Science Discovery
2019; 7(3): 140-146
http://www.sciencepublishinggroup.com/j/sd
doi: 10.11648/j.sd.20190703.12
ISSN: 2331-0642 (Print); ISSN: 2331-0650 (Online)

A Method for Improving Test Coverage by Improving Test Cases

Hu Xiaoxi*, Shen Xiaohe, Niu Jing

Beijing Institute of Aerospace Control Devices, Beijing, China

Email address:
Huxx615@126.com (Hu Xiaoxi), shenxiaohe2008@163.com (Shen Xiaohe), ruomuzi608@163.com (Niu Jing)

*Corresponding author

To cite this article:
Hu Xiaoxi, Shen Xiaohe, Niu Jing. A Method for Improving Test Coverage by Improving Test Cases. *Science Discovery*. Vol. 7, No. 3, 2019, pp. 140-146. doi: 10.11648/j.sd.20190703.12

Received: April 20, 2019; **Accepted**: May 23, 2019; **Published**: June 15, 2019

Abstract: With the development of aerospace industry, the complexity of aerospace software is increasing, and the statue of software testing is becoming more and more important. The coverage test is an important method of testing the code logic defects. In order to meet the particularity of embedded aerospace software, this paper proposes a method to improve test coverage by improving test cases, which reducing the number of test cases and improving the efficiency of testing, thereby improving the effectiveness of test coverage. Based on five kinds of optimization types, the full text reduces the number of test cases and removes redundancy in order to improve the efficiency of test cases. While improving the test coverage, the method improves overall test efficiency, finally achieves the expected effect and reduces the cost of software development.

Keywords: Aerospace Software, Test Coverage, Test Cases

一种通过改进测试用例来提高测试覆盖率的方法

胡晓曦*，申小禾，牛静

北京航天控制仪器研究所，北京，中国

邮箱
Huxx615@126.com（胡晓曦），shenxiaohe2008@163.com（申小禾），ruomuzi608@163.com（牛静）

摘要：随着航天事业的发展，航天软件的复杂性与日俱增，软件测试的地位也愈加重要。覆盖率测试是检验代码逻辑缺陷的重要手段，为了满足嵌入式航天软件的特殊性，本文提出一种通过改进测试用例来提高测试覆盖率的方法，该方法通过优化测试用例，减少了测试用例设计的数量，提高测试用例的效率，从而达到提高测试覆盖率的效果。全文从优化测试用例着手，根据五种优化类型，降低测试用例数量，去除冗余，提高测试用例效率，在提高测试覆盖率的同时，也提高了整体测试效率，最后达到了预期效果，降低了软件开发的成本。

关键词：航天软件，测试覆盖率，测试用例

1. 引言

航天产品软件因其领域的特殊性，软件的质量和可靠性尤为关键，而软件测试是保证航天软件安全性的重要以及必要手段。其中，为了实时监控软件代码的覆盖率，如何实现有效的覆盖率测试成为航天软件测试中的重要问题。通过设计一系列的覆盖率测试用例，找出没有被执行到的代码，统计软件中各个模块、各种不同功能程序语句

执行的覆盖率，最后根据统计出的代码覆盖范围数据对整个程序进行分析。

随着计算机技术逐渐渗透到我们的生活中，软件开发技术也发展得愈加迅速，在大型软件开发过程中，软件持续集成技术得到了人们的重视。我们可以通过软件持续集成工具实现方便快捷的软件覆盖率测试，实现实时监测软件开发过程中的各类代码覆盖率变化，从而减少甚至避免了软件代码的漏洞和安全问题，行之有效地保证了航天软件的质量和可靠性。

针对航天系统软件的特殊性，统计监测代码覆盖率的方法和工具需要具备如下几个特点：

1）需要代码覆盖率统计模型，实现覆盖整个项目，并兼容项目中功能模块之间的交互作用；

2）代码插桩时，做到不影响软件运行，不影响实时结果，否则统计监测到的覆盖率情况不能反映软件的实际运行情况，没有分析评估的价值和意义；

3）支持多种编程语言及算法，支持多种编译器；

4）实现大量自动化插桩，可适用于航天软件的大批量源代码情况，且插桩具备高可读性，同时可实现人工修改功能。

基于以上航天软件代码覆盖率监测的特殊性，当前现有的测试覆盖率监测统计方法及工具仅仅差强人意，在很多方面还是无法满足航天软件的特殊要求，不能普及应用在大型项目中。随着航天事业的飞速发展，航天相关软件的规模和复杂性也成指数增长，而随着软件测试在软件工程中地位的稳步提升，软件测试成本在整个航天软件开发过程中也不断增加，价格昂贵且性能不能同航天软件匹配的测试覆盖率统计产品早已不能适用于实际航天软件工程中。

目前现有文献多用于寻求提高测试覆盖率的方法而忽略了兼顾测试效率的提高。文献[1]和文献[2]采用LDRA Testbed测试工具进行嵌入式软件覆盖率分析，在获得代码覆盖特征值后，自动分析被测系统在动态执行时的代码覆盖率情况；文献[3]和文献[4]针对程序中的不可达路径进行研究分析，给出了不可达路径检测方法；文献[5]用目标码来分析不可达代码；文献[6]通过二进制代码的静态与执行轨迹动态两种分析，评估模糊测试对目标程序二进制代码的测试覆盖；文献[7]用DFT设计测试覆盖率用例来改善测试效果；文献[8]用矩阵测试提高测试覆盖率，但是在很多情况现需要单独编写测试用例，否则测试代码很难分解。这些方法均只是侧重于提高覆盖率而忽视了测试效率的提高，甚至在某种程度上增加了测试的复杂度。因此，本文提出一种通过改进测试用例来提高测试覆盖率的方法，有助于动态监测并分析代码覆盖率，降低测试成本的同时提高测试效率，改进了实际航天软件工程中软件测试的流程。

2．软件覆盖率测试技术

2.1．覆盖率测试的概念

覆盖率测试，也称逻辑测试，是一种代码层面的测试。在覆盖率测试时，需要人工或机器对代码进行插桩操作，同时考虑整体程序和模块之间的依赖关系来执行测试用例。由于需要通过在软件代码中插桩执行覆盖率测试，因此覆盖率测试等同于执行所有代码，不仅如此，所有代码执行的同时，还要分析其中的分支及语句，考虑所有的逻辑结构，也因此，执行覆盖率测试可以发现软件代码中隐藏的缺陷、错误及问题。有时，为了保证软件测试的充分全面，要随时对测试手段进行调整修改，即补充测试。由于覆盖率测试插桩是直接面向软件代码的，所以还能够有助于开发人员发现其中的冗余代码，进而优化程序代码。

2.2．覆盖率测试的方法

如上述，覆盖率测试也是白盒测试的一种。代码覆盖率，意指已测试的代码在所有可测代码中所占的比例。在软件测试过程中，代码覆盖率越高，测试得越充分全面。但是针对不同类型的代码，要采取不同的手段和方法进行代码的覆盖率测试。目前，覆盖率测试常用的方法有语句覆盖、分支覆盖、条件覆盖、分支/条件覆盖以及路径覆盖等。语句覆盖，顾名思义，覆盖代码全部的语句，即所有的语句均可被执行到，这是软件覆盖率测试中最基本的覆盖标准。所有的覆盖率测试方法都是基于代码的逻辑结构，结合模块之间的依赖关系，监测评估代码的相关覆盖率信息。

2.2.1．语句覆盖

在程序代码中，语句是最基本的单位，基本的语句构成了程序的逻辑、函数以及文件。语句如同人们的姓名，是一种有独特专一性的标识，并且不同的类型表示不同的含义。因此，基本的语句覆盖率是覆盖率测试最基本的形式。语句的覆盖率是指被测试的语句占所有被执行代码语句的百分比。但是语句覆盖率测试仅仅是基本手段，并不充分，即使语句覆盖率达100%，也不代表覆盖率测试充分，因为这只是说明语句被执行了，但是不能看出其中的逻辑缺陷。

2.2.2．分支覆盖

我们的软件有三种基本结构，分别是顺序、循环以及分支。程序的分支与逻辑成正比，即分支数量越多，代码的逻辑越复杂。覆盖率测试重点在于程序的逻辑，因此分支覆盖率测试的效果要优于简单的语句覆盖率测试。分支覆盖率测试，是计算已经执行的分支数占所有可执行的分支总数的百分比。但是在实际测试过程中代码的分支数量过多，直接导致了测试成本的大大增加，所以通常我们只选取一些重点分支语句来进行覆盖率测试，提高了测试效率的同时也可以保证测试质量。

2.2.3．条件覆盖

条件覆盖率测试是基于代码中条件表达式的测试，它是计算已经执行的条件布尔值数占所有可执行的布尔值总数的百分比。条件表达式有“真”值和“假”值两种取值，如果针对每个条件表达式的两种取值设计测试用例，那么覆盖率的测试执行会比较全面。但是条件表达式的不同取值导致的结果变化，并不等同于分支发生变化，因此条件覆盖率测试不

能保证覆盖率测试的充分性。同分支覆盖，执行条件覆盖率测试时也同样选取部分关键条件语句来执行测试。

2.2.4．其他覆盖方法

除了以上覆盖率测试方法，我们还可以根据程序的编写结构来执行覆盖率测试，但是这个方法也不能保证测试的充分性。随着航天软件的复杂度提升，软件代码的数量以及随之而来的测试用例数量也不断增加，因此，选取合适并且合理的覆盖率测试方法非常重要，保证尽可能快并且多地发现软件的缺陷。

2.3．代码插桩

插桩，是指在程序源代码中的某些位置写入额外的语句代码以便于进行测试，但是这些额外写入的代码要保证不对程序的运行带来影响，这样才能保证软件测试的效果。插桩的位置要根据具体情况，程序的结构、测试的具体要求或目的都会对其造成影响。代码插桩技术能够根据不同的需求来获取软件的相关信息，是覆盖率测试实现的最关键技术之一。目前大多数的覆盖率测试工具都是基于插桩技术。

3．嵌入式软件覆盖率测试原理

嵌入式软件在航天领域应用甚广，这种本身嵌入到计算机系统内部的软件非常特殊，一般的软件测试方法并不适用于嵌入式软件。他们与硬件联系十分紧密，在测试时要时刻考虑到硬件的特点。嵌入式软件的最终测试效果与相关的硬件性能和环境息息相关，并且硬件的实时性很强，经常需要在特定的时间内完成对某项任务的处理工作，因此测试过程也要考虑到时序的处理。由于嵌入式软件的诸多特殊之处，其覆盖率测试也要采取相应的改变措施，这些改变如下：

1） 由于硬件的实时性处理要求，需要重新考虑代码插桩技术给程序运行结果带来的影响，要对其加以控制；

2） 由于嵌入式软件与硬件的不可分割性，需要采取一定的措施通过物理或逻辑的手段完成数据的传输过程。

嵌入式软件的测试基本原理如图1所示。

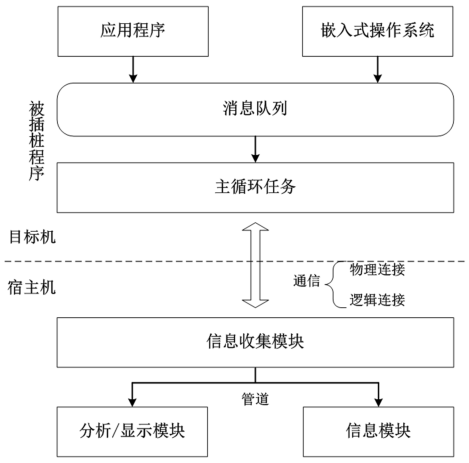

图1 嵌入式软件测试基本原理。

我们使用LDRA Testbed中的覆盖率测试功能对嵌入式软件进行测试，在函数的入口、出口以及分支处进行代码插桩后，加载修改后的代码，代码运行在目标机中。然后运行设计好的测试用例，采集相关的覆盖率信息，之后对所有的覆盖率信息进行统计计算，从这些数据中可以看出代码的逻辑结果和测试进度情况。

4．覆盖率测试的实现过程

4.1．测试覆盖率实现过程

在 LDRA Testbed软件测试环境中，McCabe自动完成覆盖率测试的代码插桩工作。通过对测试用例进行执行，把McCabe搜集的覆盖信息提取出来，再对其进行分析，即完成了覆盖率测试的工作。图2显示了使用LDRA Testbed进行覆盖率测试的完整流程。

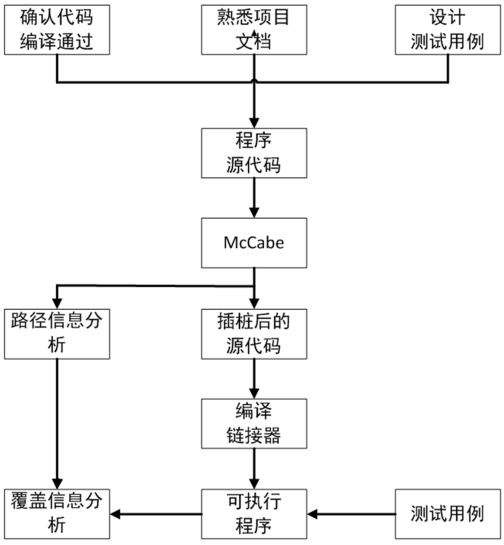

图2 嵌入式软件测试基本原理。

下面以gx.c为例，分析这段代码的语句覆盖、分支覆盖等覆盖率情况。

gx.c的源代码如下：

```
int main( void )
{set_com();
   ……
if(ucFlag_int==0x55)
{ ucFlag_int=0;
        ……
if(iTime_electrify>iTime_wacalc)
                { if(ucFlag_cali!=0xaa)
                        {for(i=0;i<3;i++)
                                {……
}
}
}
}
          else
                {
                    if(ALignOKflag==0x0f)
                      {task_nav();
              ……
```

```
if(iTime_Klm%50==0)
                    {ucFlagGpsValid=1;
                      ……
                     }
                  task_Klm5w();
                  ……
                   }
  }
       return 1;
}
```

载入文件后进行测试，使用工具对源代码进行插桩编译，执行编译程序时，工具会自动分析覆盖率的具体情况。

利用Testbed工具对嵌入式软件进行静态测试。覆盖率分析过程如下：

1) 测试准备，即在项目测试之前，首先要保证所测试的项目是可以编译通过的，也就是代码在语法上不存任何问题。
2) 要完全熟悉所测试项目的需求规格说明书、项目任务书、总体设计和详细设计等一系列项目文档，保证测试人员的理解不会偏离项目的任务目标。
3) 设计好所必须的测试用例，保证测试用例的完整、合理和可靠。
4) 使用LDRA Testbed中的Select File菜单，选择要测试的源程序，然后在Analysis菜单下，选择Select Analysis子菜单，在弹出的选项框中，选择Main Static Analysis。接下来进入静态分析状态，分析结束后，在Code Review Report界面查看静态分析情况。
5) 在静态分析后，能够了解到项目中各函数之间的调用关系，并提供评估现有测试过程质量的Stand Kiviat图，使软件得到更好的维护，见图3。该图以扇形结构表示，每个扇骨表示一类度量维度，轴上的不同点代表不同的结果值，结果和限制值之间的关系以不同颜色加以区分。

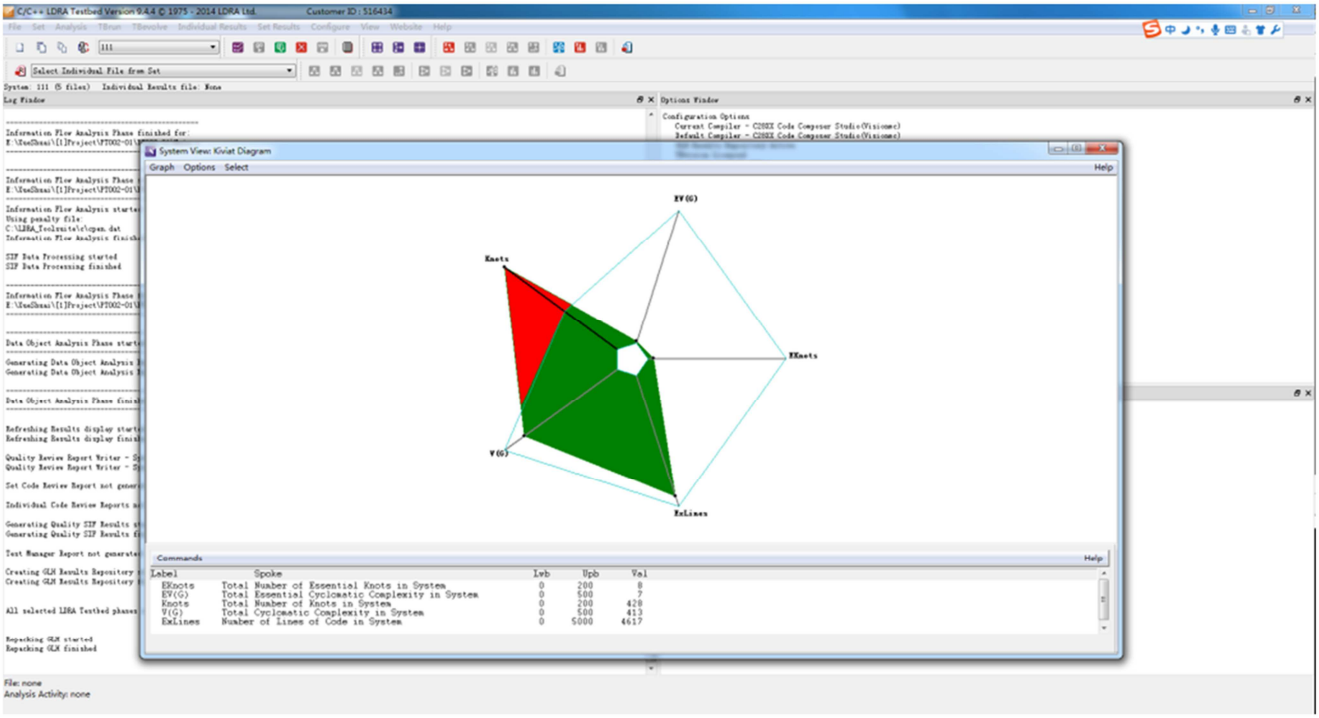

图3 被测代码的Stand Kiviat图。

6) 选择Configure菜单下的Dynamic Coverage Report Configuration选项，选择覆盖测试类型-Level B，即语句和分支要求100%覆盖。

4.2. 测试覆盖率分析

通过建立测试用例，执行测试程序，逐步完成对程序的覆盖。在执行测试用例的过程中，LDRA Testbed进行逐次累计相加，语句覆盖率等于执行语句的数目除以执行语句的总数目，在不同的测试用例下计算出软件的语句覆盖率以及分支覆盖率。以gx. c源程序下的第一个子函数compen_dimension（fWibb,xs_lever,fAibb）为例，当第一测试用例执行后，软件完成34%的语句覆盖率和8%的分支覆盖率。覆盖率在Static Coverage Analysis Report界面查看。再设计第二个测试用例，程序执行25%的语句覆盖率和22%的分支覆盖率。按照这样步骤，不断创建新测试用例，执行测试用例，累计语句覆盖率和分支覆盖率，直至软件的语句覆盖率和分支覆盖率都达到100%。

需要指出的是，并不是所有的软件语句覆盖率和分支覆盖率都可以通过LDRA Testbed达到100%。主要有三点原因：

1）源程序代码存在问题，有的函数在设计时存在逻辑错误，测试用例无论如何也无法执行到所有的语句覆盖，这种情况或是修改源代码，或是修改测试用例，才能解决无法执行到的问题。

2）由于Testbed覆盖率测试是在单进程中完成，而程序可能在多进程中执行，所以执行测试用例无法达到语句覆盖和分支覆盖的100%。

3）由于机载软件是与硬件交互的，软件代码的实现有时根据硬件的需要，做一些修改。

5．测试用例的改进方法

5.1．航天软件测试用例优化过程

航天软件由于自身运行环境的特殊性，要求测试精度之高，测试过程之复杂，远非一般应用软件可比，即便功能较少的航天软件在测试时，也会产生数量巨大的测试用例，因此需要从多个方面考虑优化测试用例。

1）在了解测试需求后，需要对测试需求进行优化处理。通过增加不完善的需求、删除重复需求、修改相似需求、归纳同类需求等手段，对全部测试需求进行优化，不但可以更好的了解测试的目的、测试的内容、测试的方法，还可以大幅度降低后续的工作量。

2）在设计具体的测试用例时，要充分考虑可容忍的最低覆盖率，测试用例在设计完成后，对完成的用例进一步进行优化。对于长期从事测试工作的专业人员，还应考虑采用测试用例的复用技术，进一步从技术层面减少工作量，提高测试的效率。

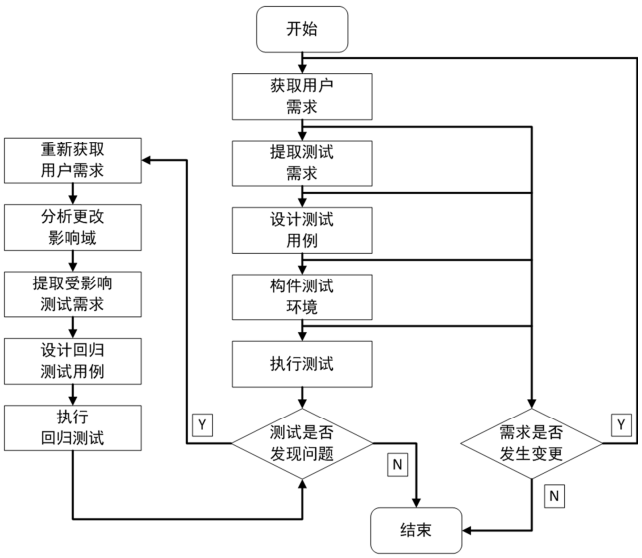

图4 航天软件测试流程图。

3）一旦用户的需求发生变化，不要马上重新设计测试用例，而要首先分析变更的需求对整个测试工作的影响，只对受到影响的测试需求和测试用例进行重新设计，保留未受到影响的部分，但是如果测试已经执行到中途则需要停止，归零后重新测试。

4）在进行回归测试之前，重点分析用户的需求变更带来的影响，在设计和选择回归测试用例时，也可以对回顾测试用例进行优化。

5）在完成一个阶段的测试工作后，将原测试用例和回归测试用例进行一次比对，确认用例无误并且代码能够顺利执行后，替换掉上一个过程的测试用例，最终能够形成完整的测试用例集合。将这些用力集合归档留存，以备将来查看和修改。

5.2．基于需求的测试用例改进方法

航天软件测试的用户需求包含多个方面，包括项目任务书、系统需求说明书、总体设计文档、详细设计文档和源程序等。不同的需求之间并不完全是相互独立的，也会存在一些关联关系。比如功能需求需要调用接口需求，功能需求会对性能需求产生影响，各个模块之间的通信关联等。因此，要对各模块的测试需求加以归纳总结，并进行分类划分。

将上述测试需求划分为以下五种关系：等价关系、包含关系、独立关系、交集关系和耦合关系。所有的测试用例都可以按照上述五种关系进行分类设计。具体操作方法如下：

1）等价关系的改进方法。如果两个测试需求集合相等，可以互相包含，那么可以删除其中一个测试需求或测试集合，并不会对整体的测试内容产生任何影响。在设计测试用例时，只需要对其中的一个测试需求或需求集合设计测试用例就能够保证全覆盖，这样就能够达到事半功倍的效果。

2）包含关系的改进方法。如果两个测试需求集合属于包含关系，即当其中一个测试需求或需求集合属于另一个测试需求或需求集合时，保留其中描述内容更全面的作为整个测试的需求或需求集合，就能够符合测试的全部要求。

3）独立关系的改进方法。如果两个测试需求集合相互独立，互不隶属，不存在任何关系时，要对两部分测试的需求覆盖率都达到100%，就必须完全保留两部分测试用例的设计。

4）交集关系的改进方法。如果两个测试需求集合存在互相重合的部分，在设计测试用例时，只需将重复的部分保留一份，再合并上互相不重复的部分，对这些需求或需求集合进行测试用例设计，即可满足要求。

5）耦合关系的改进方法。当两个测试需求或需求集之间虽然相互独立，但功能各不完全，必须结合使用方能实现某些功能时，可将它们合并为一个测试需求集，这样设计一个更为连贯的测试用例以实现对合并后需求的测试，从而减少用例数量，提高测试效率。

5.3．基于需求的测试用例改进方法

在对某航天软件测试过程中采用上述方法对测试用例进行改进。测试步骤如下：

1）提取测试需求。在进行测试之前的准备阶段，应该获取的用户需求为《需求规格说明书》和《详细设计文档》等，并根据这些文档提取出测试需求。

2）对测试需求进行优化。按照上述所叙五个类型将各个模块的测试需求进行优化处理。

3）根据各模块的需求简化测试用例。根据相关项目经验预计达到需求覆盖率需300个左右的测试用例，但实际设计了204个，测试工作量约减少了36.25%，如下表所示。

表1 测试用例简化结果对比。

关系	测试用例数量	
	改进前	改进后
等价关系	85	43
包含关系	61	48
相互独立	43	43
交集关系	57	31
耦合关系	74	39

5.4. 结果对比

通过累计记录的方式，LDRA Testbed可以得到分别以代码和图表的形式显示软件执行的调用覆盖率累计情况，使测试人员可以随时掌握软件的测试进程，以及在测试用例实施情况下软件分支执行的对错。通过测试用例的注入，本次测试分支覆盖率为94.92%。与李树芳等人的测试覆盖率快速分析方法[9]以及张臻阳等人的wrapper设计方法[10]相比不仅提高了测试覆盖率，还大大提高了测试效率，减少了测试工作量。

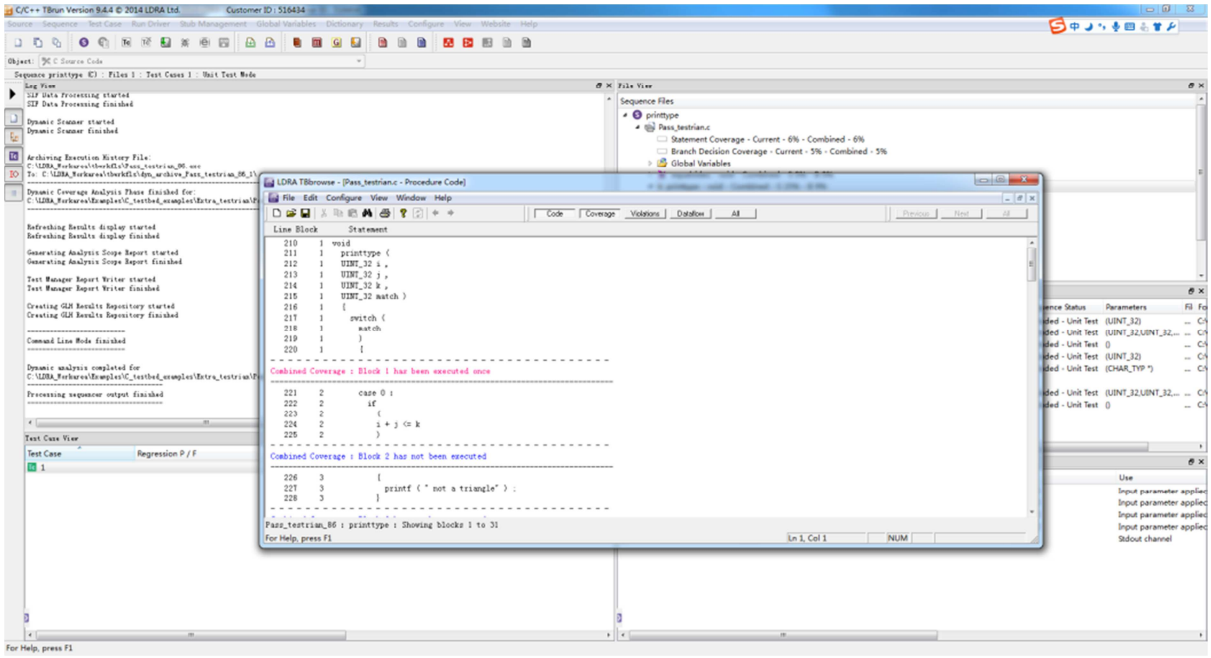

图5 LDRA Testbed覆盖率测试代码结果显示。

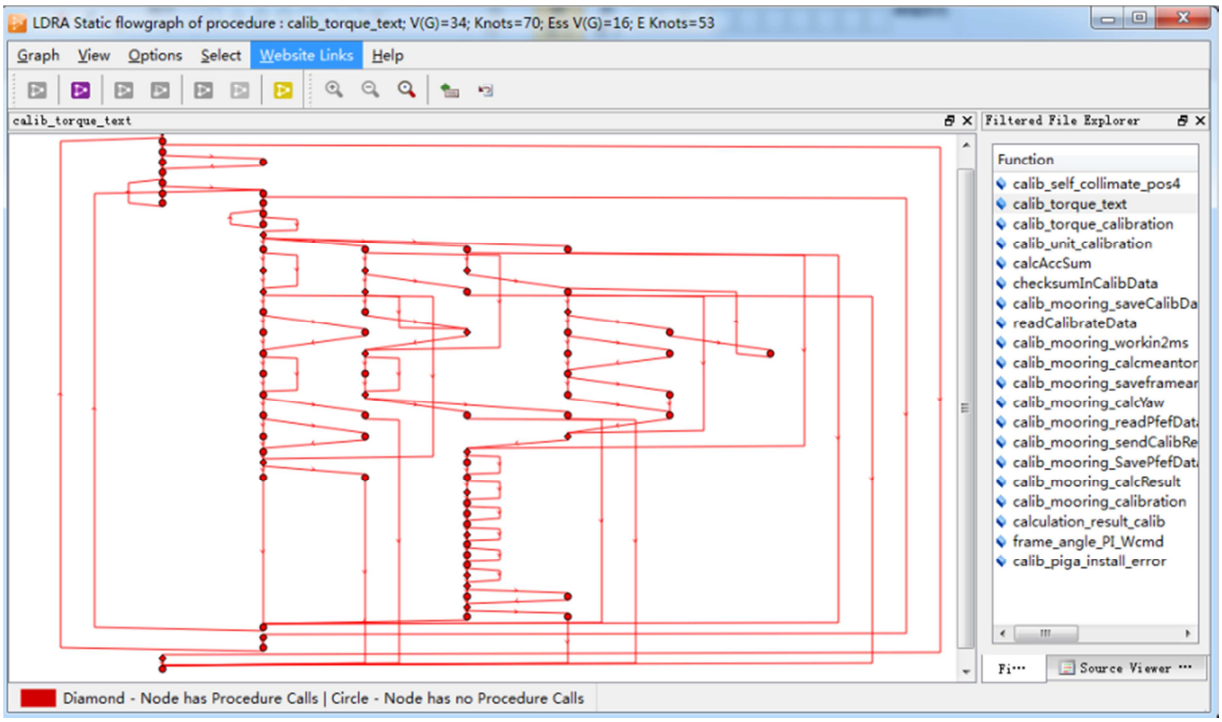

图6 LDRA Testbed覆盖率测试逻辑结果显示。

从下表统计可以看出，在测试用例改进后，无论是代码覆盖率还是测试所耗费时间，均有较大幅度的提升和改进。

表2 测试用例改进前后代码测试覆盖率变化情况。

测试用例数量			代码行数	代码覆盖行数	代码覆盖率/%	测试耗时
改进前	模块1	85	1425	1283	90.04	32h
	模块2	24	489	421	86.09	
	模块3	15	350	331	94.57	
	模块4	47	527	486	92.22	
	模块5	12	284	250	88.03	
	模块6	56	706	625	88.53	
	模块7	81	1239	1140	92.01	
改进后	模块1	31	1425	1323	92.84	20h
	模块2	20	489	475	97.14	
	模块3	15	350	331	94.57	
	模块4	41	527	501	95.07	
	模块5	12	284	273	96.13	
	模块6	39	706	664	94.05	
	模块7	46	1239	1198	96.69	

6．结论

本文基于航天软件自身的特点，提出一种通过改进测试用例来提高测试覆盖率的方法，该方法在熟悉各类文档明确项目需求后设计测试用例，通过对测试需求的优化来减少测试用例设计的数量，与其他方法[1-14]相比，在改善测试覆盖率的同时提高了测试效率，保证了测试的有效性和高效性。

全文从测试需求着手，在确保需求覆盖率的同时，化繁为简，提高测试效率。最后，将该设计方法应用于某软件测试过程中，使得测试工作量减少了约36.25%，达到了预期效果，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] 刘颖,王英,刘漫丹.嵌入式软件的测试覆盖[J] .自动化仪表,2012,33(6):63-66。

[2] 张丽.基于嵌入式系统的软件结构覆盖测试技术[J] .舰船电子工程,2005,25(3):63-66,83。

[3] 陈蕊,张广梅,李晓维.程序中不可达路径的检测方法[J]. 计算机工程,2006,32(16):86-88。

[4] 张艳梅,姜淑娟,王庆坛,等.不可达基路径的静态检测方法[J]. 计算机科学与探索,2012,6(2): 144-149。

[5] 黄晨,董燕,于倩,虞砺琨.基于目标码的测试覆盖不可达分析方法[J].测控技术,2017,36(01):100-103+107。

[6] 张垚,张超容,林腾,董芳泉.二进制代码测试覆盖率评估系统设计与实现[J].指挥信息系统与技术,2015,6(06):13-17。

[7] 王宏伟.提高DFT设计测试覆盖率的一种有效方法[J].今日电子,2008(04):41-42。

[8] 董彬.用矩阵来提高测试覆盖率[J].程序员,2007(10):108-110。

[9] 李树芳,安金霞,郑鹏飞,王猛.面向大型实时软件的测试覆盖率快速分析方法[J].西南科技大学学报,2013,28(03):89-94。

[10] 张臻阳. 一种提高测试覆盖率的wrapper设计及其优化[A]. 中国计算机学会.第二十一届计算机工程与工艺年会暨第七届微处理器技术论坛论文集[C].中国计算机学会:中国计算机学会计算机工程与工艺专业委员会,2017:4。

[11] 赵海秋. 考虑测试覆盖率和故障检测率的软件可靠性模型[A]. 中国机械工程学会可靠性工程分会.2010年全国机械行业可靠性技术学术交流会暨第四届可靠性工程分会第二次全体委员大会论文集[C].中国机械工程学会可靠性工程分会:中国机械工程学会可靠性工程分会,2010:3。

[12] Muhammad Shahid. An Evaluation of Test Coverage Tools in Software Testing [A]. 新加坡国际计算机科学与信息技术协会(IACSIT—International Association of Computer Science and Information Technology). Proceedings of International Conference on Computer Communication and Management（ICCCM 2011）[C].新加坡国际计算机科学与信息技术协会(IACSIT—International Association of Computer Science and Information Technology):成都亚昂教育咨询有限公司,2011:7.

[13] 张波. 基于测试覆盖的安全关键软件测试策略研究[D].中国科学院研究生院（长春光学精密机械与物理研究所）,2012。

[14] Muhammad Shahid. A Study on Test Coverage in Software Testing[A]. 新加坡国际计算机科学与信息技术协会(IACSIT—International Association of Computer Science and Information Technology).Proceedings of International Conference on Computer Communication and Management（ICCCM 2011）[C].新加坡国际计算机科学与信息技术协会 (IACSIT—International Association of Computer Science and Information Technology):成都亚昂教育咨询有限公司,2011:9.