软件调试问题研究

姜 文,刘立康
(西安电子科技大学 通信工程学院,陕西 西安 710071)

摘 要:对于软件开发,软件调试是非常重要的环节。尤其是对于多个开发部门一起进行开发的大型软件系统,如何正确制定调试策略,进行软件缺陷定位、诊断和纠错是一项十分重要的工作。结合软件开发的基本概念和过程,叙述了软件调试的概念和软件调试的基本过程。根据软件规模,应用软件开发可以分为大、中、小三种模式;针对三种软件开发模式,分别介绍了软件开发的角色,给出了用例图,归纳分析了开发过程中软件调试的特点。之后归纳出了软件开发过程中有效调试的先决条件、软件调试原则和策略、大型软件调试技术。最后介绍了三个典型的软件调试案例。工作实践表明,深入研究软件调试的理论与技术,有助于提高软件开发效率和软件质量,更好地满足客户对软件产品的需求。

关键词:软件测试;软件调试;软件配置管理;持续集成;基线
中图分类号:TP391.41 **文献标识码**:A **文章编号**:1673-629X(2017)11-0001-06
doi:10.3969/j.issn.1673-629X.2017.11.001

Research on Software Debugging

JIANG Wen,LIU Li-kang
(School of Telecommunication Engineering,Xidian University,Xi'an 710071,China)

Abstract:Software debugging is very important for software development. Especially for large software system developed by many development departments together,how to make correct debugging strategy for fixing,diagnosing and recovering of the software defect is very important work. In combination with the basic concept and process of the software development,the concept and the basic process of software debugging are described. According to the scale of software,the application software can be divided into three types,the large,the middle and the small. For them,the roles in software developing are introduced,the case charts are given respectively,and the features of software debugging during developing are inducted and analyzed. Then the preconditions of effective debugging,the principle and strategy of software debugging,the debugging technique for large software are concluded during the software development. At last,three typical cases of software debugging are introduced. Practice shows that deep research of the technique and theory of software debugging is contributed to improve the developing efficiency and quality of software and make the customer satisfied the demand of software better.

Key words:software testing;software debugging;software configuration management;continuous integration;baseline

0 引 言

在程序开发过程中,一个公认的事实是编写代码并不难,但是排除代码中的错误,却不是一件容易的事情。排除代码中的错误,涉及到软件调试。目前软件调试缺乏实用强大的调试支撑技术和辅助工具,主要以手工方式为主,具有很强的技巧性。随着计算机技术的发展,需要进一步研究软件调试的理论与技术。

1 软件开发过程中的基本概念

软件开发工作内容大致可分为五种:软件设计、程序编码、程序构建、软件测试和软件调试。

1.1 软件设计

给出可行的设计方案,将客户的需求转换成可以实现的设计规格文档。完成需求设计、架构设计和详细设计。将设计规格文档提供给软件开发人员进行软件源代码编码。

1.2 程序编码

根据设计规格文档,软件开发人员进行程序编码。

1.3 程序构建

程序构建通常包括对程序代码进行编译和链接,

收稿日期:2016-12-10 **修回日期**:2017-04-13 **网络出版时间**:2017-08-01
基金项目:国家部委基础科研计划:国防预研基金项目(A1120110007)
作者简介:姜 文(1986-),女,工程师,硕士,CCF 会员(E200032324M),研究方向为图像处理、模式识别、数据库应用和软件工程等;刘立康,副教授,研究方向为数字通信、图像处理与传输等。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20170801.1555.058.html

从而生成可执行文件。

(1)程序编译。

编译器的基本功能就是将一种语言编写的程序(源代码)翻译成用另一种语言表示的等价程序(目标代码)。编译器的出现为编程语言的繁荣,软件产业的形成奠定了基础。

(2)程序链接。

链接是将编译器产生的多个目标文件合成为一个可以在目标平台上执行的软件模块。

1.4　软件测试

1.4.1　软件测试定义

迄今为止,软件测试[1-2]没有一个公认的准确定义,以下是两个具有代表性的定义。IEEE 给出的定义:使用人工和自动手段来运行或测试某个系统的过程,其目的在于检验它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。该定义明确提出了软件测试以检验是否满足需求为目标;著名的软件工程专家 Glenford J. Myers 给出的定义:测试是为了发现程序中的错误而执行程序的过程。该定义强调测试的目的是检测软件中存在的错误。

1.4.2　静态测试

静态测试[3]是一种不通过执行程序而进行测试的技术。起初程序开发人员通过编译器来检查程序代码的各种语法错误,现在通常采用静态检测工具检查代码的各种错误。静态检测工具通常具有编译器的功能。这种测试能发现 60% 以上的错误。

1.4.3　动态测试

动态测试是通过执行程序,测试一个系统或组件的过程。程序在受控环境下运行,通过运行结果和特定期望结果进行对比,确定软件程序在检查状态下是否正确。此种测试通常发现 40% 的错误。

2　软件调试

2.1　软件调试的概念

软件调试[2,4-7]是泛指重现软件缺陷问题,定位和查找问题根源,最终解决问题的过程。

软件调试通常有如下两种不同的定义:

定义 1:软件调试是为了发现并排除软件程序中的错误,可以通过某种方法控制被调试程序的执行过程,以便随时查看和修改被调试程序执行状态的方法。

在该定义中,软件测试属于软件调试的一部分,与牛津词典中的调试定义类似。在牛津词典中调试定义为:“识别和排除计算机硬件或软件中错误的过程。”

定义 2:调试是执行一次成功的测试之后所要进行的工作。所谓成功的测试,是指它可以证明程序没有实现预期的功能。调试包含两个步骤,从执行了一个成功测试用例,发现问题后开始;第一步,确定程序中可疑错误的准确性质和位置;第二步,修改错误。

在该定义中软件测试从调试工作中分离出来。

2.2　软件调试的基本过程

按照定义 2,软件系统调试的基本过程如下:

(1)重现问题:重现软件测试发现的问题;

(2)问题定位:确定可能发生问题的程序段位置;

(3)查找原因:分析相关代码,确定导致缺陷问题的内在原因;

(4)设计方案:提出软件缺陷问题解决方案;

(5)修改代码:根据设计方案修改程序代码;

(6)验证和确认:采用审查、分析和测试等技术来确定错误是否被排除,是否引入了新的错误。

上述 6 个步骤不断迭代进行,直至问题解决。

软件调试基本过程如图 1 所示。

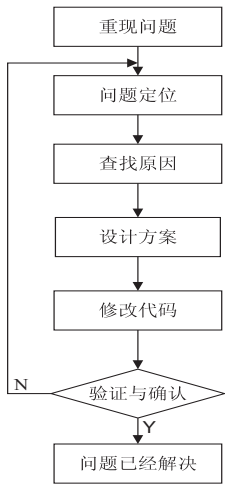

图 1　软件调试的基本过程

在这些步骤中,问题定位和查找原因是软件调试的关键环节,其工作量约占总工作量的 90% 以上。软件调试是一项既耗时又费力,同时又富有技巧性的工作。目前软件调试中的问题定位[8-9]研究的比较多。

2.3　软件调试和版本管理的关系

软件调试和版本管理[10-11]的关系非常密切,通常采用软件配置管理工具进行版本管理。

(1)在软件调试过程中可能有多种算法都可达到预期的目标,但只能选择其中一种,这时需要保留各种有价值的算法版本;软件调试完成后,需要进行代码优化,在代码优化的过程中需要保留各种不同的版本;软件调试完成后,需要增加功能和提升性能,在此基础上开展下一步调试工作,需要保留调试好的软件版本。

(2)在多人开发同一个软件系统的过程中,需要通过版本管理调试解决代码冲突问题。

(3)软件产品实际上是某个版本的软件产品,从某种意义上来讲,软件产品打补丁和开发软件的新版本是更高层次的软件开发调试工作。

3　应用软件开发模式

根据软件代码规模,应用软件的开发大致分为三种模式。

3.1　程序员个人开发的小软件

3.1.1　用例图

这种模式和早期的软件开发模式类似。小软件开发用例图如图 2 所示。

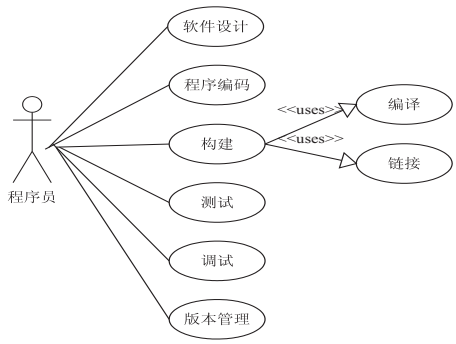

图 2　程序员开发的小软件用例图

3.1.2　软件调试的特点

发现问题(测试)、定位问题和提出解决问题方案、修改程序代码、验证全部由程序员负责。软件调试可以分为静态调试与动态调试。

1)静态调试。

源程序代码编译时同时对源代码进行静态检查,编译器提供了源代码各种编程错误和错误所在的位置。静态调试就是程序员逐条修改编译器提示的错误,通过代码编译这一关。

2)动态调试。

动态调试分为查错和纠错。查错就是对程序进行功能性能测试,查找各种不符合设计要求的各种问题;纠错就是根据发现的问题,查找原因,修改程序源代码。这里的软件调试工作包括软件测试。动态调试通常采用以下两种方式:

(1)仔细分析发现的问题,通过推理来查找发生问题的原因。程序员对程序的架构设计、编码实现十分熟悉,往往能比较快地定位和处理问题。这种方式通常具有全局观念,可以避免解决问题过程中诱导出现其他问题。

(2)通过调试工具采用人机交互方式调试代码。这种方式是逐条执行和跟踪程序代码,观察各种状态和变量的变化,检查是否符合程序设计的要求来定位问题。这种方式有助于查找程序代码的微观错误,要求程序员对程序代码的实现十分熟悉。

这两种方式是互补的,综合应用调试程序代码。

3)版本管理。

版本管理通常采用小型软件配置管理工具 VSS,也可采用文件存储的方式。

3.2　程序组软件开发

这种模式与软件开发中期的开发模式类似。通常软件分为多个软件模块,每个程序员仅负责自己开发的软件模块。这种开发模式通常用于中、小型软件的开发。

3.2.1　角色和用例图

软件设计人员:负责软件设计,提供设计规格文档。

程序开发组:程序代码编写和程序调试,负责软件的版本管理和集成构建。

测试人员:负责软件功能、性能测试。

程序组开发的软件用例图如图 3 所示。

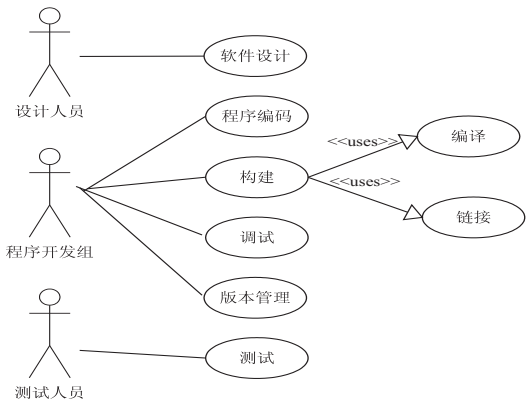

图 3　程序组开发的软件(中、小型软件)用例图

3.2.2　软件调试的特点

1)软件的设计工作和大部分测试工作从程序组工作中分离出去。设计人员负责软件设计,程序员负责程序代码的实现,定位问题和提出解决问题方案往往由设计人员和程序组共同合作处理,程序员负责软件纠错(程序代码修改),测试人员负责测试工作。

2)调试分为两个阶段:

(1)开发组自己测试软件。

程序员完成程序源代码的编写,程序代码的静态检查,使用调试工具对程序代码进行功能点的调试。所有的功能点都调试完成后,通过组内代码评审之后,将源代码合入版本库。

开发组组长指定某个程序员负责程序代码的集成构建,编译过程中发现的问题,反馈给相关的程序员进行处理。源代码完成集成构建之后,打包提交给测试人员测试。

(2)测试人员测试软件。

测试人员根据设计规格文档设计测试用例,测试提交过来的软件包。测试人员发现的各种问题反馈给程序开发组进行软件调试处理。程序开发组和设计人员确定发生问题的原因,确定修改方案,分配给相关的程序员进行代码纠错处理。对于比较复杂的问题,软件设计人员需要提供实现编码的设计文档。程序代码

修改后,进行验证和确认。

3)版本管理通常采用软件配置管理工具 SVN。通过版本管理工具对程序员提交的代码进行冲突检查,通过调试处理保证代码的兼容性和一致性。

3.3　项目组开发的软件

软件通常由多个模块组成,每个模块由若干开发单元组成。开发单元分配给程序员编写程序代码。这种开发模式通常用于大型软件的开发。

3.3.1　角色和用例图

(1)软件设计组:提供总体和各模块的设计规格文档。

(2)软件开发组:按模块分为开发小组,开发小组将开发单元分配给程序员进行程序编码。

(3)软件配置管理员:负责基线和版本库的管理。

(4)持续集成工程师:负责软件的持续集成[10-12]工作,搭建的集成构建工程,通过制定定时任务来自动完成从版本库更新代码、静态检查、编译、出包、冒烟测试等任务。冒烟测试也称为预测试,对集成构建成功的软件包的主要功能进行快速自动化测试。构建成功,可以获得最新 Build 版本,建立新的编码基线。持续集成工程师进行全量构建生成内部转测试版本,提交测试组进行的测试工作。

(5)软件测试组:对软件转测试版本进行功能、性能测试,通过后产生测试(Tested)基线。为持续集成工作提供进行冒烟测试的自动化测试用例脚本包,搭建相应的测试环境。

项目组开发软件(通常为大型软件)用例图如图 4 所示。

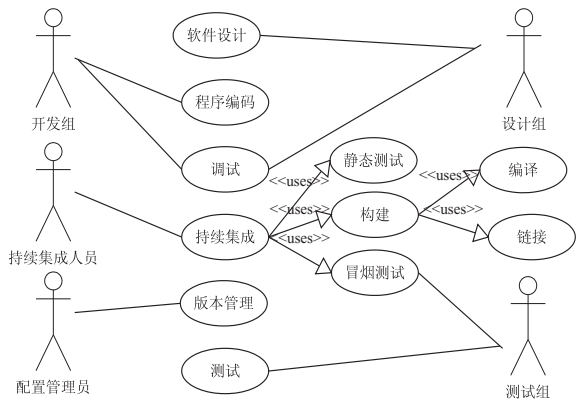

图 4　项目组开发软件(大型软件)用例图

3.3.2　软件调试的特点

1)软件设计人员和软件测试人员增加了,有的软件项目测试人员比开发人员还要多。软件测试不仅要发现程序编码中的问题,而且测试软件设计中的问题。设计中的问题自然由设计人员处理,程序编码中的问题由设计人员和开发人员共同处理。

2)软件的版本管理和集成构建工作由专人负责,实行基线和版本库的管理。基线管理[13-14]为全体开发人员提供统一的开发基点,统一的程序接口。通过控制集成构建的频率,有助于及时发现程序代码问题。

3)软件调试分为三个阶段:

(1)开发人员调试自己开发软件单元。

程序开发人员每天从版本库检出需要的文件,放在本地作为工作副本开始工作。在工作副本上进行查看、修改、编译、运行、调试等操作。为了提供高质量的代码,需要对编写好的代码进行单元测试,静态走码检查,冲突处理和本地构建工作,处理发现的各种问题。最后将评审过的代码提交到版本库。开发人员向版本库提交时,要添加注释、说明、CR 单号、修改原因等,以保证可追溯。

(2)处理持续集成工程师发现的问题。

通常持续集成工作包括静态测试、编译、链接和冒烟测试,每一步发现问题都要反馈给相关开发人员处理,直到通过集成构建。

(3)处理测试组发现的问题。

测试组测试转测试版本,将测试结果反馈给相关人员,对存在的问题逐一定位,查找原因,修改程序,对每个软件缺陷问题进行跟踪管理,直到问题解决为止。

4)设计人员和程序员共同处理反馈的各种问题,定位问题和提出解决问题方案。程序员负责程序代码修改,测试人员负责验证和确认工作。

5)版本管理可以选择 SVN、ClearCase、Git 等软件配置管理工具。通过版本管理工具进行软件的版本管理、基线管理和代码冲突检查。

6)软件开发过程中采用持续集成和基线管理技术,可以更有效地进行开发和调试工作。

4　软件调试的方法与技术

4.1　软件有效调试的先决条件

开展调试工作前要做好必要的准备工作。

1)了解设计和算法。

调试一个软件模块,需要了解它的设计和实现算法,了解各个函数之间的调用关系,该模块与其他模块之间的接口关系。

2)高质量的程序代码。

程序开发者应该提供高质量的程序代码,包括规范的代码和必要的注释,对开发的代码进行单元测试,经过同行严格的代码评审。一个复杂的模块,代码被分成许多子程序,其中大部分不超过 10 至 15 行。

3)熟悉软件运行环境。

随着网络技术、云技术、大数据技术的发展,软件运行环境越来越复杂,调试者只有熟悉这些环境和环境配置,才能实现在线软件调试。

4)熟悉调试工具。

以 Linux 环境下 C/C++程序为例,调试者需要熟悉 GCC 编译器和 GDB 调试工具。

5)动态软件调试的条件。

动态调试软件模块需要正确配置调试环境,实现两个功能:可以调用模块所提供的功能;能够得到调用结果的信息,模块内部状态的变化,当一个错误发生时该模块的动态。

6)软件开发过程的管理。

对于大型软件项目调试,管理工作十分重要。

(1)程序代码和文档管理:在软件开发过程中,底层模块和被多个开发者调用的函数不能修改,对其更新要经过严格的程序和审定。完善的软件文档可以为软件调试提供有用的信息。

(2)人员管理:软件调试过程中,需要查找模块的设计人员和开发人员,需要他们参与调试工作或者为调试工作提供帮助,因为他们对相关的模块和程序段最熟悉。

4.2　软件调试原则和策略

(1)熟悉软件设计和编码。

让熟悉软件设计和编码的人参与调试工作,修改错误也是程序设计的一种形式。在程序设计阶段使用的方法都可以应用到程序错误的修正工作中。

(2)从软件模块的最小集成包开始。

在增量式软件开发过程中,软件模块最初的起始可能是一个最低功能限度的集成包,随后新的代码不断加入到系统中。调试工作可以从最小的集成包开始,不断增加代码和模块来查找、定位问题。

(3)分而治之。

每次只处理一个问题,把被调试组件从其上下文组件之中孤立出来,通过设计驱动模块和桩模块进行调试。

(4)发现问题及时反馈和处理。

检测到的错误越早,就越容易找到原因。如果等到问题症状出现在客户端接口,那么可能很难缩小发生问题的原因列表。

(5)兼顾全局。

程序代码错误修改兼顾全局,确保修改错误没有影响软件的其他部分。

(6)彻底修改。

如果提出的修改方案不能解释与该错误有关的全部线索,那就表明只修改了错误的一部分,必须对错误进行彻底修改。

(7)关注相关问题。

在查找问题根源时,对可能发现的一些相关问题也要做处理。暂时不能处理的相关问题应该记录在案,为以后的调试工作保留相关信息。

(8)自顶向下的方法。

通常采用自顶向下或模块化方法来编写代码。采用自顶向下或模块化的方法来调试代码,可以缩小软件问题定位的范围,提高调试效率。

4.3　大型软件调试技术

大部分文献资料都是介绍传统的软件调试方法,主要关注小型软件或者软件模块中组件的调试方法。文中介绍一些大型软件开发过程的调试技术。

1)分析和推理。

设计人员和开发人员根据软件缺陷问题的信息,分析和推理调试软件。

(1)根据软件程序架构自顶向下缩小定位范围,确定可能发生问题的软件组件。

(2)根据软件功能,软件运行时序定位软件问题。

(3)根据算法原理,分析和确定缺陷问题发生的根源。

2)归纳类比法。

该方法主要是根据积累的工作经验和案例处理调试工作。

(1)根据工作经验和比对程序设计中类似问题的处理方式进行调试工作。

(2)咨询相关部门和有经验的相关人员。

(3)查找相关文档和案例,为处理问题提供思路和方法。在大型软件开发过程中,通常对每个缺陷问题进行跟踪管理,将解决问题的方案和过程详细记录。

3)跟踪回朔。

大型软件开发通常采用基线与版本管理。基线为程序代码开发提供统一的开发基点,基线的建立有助于分清楚各个阶段存在的问题,便于对缺陷问题定位。软件版本在软件产品的开发过程中生成了一个版本树。软件产品实际上是某个软件版本,新产品的开发通常是在某个软件版本的基础上进行开发。

(1)开发过程中发现有问题,可以回退至版本树上的稳定版本,查找问题根源。

(2)通过基线版本序列可以追踪产品的各种问题,可以重新建立基于某个版本的配置,可以重现软件开发过程中的软件缺陷和各种问题,进行定位并查找问题根源。

4)增量调试。

大型软件开发大多采用软件配置管理和持续集成技术。开发人员每天将评代码提交到版本库。持续集成人员完成集成构建工作。可以通过控制持续集成的粒度(构建时间间隔),控制开发人员提交到版本库的程序代码量,从而便于对缺陷问题定位。通常每天晚上进行持续集成工作,发现问题时,开发人员实际上只

需要调试处理当天编写的代码。

5)写出能重现问题的最短代码。

采用程序切片和插桩技术写出能重现问题的最短代码调试软件模块。

(1)程序切片[15]。

程序切片是通过在特定位置消除那些不影响表达式计算的所有语句,把程序减少到最小化形式,并仍能产生给定的行为。使用切片技术,可以把一个规模较大并且较复杂的软件模块转换成多个切片程序。这些切片程序相对原来的程序,简单并且易于调试和测试。

(2)程序插桩。

程序插桩方法是在被测程序中插入某些语句或者程序段来获取各种信息。通过这些信息进一步了解执行过程中程序的一些动态特性。一个软件组件的独立调试和测试需要采用插桩技术,该组件调用或运行需要桩模块。

在软件模块的调试过程中程序切片和程序插桩可以结合起来使用。

6)日志追踪技术。

日志是一种记录机制,软件模块持续集成构建过程中,日志文件记录了有用信息。若构建失败,通过查看日志文件,将信息反馈给相关人员进行软件调试。

7)调试和测试融合的技术。

(1)测试驱动开发[16]。

测试驱动开发是一种不同于传统软件开发流程的开发方法。在编写某个功能的代码之前先编写测试代码,然后编写测试通过的功能代码,这有助于编写简洁可用和高质量的代码。

(2)开发与测试融合。

程序开发人员除了进行程序代码的编写,白盒测试,也要完成基本的功能测试设计和执行。这样有助于程序开发人员更好地开展调试工作。程序开发人员可以通过交叉测试来解决测试心理学的问题(不能自己测试自己)。

采用这种模式测试人员的数量会减少,专业的测试人员去做其他复杂的测试工作。研发中的很多低级缺陷会尽早在开发过程中被发现,从而减少缺陷后期发现的成本。

5 典型案例

5.1 手写体维文字符识别算法研究

该软件程序由研究人员个人开发和调试。软件注重算法的实现,研究人员在设计完成算法设计后,采用C++语言进行编程和跟踪调试。

5.2 工具软件的开发

在软件产品开发过程中,需要开发各种专用工具,这些工作通常由工具组来完成。工具组的开发模式类似于3.2节的中、小型软件开发。

5.3 大型软件产品开发

某公司的一个大型软件开发项目,涉及50多人的软硬件设计、开发和测试工作。软件开发过程中使用的软件配置管理工具为SVN,持续集成工具为ICP-CI。开发模式类似于3.3节的大型软件开发。

6 结束语

软件调试作为软件开发的重要组成部分,对于大型软件系统,不仅需要做好软件代码的调试工作,也要做好软件模块集成的调试工作。需要开展大量的研究工作,从而提高软件产品的开发效率,提高软件的质量,以及软件的安全性与稳定性,更好地满足客户对软件产品的需求。

参考文献:

[1] Myers G J, Badgett T, Sandler C. Art of software testing[M]. 3rd ed. [s. l.]: John Wiley & Sons, Inc. ,2012.

[2] 林科学. 软件测试/调试技术应用研究[D]. 南京:南京气象学院,2004.

[3] 姜 文,刘立康. 基于持续集成的PC-Lint静态检查[J]. 计算机技术与发展,2016,26(11):31-36.

[4] 张银奎. 软件调试[M]. 北京:电子工业出版社,2008.

[5] Metzger R C. 软件调试思想[M]. 尹晓峰,马振萍,译. 北京:电子工业出版社,2004.

[6] Telles M, Hsieh Y. The science of debugging[M]. [s. l.]: Coriolis Group,2001.

[7] 蔡 铭,程 胜,王 瑞. 航天型号高可靠软件系统调试原理与技术[M]. 北京:中国宇航出版社,2008.

[8] 曹鹤玲,姜淑娟,鞠小林. 软件错误定位研究综述[J]. 计算机科学,2014,41(2):1-6.

[9] Huang Linzhi, Ai Jun. Automatic software fault localization based on artificial bee colony[J]. 系统工程与电子技术:英文版,2015,26(6):1325-1332.

[10] 姜 文,刘立康. 基于ClearCase的软件配置管理与持续集成[J]. 计算机技术与发展,2016,26(1):10-17.

[11] 姜 文,刘立康. 基于SVN的软件配置管理和持续集成[J]. 电子设计工程,2016,24(2):1-5.

[12] Duvall P M, Matyas S, Glover A. 持续集成软件质量改进和风险降低之道[M]. 北京:电子工业出版社,2012.

[13] Tykal J. Best practices for using composite baselines in UCM [R]. [s. l.]: [s. n.],2004.

[14] 姜 文,刘立康. 软件配置管理中的基线问题研究[J]. 计算机技术与发展,2016,26(6):6-10.

[15] 李必信. 程序切片技术及其应用[M]. 北京:科学出版社,2006.

[16] Beck K. 测试驱动开发(注释版)[M]. 孙 方,译. 北京:电子工业出版社,2007.