无线电调谐单元软件语句覆盖目标的实现①

宫伟祥, 崔诗娴, 陈春晓

(中电科航空电子有限公司, 成都 611731)

摘 要: 本文以无线电调谐单元设备中的无线电调谐软件为例, 通过引入 LDRA TESTBED 自动化测试工具提高测试效率, 实现满足 DO-178C 中的语句覆盖数据的目标. 语句覆盖是民用飞机软件研制中最基本的覆盖率测试目标. 语句覆盖可以有效检测出源代码中的多余代码, 提高代码质量. 通过实验表明 LDRA TESTBED 可以快速完成源代码插桩以及代码覆盖率数据分析任务, 同时, 该工具自动标识源代码中未执行的代码, 为开发者提供了分析的基础. 工具生成的语句覆盖率数据满足DO-178C标准目标要求, 可以作为证据提交给局方进行审查, 提高软件的置信度.

关键词: 语句覆盖; 机载软件; 无线电调谐单元; 自动化测试工具; 局方

Achieving of Statement Coverage Object for Radio Tuning Unit Software

GONG Wei-Xiang, CUI Shi-Xian, CHEN Chun-Xiao

(China Electronics Technology Group Avionics Corporation, Chengdu 611731, China)

Abstract: Taking the Radio Tuning Unit software hosted on Radio Tunning equipment for instance, this paper discucces how to achieve the statement coverage object of DO-178C by LDRA TESTBED automatic test tool which can improve the test efficiency during the statement coverage test. Statement Coverage is the basic test coverage object for airborne software and could effectively detect the extraneous code from the source code to improve the quality during the test. The test report indicates the TESTBED can process the code instrumentation and analyse the coverage data correctly. And it can identify the uncovered codes information that provides a source for developer to analyze. The report of TESTBED conforms to the DO-178C object requirement and be provided to the certification authority for review, which can increase the certification credits.

Key words: statement coverage; airborne software; radio tuning unit; automatic test tool; certification authority

民用机载软件作为高可靠和高安全性软件, 机载软件的研发过程严格受控. 目前, FAA 正式发布咨询通告 AC 20-115C 将 RTCA DO-178C[1]作为机载软件的审定基础之一. DO-178C 标准注重对高等级软件研制过程中对源代码追溯的要求, 所有的代码都是需要基于软件低层需求和设计描述文档开发并建立源代码与低层需求之间的链接追踪[2]. 软件验证过程中验证者基于需求来开发和执行测试用例和规程, 获取和分析语句覆盖率数据[3].

上世纪 90 年代, 代码覆盖率获取方式是通过人工在源代码中插入相关的标识语句, 通过文件或者驱动向宿主机发送覆盖率数据. 但是此方式存在效率低下, 工作量大的缺点, 严重制约项目的进度, 增加研制成本. 随着自动化测试技术的普及, 目前机载软件覆盖率测试通过专业的自动化测试工具完成. 测试工具对源代码中的语句进行处理, 用以记录代码的执行情况. 通过使用自动化工具, 可以极大减软件轻验证方面的工作量[4], 提高效率并且利用工具生成的数据可以直接作为证据提交给局方进行审查, 可以提高软件的置信度. 本文根据 RTCA DO-178C 中对于民用飞机机载软件结构测试覆盖率的要求并结合无线电调谐单元项目中的实际需求, 描述使用 LDRA TESTBED[5]自动化

① 收稿时间:2015-09-18;收到修改稿时间:2015-11-02

测试工具完成语句覆盖测试目标.

1 相关介绍

1.1 DO-178C 介绍

在民用飞机的研制任务中, 软件与硬件研发的工作量所占用的比例已经由 20 年前的 40%比 60%提升至今天的 60%比 40%. 越来越多的功能实现由传统的硬件实现转变为由软件实现. 机载软件具有替代硬件的趋势.

DO-178C 是一套民用航空机载软件研制和审定的指南, 该指南的核心为确保研制的软件需要实现预期的功能, 满足试航的要求并且提供足够的安全信心. 该指南标准在 2011 年发布实施, 正式替代 DO-178B[6]. DO-178C 对机载软件按照等级由高到低为五个等级, 即 A 级、B 级、C 级、D 和 E 级. DO-178C 标准的目标中要求 A 级、B 级和 C 级软件满足软件测试结构覆盖要求. 结构覆盖率目的在于发现基于需求的测试过程中未被执行的代码结构, 并且对产生的多余代码进行分析. C 级软件应完成语句覆盖测试目标, 语句覆盖应基于需求进行测试, 而非单元测试[7]. B 级软件除完成 C 级软件的要求之外应完成判定覆盖测试. A 级软件除完成 B 级软件的要求之外应完成 MC/DC 覆盖测试[8].

1.2 无线电调谐单元介绍

无线电调谐单元设备是飞机通信导航(CN)系统的核心设备之一, 实现对飞机通信导航设备的无线电调谐、模式控制和状态显示功能, 并且为主副驾驶员提供了主要的调谐控制界面.

无线电调谐单元设备按照满足 CCAR-25-R4 标准飞机的技术要求为背景进行研制, 同时该设备需要取得中国民航局(CAAC)的 CTSOA 适航认证.

无线电调谐单元软件驻留在无线电调谐设备中的嵌入式软件[9], 无线电调谐软件根据系统安全性分析的要求, 软件研制保证等级为 C 级. 根据 DO-178C 中对于民用飞机机载软件结构测试覆盖率的要求, 无线电调谐单元应完成 DO-178C 中的语句覆盖测试要求. 即无线电调谐单元需要完成 100%的语句覆盖目标.

2 插桩原理

自动化测试工具 LDRA TESTBED 是由英国 LDRA 公司开发研制的一款软件测试产品, 如图 1 所示. 它很对源代码具有强大的测试和分析能力. 该产品在机载软件研制过程中被广泛使用并具备通过 DO-178C 工具鉴定的可鉴定的能力.

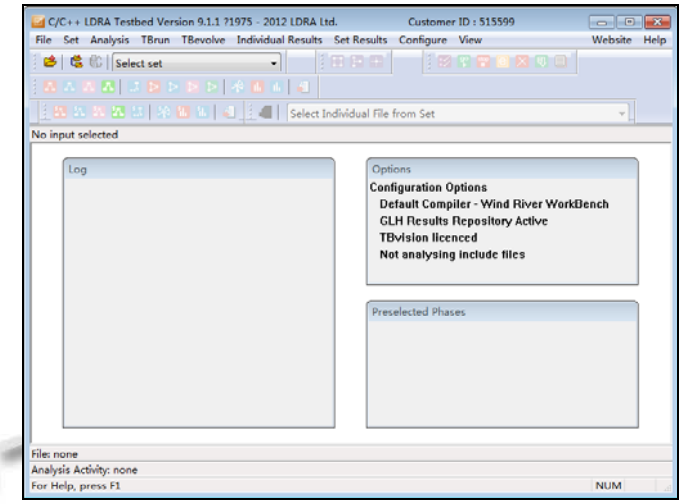

图 1 TESTBED 工具主界面

TESTBED 工具实现语句覆盖率的基本原理是对源代码进行插装. 目的是在源代码中自动插入代码检查点, 检查点用于记录测试用例执行过程中是否执行到该语句. 编译插桩后的代码, 可生成具有记录语句执行率功能的可执行目标文件. 目标文件下载到目标计算机上并运行设计的测试用例可获取覆盖率历史数据. 利用 TESTBED 工具分析覆盖率历史数据, 生成覆盖率结果. TESTBED 工具的插装方式分为基于文件系统和基于 BITMAP 数组的插装方式.

基于文件系统的插桩主要适用于主机平台开发的软件以及支持文件系统并且对实时性要求不高的嵌入式平台软件. 插桩原理如图 2 所示. 基于文件系统的插桩方式对系统的要求比较低, 便于实施但存在以下问题:

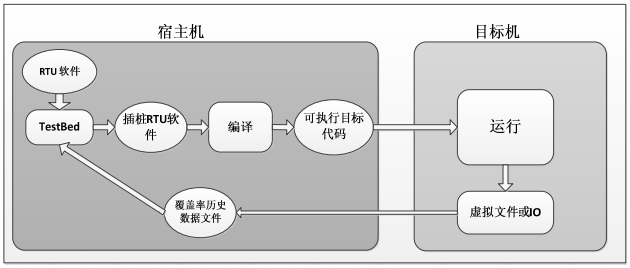

图 2 基于文件系统插桩模式

① 覆盖率数据保存在文件中
② 插装后代码膨胀比较大
③ 频繁读写文件, 对系统性能会有一定影响

与基于文件系统的插桩模式相比, 基于 BITMAP 数组的插装方式适用于对目标文件大小严格控制以及实时性要求高的嵌入式系统软件, 被插桩的代码在运行时将自动在内存中申请一块固定大小的内存空间, 用于存放覆盖率数据. 覆盖率的获取通过通信驱动(网口, 串口等)获取. 插桩原理如图 3 所示.

① 覆盖率数据存放在大小固定的数组
② 插装后的代码膨胀很小
③ 对系统的实时性影响很小

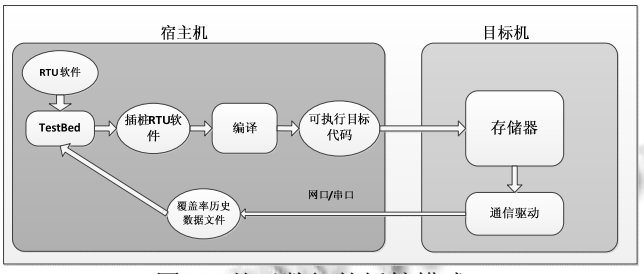

图 3 基于数组的插桩模式

3 插桩配置

3.1 添加源文件

TESTBED 工具可以为多个源文件进行统一进行插桩处理. 插桩时需要将无线电调谐单元的需要插桩的源代码文件加入到基于系统(*System*)的 TESTBED 工程集合(*Set*)中. 插桩之前应确保所有的源代码确保没有语法错误, 并且能通过编译过程. 这样可以保证插桩后的代码可以正确被编译.

源代码添加到 TESTBED 工程后, 需要配置源代码的插桩模板. 即将 TESTBED 中当前的编译器设置更改为项目使用的编译器. 编译器设置主要是影响到插桩模版的使用. 本项目使用的编译器为 *Wind River WorkBench*, 如图 1 所示.

3.2 插桩配置

源代码进行插桩时, TESTBED 需要对源代码进行静态分析和静态配置. 静态分析将源代码的格式按照 TESTBED 的要求进行重新调整, 完成源代码的复杂度的分析. 静态配置主要设置是否分析头文件以及对源代码中的宏进行处理. 宏定义存放在 *Sysppvar.dat* 文件中.

TESTBED 插桩设置需要配置, 主要包括设置插桩文件命名, 插桩代码存放位置, 插桩代码模版以及插桩模式等. TESTBED 工具可以设置插桩后的代码文件名. TESTBED 默认生成插桩后的文件命名规则是原文件名前添加前缀符号“*inszt_*”.

无线电调谐单元对实时性要求比较高, 选用基于数组格式的插桩模式. 其他的配置选择包括:

① 预先完成源代码的插桩
② 只生成一个历史数据文件
③ 采用数组模式记录覆盖率数据

TESTBED 需要对动态覆盖率数据文件进行配置. 主要包括设置覆盖率历史文件生成路径和处理覆盖率历史文件选项.

① 确认历史数据文件
② 运行被多个任务交叉执行
③ 累计覆盖率数据

4 覆盖率数据分析

4.1 执行插桩

TESTBED 完成上述配置项后可以执行源代码的插桩工作. TESTBED 插桩选项应包含以下的选项:

① 静态分析
② 复杂度分析
③ 根据插桩设置进行插桩

上述选项为开展源代码插桩的最基本选项, TESTBED 将为该工程生成工程文件(**.tcf* 格式), 该工程文件中包含了上述所有的设置, 包括插桩的文件, 插桩模板选择, 宏定义文件, 编译器设置等.

TESTBED 会调整代码的结构以及在代码中插入监测点, 因此插桩后的代码将与原有的代码有很大的不同. 图 4 为代码插桩前的代码, 图 5 为插桩后的代码.

TESTBED 会自动根据代码的结构进行插桩. 从上图中由于该函数中不包含语句分支等相关的判定, TESTBED 只在该函数中插入两个监测点.

```
bool_t Receive_A429_Word(uint16_t a429Port,
                         uint32_t *recvBuffer,
                         uint32_t *recvCount)
{
    bool_t retVal = S_FALSE;
#ifdef IDE_ENV_VXWORKS
    retVal = Receive_A429_Data_From_Adapter(a429Port, recvBuffer, recvCount);
#endif/*IDE_ENV_VXWORKS*/

#ifdef IDE_ENV_WINDOWS
    retVal = Receive_A429_Data(a429Port, recvBuffer,recvCount);
#endif/*IDE_ENV_WINDOWS*/
    return retVal;
}
```

图 4 插桩前代码

```
bool_t
  Receive_A429_Word (
  uint16_t a429Port ,
  uint32_t * recvBuffer ,
  uint32_t * recvCount )
 {
   int iA429IODriver_199zzqqzz
 = A429IODriver_199zzqqzz          (       5 ) ; /* 37 */
    bool_t
      retVal = S_FALSE ;
      /*IDE_ENV_VXWORKS*/
    retVal = Receive_A429_Data ( a429Port , recvBuffer , recvCount ) ;
      /*IDE_ENV_WINDOWS*/
    A429IODriver_199zqendz          (       6 ) ; /* 30 */
    return
    retVal ;
 }
```

图 5　插桩后代码

4.2 覆盖率数据获取

编译插桩后的文件并将可执行程序加载到无线电调谐单元设备. 执行测试用例和规程后代码覆盖率将记录在内存数组中. 串口可用情况下可通过 shell 模式调用“*gethistory*”函数即可获取覆盖率历史数据, 将数据保存为“*history.exh*”文件. *gethistory*()函数实现如下:

```
void get_history(void)
{
    int i=0,j=0;
unsigned int * pbitmaparr;
    while( qqzzglobbrns[i] != 0 )
    {
        for(j=0;j<(qqzzglobbrns[i]/8)+1;j++)
{
            pbitmaparr = qqzzglobarrb[i];
            qqoutput2( Tbsdem1zzhfil,"%8d%8d\n",
    pbitmaparr[j],qqzzglobfileid[i]);
        }
        i++;
     }
} /* endof zqzqzq */
```

由上代码可见, TESTBED 使用 qqzzglobarrb 数组存储覆盖率数据, 通过 TESTBED 自定义的 qqoutput2() 函数按照指定的格式输出.

除串口外, 还可以通过以太网采用 TCP 协议将覆盖率数据通过 send()函数输出. 无线电调谐单元设备作为客户端, 通过网络将数据发送到指定的主机上. 代码实现如下:

```
unsigned char sendBuf[32];
while( qqzzglobbrns[i] != 0 )
{
    for(j=0;j<(qqzzglobbrns[i]/8)+1;j++)
    {
        pbitmaparr = qqzzglobarrb[i];
memset(sendBuf, 0, 32);
sprintf(sendBuf,"%8d%8d\n",pbitmaparr[j],
    qqzzglobfileid[i]);
    realLen = send(clientSocket, sendBuf, 32, 0);
    }
    i++;
}
```

4.3 覆盖率结果获取

当获取到生成的“*history.exh*”文件后. 选择动态覆盖率分析选项对覆盖率历史文件进行进一步分析以获取覆盖率数据.

在 TESTBED 分析完成后, 语句覆盖测试报告测试覆盖率记录保存在 TESTBED 生成的工程目录下. 测试报告显示整个集合的语句覆盖率测试数据.

测试报告页面中包含整个测试集中所有文件的覆盖率信息, 包括函数语句覆盖, 分支覆盖, LCSAJ 等数据. 测试报告页面详细说明每个函数的执行覆盖率情况, 本项目的初步覆盖率是 48%. 如图 6 所示.

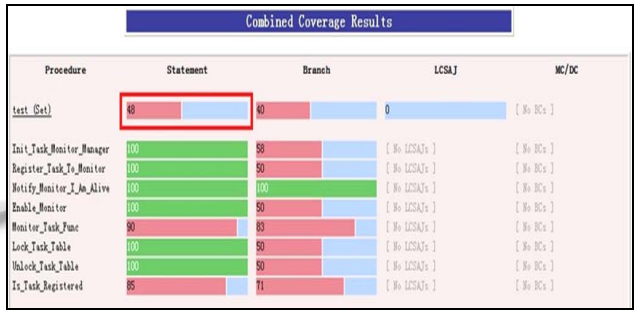

图 6　整个文件的覆盖率

在上图中列举了这个项目的总的覆盖率以及涉及到的部分函数的语句覆盖率, 详细情况如表 1 所示.

表 1　函数的语句覆盖

NO.	函数名称	覆盖率(%)
1	*Init_Task_Monitor_Manager*	100
2	*Register_Task_To_Monitor*	100
3	*Notify_Monitor_I_Am_Alive*	100
4	*Enable_Monitor*	100
5	*Monitor_Task_Func*	90
6	*Lock_Task_Table*	100

7	*Unlock_Task_Table*	100
8	*Is_Task_Registered*	85

从上表中可以发现, “*Init_Task_Monitor_Manager*”函数的覆盖率为达到 100%, 表明该函数中的代码被全部执行. 同时, “*Monitor_Task_Func*”函数的覆盖率为 90%, 表明该函数中存在部分代码未被执行. 下图以“*Is_Task_Registered*”为例, 说明函数中的代码执行情况.

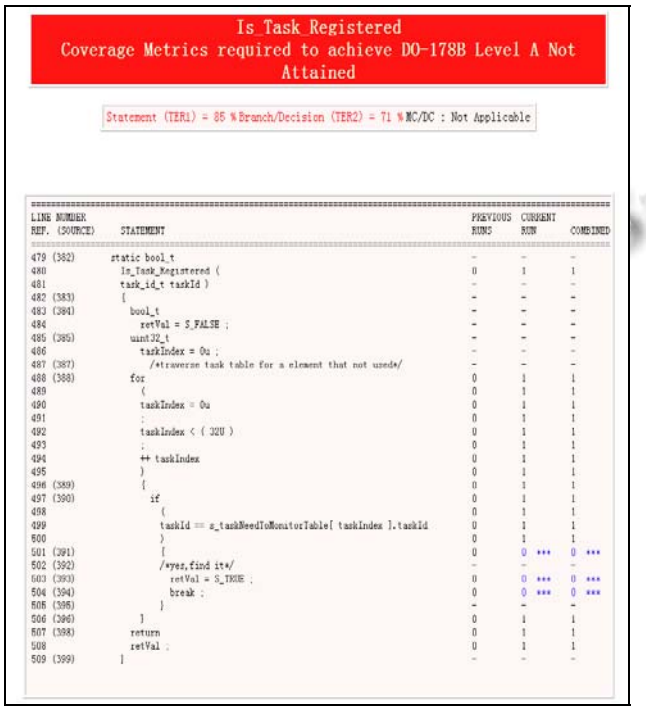

图 7 代码覆盖说明

从上图中可以发现, TESTBED 工具标识出源代码中代码执行的标识. 上图中 1 代表该行的代码被执行到, 0 代表为被执行. 上述的代码中 if()语句作为防御性编程, 正常操作无法执行到该语句, 因此 if()语句只完成了判定, 但没有执行其中的代码.

4.4 单个文件覆盖率

TESTBED 除具备分析整个集合内所有文件的覆盖率数据能力外, 还为每个文件生成覆盖率数据文件. TESTBED 可生成 ASCII 以及 HTML 两种格式的覆盖率数据报告. 图 8 为 HTML 格式的文件的覆盖率报告文件, 该源文件的整体语句覆盖率为 73%.

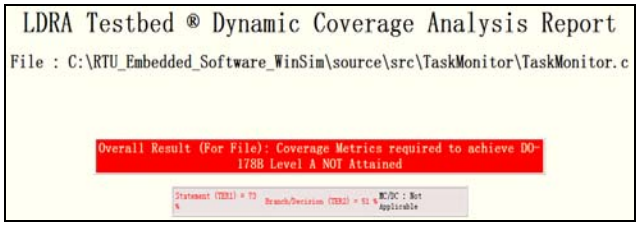

图 8 单个文件覆盖率

4.5 覆盖率说明

虽然 DO-178C 的要求 C 级软件需要完成 100%的语句覆盖率目标. 但实际的项目的开展过程中, 测试覆盖率达不到 100%的目标要求[10]. 实际项目研制中, 代码覆盖率一般在 90%以上. 对于未实现 100%覆盖率目标的函数, 需要应根据覆盖率结果分析并且定制解决方案, 这些方法包括:

① 检查是否缺少需求, 若缺少则补充需求

② 基于需求补充测试用例

③ 检查代码是否是死代码或多余代码, 经过分析后确认是否移除, 死代码为程序错误

④ 确认代码是否是非激活代码, 采用相应的方式确认非激活代码满足需求和设计, 非激活代码为可以追溯到需求的代码, 但是未被执行或使用的数据

若通过上述过程, 多次开展测试, 最终仍有相关的代码没有被执行, 则该代码判断为不可执行代码. 对于不可执行代码, 应提供相应的说明, 说明包括该为执行代码所在的模块, 文件名, 函数名称以及行号, 同时, 需要提供未执行原因. 以图 7 为例, 若通过上述的方法都没有执行的代码则应提交未执行代码说明. 如表 2 所示.

表 2 未执行代码说明

模块	文件	位置	代码	原因
任务管理模块	Task_Manager.c	391 行 ~395 行	*{ retVal= S_TRUE; break; }*	防御性编程

已经完成覆盖率的代码覆盖率加上未执行代码的解释说明需要完成 100%的覆盖率. 该说明文件和 TESTBED 生成的测试覆盖率数据应经过评审和分析, 作为提交给局方作为满足 DO-178C 测试覆盖目标的证据.

5 结语

语句覆盖测试目标是 DO-178C 标准中规定的机载软件测试结构覆盖中最基本的目标, 项目开展过程中使用自动化测试工具来实现. 本文介绍了通过 LDRA TESTBED 工具完成无线电调谐单元软件的语句覆盖率的测试. 在机载软件研制过程中需要开展多次迭代开展语句覆盖实现较高的语句覆盖率. 并且程序的语句覆盖达不到 100%的要求, 对于未执行的代码, 需要经过标识并提供理由. 最终达到 100%的覆盖率目标要求. 通过 TESTBED 工具提供的语句覆盖率报告以及未执行代码的说明文档, 可以向局方提供可信的证据表明软件的语句覆盖满足 DO-178C 目标.

参考文献

1 Software Considerations in Airborne Systems and Equipment Certification. DO-178C, Washington: RTCA, Inc, 2011: 34−51.

2 沈小明,陆国荣,王云明,蔡喁,欧阳坡.机载软件研制流程最佳实践.上海:上海交通大学出版社,2013.12:75−85.

3 李华飚译.核心测试过程:计划、准备、执行和完善.北京:中国电力出版社,2007.1:42−87.

4 蔡为东.软件测试管理全程实践.北京:电子工业出版社,2009.1:66−75.

5 黄荦,丁立冬.基于LDRA Testbed的民用机载软件结构覆盖率分析流程研究.航空标准化与质量,2014,4:26−28.

6 Software Considerations in Airborne Systems and Equipment Certification. DO-178B, RTCA, Inc, 1992:31−49.

7 张晓明,黄琳译.软件测试的艺术.北京:机械工业出版社,2013,12:34−87

8 Beizer B. Software Testing Techniques, Second Edition . USA: The Coriolis Group, 2010.1

9 杨珂瑶.基于 DSP 的嵌入式软件测试方法.计算机与现代化,2014,10:61−66.

10 陈邵英,张河涛,刘建华.软件测试与持续质量改进.北京:人民邮电出版社,2008,2:25−38.