文章编号:1671-4598(2019)07-0107-04 DOI:10.16526/j.cnki.11—4762/tp.2019.07.023 中图分类号:TP319;V240.2 文献标识码:A

基于 LDRA Testbed 的民用机载软件静态测试方法

周 培
(中航雷华柯林斯(无锡)航空电子设备有限公司,江苏 无锡 214063)

摘要:静态测试作为软件测试的重要方法,是保证民用机载软件中安全关键软件质量的关键步骤;介绍静态测试的概念和方法,采用自动化分析方法,基于软件分析工具 LDRA Testbed 从主要静态分析、复杂度分析、静态数据流、交叉索引、信息流和数据对象分析六大部分完成软件的静态测试过程,探究其测试原理和关键标准文件的配置,生成相应的代码审查和测试度量报告,以有效提高民用机载软件质量。

关键词:软件测试;LDRA Testbed;静态测试;民用机载

Civil Airborne Software Static Testing Method Based on LDRA Testbed

Zhou Pei
(AVIC Leihua Rockwell Collins Avionics Company, Wuxi 214063, China)

Abstract: Software testing is a key step to ensure the quality of safety critical software in civil airborne software. Static testing is an important part of software testing. Introduce the concept and classification of static testing. Using automated analysis method, Based on the test tool LDRA Testbed, the static testing process is completed from six major parts: static analysis, complexity analysis, static data flow, cross-index, information flow and data object analysis. Investigate its testing principles and configuration of key standard files. Generate corresponding code review and test measurement reports to improve the quality of civil airborne software effectively.

Keywords: software testing; LDRA testbed; static testing; civil airborne

0 引言

当今信息化社会中,对软件的质量要求随着软件应用的普及变得越来越高,特别是在航空等许多涉及人类生命安全的领域,对软件质量和安全的要求更是提高到了一个不容忽视的程度。为了充分保证民用机载软件中安全关键软件的质量,软件测试是关键步骤,其中静态测试是重要的环节。本文基于软件分析工具 LDRA Testbed,探究民用机载软件的静态测试方法。

1 静态测试概述

1.1 静态测试的概念和方法

静态测试也被称作静态分析,传统意义上是指不实际编译和运行被测软件,只静态地解析软件源代码,查找错误或搜集相关度量数据。静态测试的对象有代码、界面和文档 3 种,以测试是否符合相应的标准规范和需求说明为主要目的[7]。

静态测试的检测方法分为人工检测和计算机辅助静态分析 2 种:1)人工检测:指不依赖计算机而完全靠人工审查或评审软件来完成静态测试。以检查编码风格和检验质量为主,也检验各阶段的软件产品。该方法能够有效发现逻辑设计和编码错误[8]。2)计算机辅助静态分析:基于静态分析工具对被测软件进行特性分析,提取程序中的有效信息,从而检查各种逻辑缺陷和可疑程序构造[8]。

收稿日期:2019-01-28; **修回日期**:2019-02-20。

作者简介:周 培(1990-),女,宁夏固原人,初级工程师,主要从事民用机载软件的测试开发验证工作方向的研究。

在民用机载软件测试领域,对于文档常采用人工检测的方法,而对于代码,由于其规模庞大且具有高安全性和强时序性的要求,则必须采用计算机辅助静态分析的方法以尽可能多地发现程序中的隐式错误。市场上的大部分测试工具只能识别源代码与标准的偏差,而 LDRA Testbed 除了检测编码标准符合性,可以提供调用图、程序流程图和各种分析报告来可视化代码的可测试性、可维护性和清晰度,是航空领域自动化测试的主要工具。

1.2 LDRA Testbed 静态测试方法

LDRA Testbed 的静态分析支持代码程序设计标准的规定,生成关于复杂度和标准违反情况的分析报告,同时解析源代码,生成后续插桩和动态分析阶段的内部工作文件,其原理类似于编译器执行的语法分析。因此必须首先执行静态分析。

LDRA Testbed 的静态测试结构如图 1 所示,由图可知包含以下 5 种分析,并且这些分析方法必须按照顺序依次执行:

1)主要静态分析(Main Static Analysis)

2)复杂度分析(Complexity Analysis)

3)静态数据流分析(Static Data Flow Analysis)

4)交叉索引(Cross Reference)

5)信息流分析(数据对象分析)(Information Flow Analysis)

2 主要静态分析

2.1 主要静态分析概述

主要静态分析是 LDRA Testbed 测试系统的核心模块,

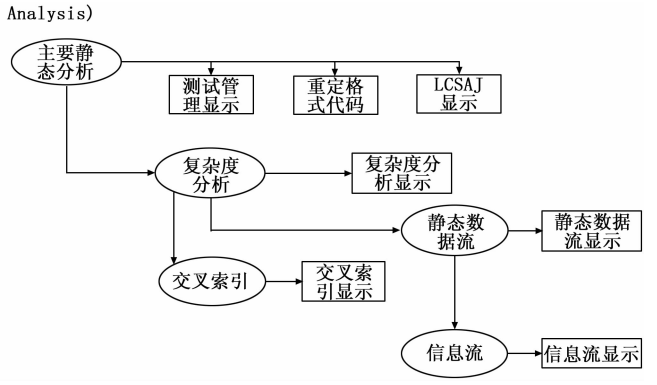

图 1 LDRA Testbed 静态测试结构

所有要求进行 LDRA Testbed 测试分析的软件必须首先执行主要静态分析。分析源代码，重新格式化源文件的副本方便后续进一步的分析，在分析运行期间会搜索违反编码标准的源代码并确定当前存在的所有 LCSAJs。主要静态分析最终会映射源代码的结构以生成静态调用图。LDRA Testbed 将通过 Log 窗口记录正在执行的主要静态分析的哪个阶段，当主要静态分析完成后，才可以继续复杂度分析和动态覆盖率分析。

主要静态分析完成以下目标：

1）基于源代码（C/C++）的语法分析和宏扩展；

2）解析源代码，生成参考列表；

3）计算控制流分支；

4）验证编码标准；

5）C++面向对象的度量；

6）注释分析报告（Ada，C/C++和 Cobol）；

7）计算 Halstead 度量（C/C++，Cobol 和 PL/Mx86）；

8）运行线性代码序列和跳转（LCSAJ）。

主要静态分析生成以下结果或图表：

1）分析范围报告；

2）LCSAJ 报告；

3）代码审查报告；

4）质量审查报告。

2.2 主要静态分析原理过程

在 Main Static Analysis 前的确认框中打勾，然后点击 Start Analysis 来开始分析。主要静态分析对源代码系统生成相应的 source _ file _ name. man 文件，该文件提供了 LDRA Testbed 测试分析的管理概述，一旦信息可用它就会用各分析模块的结果进行更新。编码规则的违反情况、注释的使用和 Halsteads 度量（只针对 C/C++和 Cobol）这些信息只能在该文件中找到，这是构成质量评价报告的主要部分。在主要静态分析完成后点击 System Results 菜单，选择 Test Manager Report（HTML）会打开测试管理报告。进入报告中左列的各节点程序，会显示导致程序失败的具体标准细节，其中违反的编程标准以红色显示。

从 Individual Results 菜单选择 Reformatted Code，会显示重新格式化的源代码。主要静态分析生成 source _ file _ name. ref 文件，该文件包含最终重新格式化的源代码清单，这是所有 LDRA Tsetbed 测试报告和测试结果的基础。重新格式化的源代码按照 LDRA 的标准显示分支结构，并标记每一行代码，这个代码版本的生成是 LDRA Tsetbed 测试分析的第一步。Reformatted Code Report 提供了跨系统进行一致度量的能力，对于以宏扩展的形式显示源代码特别有用。图 2 为 Reformatted Code Report（部分），其中 T 为可执行代码，F 为不可执行代码，如变量声明、注释等。

```
26F//=========================================
27F// FUNCTION: CSelfTest::getInstance()
28F//=========================================
29F
30FCSelfTest &
31TCSelfTest::getInstance()
32F{
33F   // The only instance of the class
34F  static CSelfTest theInstance ;
35T  return
36T  theInstance ;
37F}
38F  // instance
39F//=========================================
40F// Constructor
41F//=========================================
42F
43T  CSelfTest::CSelfTest() :
```

图 2 Reformatted Code Report（部分）

从 Individual Results 菜单选择 LCSAJ Report，会显示代码与其相关的 LCSAJ 定义。主要静态分析生成 source _ file _ name. lcj 文件，该文件右侧记录 LCSAJ 密度，底部记录关于 LCSAJs 当前数量信息以及不可达性的总结，其中可以通过查找字符串 UNREACHABLE * * * * 来定位不可达的代码。图 3 为 LCSAJ Report（部分）。

```
START              30  CSelfTest &
                   31  CSelfTest::getInstance()
                   32  {
                   33     // The only instance of the class
                   34    static CSelfTest theInstance ;
                   35    return
       FINISH      36    theInstance ;
-------------------------------------------------------------
                   37  }
                   38    // instance
                   39  //=================================
                   40  // Constructor
                   41  //=================================
                   42
```

图 3 LCSAJ Report（部分）

2.3 关键技术——配置 cpppen. dat 文件

提高代码的一致性、可维护性和整体质量的常用方法是根据行业或公司定义的标准评估代码，以便快速识别和纠正缺陷。虽然通常是通过手动的同行评审来实现的，但这个过程可能会经历长时间历程并消耗大量人力，尤其是在代码规模巨大的民用机载软件中。

LDRA Testbed 支持客户自定义标准来自动执行静态分析和规则检查，所以需要测试人员配置自定义编程标准 cpppen. dat 文件。

cpppen. dat 文件的格式为 1 1 0 1S Identical name in scope。第一列表示本条标准在 cpppen. dat 文件中出现的

顺序；第二列表示是否要被检测到；第三列表示该标准值的限制范围；第四列表示该标准的内部定值；第五列的字母有如下含义：S 表示静态分析；C 表示复杂度分析；D 表示静态数据流分析；I 表示信息流分析；X 表示交叉索引；Q 表示代码审查报告。后面即为该标准的描述。按照上述格式，测试人员可根据不同的测试对象，从行业标准的超集中筛选出所需的子集，然后逐行编写与标准对应的特定.dat 文件以缩短静态测试时间。

3　复杂度分析

3.1　复杂度分析概述

复杂度分析通常是主要静态分析后的下一步，因为它是其他静态分析部分（如数据流分析）的先决条件。复杂度分析报告源代码的低层结构，通过 Kiviat 图计算各种用来执行程序复杂性标准的指标。对于非 C/C＋＋代码，复杂度分析是在逐个程序运行的基础上分析代码，分析范围是用户可配置的。在分析运行期间，LDRA Testbed 将通过 Log 窗口记录复杂度分析的哪个阶段，当复杂性分析完成后，才可以继续静态数据流分析和其他分析。

复杂度分析完成以下目标：

1）将源代码分成基本块为后续进一步测试做准备。

2）生成静态数据流图用以表示源代码。

3）计算各种复杂性度量。

（1）控制流节点度量：节点度量对于程序结构的顺序高度敏感，它是程序复杂度的一个因素。节点分析度量了代码中的混乱程度，因此代码阅读器需要进行大量的“跳跃”操作。过多的节点可能意味着程序可以重新排序以提高可读性和降低复杂性。

（2）圈复杂度：圈复杂度反映程序的决策结构。对于任何给定模块，建议圈复杂度不超过 10。该值可用作模块从重新设计中获益的一个指标。这是一种测量定向图大小的方法。

（3）循环深度：循环深度也称为区间分析，用于探究程序循环结构。控制流循环的最大深度是影响代码整体可读性、复杂性和效率的一个因素。

（4）过程输入和输出计数：该参数基于静态调用图并度量过程的调用。输入计数是对任一程序的调用次数，输出计数包括 return 语句、程序出口或从一程序跳转到其他程序的次数。该参数可从测试管理报告中获得。

4）生成使用结构化编程结构的验证报告（SPV）：LDRA Testbed 使用结构化编程验证作为第 2 种方法来决定程序的结构是否合理。LDRA Testbed 通过逐个模块地将结构化模板与基本块的有向图匹配来执行 SPV，结构化模板（包括文件 c/cppplt. dat）是可配置的，以适应本地编译环境。SPV 包括基本圈复杂度、基本节点数 2 种度量指标。

5）在测试管理中生成关于复杂度分析标准违反情况的报告。

3.2　复杂度分析原理过程

在 Complexity Analysis 前的确认框中打勾，然后点击 Start Analysis 来开始分析。复杂度分析完成后，单击 System Results 菜单，选择 Text Results，选择 Quality Review Report（HTML）会打开质量审查报告，该报告提供关于复杂度度量的深入信息。此外，代码审查报告中会增加复杂度分析信息。

复杂度分析生成 source _ file _ name. cmp 文件，该文件显示复杂度度量和源代码的相关信息，文件的摘要处显示基于过程的复杂度度量，这些度量包括：圈复杂度度量、基本块、结点度量、过程的输入/输出、循环次数/嵌套深度和 SPV。因此选择 Static Bar Charts 可调出以下 9 中静态柱状图：

1）可执行的重新格式化行（Executable Reformatted Lines）；

2）基本块（Basic Blocks）；

3）圈复杂度（Cyclomatic Complexity）；

4）节点（Knots）；

5）间隔分析（Order 1 Intervals）；

6）基本圈复杂度（Essential Cyclomatic Complexity）；

7）基本节点数（Essential Knots）；

8）区间嵌套（Max. Interval Nesting）；

9）系统总体复杂度（Total System Complexity）。

4　静态数据流、交叉索引、信息流和数据对象分析

4.1　静态数据流、交叉索引、信息流和数据对象分析概述

静态数据流分析在源代码中寻找各种类型异常使用的变量情况，生成以下类型的关于源代码的信息以突出显示所有可能的程序缺陷：

1）过程调用信息；

2）数据流异常信息；

3）程序接口分析；

4）数据流标准违反情况；

5）LDRA Testbed 的插桩；

6）静态数据流生成的警告。

复杂度分析完成后，LDRA Testbed 能够在源代码的所有使用数据项上生成交叉索引，提供关于所有数据项（全局变量、局部变量或参数）的交叉引用信息，并说明数据项的使用情况。在需要源代码文档时交叉索引尤为重要。

信息流分析是对程序变量相互依赖关系的研究，该方法最初是由 Balzer 在 1969 年于 EXDAMS 系统中发明的。信息流是一种基于路径的分析技术，利用图论方法对单一出口图进行分析，不需要只是平面图。任何具有多个入口或多个出口的图能够通过附加头或追踪节点的方法转化为适当形式。因此，这些小扩展的分析适用于从任何源代码程序得到的有向图。

数据对象分析使用户能够从静态数据流分析、交叉索引和信息流分析中过滤信息，从而创建关于特定变量的报告，这对于正确跟踪基于变量和常量的日期非常重要，在测试中也非常实用。数据对象分析生成一个分析范围内的

完整变量交叉索引，其中的变量都是满足指定的筛选条件，在配置后报告包含一系列特定变量和状态的表格。

4.2 静态数据流、交叉索引、信息流和数据对象分析过程

静态数据流、交叉索引、信息流和数据对象分析的操作和图 2 中主要静态分析的操作一致。静态数据流分析生成 source _ file _ name. dfl 文件，该文件给出源代码数据流的完整信息，包括过程调用、递归调用、使用全局变量作为参数调用、未使用的变量、数据流异常、过程接口和未声明的变量 7 种信息。并且可通过搜索字符串“URANOMALIES”，“DECLARED BUT NEVER USED”和“＊＊＊”等查找特定信息。在 Data Flow Analysis Report 检查静态数据流的详细信息，该报告提供调用结构、数据流异常以及过程接口的相关问题。

交叉索引生成 source _ file _ name. xrf 文件，该文件生成所有变量用法的交叉引用，并指明变量用于局部、全局或是作为过程参数。由于该文件是基于过程逐渐累积并对过程中的每个变量逐个分析，通过 Cross Reference Report 查看交叉索引信息。

信息流分析生成 source _ file _ name. inf 文件，该文件包含源代码中变量依赖关系的信息，并详细说明变量是否被定义，其他变量是否直接或间接影响它的取值。通过 Information Flow Analysis Report 查看信息流。

上述 4 种分析完成后，单击 System Results 菜单，选择 Text Results，选择 Data Object Analysis Report（HTML）会打开数据对象分析报告，除显示交叉索引和静态数据流分析信息外，还记录变量间的复杂关系，该报告可以配置为仅记录特定筛选过滤的变量详情。

5 测试与验证

以民用机载软件中某航电设备的自检系统为测试对象，将 8 个. cpp 文件组成目标测试系统，完成 LDRA Testbed 静态测试过程。以其中的 CSelfTest. cpp 为例，该源文件的细节在分析范围报告中如图 4 所示。由图 4 可知 CSelfTest. cpp 的分析路径，LDRA Testbed 标记的模块号为 40，以便于后续查看该文件在系统调用图中的位置，分析时间戳，文件大小和是否可读。

图 4 分析范围报告中的 CSelfTest. cpp 信息

测试管理报告（Test Manager Report）如图 5 所示，该报告显示代码审查和质量审查的总览，显示目标系统违反编码规则的情况，从可测试性、可维护性和清晰性三方面显示出目标系统的质量情况。由图 5 可知，CSelfTest. cpp 违反编码规则的占比为 30%，可测试性为 47%，可维护性为 55%，清晰性为 71%，该文件还有很大提高质量的空间。

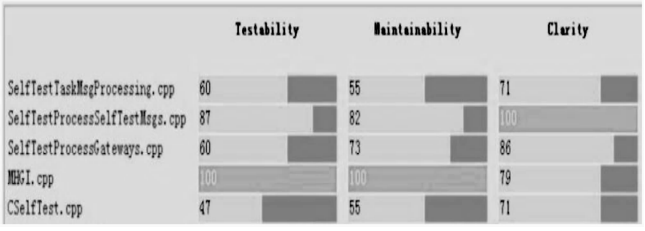

图 5 测试管理报告

代码审查报告（Code Review Report）提供目标系统代码质量的即时图，该报告反映整个系统中每个文件的代码质量，以 CSelfTest. cpp 为例，定位到 CSelfTest::init 中，可以获得违规代码行的位置，例如 534 行违反 MISRA 76，如图 6 所示。

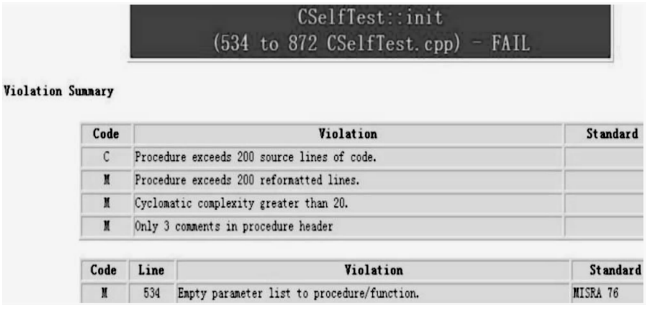

图 6 CSelfTest. cpp 中 CSelfTest::init 代码审查详情（部分）

复杂度度量：如图 7 所示，复杂度分析完成后在质量审查报告中生成复杂度度量：包括节点数、圈复杂度、基本节点数、基本圈复杂度和 SPV 判断。最后一行可得 CSelfTest. cpp 的节点数是 197，圈复杂度是 319，基本节点数是 16，基本圈复杂度是 9，SPV 判断为 No。说明该文件

Knots	Cyclomatic Complexity	Essential Knots	Ess. Cycl. Complexity	Structured Proc (SPV)
0 (P)	1 (P)	0 (P)	1 (P)	Yes (P)
24 (F)	7 (P)	0 (P)	1 (P)	Yes (P)
36 (F)	31 (F)	0 (P)	1 (P)	Yes (P)
0 (P)	1 (P)	0 (P)	1 (P)	Yes (P)
1 (P)	4 (P)	0 (P)	1 (P)	Yes (P)
19 (F)	31 (F)	0 (P)	1 (P)	Yes (P)
25 (F)	13 (F)	0 (P)	1 (P)	Yes (P)
4 (P)	71 (F)	0 (P)	1 (P)	Yes (P)
1 (P)	4 (P)	0 (P)	1 (P)	Yes (P)
0 (P)	1 (P)	0 (P)	1 (P)	Yes (P)
0 (P)	2 (P)	0 (P)	1 (P)	Yes (P)
13 (F)	11 (F)	8 (F)	5 (F)	No (F)
13 (F)	11 (F)	8 (F)	5 (F)	No (F)
12 (F)	20 (F)	0 (P)	1 (P)	Yes (P)
2 (P)	27 (F)	0 (P)	1 (P)	Yes (P)
0 (P)	5 (P)	0 (P)	1 (P)	Yes (P)
3 (P)	9 (P)	0 (P)	1 (P)	Yes (P)
13 (F)	68 (F)	0 (P)	1 (P)	Yes (P)
29 (F)	15 (F)	0 (P)	1 (P)	Yes (P)
1 (P)	5 (P)	0 (P)	1 (P)	Yes (P)
1 (P)	2 (P)	0 (P)	1 (P)	Yes (P)
197 (F)	319 (F)	16 (P)	9 (P)	No (F)

图 7 CSelfTest. cpp 的复杂度信息

（下转第 149 页）