基于持续集成的 C/C++软件覆盖率测试

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:覆盖率测试是一种白盒测试方法,软件代码的覆盖率指标是软件开发过程中重要的度量指标。覆盖率测试主要分为两部分:对程序代码进行插桩;编译插桩后的进程文件进行用例测试。首先介绍了 Linux 环境下 C/C++代码覆盖率测试的原理与流程,在此基础上叙述了基于持续集成的软件覆盖率测试原理与流程。详细叙述了覆盖率工程的实现,包括持续集成系统上进行插桩编译、集成构建、插桩数据预处理、HLT 测试检查和覆盖率数据处理。代码覆盖率报告提供了软件代码整体覆盖率与新增代码覆盖率的统计数据,反映了软件代码宏观覆盖率信息。测试过程中生成的 VBS 数据库提供了微观的覆盖率信息。最后介绍了一个典型工作案例,工作实践表明软件开发过程中做好代码覆盖率统计工作,可以有效监控和改进软件源代码的质量,提升软件开发和测试工作。

关键词:覆盖率测试;插桩技术;持续集成;构建;VBS
中图分类号:TP311.56 **文献标识码**:A **文章编号**:1673-629X(2018)03-0037-05
doi:10.3969/j.issn.1673-629X.2018.03.008

Code Coverage Test of C/C++ Software Based on Continuous Integration

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

Abstract:Coverage testing is a kind of white-box testing method. Software code coverage indicators is an important metrics in the development of software. Coverage testing is mainly divided into two parts:the instrumentation of program code;testing the module files after instrumentation with the test case. First we introduce the principle and process of C/C++ code coverage testing in the Linux,basd on which the principle and process of software coverage testing based on the continuous integration is specified. We describe the implementation of coverage project in detail,including instrumentation compiling on the continuous integration system,integration establishment,instrumentation data preprocessing,HLT test and coverage data processing. Code coverage report provides the statistics of overall software code coverage and new code coverage,which reflects the macro software code coverage information. The VBS database generated in the test process provides the micro coverage information. Finally introduction of a typical case,the practice shows that doing the statistics of code coverage well during the software development can effectively monitor and improve the quality of the software source code,and also promote software development and testing work.

Key words:coverage testing;instrumentation technology;continuous integration;building;validation before submission

0 引 言

软件测试[1-2]是保证软件质量和可靠性的重要手段。其中覆盖率测试就是监控软件代码覆盖率的一种有效的测试方法,通过一系列的测试集来找出哪些代码没有被执行到,统计程序各类语句执行的覆盖率,并对代码的执行路径覆盖范围进行评估、分析。

随着软件开发技术的不断发展,软件持续集成[3-5]技术已经成为大型软件开发过程中重要的组成部分。通过持续集成工具构建覆盖率工程可以方便地进行软件代码的覆盖率测试,及时提供软件开发过程中的覆盖率数据,从而有效监控和改进软件源代码的质量,改进和提升软件开发和测试工作。

1 代码覆盖率测试与软件插桩技术

1.1 代码覆盖率测试

覆盖率测试[6-10]是一种白盒测试方法。覆盖率度

收稿日期:2017-03-04 **修回日期**:2017-07-11 **网络出版时间**:2017-11-15
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与分析、数据库应用和软件工程;刘立康,副教授,研究方向为数字通信、图像传输与处理等。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20171115.1425.040.html

量指标有多种,包括语句覆盖、分支覆盖、函数覆盖、条件覆盖、分支/条件覆盖、路径覆盖等。其中语句覆盖是最基本的覆盖标准。覆盖率测试以程序内部的逻辑结构为基础,设计若干测试用例,在这些测试用例运行时,提取相关的覆盖率信息。

1.2 软件插桩技术

插桩技术[11-14]是在源程序的某些位置插入语句或程序段,但并不破坏程序的完整性的一种技术。在插桩过程中,插桩的位置根据程序的结构,测试的要求以及实现的测试目的进行设定。软件插桩技术能够根据实际需要,获取软件的各种信息。代码插桩是实现覆盖率测试的关键技术之一,如今大多数的覆盖率测试工具均采用代码插桩技术。

1.3 代码覆盖率测试的应用

代码覆盖率测试主要应用在两个方面:

(1)评估测试质量:根据软件的代码覆盖率检查报告,发现漏测场景,开发工程师分析之后给出改进建议,测试工程师补充新的测试点和新的测试用例;

(2)帮助识别冗余代码:开发工程师对代码覆盖率检查报告进行分析之后,发现冗余代码,对程序代码进行优化与重构。

2 Linux 环境下 C/C++代码覆盖率测试的原理与流程

2.1 C/C++代码覆盖率测试的原理

在 Linux 上的 C/C++软件开发一般使用 gcc/g++作为编译器。gcov 工具[15-16]是伴随着 gcc 发布的,是 gcc 附带的一个代码覆盖率分析工具,配合 gcc 共同实现对单个或多个 C/C++文件进行语句和分支覆盖率测试。

gcc 编译应用程序时加选项-fprofile -arcs 和-ftest -coverage。这样,在编译过程中,gcc 对每个 c/cpp 文件生成一个 *.gcno 数据文件,并且同时对应用程序进行插桩。运行插桩后的程序时,每个 c/cpp 程序的动态信息会保存在 *.gcda 数据文件中。

*.gcno 文件包含了基本块图和相应块的源码的行号信息。包含基本块的信息,代码行和块的映射关系。

*.gcda 文件包含了程序分支跳变的次数和其他概要信息(gcda 只能在程序运行完毕后产生)。包含代码行执行的情况以及覆盖率的信息归纳。

gcov 工具利用编译时生成的.gcno 数据文件以及运行时生成的.gcda 数据文件,得到程序的代码覆盖信息,并且以注释源代码的形式显示执行过和未执行过的代码。

2.2 程序代码覆盖率测试流程

gcc 和 gcov 进行应用程序覆盖率测试的流程如图 1 所示。

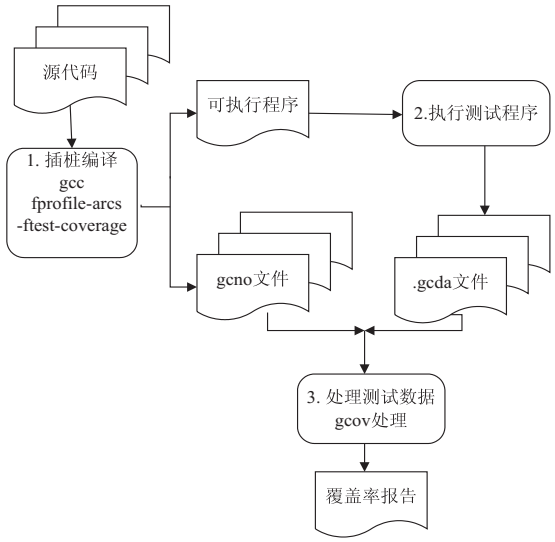

图 1 gcov 覆盖率测试流程

覆盖率数据产生的过程如下:

(1)插桩编译程序源代码;生成可执行文件和源代码文件相应的 *.gcno 文件。

(2)运行测试程序,生成源代码文件相应的 *.gcda 文件。

(3)使用 gcov 获取文本形式的覆盖率数据文本文件 *.gcov;也可以使用 lcov 工具获取 html 形式的覆盖率数据。

gcov 工具计算速度快,可以准确计算覆盖率。被测程序的编译插桩是由 gcc 完成,gcov 负责数据的分析和显示。但是 gcov 工具只能支持语句以及分支两种覆盖率测试功能,不适用于大型软件和复杂开发环境的覆盖率测试,不能给出软件开发过程中新增代码的覆盖率测试。

3 基于持续集成的软件覆盖率测试原理与流程

3.1 代码覆盖率测试原理

软件代码覆盖率测试通过持续集成工具,完成对软件各模块的插桩编译、版本包出包。在版本包测试过程中完成代码覆盖率数据采集,对相关数据进行加工处理后生成覆盖率报告。

覆盖率数据分为三种类型:

(1)当前软件代码的覆盖率。

通过覆盖率测试获得当前软件代码的覆盖率数据和各模块的覆盖率数据。这个数据反映了软件开发过程中程序代码总体的代码覆盖率状况。

(2)新增代码的覆盖率。

通过基线版本代码和当前版本软件代码的比较分析,得到新增与修改的代码的列表文件,结合测试过程中得到的覆盖率数据,获得新增代码的覆盖率。这个数据反映了软件开发过程中新增程序代码或修改了的程序代码覆盖率状况。

(3)生成 VBS(validation before submission)数据库。

VBS 数据库由测试用例覆盖软件模块代码行的数据组成,反映了软件模块代码行和测试用例的对应关系。

3.2　代码覆盖率测试流程

基于持续集成的覆盖率测试流程如图 2 所示。

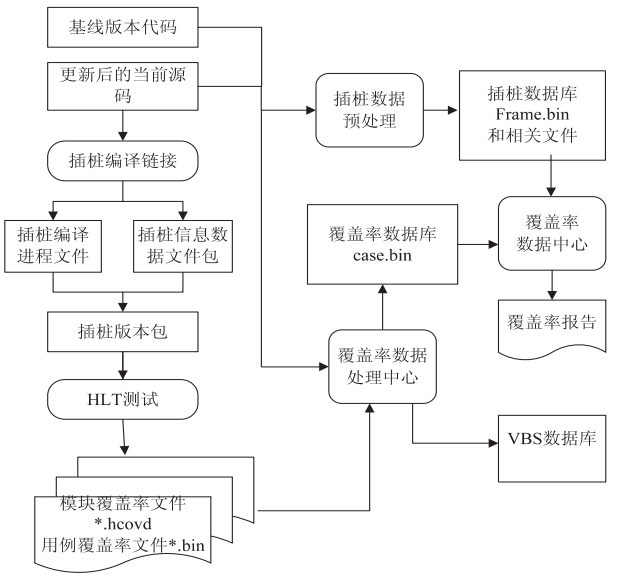

图 2　基于持续集成的覆盖率测试流程

(1)下载代码:从版本库下载基线版本代码和更新当前代码版本包。

(2)插桩编译链接:对当前版本代码进行插桩编译,生成插桩编译进程文件和插桩信息文件包,出插桩版本包。

(3)软件代码预处理:对基线版本代码、更新后的当前代码和插桩信息文件包进行预处理,生成插桩数据库 Frame. bin(里面包含了插桩文件、函数、有效行信息)和相关文件。

(4)HLT(HIGH LEVEL TEST)测试:在测试环境中对插桩版本包进行 HLT 测试,测试过程中收集相关数据,每个软件模块生成覆盖率数据文件为 *. hcovd;每个测试用例生成用例 ID 命名的 *. bin 覆盖率数据文件。

(5)处理覆盖率数据文件:将插桩信息文件包、覆盖率数据文件 *. hcovd 和用例的覆盖率文件 *. bin 提交覆盖率数据处理中心,软件模块的覆盖率数据文件 *. hcovd 汇总处理生成 Case. bin 数据库;用例的覆盖率文件 *. bin 汇总处理生成 VBS 数据库。

(6)生成覆盖率报告:将插桩数据库 Frame. bin 和相关文件提交覆盖率数据中心;将覆盖率数据库 Case. bin 提交覆盖率数据中心。覆盖率数据中心依据提交的相关数据生成可以阅读的覆盖率报告。

4　覆盖率工程的实现

工程采用软件配置管理工具 SVN 和持续集成工具 ICP-CI 开展持续集成工作。软件产品开发阶段每天都要把源代码合入版本库中,通过持续集成(CI)系统进行集成构建。通常每周进行一次覆盖率检查。覆盖率检查工程子系统关系图如图 3 所示。

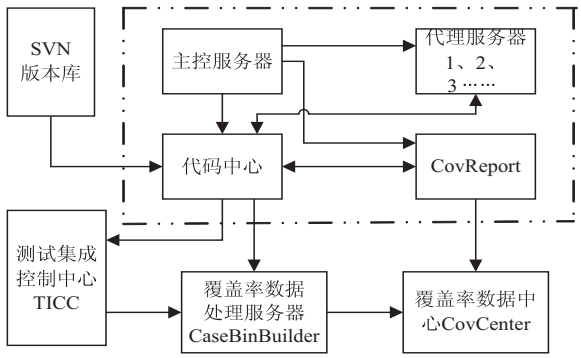

图 3　覆盖率检查工程子系统关系图

4.1　CI 系统上进行插桩编译和集成构建

大型软件系统通常采用分布式 CI 系统。CI 系统由主控服务器(CI Master)和代理服务器(CI Agent)组成,代码中心和 CovReport 均为特殊的代理服务器,如图 3 虚线方框所示。

4.1.1　搭建覆盖率检查工程

在主控服务器的 ICP-CI 工具页面上搭建覆盖率检查工程,工程命名为:“产品名_版本号_COV”,通过代理服务器标签将各种任务下发代理服务器执行。

4.1.2　代码更新和下载基线版本代码

代码中心登陆 SVN 版本库,完成当前代码更新和下载基线版本代码。

(1)当前代码更新。

编写代码更新的批处理脚本,并把代码更新的脚本配置在任务中,执行代码更新任务。对于配置管理工具 SVN,调用 SVN 的“update”命令更新代码中心的软件代码。

(2)下载基线版本包代码。

通常每周进行一次全量集成构建,生成转测试版本包,提交测试组进行全面测试,测试通过后产生测试(Tested)基线。覆盖率检查工程需要从版本库下载某个特定的测试基线版本代码,下载的版本代码的文件压缩包为 BaseCode. zip。

4.1.3　插桩编译

CI 系统的代理服务器从代码中心下载更新后的

当前代码压缩包 CurrentCode. zip,采用功能更为强大的插桩工具 Avatar 进行软件代码的插桩编译。

(1)makefile 文件的修改。

使用 Avatar 进行插桩编译时,需要链接 Avatar 的静态库:

-L /opt/Linux_avatar_64 - INCSCore_suse_10_x86-64;

插桩编译脚本增加插桩工具的初始化部分,清理之前插桩工程遗留的文件和目录,脚本内容如下:

```
if [ -e /opt/linux_avatar_64/HLLT_init. sh]; then
source /opt/linux_avatar_64/ HLLT_init. sh
rm - rf /opt/linux_avatar_64/llt
```

(2)插桩编译成功的判断标准。

完成插桩编译之后,如果模块的进程文件与插桩信息文件包都生成了,则表示插桩编译成功;如果仅有插桩信息文件包生成,则表示插桩编译没有成功,插桩信息文件包中的插桩信息文件名必定是乱码。

4.1.4 生成插桩版本包

插桩编译任务完成后,在 ICP-CI 工具的任务页面配置出包任务 package,将出包脚本配置到任务中。运行出包脚本,生成包含各模块的进程文件和插桩信息文件包的插桩版本包。代理服务器将插桩版本包提交到代码中心。

4.2 插桩数据预处理

在标签为 CovReport 的 CI 代理服务器上进行插桩数据预处理,从代码中心下载基线代码版本 BaseCode. zip、当前代码版本包 CurrentCode. zip 和插桩信息文件包。覆盖率工程需要基线版本代码作为统计新增代码覆盖率的参考基准。

插桩数据预处理过程如下:

(1)解压 BaseCode. zip、CurrentCode. zip 和插桩信息文件包,采用专用的统计分析工具处理相关文件。

(2)生成插桩数据库 Frame. bin,Frame. bin 文件合并了所有插桩编译生成的数据文件,给出插桩文件函数的代码行信息。数据记录了文件名、文件路径、函数名、函数起始行号、函数终止行号等信息。

(3)当前代码版本包中的非新增与修改的代码文件的信息写入相关文件;所有插桩编译模块的源文件名称与源文件路径写入相关文件。

4.3 HLT 测试检查

代码中心将插桩版本包提交到测试任务集成控制中心(test integration control center,TICC),TICC 自动完成测试用例管理、测试环境配置、测试版本安装与卸载、自动化测试用例执行等任务。

HLT 测试检查过程如下所示:

(1)在自动化工厂环境中卸载前一次的插桩版本包,安装新的插桩版本包,下发自动化测试用例连跑任务,并打开测试套。

(2)测试套在用例执行开始前清除前一次的覆盖率数据,用例执行结束之后,从执行环境上收集用例对应的覆盖率数据。将每个用例的覆盖数据整理,合并成一个用例 ID 命名的 *. bin 文件。

(3)执行测试结果回收整理,完成 HLT 测试检查后,会生成模块的覆盖率数据文件 *. hcovd。覆盖率数据文件的文件名称通常由以下格式组成:

模块名_模块链接的时间_插桩编译时间戳_代码覆盖率数据文件生成的时间_生成覆盖率文件的测试执行机 IP 地址. hcovd

4.4 覆盖率数据汇总处理

将软件模块的覆盖率数据文件 *. hcovd 和用例的覆盖率数据文件 *. bin 提交到覆盖数据处理服务器(标签为 CaseBinBuider),代码中心将插桩信息文件包也提交到覆盖率数据处理服务器。对覆盖率数据文件 *. hcovd 进行加工处理,生成覆盖率数据库 Case. bin。对用例的覆盖率数据文件 *. bin 加工处理,生成 VBS 数据库。

4.5 数据统计和出覆盖率报告

代码覆盖率数据中心(标签为 CovCenter)进行覆盖率数据统计工作。将插桩数据库 Frame. bin、覆盖率数据流 Case. bin 和其他相关的数据文件提交到 CovCenter,开展如下工作:

(1)代码覆盖中心将各种覆盖率信息写入 MYSQL 数据库。

(2)生成可以阅读的覆盖率报告,覆盖率结果文件为 Cover. xml。代码覆盖率报告提供了宏观的软件代码覆盖率信息。将这些数据提交公司质量部门,作为软件开发过程的度量指标之一,监测软件开发过程。

5 VBS 数据库

VBS 数据库提供了微观的覆盖率信息,提供代码轨迹与测试用例的对应关系。

5.1 VBS 数据库内容

在 HLT 检查过程中,执行测试结果回收整理的环境将每个用例的覆盖数据进行整理,合并成一个用例 ID 命名的 *. bin 文件。将这些文件汇总处理生成 VBS 数据库,数据库内容如表 1 所示。

表 1 VBS 数据库内容

TestCase	CodeLine	Codefile
TC1	323-329	sipApp. c
TC2	571-578	imgApp. c
……	……	……

数据库字段主要由 TestCase(用例名称)、CodeLine

(执行用例覆盖的代码行)和 Codefile(代码所在的文件名)组成。

5.2　VBS 数据库的功能

(1)程序代码、测试用例、功能特性之间的对应关系。

在修改程序代码时,通过 VBS 数据库筛选可以查到代码行相关联的测试用例;通过 TMSS 数据库可以查到测试用例相关联的软件功能特性,如图 4 所示。

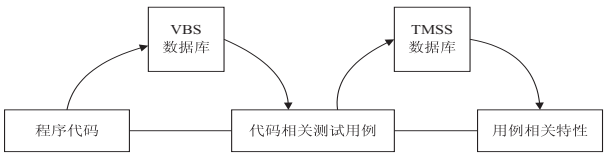

图 4　代码、测试用例、功能特性关联图

TMSS(test management service system)测试信息管理中心,负责测试用例、自动化测试脚本以及测试任务相关信息的管理。TMSS 中的测试用例信息库提供测试用例与软件功能特性之间的对应关系。

(2)帮助了解开发单元的功能特性。

长期的增量开发,功能越来越多,特性交互场景也越来越多,即使编一行代码也要花费很多时间去分析和梳理交互场景,对开发人员经验的要求逐渐提高。提交文件名和代码行号,查看影响本行代码的测试用例,查看本行代码涉及的功能特性、场景,可以帮助理解代码。

(3)方便问题单修改。

修改完问题单后,通过筛选给出影响代码的用例集进行回归测试,快速验证修改代码的兼容性。

(4)提高开发质量和效率。

将 VBS 筛选功能集成到研发桌面上,开发人员一键便可完成代码的影响分析,从而提高程序代码开发的效率和质量。

(5)提升测试用例。

测试人员通过对用例对应的代码和软件功能特性的对比分析,可以发现测试用例存在的一些问题,从而修改优化测试用例或者增加新的测试用例。

6　典型案例

某公司的一个软、硬件结合的大型开发项目,总的代码量超过两百万行。软件产品采用 C/C++进行软件开发。采用的配置管理工具为 SVN;持续集成工具为 ICP-CI。CI 系统由 1 台主控服务器、14 台代理服务器组成,1 台用作代码中心,2 台用于插桩数据预处理。每周进行一次覆盖率检查。表 2 给出其中一次检查的覆盖率数据。

在软件开发过程中,覆盖率测试主要关注新增代码的覆盖率情况,选择测试用例注重新增代码的测试,要求新增代码覆盖率大于 65%,对于整体的代码覆盖率,各软件产品根据产品自身情况灵活分析和处理。

(1)表 2 中软件产品版本包新增代码覆盖率数据大于 65%,满足度量指标要求。

(2)表 2 中模块 MK1、MK2、MK4 和 MK5 新增代码覆盖率数据大于 65%,满足度量指标要求。

(3)表 2 中模块 MK3 和 MK4 新增代码覆盖率数据低于 65%,需要根据模块的测试报告分析该模块未被测试用例覆盖到的场景,需要对未覆盖到的场景增加自动化测试用例。

表 2　软件产品版本包与版本中模块的代码覆盖率

版本包/模块名称		代码行数	代码覆盖行数	代码覆盖率/%	新增代码行数	新增代码覆盖行数	新增代码覆盖率/%
版本包	MKL 5.9.0	335 285	147 978	44.14	10 601	6 973	65.78
模块	MK1	4 022	2 383	59.25	220	156	70.91
	MK2	15 4948	68 159	43.99	7 261	4 860	66.93
	MK3	40 412	14 213	35.17	187	96	51.34
	MK4	62 326	25 708	41.25	201	171	85.07
	MK5	56 062	29 376	52.40	2 696	1 681	62.35
	MK6	17 488	8 139	46.54	36	9	25

7　结束语

工作实践表明,基于持续集成的代码覆盖率检查在软件开发过程可以发挥重要作用,有助于及时发现开发和测试过程存在的各种问题。根据代码覆盖率的检查报告分析测试场景,及时补充测试用例和优化代码。代码覆盖率作为软件度量指标,给软件开发过程管理提供了一项重要的评价指标。软件开发过程中做好代码覆盖率检查工作,有助于提高软件产品的质量,降低软件开发的成本。

参考文献:

[1]　MYERS G J,BADGETT T,SANDLER C. Art of software tes-

(下转第 46 页)

实导致了节点间的数据不一致,而实现了表广播机制的 MyCat,在异常节点恢复正常后,成功同步数据,保证与主节点的数据同步。

以上实验表明,实现表广播机制的 MyCat 的可靠性较高,在保证数据高效插入的同时,也保证了数据一致性。

4 结束语

提出了一种表广播机制在 MyCat 中的实现方案,通过对 MyCat 提供的操作全局表的相关功能进行优化,把对多个节点的操作修改为对单个节点的操作,有效解决了 MyCat 面临的弱 XA 问题。实验结果表明,该方案在实现全局表原有功能的同时,有效提高了全局数据表中各节点数据的一致性。

目前,对于分库表的查询操作,MyCat 只会返回各分库合并后的数据,在排序、分页等功能上做得还不够完善,因此,未来计划运用其他的方法解决这类问题。

参考文献:

[1] 庄天红. 常用数据库系统性能解析[J]. 微型电脑应用,1999,15(2):52-54.

[2] BOYD D, CRAWFORD K. Critical questions for big data [J]. Information Communication & Society, 2012, 15(5): 662-679.

[3] 项 凯. 面向海量高并发数据库中间件的研究与应用[D]. 上海:上海交通大学,2015.

[4] 邱 硕. Cobar 的架构与实践[J]. 程序员,2012(9):90-93.

[5] 祝雄锋. 数据库集群中间件 MySQL Proxy 研究与分析[D]. 武汉:武汉理工大学,2011.

[6] 王 葱. 基于 MyCAT 的分布式数据存储研究与应用[D]. 上海:东华大学,2016.

[7] HAYES B. Cloud computing[J]. Communications of the ACM,2008,51(7):9-11.

[8] 安延文. 数据库审计系统中 MySQL 协议的研究与解析[D]. 北京:华北电力大学,2016.

[9] 杨 晶,刘天时,马 刚. 分布式数据库数据分片与分配[J]. 现代电子技术,2006,29(18):119-121.

[10] 赵 艳,李 钧. 异构数据源分布式事务处理研究[J]. 计算机工程,2009,35(4):69-71.

[11] 胡百敬,陈俊宇,杨先民,等. SQL Server 2005 T-SQL 数据库设计[M]. 北京:电子工业出版社,2008.

[12] 黄雅萍,刘晓强,吴成义. 基于 MySQL 和 PHP 的分布式事务处理[J]. 东华大学学报:自然科学版,2011,37(1):81-85.

[13] 张旭刚,李东辉,俞 俊,等. 基于 zookeeper 和强一致性复制实现 MySQL 分布式数据库集群[J]. 微型电脑应用,2016,32(1):77-80.

[14] 徐 恪,刘亚霄,刘卫东. 数据库应用系统中的安全访问代理的设计与实现[J]. 计算机工程与应用,2000,36(1):105-107.

[15] 李曙强,蒋树春,吕 兵. 一种基于 mysql 数据库的 sql 信息采集审计系统:CN,CN103488797A[P]. 2013-10-14.

[16] 黄春华. 常用数据库的比较[J]. 科技信息,2011(14):200.

[17] 李占波,李 娜. XML 数据在关系数据库中的存储[J]. 微计算机信息,2007,23(27):192-194.

(上接第 41 页)

ting[M]. 3rd ed. Hoboken, New Jersey, U. S: John Wiley& Sons,2012.

[2] 赵 翀,孙 宁. 软件测试技术:基于案例的测试[M]. 北京:机械工业出版社,2013.

[3] 姜 文,刘立康. 基于 SVN 的应用软件持续集成[J]. 计算机测量与控制,2016,24(3):109-113.

[4] DUVALL P M, MATYAS S, GLOVE 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.

[5] 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.

[6] 于全喜. 嵌入式软件路径覆盖测试数据采集研究与实现[D]. 西安:西安理工大学,2009.

[7] DELAMARO M E, VINCENZI A M R, MALDONADO J C. A strategy to perform coverage testing of mobile applications[C]//Proceedings of the 2006 international workshop on automation of software test. New York, NY, USA: ACM, 2006:118-124.

[8] WOODWARD M R, HENNELL M A. On the relationship between two control-flow coverage criteria all JJ-paths MC-DC[J]. Information and Software Technology, 2006, 48(7): 433-440.

[9] MALEVRIS N, YATES D F. The collateral coverage of data flow criteria when branch testing[J]. Information and Software Technology, 2006, 48(8): 676-686.

[10] WU X, LI J J, WEISS D, et al. Coverage-based testing on embedded systems[C]//Proceedings of the 29th international conference on software engineering workshops. Minneapolis, Minnesota, US: IEEE, 2007: 204-209.

[11] 郭 锐,李 博,彭宝新. 用于覆盖测试的代码插桩程序设计与实现[J]. 科学技术与工程,2013,13(30):9072-9077.

[12] 张 荣,王曙燕. 基于插桩技术的动态测试研究与实现[J]. 现代电子技术,2011,34(4):50-52.

[13] 路 翠. 嵌入式软件白盒测试中插桩技术的研究与应用[D]. 北京:北京工业大学,2010.

[14] 王 伟. 基于多 Agent 的嵌入式软件测试系统研究与实现[D]. 南京:南京航空航天大学,2011.

[15] 王叔娥. 嵌入式软件动态分析技术的研究[D]. 成都:电子科技大学,2011.

[16] 朱 丽. 嵌入式软件动态测试平台的研究与实现[D]. 福州:福建师范大学,2013.