文章编号:1671-4598(2017)03-0119-04 DOI:10.16526/j.cnki.11—4762/tp.2017.03.033 中图分类号:TP311.5 文献标识码:A

浮力调节软件的单元测试技术研究

李 磊
(中国船舶重工集团公司第七一〇研究所,湖北 宜昌 443000)

摘要:为了提高嵌入式软件的单元测试效率,同时能达到很好的测试效果,针对嵌入式软件制定单元测试充分性准则和单元测试策略,并使用测试工具对嵌入式软件进行单元测试具有实际意义;以浮力调节软件为例,通过研究基于控制流的单元测试充分性准则,结合浮力调节软件的单元测试需求,提出了针对浮力调节软件的单元测试充分性准则;同时,考虑到圈复杂度和函数节点数对函数正确实现的影响,制定了基于优先级的单元测试策略;利用自动化单元测试工具 Testbed 搭建了浮力调节软件动态测试环境,通过代码覆盖率分析,帮助创建测试用例以达到单元测试充分性要求,从而实现了浮力调节软件单元测试自动化;在自动化测试工具的帮助下,结合单元测试充分性准则和单元测试策略,最终实现严格而高效的单元测试。

关键词:浮力调节软件;单元测试;单元测试充分性准则;单元测试策略;Testbed

Research on Unit Testing Techniques of Buoyancy Adjusting Software

Li Lei
(No. 710 Research Institute of CSIC, Yichang 443000, China)

Abstract: In order to improve the efficiency of embedded software unit testing, and achieve good testing result in the meantime. It has practical significance to make adequacy criteria and strategy of unit testing and use automatic unit testing tool to accomplish unit testing. Take the buoyancy adjusting software for example, by studying the adequacy criteria of unit testing based on control flow and unit testing requirement of the buoyancy adjusting software, the unit testing adequacy criteria of buoyancy adjusting software has been put forward. Meanwhile, considering the influence of cyclomatic complexity and knots on realizing functions correctly, the unit testing strategy based on priority has been made. The dynamic testing environment is established by Testbed, which is an automatic unit testing tool. By analyzing code coverage, Testbed helped to create test cases to meet the adequacy criteria requirements, thereby realized the automatic unit testing of the buoyancy adjusting software. With the help of automatic unit testing tool, adequacy criteria and strategy of unit testing, rigorous and efficient unit testing has been accomplished in the end.

Keywords: buoyancy adjusting software; unit testing; adequacy criteria of unit testing; unit testing strategy; Testbed

0 引言

近年来,嵌入式系统的发展异常迅速,深入渗透到航空、航天、核电、汽车、电子等关键领域,导致嵌入式系统复杂度越来越高,安全性要求也越来越高。出于成本及安全性等多方面的考虑,原来用硬件实现的功能现在都尽可能地改用软件来实现。现在嵌入式系统中的增值功能也主要靠软件完成,导致软件所占的比重越来越大。在软件生命周期中随着软件开发的进行,发现错误以及修正错误的成本越来越大。经验表明,对代码开发早期进行集中测试可以减少测试的工作量与开发成本。单元测试在初始编码阶段提供了识别和改正错误的方法,从而降低了开发软件的风险,帮助开发人员优化设计并鼓舞了软件开发团队的士气[1]。

由于时间和成本的约束,需要明确应该对哪些模块进行重点测试,通过分析和计算影响模块的复杂度的度量元,建立单元测试的优先级,合理划分单元测试的时间,使单元测试更加具有针对性。穷举的单元测试是不可能的,需要确定合理的单元测试充分性准则,并根据该准则决定何时停止设计更多的测试用例。

当前,软件已成为嵌入式系统的重要部分,由于软件自身的发展和市场成本的压力,软件必然是越来越复杂,要求软件的开发时间却越来越短,以至于对测试产生巨大压力。充分结构化的测试过程已无法独立在短时间内满足质量要求,因此必须使用测试工具,以保证测试进度和质量[2]。使用自动化测试工具可以降低我们对驱动模块的需求,流程分析工具可以列举出程序中的路径、找出从未被执行的语句,以及找出变量在赋值前被使用的实例[3],从而使测试过程中的枯燥劳动减至最小,测试效率和效果得到提升。

本文以浮力调节软件为例,研究并实践了控制流单元测试充分性准则,并且根据不同函数的复杂度,制定了基于优先级的单元测试策略。最后,结合 Testbed 自动化测试工具,针对浮力调节软件,设计并完成了严格而高效的单元测试。

1 浮力调节软件单元测试充分性准则

单元测试就是验证软件单元的实现是否和单元的说明完全一致的相关联的测试活动组成的。根据软件单元的说明文档(在实践环境中,该文档可能是一种说明语言,或者是一种自然语言或是状态转换图)编写测试用例,对重要的接口、局部数据结构、边界条件、独立路径和错误处理路径,通过代码检

收稿日期:2016-09-18; **修回日期**:2016-11-21。

作者简介:李 磊(1987-),男,湖北宜昌人,硕士,主要从事软件测试方向的研究。

查或执行测试用例有效地进行测试[4]。

单元测试充分性准则被用来判断设计的测试用例是否有效，并决定何时停止设计更多的测试用例，从而使得单元测试过程严格而高效。此处描述的单元测试充分性准则实质上是设计测试用例时常用的白盒测试方法，即逻辑覆盖准则。在使用 Testbed 自动化测试工具进行代码覆盖率测试时，每完成一个测试用例，都会直观的反映在各种覆盖率变化上，如果设计的测试用例没有导致覆盖率的任何变化，则该测试用例无效，当符合该模块单元测试充分性准则对应的覆盖率达到 100％时，就不需要再增加新的测试用例。根据实际进行单元测试所使用的单元测试充分性准则，仅介绍最常用的 3 种基于控制流的单元测试充分性准则。

1.1 基于控制流的单元测试充分性准则

语句覆盖（Statement Coverage，SC）：设计足够的测试用例，使被测试程序中每条语句至少执行一次。语句覆盖是较弱的准则，对于组合逻辑判断条件，某些判断条件出错不会被发现。

判定覆盖（Decision Coverage，DC）：设计足够的测试用例，使得每一个判断至少有一个为真和为假的输出结果，或者说每条分支路径都必须至少遍历一次。判定覆盖通常可以满足语句覆盖，但并不是所有情况都满足。判定覆盖是较强一些的准则，但仍然相当不足，其原因是“与”和“或”表达式中某些条件的结果可能会屏蔽掉或阻碍其他条件的判断。比如，如果“与”表达式中有个条件为“假”，则无须计算该表达式中的后续条件。同样，如果“或”表达式中有个条件为“真”，那么后续条件也无须计算。因此判定覆盖准则不一定会发现逻辑表达式中的错误。

修正条件/判定覆盖（Modified Condition/Decision Coverage，MC/DC）：该准则要就设计足够多的测试用例，使入口和出口的每个点至少被调用一次，从而每个判定到所有可能的结果值都要至少转换一次，判定中的每个条件可以独立地显示出它对判定结果的影响。该准则能够部分解决判定覆盖的问题，满足多重条件覆盖准则的测试用例集，同样满足语句覆盖和判定覆盖。

1.2 浮力调节软件单元测试充分性准则

软件测试领域存在 2－8 原则，即 80％的错误是由 20％的错误引起的，80％的测试成本和时间花在 20％的模块中[5]。所以根据单元测试充分性准则合理分配测试投入，才能达到更快的进度，更低的成本，更高的质量。

根据软件的要求程度，设计满足覆盖率要求的测试用例。对于嵌入式应用中的一些要求可靠性极高的关键模块，需要满足修正条件/判定覆盖。开展修正条件/判定覆盖测试难度很大，会占用大量成本，所以对于一般模块，只需要满足语句覆盖和判定覆盖即可。

浮力调节控制器包括主、从控制器两部分，控制器架构为一主二从模式，主控制器包括主控制板和从控制板，从控制器只包含一个从控制板。浮力调节控制器主控制板和从控制板处理器均采用 TI 公司单片机 MSP430F5438A 单片机，设计上考虑兼容性，以单片机自带的串口通信模块、定时器模块等为基准，软件基于 TI 公司单片机的 IAR 编译环境开发。浮力调节控制器主控软件主要实现与自动驾驶仪通信、从控软件通信、CTD 数据采集、受控模式、浮力自适应模式、定深模式及系统状态信息合成反馈等功能，主控软件程序流程图如图 1 所示。

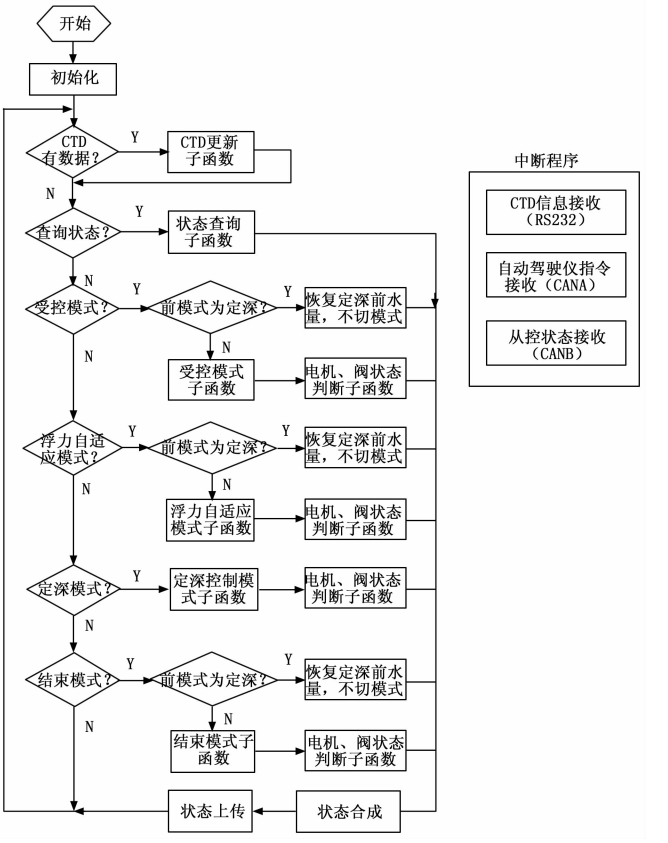

图 1 主控软件程序流程图

根据软件规格说明中描述的各模块实现功能的重要性，得到浮力调节软件单元测试充分性准则详见下表 1。

表 1 浮力调节软件单元测试充分性准则

单元测试需求	被测试模块	模块功能	单元测试充分性准则
1 级核心代码	受控模式函数，浮力自适应模式函数，定深模式函数	识别、发送控制指令，解析、反馈系统状态；计算补偿水量；定深控制目标深度	修正判定条件覆盖 100％
2 级常用但易出错代码	主函数，系统状态合成函数，工作状态合成函数，指令发送和状态回收函数	软件主框架；处理并合成系统状态；合成工作状态；发送控制指令并回收从控状态	语句覆盖 100％ 判定覆盖 100％

2 基于优先级的浮力调节软件单元测试策略

2.1 浮力调节软件关键模块优先级的度量元

经验表明，程序中一个模块越复杂，其存在缺陷的可能性也就越大。根据浮力调节软件各个模块的复杂程度，可以对所

有模块进行优先级排序，并以此建立基于优先级的单元测试策略。程序模块的复杂程度主要体现在以下两个方面。

2.1.1　环路复杂性

环路复杂性可以采用 McCabe 复杂性度量法[6]来计算，一般用圈复杂度描述，记为 $V(G)$。圈复杂度用来衡量一个程序模块所包含的判定结构的复杂程度，如果一个程序模块的圈复杂度较大，则说明该模块的代码质量可能较低，对其进行测试和维护的难度也较大。经验也表明，程序可能存在的缺陷数量，与圈复杂度有着很大相关性[7]。

圈复杂度计算公式如下：

$$V(G)=e-n+2$$

其中：e 表示程序控制流图中边的数量，n 表示程序控制流图中节点的数量。圈复杂度还有更直观的计算方法，因为圈复杂度所反映的是“判定条件”的数量，所以圈复杂度实际上就是等于判定节点的数量再加 1，也即控制流图的区域数。其对应的计算公式如下：

$$V(G)=\text{区域数}=\text{判定节点数}+1$$

2.1.2　控制流节点数

在程序中，当两个控制流跳转发生交叉时，就产生了控制流节点。控制流节点的数量可以用来衡量程序模块的复杂程度，记为 H_{knots}。如果程序中一个模块的控制流节点数越多，则意味着该模块的复杂度可能越高。通过对代码结构进行重构可以减少控制流节点数目，从而使该模块的复杂程度得以降低。

2.2　浮力调节软件单元测试模块优先级度量数学模型

浮力调节软件单元测试模块优先级度量数学模型如下：

$$H_{\text{priority}}=V(G)+H_{\text{knots}}$$

可以看出，H_{priority} 是被测模块复杂度度量元的加权值，H_{priority} 越大，意味着该模块的代码复杂性越大，存在缺陷的可能性也就越大。通过计算程序中各模块的优先级度量 H_{priority}，并按该度量值大小进行排序，建立基于优先级的单元测试策略，从而合理地分配各个模块的测试资源。

2.3　基于优先级的浮力调节软件单元测试策略应用

2.3.1　基于优先级的浮力调节软件单元测试策略应用过程

（1）首先利用 Testbed 测试工具对将要进行单元测试的所有函数进行静态分析，得到每一个被测函数的圈复杂度 $V(G)$ 和节点数 H_{knots}；

（2）根据浮力调节软件单元测试模块优先级度量数学模型公式计算得到每一个函数的加权值 H_{priority}，然后按加权值 H_{priority} 大小进行优先级排序，得到浮力调节软件代码的复杂度分布；

（3）合理分配各个函数的测试工作量和时间，保证测试的进度，尽可能多的对重要并且复杂度高的函数进行测试。

2.3.2　基于优先级的浮力调节软件单元测试策略应用实例

经过 Testbed 自动化测试工具对浮力调节软件进行静态分析，在软件质量分析报告中，可以看到每一个要进行单元测试的函数圈复杂度和节点数[8]，如图 2 所示。

根据前面提出的浮力调节软件单元测试模块优先级度量数学模型公式，加权确定每一个函数单元测试的优先级，并按优先级排序，得到浮力调节软件各函数优先级顺序如表 2 所示。

表 2　浮力调节软件各函数优先级顺序

函数名	圈复杂度	节点数	加权值	优先级
buoyancyCmpMode	181	101	282	1
SendMsg_RecheckStat	81	58	139	2
ModeStop	82	45	127	3
main	38	25	63	4
Creat_Moto_Stat	17	12	29	5
seccpre	14	9	23	6
CTD_init	21	2	23	7
CTDupdate	13	9	22	8
ControlledMode	19	2	21	9
SecCalibration	17	4	21	10
CreateState	13	4	17	11
Init_PWM	4	2	6	12
fzxqwaterbak	3	0	3	13
Create_Water_Level	3	0	3	14

3　浮力调节软件自动化单元测试实施过程

根据浮力调节软件设计说明，该软件基于 TI 公司单片机的 IAR 编译环境，其单元测试实施过程如下：

（1）首先需要使用 TBconfig 来配置 IAR 编译环境下的仿真单元测试环境，然后再 Testbed 上设置编译器为 IAR；

（2）打开单元/集成级测试工具 TBrun，该工具可以自动生成软件测试驱动和桩模块，从而节省时间，使测试人员将重点放在设计测试用例上，提高软件测试效率，同时提高软件测试人员积极性[9-10]。针对要测试的函数生成测试用例，再对测试用例中调用的函数进行打桩操作。

Procedure	Knots	Cyclomatic Complexity
main	25 (F)	38 (F)
fzxqwaterbak	0 (F)	3 (F)
seccpre	9 (F)	14 (F)
CTDupdate	9 (F)	13 (F)
BuoyancyCmpMode	101 (F)	181 (F)
CreateState	4 (F)	13 (F)
ControlledMode	2 (F)	19 (F)
ModeStop	45 (F)	82 (F)
CTD_init	2 (F)	21 (F)
SecCalibration	4 (F)	17 (F)
Init_PWM	2 (F)	4 (F)
SendMsg_RecheckStat	58 (F)	81 (F)
Create_Moto_Stat	12 (F)	17 (F)
Create_Water_Level	0 (F)	3 (F)
clear_var	0 (F)	1 (F)
delay	1 (F)	2 (F)
timerout	0 (F)	1 (F)
math_dianyao	0 (F)	1 (F)
midujisuan	0 (F)	1 (F)

图 2　函数圈复杂度和节点数

（3）查看被测函数的控制流图，结合程序规格说明并根据控制流图设计对应的测试用例，得到输入值以及预计输出值。以函数 fzxqwaterbak（void）为例，该函数代码如下所示：

```
void fzxqwaterbak(void)
{
```

```
comandconflict=1;
Seepage_num=0;
CreateState();
if (((cong1_state.dta[1] & 0x1F)==0x00) &&(cong2_state.dta[1] & 0x1F)==0x00))
{
fzxqstart=0x00;
}
cana_sendMessage(cong_state);
cana_sendMessage(cong_state2);
cana_sendMessage(cong_state3);
}
```

在输入输出窗口中仅留下对测试结果有影响的变量，并将前面得到的输入和输出值填入到相应位置。执行该测试用例，最终会显示该测试用例是否通过，测试用例输入输出及测试结果如图 3 所示。

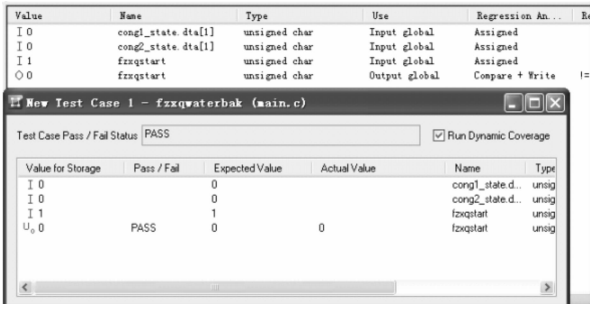

图 3 测试用例输入输出及测试结果

(4) 在 File View 窗口中查看覆盖率，通过覆盖率变化可以直观了解对该函数的测试是否已经达到充分性准则的要求，如果没有达到，则查看当前的控制流图，查看有哪个分支没有覆盖到，控制流图覆盖情况如图 4 所示。

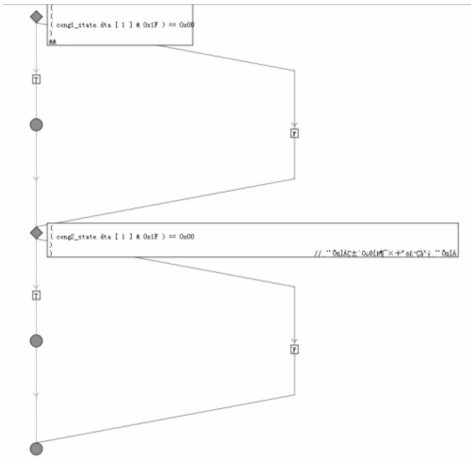

图 4 控制流图覆盖情况

实心表示已经覆盖到的语句和分支，线段面积表示未覆盖的部分。如果未覆盖到的原因是缺少相应测试用例，则新增测试用例，重复步骤 2～4，否则分析原因并记录；

(5) 分析测试结果，记录相关问题，完成测试报告。

4 总结

单元测试技术对大型程序尤其有用，通过这种技术来测试程序的组件如子程序、子函数、类以及过程。单元测试和其它类型的测试有着同样的目标：找出程序不满足规格说明书的地方。

单元测试中测试用例的设计过程应该是，使用一种或多种白盒测试方法分析模块的逻辑结构，然后使用黑盒测试方法对照模块的规格说明以补充测试用例。

一个测试用例除了需要对输入数据进行描述外，还需要对在相应输入数据下的正确输出结果做出精确描述。这样才能避免某个似是而非、实际上是错误的结果可能会被解释成正确的结论的事情发生。

当测试用例造成模块输出的实际结果与预期结果不匹配的情况时，存在两个可能的解释：要么该模块存在错误，要么预期的结果不正确。为了将这种混乱降低到最小程度，应在测试执行之前对测试用例集进行审核或检验。

在进行单元测试之前，需要仔细阅读程序规格说明书和模块的源代码。对测试系统有了一个大致了解以后，提前针对各个模块制定对应的单元测试充分性准则和基于优先级的单元测试策略，从而更为合理地分配单元测试工作量，达到很好的测试效果。

5 结束语

通过应用本文所论述的单元测试充分性准则和基于优先级的单元测试策略，并借助自动化测试工具 Testbed，测试人员目标性更强，人员分配和时间分配更为合理，单元测试的效果得到了提升，从而高效而严格的完成了对浮力调节软件进行的单元测试。

参考文献：

[1] 冷知见. 软件自动化测试方法的研究与应用 [D]. 武汉：武汉理工大学，2011.

[2] 杨一波. 无人机飞行控制软件测试技术研究 [D]. 南京：南京航空航天大学，2008.

[3] Glenford J. Myers，Tom Badgett，Corey Sandler. 软件测试的艺术 [M]. 北京：机械工业出版社，2012.

[4] 袁玉宇. 软件测试与质量保证 [M]. 北京：北京邮电大学出版社，2008.

[5] 肖利琼. 软测之魂：核心测试设计精解 [M]. 北京：电子工业出版社，2011.

[6] 金维佳，施小敏. 基于嵌入式软件的覆盖测试问题研究 [J]. 信息技术，2011 (4)：117－120.

[7] 刘媛媛. 飞行控制软件单元测试的实施 [D]. 北京：北京邮电大学，2010.

[8] 胡丹瑞. 基于 LDRA Testbed 的软件静态测试研究与实现 [J]. 计算机安全，2012 (6)：69－71.

[9] 朱昭俊，蒋文丹，苏 赛. TBrun 在单元测试中的应用 [J]. 信息安全与技术，2013 (2)：85－96.

[10] 徐润德，陈 亚，赵慕奇. 基于 LDRA Testbed 的软件单元测试 [J]. 海军航空工程学院学报，2011，26 (3)：356－360.