# 嵌入式系统软件测试

张成志[1] 谢 涛[2] 曹东刚[3] 张 路[3]
[1]香港科技大学
[2]美国北卡罗莱纳州立大学
[3]北京大学

**关键词：**测试覆盖标准 测试预言

## 引言

嵌入式系统已应用于工业、国防、医疗、教育及日常生活等领域，渗透到社会的各个角落，几乎达到了无处不在的地步。无论是简单的机顶盒控制系统，还是大型分布式飞机控制系统。所有嵌入式系统在结构上都是由硬件设备和软件组成的，并且具有实时性。实时性要求系统在限定的时间内对事件作出响应。由于嵌入式系统失效可能会造成严重后果，因此测试非常重要，尤其要对嵌入式系统软件进行严格的和充分的测试，以保证嵌入式系统的质量。

由于嵌入式系统的如下特点，它的测试技术[1]也有所不同：

**微型化** 嵌入式系统在系统功能、体积等方面均呈现微型化特征。嵌入式硬件一般针对某项特定应用来设计，通常具有体积小、集成度高的特点，呈现微型化态势；嵌入式系统软件则针对特定领域、特定用户的个性化需求，与硬件紧密结合，恰当地满足应用需求。

**紧耦合** 嵌入式系统通常面临受限的软/硬件资源条件，同时在实时性、安全性、可靠性、功耗等非功能属性方面有着严格的要求。这些限制条件使得嵌入式系统呈现紧耦合特征，嵌入式系统软件的各功能模块应紧密结合满足应用的功能及非功能需求。

**专用性** 嵌入式系统的应用领域广，用户需求多样。为满足用户对产品功能、性能、功耗等指标的特定需求，嵌入式系统在软、硬件两方面均呈现出专用性。

上述特点使嵌入式系统软件的测试面临诸多难点，例如，如何仿真或模拟嵌入式硬件的能力，为测试提供接近真实环境的输入和输出数据；如何验证实时性、安全性、可靠性、功耗等非功能指标；如何进行压力测试等等。此外，测试人员需要更成熟的测试工具和方法的指导。

## 无线传感器软件实例

nesC[4]是一种用于无线传感器网络的常见编程语言。它广泛应用于跟踪移动目标和监测生态系统。这些系统中的传感器运行于一个很轻量级的TinyOS操作系统[3]上，可依靠电池电量运行很长时间。为了节省能源，这些传感器大部分时间在休眠，只是在有相关的环境事件发生时才被中断唤醒以做出反应。为了保证高响应能力，中断可以强制取代正常任务的执行。中断的广泛使用、其对正常任务执行的强制取代以及可延迟的任务执行都给无线传感器网络编程带来了挑战。无线传感器网络中会碰到的各种各样的问题，例如数据争用[5]、接口合同违反[6]以及内存安全[7]。

nesC应用程序主要由中断驱动。当一个中断发生，TinyOS将其视为一个事件并把这个事件传播给目标nesC应用程序。在应用程序中，一种用关键字

async（代表异步）标识的子例程以异步方式响应这个中断事件。这种子程序叫做事件处理程序。为了保证高响应能力，事件处理程序通常会设计成比较轻量级的。在中断触发复杂的计算时，响应这个中断的事件处理程序会衍生（post）额外的任务来进行复杂的计算。与传统的函数调用/返回的语义不同，任务并不是在它衍生时就立刻执行的。相反，衍生机制只是简单地将任务添加到任务队列的末尾，并立即将控制返回给该事件处理程序。直到没有未完成的中断时，被衍生任务才开始执行。另外，衍生的任务是基于衍生的顺序依次被执行的。一个事件处理程序可抢占一个任务的执行。在事件处理程序完成之后，被抢占的任务的执行才得以恢复。

```
 #1: module OscilloscopeM {
#2:       providesinterface StdControl;
 #3:       uses{
 #4:            interface Leds;
 #5:            interface ADC as PhotoSensor;
 #6:            interface ADC as TempSensor;
#7:           interface SendMsg; /* other interfaces */
#8:       }
 #9: } implementation{
#10:       uint8_t packetReadingNumber;                    // shared variable
#11:       // uint8_t lock = 0;
#12:       TOS_Msg msg;// shared variable
#13:       taskvoid task4() {/* do something */}
#14:       taskvoid task3() {/* do something */}
#15:       taskvoid task2() {
#16:            OscopeMsg *pack;
#17:            static uint32_t i;
#18:            uint32_t start = i;
#19:   atomic{                                          // atomic block
#20:                 pack = (OscopeMsg *)msg.data;
#21:                 packetReadingNumber = 0;
#22:                 for (; i < start + 100000 && i < 200001; i++) { /* process sensing data */ }
#23:            }
#24:            if (i < 200001) {
#25:                 post task2(); // post another instance to process remaining data
#26:                 return;
#27:            }// else (finish processing the data): send the packet
#28:            i = 0;
#29:            pack->channel = 1;
#30:            pack->sourceMoteID = TOS_LOCAL_ADDRESS;
#31:            if (call SendMsg.send(TOS_UART_ADDR, sizeof(OscopeMsg), &msg)) {
#32:                 call Leds.yellowToggle();
#33:            }
#34:       }
#35:       taskvoid task1() { /* do something */
#36:            post task3();
#37:            post task4();
#38:       }
#39:       asyncevent result_t PhotoSensor.dataReady(uint16_t data) {
#40:            OscopeMsg *pack;
#41:            // atomic { if (lock == 1) return; }
#42:            pack = (OscopeMsg *)msg.data;
#43:            atomic{
#44:                 pack->data[packetReadingNumber++] = data;
#45:                 if (packetReadingNumber == BUFFER_SIZE) {
#46:                      // lock = 1;
#47:                      post task2();
#48:                 }
#49:            }
#50:            return SUCCESS;
#51:       }
#52:       event result_t SendMsg.sendDone(TOS_MsgPtr sent, result_t success) {
#53:            // atomic lock = 0; /* notify the availability of the message buffer */
#54:            return SUCCESS;
#55:       }
#56:       asyncevent result_t TempSensor.dataReady(uint16_t data) { // do something
#57:            post task1();
#58:            return SUCCESS;
#59:       } /* other commands, events, and tasks */
#60: }
```

图1 OscilloscopeM模块

图1模块改编自TinyOS系统中的OscilloscopeM模块（用OM来表示）。OM可用于控制室内环境条件。在图中，我们缩减了其中的代码，以便解释一个nesC故障。在nesC中实现程序逻辑的模块叫做组件。在#10和#12行，OM声明了两个共享变量packetReadingNumber 和msg，在#13和#38行之间定义了四个任务task1、task2、task3、task4，在#39和#59行之间实现了三个事件处理程序PhotoSensor.dataReady、SendMsg.sendDone、TempSensor.dataReady。OM请求底层硬件平台来采集照片和温度传感器的读数，把读入的数据置于一个缓冲区，然后处理缓冲区的数据，并把处理后的结果发到一个指定的基站（base station）。

典型的应用场景是：新的感知数据到来后，OM通过事件处理程序dataReady采样数据，并将它们存储在一个共享缓冲区。此事件处理程序可以被多次

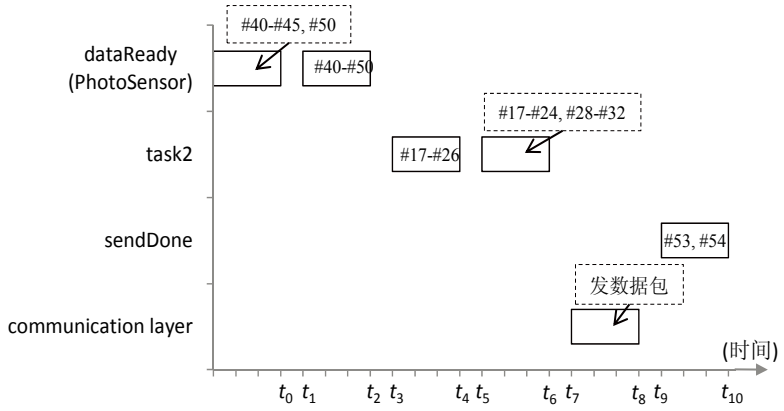



图2 一个典型应用场景的时间线图

调用，直到缓冲区的大小达到预定的阈值。当达到阈值时，task2被衍生用于处理缓冲区的数据。在执行时，task2调用send命令来发送一个存有处理结果的数据包到一个基站。send命令是由底层通信层来执行的。成功发送该数据包后，事件处理程序sendDone将被调用通知相关的用户应用程序。图2描述了BUFFER_SIZE是2时的应用场景。

这个OM代码至少包含一个故障：在task2处理数据的过程中，#39行的事件处理程序PhotoSensor.dataReady可以填充缓冲区的数据。具体而言，假定处理结果被存储在缓冲区的第一个槽，如果在task2完整地发送数据包之前执行到#29行，dataReady强行抢占执行，那么一个新的感知数据将覆盖缓冲区该槽中尚未完整发送出去的处理结果。这样，在task2恢复执行后，它会将错误的数据包发送出去。正确的OM代码应该忽略在t3和t10期间所有的dataReady强行抢占信号。在组件中采用锁定机制是一种解决方案：我们可以把在图1中#11、#41、#46和#53行带下划线的语句从注释中恢复成正常语句来实现这个锁定机制。

## 测试技术基础

从系统逻辑复杂度的角度看，嵌入式系统软件（如上述nesC应用程序）与传统应用软件有着类似的特点。因此，测试也是保障嵌入式系统软件质量的重要手段。事实上，嵌入式系统软件的测试与传统软件测试也有着相近的内容[1]。

一般来说，软件测试有两个目标：（1）发现错误和缺陷，帮助开发人员排除这些错误和缺陷；（2）为开发人员发布系统和用户使用系统提供信心（测试中发现错误的数量变化可能预示着系统中仍然残存的错误数量）。为了达到这些目标，可以采用黑盒测试与白盒测试。黑盒测试是将被测软件看成一个黑盒子，检测其外部特征是否满足需要，而不考虑其内部结构。由于黑盒测试是从使用者的角度看待被测软件，黑盒测试能够在一定程度上反映被测软件的实际使用情况，但由于相同的特征可以有多种不同的实现方式，除非进行穷举测试，黑盒测试难以发现许多程序实现中引入的错误。白盒测试是考查被测软件的内部，检测软件结构和逻辑上的错误。与黑盒测试相反，白盒测试有助于发现黑盒测试难以发现的错误，但白盒测试并不能替代黑盒测试。在实践中，一般要求既进行黑盒测试也进行白盒测试。

与传统软件测试相比，测试嵌入式系统软件时需要考虑的错误种类更多，其原因包括：（1）嵌入式系统软件涉及到大量软硬件交互问题；（2）嵌入式系统软件的开发环境与运行环境相对分离；（3）嵌入式系统软件通常包含操作系统的部分功能。另外，嵌入式系统软件的错误严重性分级也与应用软件不同。从错误种类来源上看，嵌入式系统软件的错误可分为以下几类[2]：（1）传统的软件错误，指的是在其他软件中也常见的软件错误，如算法错误、指针错误等；（2）系统软件错误，指的是通常在操作系统中需要考虑的错误，如死锁、中断错误等；（3）编译错误，指的是在嵌入式系统软件的编译过程中引入的错误，如编译环境失配、语言错误等；（4）配置错误，指的是在嵌入式系统软件的配置过程中引入的错误，如硬件配置错误、运

行环境配置错误等。

软件测试依赖于执行测试用例来暴露软件故障。一个测试用例通常包括两部分：测试输入以及测试预言（test oracles）。在测试过程中就相应的要解决两个问题：（1）如何判定已有测试输入的测试能力及其不足之处，然后对其进行增强；（2）如何创建有效的测试预言。接下来我们将阐述在嵌入式系统软件测试中的这两个问题。

## 测试覆盖标准

测试覆盖标准[8]对一组测试用例的测试能力提供了有效的度量手段。在一个嵌入式系统发布前，可以用符合预定测试覆盖标准的测试用例对其进行测试。如果这样，选取恰当的测试覆盖标准来发现特定的故障尤为重要。嵌入式系统通常在计算资源上都很吃紧。嵌入式系统的执行模型和编程语言会与常用于桌面系统的执行模型和编程语言有所不同。我们应该分析目标嵌入式系统所用执行模型的独有特性，并根据这些特性对传统的测试覆盖标准进行适应性的修改。

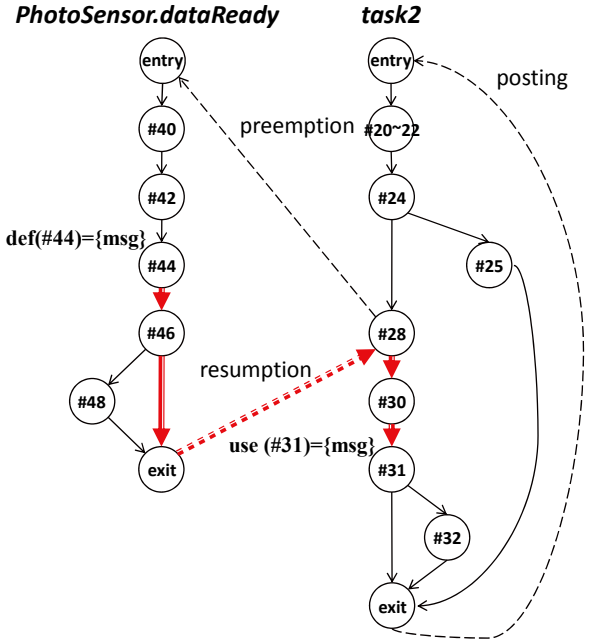



图3 为图1中代码所构建的简化流图

以下我们将用nesC应用程序为例，阐述适应化修改的重要性。

图3给出了一个为图1中代码所构建的简化流图。该图包含显式控制流（使用实线箭头边表示）和隐式控制流（使用虚线箭头边表示）。图上的节点代表语句，由其代码行数标识。中断以及由于任务衍生、强行抢占、恢复而引起的延迟任务执行都会造成隐式控制流。如果测试人员想要通过测试来暴露特定于nesC程序执行模型的故障，覆盖到这些隐式控制流能够起到重要的辅助作用。例如，如果所用的测试用例能覆盖到图3中由粗红线所示隐式定义-使用（def-use）关联（从#44行节点到#31行节点、由任务恢复引起的），那么这个测试用例就会失败并把OM代码的故障暴露出来。

传统的测试覆盖标准[8]通常不能有效地暴露与中断或任务衍生相关的故障，比如OM代码中的故障。例如，一组遍历了正常场景的测试用例会满足传统的全部使用（All-Use）覆盖标准，但是这组测试用例并不能暴露这个故障。另外，基于事件驱动的测试也不足以来暴露这个故障。例如，如果执行一组测试用例使得#39和#56行的单个事件处理程序得到了执行但是没有造成强行抢占，那么这组测试用例满足了传统的全部事件（all-event）覆盖标准，但是这组测试用例也并不能暴露这个故障。要暴露这个故障，这组测试用例的执行必须覆盖特定强行抢占的场景。具体而言，如果一个测试用例的执行使得在#39行的dataReady在图2中t3到t6间隔期间强行抢占了task2，而且收集到的新感知数据和之前的处理结果不一样，那么故障就可以被暴露。

最近，香港科技大学的研究人员（本文第一作者领导的研究小组）[9]对传统的分支和数据流测试覆盖标准进行了修订，将任务衍生、强行抢占、恢复而引起的隐式控制流包含到为测试覆盖标准所用的控制流中（如图3所示）。他们的实验结果显示修订后的分支测试覆盖标准比传统的分支覆盖标准要高21%，能更有效地帮助暴露故障；修订后的数据流测试覆盖标准比传统的数据流覆盖标准要高69%，能更有效地帮助暴露故障。

## 测试预言

测试预言是测试中的一种检测机制，它可以被用来确切地判定测试用例的执行是否通过或者失败。嵌入式系统通常涉及和应用程序层之间的较底层（包括硬件）交互。并不是较底层出现的所有异常行为都可以导致在应用程序层有可观测到的不正确输出。这些异常行为的例子包括对任务进行不正确的解锁、在内核中断服务例程中不正确地使用了一个二元操作符等。因此，创建有效的测试预言是测试嵌入式系统软件中一个具有挑战性的问题。

通常给嵌入式系统软件创建测试预言有两种方法。第一种方法是比较它可观测到的输出和预期的输出。这种类型的测试预言称为基于输出的测试预言。如果异常行为不会触发可观测到的不正确输出，这种基于输出的测试预言无效。第二种方法是插装被测嵌入式系统来生成执行跟踪（Execution Traces）。这样所收集的执行跟踪可和一组独立于应用程序的属性来做对比以检测属性的违反。这种类型的测试预言称为基于属性的测试预言[10]。由这些属性较好地来捕捉的行为包括那些用于跨层交互的数据结构所展现的逻辑行为。比如，互斥信号量、锁、计数器、消息队列是跨层交互所采用的常用数据结构。这些数据结构的逻辑行为可以用状态转换的允许序列来进行建模。当收集的执行跟踪不被这些序列接受时，就检测到了故障。基于属性的测试预言对检测如下类型的故障较为有效：不正确的任务初始化、不正确的信号量创建、不正确配对的临界区入出口、调度程序或等待列表中不正确的位操作等。

实验结果显示这两种测试预言主要方法在故障检测能力上是相辅相成的[10]。对基于输出的测试预言无法有效检测的一些故障，基于属性的测试预言能够检测到。基于属性的测试预言的优越性尤其体现在测试那些有较少可观测到输出的嵌入式系统。使用基于属性的测试预言的一个优势是：所创建的测试预言可广泛应用于在同一嵌入式系统平台上运行的多个应用程序。相比之下，要为每个应用程序来创建基于输出的测试预言。然而，基于属性的测试预言需要插装，并可能产生误报。实验结果表明如果扩展一组测试用例以获取更好的跨层和跨任务的定义-使用关联覆盖，那么基于属性的测试预言的有效性可以得到较好的增强[10]。

此外，还有其它测试预言方法。如时机测试预言（timing oracles），比较的是事件发生的时机与他们的实时需求[11]，例如“心脏起搏器软件必须永远不会触发高于规定的频率。”测试预言通常被创建检测和时间相关的故障。能源测试预言（energy oracles），比较的是嵌入式系统的实际能源消耗模式与其预期的模式，例如应用程序的电池消耗率或操作所需的能源量。另外，嵌入式系统的输入往往存在着不确定性。例如，接收GPS传感器输入的嵌入式系统需要被测试来确保即使在GPS信号受到扰动的情况下，系统仍能正常运作。与此对应，鲁棒测试预言（robustness oracles）用来检测在不确定性下系统的正常运作的程度是否超过了预期值[12]。变形测试预言（metamorphic oracles）则把一组相关的测试用例的输出和事先定义的和应用程序相关的属性进行了比较[13]。变形关系利用了网络嵌入式系统中的冗余度，这些测试用例与这些变形关系相关。具体而言，用于检测环境的无线传感器网络中所部署的邻近传感器能同时检测同一突发事件。因此，可创建一组测试用例来模拟这些邻近传感器周边的同一突发事件，然后通过可创建变形测试预言来检测这些邻近传感器在这组测试用例作用下所采取的行动是否一样。

## 结语

如何提高嵌入式系统软件的质量是一项重要且困难的任务。除了传统类型软件所面临的共同测试的挑战外，嵌入式系统软件还有着自己的特性，比如微型化、紧耦合、专用性，这些特性给测试造成了较大的挑战。本文通过描述一个具体的嵌入式系统软件（无线传感器软件）实例阐述和总结了嵌入式系统软件测试中要解决的一些重要的技术要点，比如测试覆盖标准和测试预言。■

**张成志**

香港科技大学计算机科学及工程系教授。主要研究方向为软件工程、服务计算和普适计算等。scc@cse.ust.hk

**谢　涛**

美国北卡罗莱纳州立大学（North Carolina State University）计算机科学系副教授。主要研究方向为软件工程、软件分析与测试、挖掘软件工程数据等。xie@csc.ncsu.edu

**曹东刚**

CCF会员。北京大学信息科学与技术学院副教授。主要研究方向为软件工程、网络中间件等。caodg@sei.pku.edu.cn

**张　路**

CCF高级会员、2010年CCF青年科学奖获得者。北京大学信息科学与技术学院教授。主要研究方向是软件分析与测试、软件维护与演化、软件服务工程等。zhanglu@sei.pku.edu.cn

## 参考文献

[1] 郑人杰，马素霞，麻志毅，软件工程，人民邮电出版社, 2009

[2] N. Jones. A Taxonomy of Bug Types in Embedded Systems, Oct. 2009. http://embeddedgurus.com/stack-overflow/-2009/10/a-taxonomy-of-bug-types-in-embedded-systems

[3] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. SIGOPS Oper. Syst. Rev. 34, 5 (Dec. 2000), 93~104

[4] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In Proc. of PLDI, 2003, 1~11

[5] T. A. Henzinger, R. Jhala, and R. Majumdar. Race Checking by Context Inference. In Proc. of PLDI, 2004, 1~13

[6] W. Archer, P. Levis, and J. Regehr. Interface Contracts for TinyOS. In Proc. of IPSN, 2007, 158~165

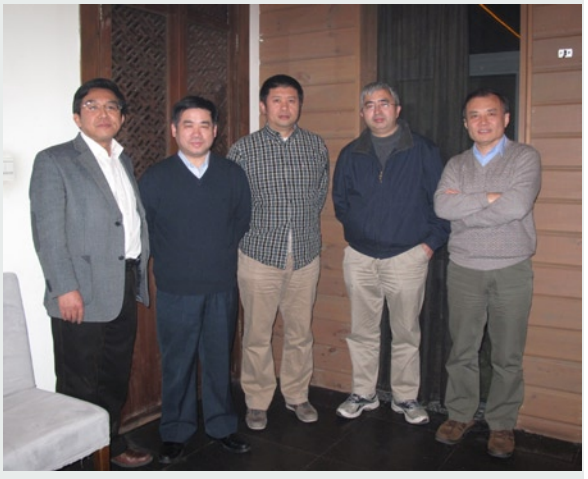
[7] N. Cooprider, W. Archer, E. Eide, D. Gay, and J. Regehr. Efficient Memory Safety for TinyOS. In Proc. of SenSys , 2007, 205~218

[8] H. Zhu, P. A. Hall, and J. H. May. 1997. Software Unit Test Coverage and Adequacy. ACM Computing Survey 29, 4 (Dec. 1997), 366~427

更多参考文献：www.ccf.org.cn/cccf

# CCF秘书长会议在京召开

2012年1月18日，CCF秘书长杜子德在京主持召开了秘书长会议，副秘书长马殿富、陈熙霖、潘柱廷、刘雨参加了会议，办公室主任朱征瑜列席了会议。



杜子德分析了CCF目前的主要情况，提出了副秘书长的分工，各位副秘书长就各自承担的工作提出了自己的想法。会议就如下几个问题进行了讨论：CCF在高等教育中的影响力和作用的发挥；CCF对企业的服务和扩充企业中的会员；对专委的服务和专委的改革；理事及会员代表作用的发挥；如何强化对会员的服务以及发展会员等。

本次会议是1月7日常务理事会批准副秘书长任职后的第一次秘书长会议。