文章编号：2096-1472(2019)-08-17-04
DOI:10.19644/j.cnki.issn2096-1472.2019.08.006

嵌入式软件单元测试方法研究

刘 佳[1,2]，杨永文[1,2]，李金华[1,2]
(1.中国船舶重工集团公司第七一一研究所，上海 201108；
2.低速机国家工程试验室，上海 201108)

摘 要：嵌入式软件通常采用交叉开发的方式，尽早进行软件测试可以及时发现软件开发初期的不足。单元测试是嵌入式软件开发过程中最基础级别的测试活动。本文对嵌入式软件的单元测试进行了分析，主要包括测试工具的选择、测试内容的梳理和测试过程的优化，重点阐述了注意事项及原则。通过总结分析，提出了适宜操作的嵌入式软件单元测试方法。实践证明，该方法大大提高了软件测试水平和软件产品代码的质量。

关键词：测试计划；单元测试；测试过程；嵌入式软件
中图分类号：TP311.5 **文献标识码**：A

Research on the Unit Testing Method for Embedded Software

LIU Jia[1,2],YANG Yongwen[1,2],LI Jinhua[1,2]
(1.*Shanghai Marine Diesel Engine Research Institute,Shanghai* 201108,*China*;
2.*Low Speed National Engineering Laboratory,Shanghai* 201108,*China*)

Abstract: As the cross-development method is usually applied for embedded software,software testing is conducted early,so as to promptly find out the defects and problems at the early stage of software development.Unit testing is the basic testing in the process of embedded software development.This paper analyses embedded software unit testing,including the choice of testing tools,the analysis of testing content and the optimization of testing process,mainly describing the notes and principles.Through the comprehensive summary and analysis,the paper proposes an easy-to-operate and suitable unit testing method for embedded software.Practice indicates that this method greatly improves the level of software testing and software product quality.

Keywords:testing plan;unit testing;testing process;embedded software

1 引言(Introduction)

由于嵌入式软件运行在嵌入式计算机系统中，且实时性强。开发人员在编写完一段代码后，很难立即有效地去运行程序，从而导致缺陷越积越多。软件单元测试是软件开发过程中的一项基本活动，也是软件验证、确认的重要手段。通过一系列的单元测试，可改进软件质量，有效地减少软件漏洞的数量，降低程序的风险，找到软件中可能潜在的问题，提高代码的规范性、稳定性、可靠性。

单元测试针对单个函数的测试，工作量大，处于施工设计的重要阶段，时间紧，且发现的软件问题“严重等级”较低，如果依靠人工进行代码审查、静态分析，代码打桩编写执行用例，成本较高，流程不清晰不可控，难以保证测试充分性。本文根据嵌入式软件的特点，基于多年的测试工作经验，参考相关的标准，提出了一套适用的单元测试流程，包含测试工具、测试内容、测试过程的标准化、规范化等要求。

嵌入式软件测试环境分为目标机环境和宿主机环境。但在单元测试层面上，对于嵌入式软件来讲，除非特别指定在目标机环境下进行，都可以在宿主机环境进行。

2 单元测试工具(Unit testing tools)

单元测试的本质是针对代码进行测试，工作量和难度都比较大。采用合适的工具及有效的方法，可或多或少地实现“自动化”，极大提高测试效率，满足单元测试的覆盖率和

准确度要求，减少工作量，降低成本。

常见的单元测试工具有PQRA QAC、C++Test、Logiscope、Tessy、LDRA Testbed、PolySpace、Cantata等。

本文采用QAC软件作为静态测试工具，该工具是针对C代码的深度流静态分析器，通过内置的标准规则，以警告消息方式突显代码中存在的违规现象，帮助开发者改善软件开发质量[1]。采用Tessy软件作为动态测试工具，针对嵌入式语言，能够快速进行驱动模块桩模块配置，用例设计可视化，并且可根据测试要求的不同，选择不同的测试环境。Tessy可以实现代码实际运行结果和测试用例中设定的预期结果的自动对比，检查代码功能正确性。在进行功能测试的同时，还对代码测试的覆盖率进行了统计。

3 单元测试内容(Content of unit testing)

在单元测试开展之前，需要对测试过程的各个阶段进行方法设计，以作为具体工作的指导依据。嵌入式软件单元测试要求可分为静态测试阶段和动态测试阶段。

3.1 静态测试阶段

静态测试是借助测试工具或人工检查的方法，对被测程序进行特性分析，查找软件编码的错误，并对软件单元的静态度量指标进行分析。由于静态测试不需要编译或运行代码，因而也不会受到软件内部其他缺陷模块的影响。

静态测试阶段可分为代码审查和静态分析两部分。

3.1.1 代码审查

代码审查主要工作为检查代码和设计的一致性、代码执行标准的情况、代码逻辑表达的正确性、代码结构的合理性、代码的可读性等[2]。根据适用的规则，项目组内部定制合适的代码审查单，规定内容及标准。

QAC作为常用的静态分析工具，可实现代码审查部分工作的自动完成，如变量初始化定义是否正确、是否存在不能执行的代码等。除此之外的审查工作均须人工完成并填写至代码人工审查单中。

3.1.2 静态分析

静态分析可分为数据流分析、控制流分析、表达式分析、相关度量指标分析和代码质量的分析等，为动态测试提供辅助参考的信息。

静态分析一般来说应满足四方面准则要求：a.通用语言编码要求；b.国家级、行业级标准规范；c.单位内部组织级编码规范；d.适用于具体软件的项目级编码要求。

QAC软件能够通过内嵌工具及插件工具完成以上a类、b类准则的自动检查，但c类、d类准则由于具有特殊性，QAC不能完全覆盖，对于该部分的静态分析工作均须人工完成，此部分工作可与代码审查同时进行，分析结果反映在人工代码审查单中。

静态测试条款详见表1。

表1 静态测试条款

Tab.1 Static testing terms

项目名称	规范要求
命名规则	禁止形参名与全局变量、类型或标识符同名
	标识符的有效字符须多于5个字符但小于31个，for循环的计数器可以是一个字符
	标识符不以“_”开头或结尾
	指针类型标识符应以“ptr”结尾
	……
注释	一般情况下，源程序有效注释率必须在20%以上
	注释格式统一，使用“/*… …*/”
	……
降低度量指标	函数参数不应超过7个
	函数的规模尽量限制在200行以内
	不得使用goto语句
	控制结构的嵌套不能超过6层
	……
程序多余物	各判断分支是否都得到处理
	是否存在执行不到的代码
	是否存在不合理的循环结构
	是否存在无效的函数参数
	……
表达式分析	表达式中的括号使用正确
	表达式的除数是否为0
	……
……	……

3.2 动态测试阶段

在动态测试中，包含的测试类型大致分为接口测试、局部数据结构测试、功能测试、边界测试、运算测试、错误处理测试、独立路径测试等。根据接口测试和局部数据结构测试的内容要求[3]，可适当并入静态测试，测试结果反映在QAC报告和人工代码审查中。

由于开发团队与测试团队相互独立，在过程中要求开发人员每完成一个单元均提交测试显然不符合实际，一般在某个功能相对独立的软件开发完毕后才进行单元测试。此情况下，采用由底向上的集成单元测试方法即可以利用现有代码省去桩函数的编写工作，也可利用测试工具完成驱动函数自动设置工作，与自顶向上、多层结构等测试方法相比，能够减少工作量并提高测试的准确性。

由底向上的集成单元测试的方法一般分为两类[4]。

第一类方法：在集成单元测试中，考虑内部调用关系，并定义其调用的各独立单元间存在两种关系——并行关系和串行关系。并行关系是指在集成单元中包含的各独立单元间无直接关系，仅是顺序排列。串行关系是指在集成单元中包含的各独立单元间存在输入输出关系，即一个独立单元的输出为另一个单元的输入。对于仅含有一个独立单元的集成单元，作为并行关系处理。对于含并行关系的集成单元，单元

测试仅测试其内部独立单元的调用关系；对于含串行关系的集成单元，在测试其内部独立单元的调用关系的同时，还应对各独立单元的输入输出关系设计测试用例进行测试。以图1的单元结构为例，其测试方法如图2所示。

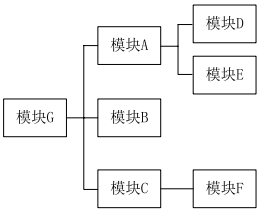

图1 单元结构图

Fig.1 Unit structure

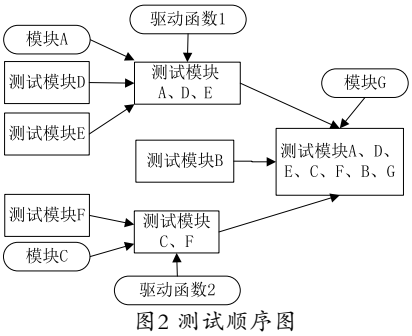

图2 测试顺序图

Fig.2 Testing sequence

第二类方法：在集成单元测试中，不考虑单元间的调用关系，仅测试单元本身的功能、边界等方面，涉及调用关系的测试待后续测试阶段进行。

在实际工作中，这两种方法对于同一软件的不同版本中均进行了应用，结果表明，第一类方法更易发现问题。分析原因为该软件调用关系复杂且单元本身逻辑关系清晰，测试重点应为基于复杂调用关系组成的复杂逻辑关系。但对于逻辑关系复杂且调用关系简单清晰的软件，第二类方法更为适合。

4 单元测试过程(Unit testing process)

单元测试过程分为编制测试计划、执行测试、编制测试报告三步[5]，如图3流程图所示。图4—图7为图3局部的详细说明。

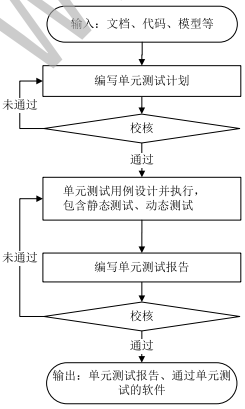

图3 单元测试流程简图

Fig.3 Unit testing flow chart

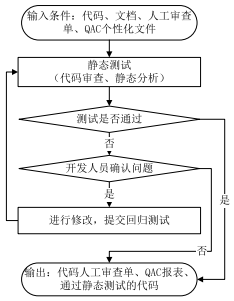

图4 静态测试流程图

Fig.4 Static testing flow chart

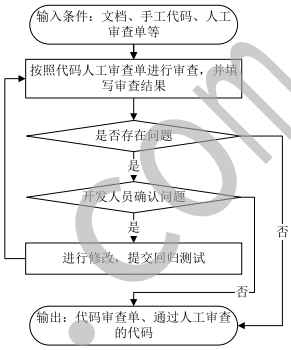

图5 代码审查测试流程图

Fig.5 Code review testing flow chart

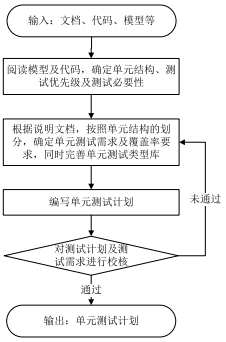

图6 单元测试计划阶段流程图

Fig.6 Unit testing-planning stage

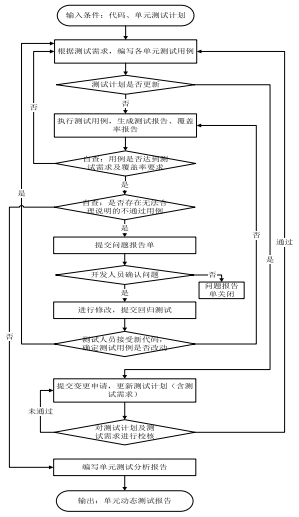

图7 单元测试执行阶段流程图

Fig.7 Unit testing implementation flow chart

4.1 测试计划

单元测试计划为单元测试工作开展提供依据，具体规定了测试环境、测试需求、测试进度等方面。在编写过程中，按如下步骤进行可有效提高工作效率。

具体步骤：

(1)明确单元测试思路，即测试方法。在测试的各阶段，确定测试方向及内容，保证全面合理。

(2)对单元整体结构进行梳理，同时进行测试必要性分析。对结构相同或相似的单元可归为一类，仅从该类中挑选代表单元进行测试即可，同时应注意此时单元测试中发现的问题不仅是本单元的问题，而是一类问题。另外，对明显没有缺陷的代码不需编写测试用例。

(3)定义与完善测试类型库。由于单元测试的常用测试类型比较固定统一，故可采用单元测试类型库的形式，对测试类型进行细化，明确相应测试方向及具体内容，对后续测试工作起指导作用，详见表2。

(4)依据详细说明或模型说明及代码，编写单元动态测试需求。测试需求的编写原则：以功能测试为基础，充分考虑边界情况，兼顾其他测试类型，尽可能提高代码测试的覆盖率，测试应充分、完整，及时发现其中的缺陷错误[6,7]。

表2 单元测试类型库

Tab.2 Type library for unit testing

测试类型	测试项目	描述
功能测试	功能测试	基于函数的功能说明，测试函数功能实现的正确性
接口测试	调用与被调函数间实参与形参的一致性分析	形参与实参对应的变量类型是否一致
		形参与实参个数和顺序是否一致
		是否把常量当作变量传递内容
	……	……
边界条件测试	控制条件边界测试	与设定值比较的边界值测试
		出现if或switch等条件判断的边界值测试
		测试执行到最后一次循环体
	……	……
逻辑测试(覆盖率测试)	语句覆盖(SC)	被测代码中每个可执行语句是否被执行到
	分支判断覆盖测试(DC)	程序中每个判定的取“真值”“假值”分支至少执行一次
	修正的条件判断覆盖(MC/DC)	程序中每个判断的使用可能的条件的取值至少执行一次
	……	……
……	……	……

4.2 测试执行

测试用例的编写在动态测试工具Tessy中完成，测试用例依据测试需求，着重测试数据的输入设计。其编写原则为：

(1)为单元运行起来而设计测试用例；

(2)正向测试；

(3)逆向测试；

(4)满足特殊需求；

(5)代码覆盖；

(6)复杂度适中；

(7)中断函数、死循环while(1)等需要对代码做适当修改；

(8)对发现错误较多的函数，应需进行更深入的测试。

此过程中，存在对测试计划进行修改的可能。执行过程及结果可通过Tessy自动报告生成。

动态测试过程中发现的软件问题以《软件问题报告单》的形式提出。其中软件问题以问题类型分类为：设计问题、程序问题、文档问题、编码问题及其他。以问题等级分类为：关键缺陷，严重缺陷，一般缺陷及建议改进。

4.3 测试报告

测试工作执行完毕后，按照文献[8]要求进行单元测试报告的编写，同时对Tessy生成的报告进行归类整理，应重点对测试中发现的软件问题进行全面分析。完成的测试报告及时发送给干系人以便下一步工作的开展。

4.4 测试校核

为了提高测试工作的有效性及全面性，在过程的恰当位置设置了两处校核步骤：测试计划校核、测试报告校核。

测试计划校核也可称为测试需求校核，需要在测试提交前校核完成[9]。校核人员应核对测试需求是否覆盖软件说明文档，测试类型是否全面，重点校核功能测试描述是否清晰，用例的前提条件、执行步骤、输入数据、预期输出是否正确。边界测试、接口测试可参考测试类型库中定义的内容，校核方便。

测试报告校核重点为核对测试用例对测试需求的覆盖是否全面，问题解决是否正确，对遗留问题处理是否恰当等。

5 结论(Conclusion)

单元测试被广泛应用于软件测试的基础环节。本文结合工作实际，提出了较为实用的嵌入式软件测试方法，包含了测试技术、测试管理等方面，改变了以往单元测试随机性、无序性等弊端。在实际工作中采用该方法进行单元测试，保证了软件测试的充分性、规范性，便于软件问题追踪闭环，提升了软件测试管理，提高了软件测试水平和软件产品代码的质量。

(下转第13页)