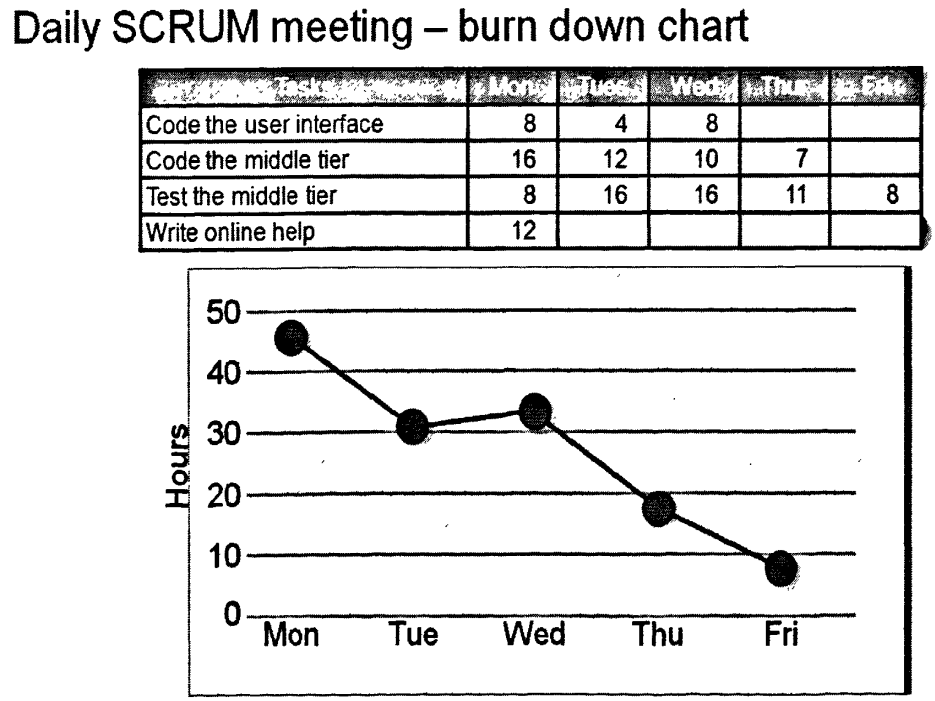


图 5-7 Sprint 每日例会时间花费图

通过上图可以看到，经过统计，每日 Sprint 例会所耗费的时间会变得越

来越少。对于每日例会所要讨论的内容，这里也提供一个范例，图 5-8 是 WCS 项目进行过程中，每日 Sprint 例会上会议记录所采用的一个模板示例：

May 16, 2008

Team Member	Status	Non-attachment Time (hours)
FVT	**Test plan:** Service test plan (combined driver3&final) was issued on 05/14, will be reviewed on 05/20 UI test plan (combined driver3&final) was issued on 05/14, will be reviewed on 05/21 **Execution:** (started on 05/14) Driver3 service: Total/Attempts/Completed/Failed/Blocked: 135/71/62/3/0 (status of unofficial testing, would revise the completed testcases according to review commends) Driver3 UI: Total/Attempts/Completed/Failed/Blocked. 155/1/0/1/0 **Hot defects:** #172140 Failed to preview the file — open #172138 Failed to drag and drop file path when creating file — open	
	What have you done since the last Daily Scrum regarding this project? 126 DSL performance 0.2 PD remaining 1.6 PD **What will you do between now and the next Daily Scrum meeting regarding this project?** Code ready. UT triggers on DB2 && Oracle. **Is anything holding you up?** need an Oracle environment **What did you do for non-project tasks?** Sony PMR 0.7 PD LI 2662 defect 0.2 PD	Sony PMR 0.7 PD LI 2662 defect 0.2 PD

图 5-8 Sprint 每日例会会议记录模板

根据研究，可以发现有效的每日例会将对测试的顺利进行具有如下的帮助：

- 避免了关于技术问题的重复讨论。
- 尽最早可能的发现问题和风险。
- 可能帮助各个团队在开会之前收集相关的信息，对项目的进展有一个良好的掌控。
- 可能帮助团队的成员们随时处于同一个起跑线上。
  - ✓ 可以在会上集中的分享各种信息。
  - ✓ 可以集中的通告产品设计上的一些修改。
- 根据会上得到的最新情况，及时地更新随后每天的资源分配。
- 对测试团队日历进行及时的更新。

5.3.3 最后阶段（Postgame）的实施与工作介绍

在经过了一系列的 Sprint 迭代阶段的开发/测试后，产品的新功能将能到很好的完善，新功能性能也将能够具备客户的考验，产品的 SCRUM 开发过程将

进入到尾声。在最后阶段的测试中，FVT 测试组会进行下面的测试工作：

- 正式的 FVT 测试
- 产品在线帮助说明测试

1.正式的 FVT 测试

正式的FVT测试包括对Sprint阶段完成测试的功能模块再次进行回归测试，同时也包括对老版本中的功能进行回归测试。这里我们设计了一个FVT正式测试阶段的进入标准，以作为参考：

- 各个功能模块已经声明开发完成（DCUT）
- 单元测试（UT）的检查已经完成
- 对于功能模块的确认测试（Acceptance testing）已经通过
- 对于功能模块的Test plan已经被批准(即生成了最终版的test plan)
- 各测试小组的具体的测试计划已经发布

正式的 FVT 测试中的测试范围（Test coverage）：

- 新开发的功能模块 + 之前版本的回归测试部分
- 更多的平台的配置（更多的测试环境）r
- 多语言测试（GVT）
- 大数据测试（LOT）
- 在线产品说明测试

2.产品在线帮助说明测试

在经过了几个Sprint阶段的迭代测试后，产品的功能已经相对比较完善，相关的产品说明帮助也应该已经接近完成，这是FVT测试组可以开始进行产品在线帮助说明测试。

5.4 对 WCS 项目 SCRUM 测试方案实施的总结和分析

5.4.1 FVT 和 SVT 实施中遇到的问题和解决办法

在任何软件开发模式中进行测试，都不会是一帆风顺的，都遇到这样或那样的问题。在SCRUM过程中也一样。作者根据自己在WCS项目的FVT组的实际测试过程，并另外对SVT组的工作进行调研和分析，总结出了FVT和SVT在

SCRUM 过程中测试时，会遇到的问题以及相应的解决办法。

1. FVT 测试在 Sprint 阶段中的困难以及解决办法的分析

.遇到的问题和解决办法

1)集成测试中的问题：

- 一个测试场景依赖于多个开发任务：我们经常会碰到这样的事情，一个测试场景依赖于多个模块才能工作，当我们开始测试时，却发现其中的一个或几个模块还没开发好，结果测试工作被迫停止，严重的的影响了项目的进度。

建议和解决办法：

- 首先测试人员要分析清楚 FVT 的测试场景和开发人员的工作任务的关系，场景中开发模块的先后顺序，以及优先顺序。然后依次为标准要求开发组调整任务的优先顺序来推动 FVT 测试的执行。
- 同时也根据开发人员的任务列表，来设计一个合理的 FVT 测试计划表，使之不与开发任务产生很大的冲突，以避免不必要的麻烦。

2)Bugs 处理上的问题：

- 由于 Sprint 阶段一般时间都很短，开发人员也快速的开发者产品的功能模块，开发出来后，有的进行简单的单元测试，有些还没怎么进行内部测试，就将他们交给测试组进行 FVT 测试，这样必然会造成有很多的 bugs，在 WCS 项目的测试中，通过统计，我们发现平均每天会有 3.75 个 bugs 被报告出来。
- 有些 bugs 还比较好处理，但是有些 bugs 就比较严重，它会严重的影响到整个测试工作无法进行下去。

建议和解决办法：

- 一旦发现问题，就要尽快地解决，在解决过程中，开发组和测试组应该保持随时的交流，以帮助问题尽快解决。
- 问题解决后，要及时的进行回归测试。
- 每天进行 bugs 的确认会议，统计和讨论当前的 bugs 的情况。使每天都对 bug 的情况有所处理和进展。

3)Test plan 的准备上遇到的问题

- 在每一个 Sprint 阶段，一般时间很短，而留给用来写设计文档和计划文档的时间更短（一般只有一周的时间），因此经常会出现设计文档很

粗略就发布出来，测试人员根据这样的设计文档很难来设计自己的 test plan。

建议和解决办法：

- 由于 Sprint 是迭代过程，每一个 Sprint 阶段都是在之前的基础上进行添加和修改，因此首次的 Sprint 阶段很关键，而且首次的 Sprint 阶段的 Test Plan 设计也将是最为麻烦和复杂的，因此一个很好的方法就是在整个 SCRUM 中的开始的 Pre-game 的阶段就开始着手设计 Test plan。
- Test plan 作者必须要对自己所负责的功能模块有深入的了解，这样才会在信息不全的情况下设计出合理的 Test Plan。
- 其次测试人员要与开发人员保持密切的交流，将问题在最短的时间内解决掉。

4)回归测试遇到的问题：

- 由于在 Sprint 阶段测试中难免会遇到 bug，但测试人员将 bug 提交，开发人员进行 bug 修复后，测试人员要进行重新的测试，这个属于回归测试了。但是同样的，这些回归测试也要被安排在 Sprint 阶段中，可想而知，时间会很紧张。但是是否会有回归测试，以及回归测试可能会占用多少的时间，这些在开始都还是未知数，因此这个对于 FVT 工作计划和人员安排都是一个很大的挑战。

建议和解决办法：

- 如果可能的话，为 bug 找到一个有效的回归测试范围，即不用之前所有的测试用例全部又测试一遍，这样很浪费时间。 如果遇到测试环境很难搭建的情况，可以考虑直接将开发人员的修复补丁打到当前的测试环境中，进行测试，以节省时间，避免不必要的浪费。

.FVT 在 Sprint 阶段测试经验的总结：

- 测试人员要和开发人员保持紧密的联系，共同进退。只有这样才能顺利按时地完成每一个 Sprint 阶段。
- 在每天的交流会议上，要及时地提出问题和疑惑，将问题尽早暴露，尽早解决。为以后的测试的顺利进行打通通道。
- 确保能及时地了解到开发人员的所进行的变化。
- 如果遇到有太多的问题，需要推迟解决，但是这样会影响到后面的测试环节，务必一定要报告出这个问题，因为测试的部分是产品记录中重要一个部分。

2. SVT 测试在 Sprint 阶段中的困难以及解决办法的分析

.为什么要将 SVT 引入到 Sprint 阶段?

在有些产品的 SCRUM 开发过程中，在 Sprint 阶段是不引入 SVT 测试的，因为一般 SVT 测试需要花费的时间很长，解决相应的问题也会很麻烦，这样的特点和敏捷开发有些冲突。但是在 WCS 项目的 SCRUM 开发过程中，却引入了 SVT 的测试，这是因为根据以往的经验，我们发现：

1) SVT 可以发现性能问题

- ✓ 一般情况下，性能问题越晚发现，所要花费的解决问题的费用会越来越昂贵。
- ✓ 性能 bugs 一般情况下都是底层设计 bugs，因此越早发现，越好解决。
- ✓ 性能 bugs 同样也影响着其他的测试组 (e.g. FVT, 开发组)，因此发现并解决性能问题，会提高整个产品的开发速度。

2) 相比于之前的测试方法，将 SVT 引入到 Sprint 阶段中，可以更早的发现性能问题。

.SVT 测试在 Sprint 阶段中的困难以及解决办法的分析

在 WCS 的开发历史中，由于将 SVT 引入到 Sprint 阶段，也是在最近的产品版本开发中进行的，在刚开始尝试中，遇到了一些问题：

1)“残缺的”功能

- 在 Sprints 的开始阶段，由于很多功能模块没有开发完成或者还没有开发，因此进行 SVT 的测试很困难，没有足够的内容来进行系统测试。
- 对于性能测试的通过标准不是很明确。

2)Test plans 仍然处于修改和变动的状态

- 以往是在最后开发完成后再进行 SVT 的测试，在 Sprints 阶段，每个功能模块都还在进行着修改，这样对于产品的稳定性会造成影响，从而也会影响到 SVT 的测试作用。
- 因此还需要很多的 Test Plan 的修改。

3)很短的测试时间

- SVT 测试由于需要测试性能，因此往往需要进行大数据量和长时间的测试工作，这是 SVT 测试和 IVT/FVT 测试的最大区别。但是每个 Sprint 阶段

的时间往往很短，因此有的时候往往时间不够用。

- 同时 SVT 测试还需要设计大量的测试数据和脚本，这些都要占用很多的时间。

4)对于大型软件项目，SVT 测试组往往是一个跨大洲的组

- 很多地方往往要相差 12 个小时的时差。
- 造成了测试人员和开发人员交流的困难。
- 长时间的交流循环，最后很容易造成 SVT 测试延期。

针对上述的问题，我们可以采取下列的解决办法来改善问题：

1）为避免功能模块短缺，可以采取延迟一个 Sprint 周期开始 SVT 测试的方法，如图 5-9 所示：

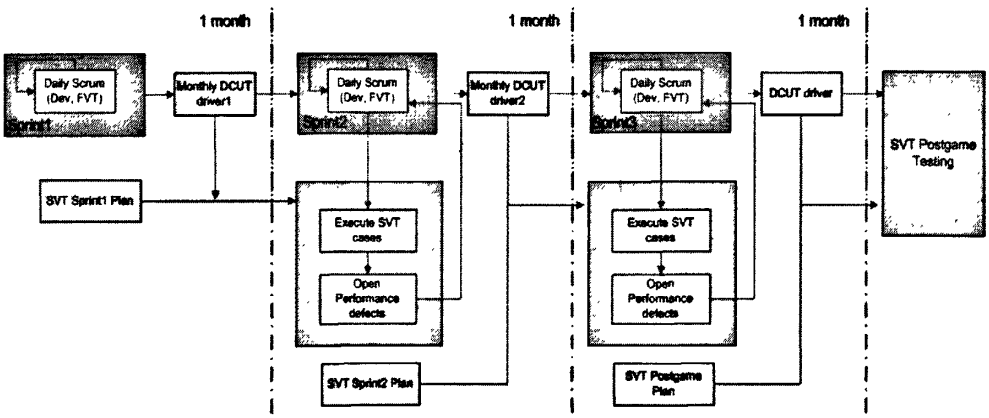


图 5-9 SVT 在 Sprint 阶段延迟执行示例图

这样，在经过了一个 Sprint 阶段后，主要的功能已经完成了一轮的开发和测试，产品得到初步的稳定，在随后的开发中，也将是基于第一轮的基础上进行，这时 SVT 测试再进入，将会有效的避免缺少功能的情况。

2）进行“架空”测试

由于 SVT 测试的问题很大程度和产品的底层以及所采用的架构有关，因此我们可以考虑在产品的新版本采用新技术前，就先考虑和研究新架构的特点，进行对于新架构的性能测试的试验和实现。这样在 SCRUM 开发过程开始后，能有所准备的应对新产品的测试。

3）进行有效的 Test case 分配

可以从整体考虑，将回归测试的测试用例分配到整个 Sprint 过程，在可能的情况下，不必将所有的测试用例分配在一个 Sprint 阶段中。同时可以将主要的测试精力放到和客户核心性能有关的功能模块上。

4）进行 24 x 7 的循环测试

测试人员实时地与开发人员进行交流，遇到问题马上沟通并解决。当开发人员修复了 bug 后，可以马上将修复补丁交给测试人员进行验证测试，以此达到在最短时间解决问题的目的。

5）对底层模块进行先行的性能测试

采用 SCRUM 开发模式进行开发，一个特点是往往底层的服务先行开发，而将表现层的部分放到最后开发。这样能够保证产品的稳定性。针对这样的特点，SVT 的测试可以在 UI 部分没有开发前，就先对底层服务进行测试，尽早的发现问题。图 5-10 是对现行性能测试的一个说明。

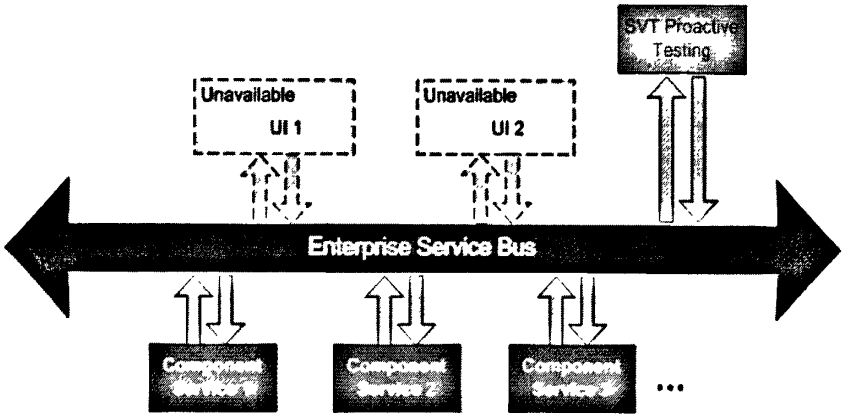


图 5-10 先行 SVT 测试

6）对用户界面（UI）进行原子测试

这个经验和上一条有些类似，是在一部分 UI 功能没有实现时，造成一些测试用例不能顺利的进行，这时我们可以对已经完成的 UI 模块进行小单元的测试。如图 5-11 所示：

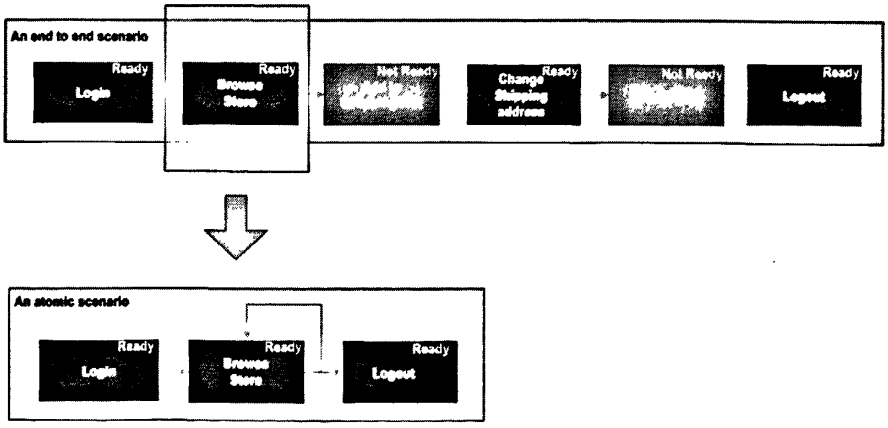


图 5-11 原子测试示例图

在图 5-11 中，我们可以看到，一个测试用例需要由六个功能模块组成，但是这时还有 2 个模块没有开发完成，所以这个测试用例还不能进行测试。这

时我们可以考虑进行一个小规模的测试，以完成对大部分模块的测试，及早发现可能的问题。

5.4.2 WCS 的 SCRUM 测试过程实施情况的总结

通过分析在 WCS 项目中实施 SCRUM 测试方案，论文认为 SCRUM 测试方案有以下特点：

1)测试活动从设计阶段就开始介入，提高了问题的发现率和提前率，实现了早发现，早修改，低风险。

2)实现了一套规范的管理流程，从设计到实现到执行，能够遵循定义好的计划。

3)自动化测试采用数据驱动和功能分解方法实现，大大提高了自动化测试的成熟度。

4)手工测试与自动化测试并发机制，确保在测试中能及时发现 Bug。

5)完善的 Bug 追踪机制。

同时论文也认为 SCRUM 测试方案具有以下不足之处：

1)SCRUM 方案中很多内容需要测试人员编码实现，对测试人员水平要求较高。

2) SCRUM 方案只适用于采用 SCRUM 开发方法开发的软件的测试。

3)需要大量的人力对测试用例库、自动化测试用例库、数据库进行升级和维护。

第六章 总结与展望

6.1 本文总结

对于软件来讲，不论采用什么技术和什么方法，软件中仍然会有错。采用先进的开发方法、完善的测试过程，可以减少错误的引入。测试是所有工程学科的基本组成单元，是软件开发的重要部分，在软件生命周期中占据了重要的地位，越来越被软件开发者重视。一段时间以来，软件测试理论研究得到了一定发展。但相对一般软件开发方法而言，SCRUM 敏捷开发方法刚刚兴起，受其自身特点限制，基于 SCRUM 方法的软件测试还有待进一步探讨，并期望着能够在业界得到更多的实际应用。

作者在本论文中的主要工作有：

1）致力于 SCRUM 测试模型的研究，通过分析现有的测试方法、测试管理方案、测试技术提出了一套基于 SCRUM 软件开发的测试方案。

2）在 IBM WCS 产品项目的实际 SCRUM 开发测试过程中，为 Catalog 组件的 SCRUM 测试，设计了测试计划和测试用例，参与了测试计划预审会议，并实现和完成了一套自动化测试用例。本用例已经通过了 WCS 产品的最终测试，并已经应用到 WCS 下一个版本的测试项目中。

3）对 WCS 项目采用 SCRUM 测试过程后的测试结果进行分析和比较，探讨 SCRUM 测试方案的可行性，有效性，并总结 SCRUM 测试方案的优缺点。

4）对项目测试中 FVT 和 SVT 遇到的问题进行了深入的分析研究，并给予解决方案或参考。

论文的研究应用于 IBM 的 WCS 项目的软件测试。

6.2 今后的研究方向

软件测试是一个不断发展的领域，包含的内容非常广、非常新。对于本文所提出的基于 SCRUM 软件开发的测试模型，还需要进一步的深入研究，从而将其更好的应用于新的产品的测试中，保证软件产品的质量。

1.通过对 WCS 测试结果的分析，论文认为无论是基于 SCRUM 方法的测试还是其他软件开发过程中的测试，自动化测试依然是一个值得深入研究的问题。

要成功地实现软件测试自动化，以下工作有待进一步完成：

1)自动化对稳定的应用进行的测试

在对某一个应用的自动化测试之前，首先应该确定该应用是否稳定。对一个在将来可能发生变化的应用的测试进行自动化是没有必要的，因为应用一旦改变，相应的自动测试代码就要随之改动，所以应该只自动化稳定应用的测试。

2)自动化重复性测试

如果一个测试经常重复使用，并且使用这个测试不方便，那么就应该考虑自动化这个测试。

3)自动化已经实现的手工测试用例

在对软件测试自动化前，通常已经实现了很多的详细的手工测试用例，从中选择可以自动化的手工测试用例自动化。

4)合理限制自动化的范围

如前所述，100%的自动化并不是追求的目标。过大追求自动化的范围只会适得其反。软件测试自动化的开发人员应该在一个合理的可以进行自动化的范围内投入精力，在能力许可的情况下，再逐步扩大测试自动化的范围。

2.通过 WCS 项目对 SCRUM 开发过程的实践，对于软件的开发过程中测试的具体应用还有以下问题需要进一步的研究：

1）如果在紧张的 SCRUM 开发过程中，更好的分配各种资源，最有效的完成产品的测试。

2）压力测试目前在 SCRUM 开发过程中还有很多问题要解决，如何更好的将压力测试融入到整个 SCRUM 测试过程中。

3）由于时间有限，本次没有对其他的一些测试方法，例如 BVT, IVT, MVT 等测试方法在 SCRUM 过程中的应用进行研究。

参考文献

[1] Frederick P. Brooks Jr. The Mythical MAN-MONTH. 中国电力出版社，2003.
[2] 朱少民. 软件测试和质量保证的关系. 2006
[3] 刘孟任等译. 能力成熟度模型(CMM)：软件过程改进指南. 电子工业出[]版社.2001
[4] 崔启亮，胡一鸣. 国际化软件测试. 电子工业出版社. 2006
[5] Robert C. Martin. 敏捷软件开发：原则、模式与实践. 清华大学出版社. 2003
[6] Ken Schwaber. Agile Project Management with SCRUM. America: Microsoft Press. February 2005
[7] 钱乐秋，张敬周，朱三元. Agile方法研究综述. http://www.iturls.com/Articles/doc/Agile Summarize.pdf. 2005
[8] Glenford J. Myers. 软件测试艺术(原书第2版). 机械工业出版社. 2005
[9] Hetzel Bill. Complete Guide to Software Testing 2nd ED. New York: John Wiley & Son. 1993
[10] Rick D. Craig, Stefan P. Jaskiel. 系统的软件测试. 电子工业出版社. 2003
[11] Louise Tamres. 软件测试入门. 人民邮电出版社. 2004
[12] International Business Machines. 软件测试自动化技-IBM Rational 技术白皮书(V1.0). America: IBM Press. November 2005
[13] 陈宏刚. 软件开发的科学和艺术之软件测试. 电子工业出版社. 2004
[14] 郑人杰. 计算机软件测试技术. 清华大学出版社. 1992
[15] 刘胜. 软件测试及其自动化. 信息技术2001年第3期
[16] Ken Sehwaber' s SCRUM web Page. http://www.controlchaos.com/. 2006
[17] Jeff Sutherland' s SCRUM Web Page. http://www.jeffsutherland.org/scrum/index.html. 2006
[18] Ken Schwaber. Controlled Chaos: Living on the Edge. American Programmer. April 1996
[19] Mountain Goat Software web site. http://www.mountaingoatsoftware.com/scrum/index.php. 2006
[20] IBM.SCRUM meets RUP. http://www-128.ibm.eom/developerworks/rational/feb05/krebs. 2005
[21] Shari Lawrence Pfleeger. 软件工程理论与实践. 清华大学出版社. 2003
[22] Jerry. 研发过程管理工作规范. http://developer.51cto.com/art/ 200610/33724.htm. 2006
[23] 陈绍英，夏海涛，金成姬. Web性能测试实战. 电子工业出版社. 2006
[24] Microsoft. 全球化和本地化的测试相关资料. 2004
[25] IBM WebSphere Commerce Server产品在线帮助说明

附录 名词解释

[1] WCS：(WebSphere Commerce Server) IBM的针对电子商务的方案解决产品。

[2] FVT：(Functionality Verification Testing) 功能测试。

[3] SVT：(Stress Verification Testing) 压力测试。

[4] SOA：(service-oriented architecture) 面向服务架构

致谢

5年的工程硕士学习就要结束了，回顾这几年的学习与实践，许许多多的老师、同事和同学都给了我非常大的帮助，使我不断地成长和提高，在此我要向所有这些老师、同事和同学表示最衷心的感谢。

首先要感谢的是我的指导老师杨文川教授，对于我的问题，杨老师总能在百忙之中给予指导，引导我把握课题的研究方向，大大增强了分析问题、解决问题的能力，为今后的发展打下了良好的基础。本论文是在他的精心指导下完成的，从论文体系的构思、论文观点的提炼，从初稿的写作和修改直至定稿，每一步都倾注了杨老师大量的心血，从而保证了论文的顺利完成。杨老师严谨的治学风范和忘我的工作精神深刻地影响着我，并使我终身受益，在此对杨老师表示崇高的敬意和真挚的感谢！

此外，我还要感谢我的同事，在项目的实施中与我并肩战斗，共同去攻克技术难关、完成琐碎的实现细节，使我得以将理论研究与项目实践较好地结合起来，也是我论文得以顺利完成的保证。

由于本人水平有限，对一些问题的看法尚显肤浅，文中难免有许多不足、缺陷乃至错误，请各位老师、读者批评指正。