基于分类树的 OSEK/VDX 操作系统一致性测试研究①

张志雄[1,2]　李　曦[1,2]　(1.中国科学技术大学 计算机科学与技术学院 安徽 合肥 230027;
2.中国科学技术大学 苏州研究院 江苏 苏州 215123)

摘　要: 在深入研究 OSEK/VDX 操作系统规范的基础上，提出了基于分类树的 OSEK/VDX 操作系统的一致性测试方法。该方法的核心思想首先是从 OSEK 规范抽取出测试目的，其次根据规范和测试目的构造分类树和划分分类树的输入域，再次为分类树添加生成规则和限制条件，最后借助 CTE XL 工具自动生成一致性测试用例。通过对 MiniOSEK 的一致性测试，验证了使用分类树方法进行 OSEK/VDX 操作系统一致性测试的有效性与正确性。

关键词: OSEK/VDX 操作系统; 一致性测试; 分类树方法

Research on Conformance Test of OSEK/VDX Operating System Based on CTM

ZHANG Zhi-Xiong[1,2], LI Xi[1,2] (1.Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China; 2.Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou 215123, China)

Abstract: After in-depth study of OSEK/VDX operating system specification, a method of conformance test for OSEK/VDX operating system based on classification tree is proposed. This method works as follows: first, test purposes are extracted from the OSEK specification; next, according to the specification and test purposes, classification tree is constructed and the input domain of the classification tree is divided; then generation and constraint rules are added to the tree; finally, conformance test cases are automatically generated with the help of CTE XL tool. The effectiveness and correctness of OSEK/VDX operating system conformance test using classification tree method are proved through the conformance test of MiniOSEK.

Keywords: OSEK/VDX operating system; classification tree method

1　引言

OSEK/VDX 操作系统(简称为 OSEK OS)规范[1]是针对汽车应用特点而专门制定的一个小型实时嵌入式操作系统规范。出于对应用程序可移植性和操作系统本身可移植性的考虑，需要对 OSEK OS 进行一致性测试(Conformance Test，又称符合性测试)。一致性测试是一种功能测试，它采用黑盒测试方法，通过观察具体实现在不同的环境和条件下的反应行为，验证实现与设计规约是否一致。一致性测试只关心实现呈现的外部特征。ISO/IEC 14515 标准[2]给出了 POSIX (Portable Operating System Interface)操作系统的一致性测试方法，而 IEEE POSIX 验证机构提供了 POSIX 操作系统的一致性测试集。

为了验证实现与规约的一致性，通常可以采用两种方式：一是形式化验证[3, 4]，即用形式化语言描述实现，利用数学方法来证明实现是否符合要求；二是设

① 基金项目:电子信息产业发展基金(操作系统安全加固软件研发及产业化文号:财建[2008]329 工信部运[2008]97
收稿时间:2010-03-18;收到修改稿时间:2010-05-09

计测试集，通过检验测试集的运行结果来判断实现是否符合要求。测试集的构造是使用测试集进行一致性测试的关键，通信领域通常采用基于模型的方法来生成测试集。文献[5]提出了使用有限状态机模型来自动生成测试集，但是有限状态机要通过 Murφ描述语言来构建；文献[6]给出了使用 UML 生成测试集的方法和流程；文献[7]提出了一种在 SDL 规范下的基于控制和数据依赖的测试集生成方法。基于模型的测试集生成方法，通常需要其它语言(UML, SDL 等)的支持，并且还可能引起状态空间爆炸问题。

本文提出了使用分类树方法[8](CTM，Classification Tree Method)对 OSEK OS 进行一致性测试的方法，该方法使用分类树来构造测试集，不需要额外语言支持，节省了构建模型的工作，充分利用了 OSEK OS 只有任务、中断、事件、资源和报警器 5 个功能模块的特点，使得只需要构造少量的分类树和测试用例，就能获得较高的测试覆盖度。API 是 OSEK OS 呈现外部特征的途径，通过分析 API 特定的执行环境、动作和结果，就可以得到分类树的输入域，使得测试集的设计变得结构化和系统化，有效地提高了测试集的设计效率。

本文的组织如下：第 2 节简要概述 OSEK OS 规范；第 3 节提出 OSEK OS 的一致性测试流程，并介绍了分类树工具；第 4 节详细说明使用分类树方法构造 OSEK OS 中断模块测试集的过程；第 5 节验证了使用分类树方法构造 OSEK OS 一致性测试集的有效性；最后一节总结了当前的工作。

2 OSEK/VDX操作系统规范

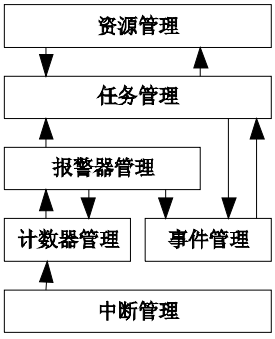

图 1 OSEK/VDX 操作系统组成

OSEK/ VDX 规范体系主要由 4 部分组成[1]：操作系统规范(OSEK OS)、通信规范(OSEK COM)、网络管理规范(OSEK NM)和 OSEK 实现语言(OSEK OIL)。OSEK OS 规范组成如图 1 所示，它包括任务管理、中断管理、事件管理、资源管理和计数器报警器管理五个组成部分。

OSEK OS 规范把任务分为四种符合类(CC, Conformance Class)，分别为基础类的 BCC1 与 BCC2，扩展类的 ECC1 与 ECC2，不同的符合类对系统的要求不同。基础类和扩展类的区别在于任务是否可以拥有事件，可以拥有事件的是扩展类任务；而 1 类和 2 类的区别在于同种优先级是否可以拥有多个任务，可以拥有多个任务的是 2 类。规范还规定，高符合类别是低符合类别的超集，即符合低类别要求的一定符合高类别的要求，反之则不一定。基础任务有就绪、运行和挂起三个状态，而扩展任务因为可以拥有事件，所以要多一个等待状态。

事件管理和资源管理是 OSEK OS 任务间的同步互斥手段。只有扩展任务才拥有事件，扩展任务在等待一个事件发生时将进入等待状态，而当事件发生时就进入就绪状态。为了防止优先级反转和死锁问题，资源管理引入了天花板优先级协议；而为了支持任务组，又引入了内部资源的概念。

中断管理将中断服务例程(ISR: Interrupt Service Routine)划分为 ISR1 和 ISR2 两类，区别在于中断处理过程中能否使用操作系统提供的服务。OSEK OS 的一致性测试不考虑实现依赖于硬件条件的 ISR1。

计数器管理是报警器管理的基础，但由于计数器管理与硬件关系密切，所以规范只对计数器管理提供了一些实现建议，如每个计数器的最大计数值。报警器可以分为两种类型：周期的和非周期的。每个报警器都有一个参考计数器，报警器的超时行为可以是激活一个任务、设置一个事件和调用回调函数中的任意一种。

除了以上五个主要组成部分之外，OSEK OS 规范还规定了一些系统执行控制函数和 HOOK 函数，以方便用户在操作系统中添加感兴趣的内容。

3 OSEK OS一致性测试流程

3.1 OSEK OS 的一致性测试流程

OSEK OS 的一致性测试流程如图 2 所示，它分为四步，分别是由规范抽象出测试目的、设计测试用例、

在实现系统上执行测试用例和分析测试记录得到测试报告。

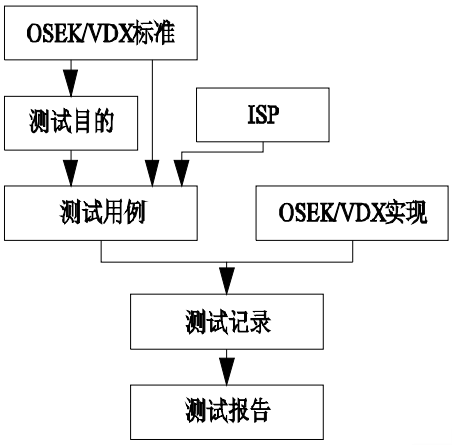

图 2　测试流程图

3.1.1 由规范抽象出测试目的

OSEK OS 规范是用自然语言描述的，为了进行一致性的测试，需要从中抽取出可以评测的内容，即测试目的。使用断言将测试目的表示成如下表格形式：

序号	断言	页码	段落/图/表	影响变量

(1) 序号：该条断言的编号；

(2) 断言：从规范中抽象的测试目的描述；

(3) 页码、段落/图/表：断言在规范中的参考位置；

(4) 影响变量：有三类，分别是任务类别(BCC1/BCC2 /ECC1/ECC2)、调度策略(Non-preempt/Mix-preempt/Full- preempt)和返回值(Standard/ Extended)。这些变量是为了适应不同的应用软件和硬件环境而设置的。

3.1.2 设计测试用例

OSEK OS 的一致性测试根据规范、测试目的和 ISP (Implementation Specific Parameter，实现的特定参数，如存储空间限制，任务个数限制等)，采用分类树的方法来设计测试用例。根据 OSEK OS 的功能模块划分，可以给每个功能模块构造一棵分类树，即任务管理分类树、资源管理分类树、事件管理分类树、中断管理分类树和报警器管理分类树。

根据 OSEK OS 每个 API 特定的执行环境、执行动作、影响因素和返回值，划分得到分类树的四个输入域，分别是执行环境 EXECUTION_LEVEL、系统服务 OS_SERVICE、影响因素 AFFECTED_TASK 和返回值 RETURN_STATUS，表示为：

EXECUTION_LEVEL = {TASK, ISR2}

OS_SERVICE = {特定模块的 API 函数}

AFFECTED_TASK = {影响的任务属性}

RETURN_STATUS = {API 函数的返回值}

特定功能模块的输入域在内容上会有区别，各个模块的输入域也要根据模块特点作出相应的调整，如中断模块的 API 没有返回值输入域。得到输入域后，根据 API 的特点，总结测试用例的生成规则，并最终生成测试用例集。

3.1.3 执行测试

执行测试是指在 OSEK OS 上执行测试用例，并得到测试记录的过程。由于硬件条件(如存储空间)或者 OSEK OS(如任务个数)本身的一些限制，可能需要分批次来执行测试用例。为了方便对测试记录的分析，每条测试记录至少需要提供以下信息：

(1) 执行级别，例如运行任务 ID，ISR 等；

(2) 指示 API 函数返回值正确与否的标记；

(3) 能辨别执行语句的编号。

同样地，为了观察测试记录，还需要以某种方式将测试记录表现出来，这些方式有文件系统、特定内存区域读取、串口输出等。

3.1.4 分析测试记录得到测试报告

OSEK OS 通过一致性测试需要满足两个条件：一是所有测试用例的 API 返回值与期望值相同，二是所有测试用例的语句都以期望的顺序执行。

测试记录体现了每个测试用例的执行情况，通过分析它，就可以得出 OSEK OS 是否通过一致性测试的结论，并最终给出测试报告。测试报告包含两方面的内容：

(1) 给出是否通过一致性测试的结论；

(2) 对于未能通过的一致性测试用例，分析不一致性的可能原因。

3.2 分类树工具

为了方便地从分类树中得到测试用例，OSEK OS 一致性测试的设计测试用例阶段将借助 CTE XL(Classification Tree Editor eXtended Logics)工具来构造分类树。CTE XL 是一个语法控制的、图形化的分类树编辑工具，它能够有效地支持分类树方法，为使用分类树方法构造测试用例提供有力的支持。

分类树方法的主要思想是从被测对象中抽取具体

的测试数据，并且根据测试需求，划分每个测试单元的数据输入域，然后通过组合不同的分类子集，产生测试用例。依据分类子集的组合形成的测试用例数量可能非常庞大，所以要在 CTE XL 工具的支持下，分析各个输入域集合的关系，通过划分等价类、定义因果关系等精简测试用例的数量。

4 中断模块的测试用例构造

OSEK OS 一致性测试的第一步是从规范中抽象出测试目的，而 OSEK OS 规范对中断管理的描述比较少，能够得到的测试目的如表 1 所示。

表 1 中断模块的测试目的

序号	断言	页码	段落/图	影响变量
1	如果被中断的任务是可抢占任务，那么在最外层的 ISR2 结束后，要进行调度	25	6	All
2	ISR2 可以调用的OS服务有限制，调用禁止的 OS 服务产生错误 E_OS_CALLEVEL	45	图 12-1	Extended

分析这两个测试目的可知，中断管理部分只能测试第一个，而第二个只有在其他功能模块才能测试(只有相应功能模块才有 ISR2 调用的 API 函数)。

为了得到中断管理的测试用例，需要为中断管理构造分类树，而构造分类树的首要任务是获得中断管理的输入域：

EXECUTION_LEVEL = {TASK, ISR2}

OS_SERVICE = {EnableAllInterrupts, DisableAllInter- rupts, ResumeAllInterrupts, SuspendAllInterrupts, ResumeOS- Interrupts, SuspendOSInterrupts, TriggerInterrupt, ReturnFrom- Interrupt}

INTERRUPTED_TASK = {Preempt, Non- preempt, NU- LL}

TriggerInterrupt 和 ReturnFromInterrupt 分别表示中断触发和中断返回两个动作；NULL 只是一个占位符，表示不需要考虑这个输入域。

因中断管理的 API 没有返回值，所以并没有 RETURN_STATUS 输入域。于是综合得出初步的分类树，如图 3 上半部分所示。如果不加任何的限制条件，自动生成测试用例，要达到 100%的测试覆盖率，至少需要 2×8×2＝32 个测试用例(NULL 不计算在内)。由于中断管理的分类存在一些依赖关系，可以用其减少潜在的测试用例数量。这些关系是：

关系 1：只有中断级别才需要考虑中断的返回。在 CTE XL 中，这种因果关系表示为 ReturnFromInterrupt => ISR2。

关系 2：只有处于中断返回时，才需要考虑被中断任务的可抢占性。在 CTE XL 中，这种因果关系表示为 preempt=>ReturnFromInterrupt 和 non-preempt=> ReturnFromInterrupt。

在 CTE XL 中声明每条约束关系，并采用全部匹配的生成规则 EXECUTION_LEVEL * OS_ SERVICE * INTERRUPTED_TASK，可以生成如图 3 的测试序列。

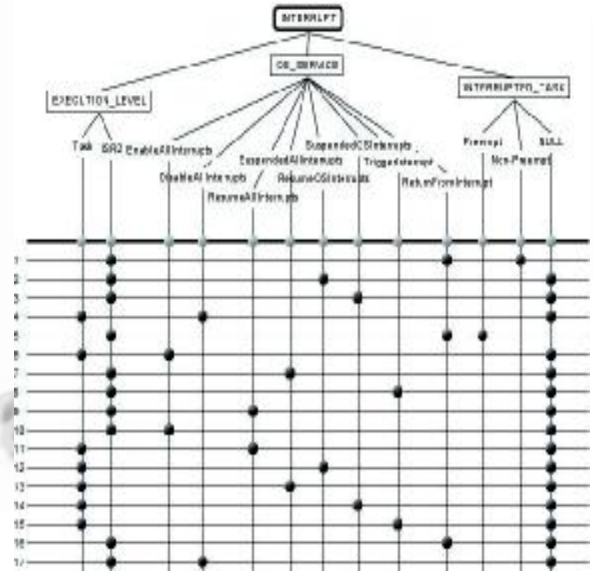

图 3 中断管理分类树

分类树联合表的每一行表示一种测试情况，可以形成一个测试用例，如图 3 的第一种和第五种测试情况如表 2 所示。

其中，n、m、f 分别表示 non-preempt、mix-preempt 和 full-preempt；B1、B2、E1、E2 分别表示 BCC1、BCC2、ECC1 和 ECC2；s、e 分别表示 standard 和 extended。这些变量表示系统的环境，不同的变量组合表示了不同的系统环境，如 n/E1/e

表示 OS 的任务是可抢占的，支持的任务类别是 ECC1，系统服务支持扩展状态。动作是表示这个测试情况需要执行的动作，而预期结果则是根据规范得到的采取了这个动作需要达到的目标。为了得到可以运行的测试用例，最后还要用实现语言 (C/C++ /Java 等)将表格的每种测试情况描述出来。

表 2　中断模块的两种测试情况

序号	影响变量	动作	预期结果
1	n, m B1, B2, E1, E2 s, e	从 ISR2 中断返回，发生中断时的任务不可抢占	被中断的任务继续执行
2	m, f B1, B2, E1, E2 s, e	从 ISR2 中断返回，发生中断时的任务可抢占	进行调度

从图 3 可以看到，添加了约束关系后，测试用例的数量变为 17，比未加约束关系时要少 15 个，而覆盖度依然为 100%。采用 CTE XL 工具来管理分类树，可以极大地减少构造测试用例的工作量，同时提高了操作的准确性和效率。

5　MiniOSEK中断模块的一致性测试

5.1 测试环境

MiniOSEK 是中国科学技术大学苏州嵌入式系统实验室依据 OSEK 操作系统规范自主开发的一个嵌入式操作系统。它基于 MPC555 平台，支持 ECC2 类任务，支持不可抢占/混合抢占/完全抢占调度方式，支持标准和扩展返回值，能根据需要对内核进行裁剪。

本次测试采用的测试环境如下：

①宿主：Debian 5.0.2 操作系统，采用 powerpc-linux (GNU 交叉编译器)交叉编译环境，QEMU 仿真器；

②目标：MiniOSEK(混合抢占，ECC2，扩展返回值)，运行在 QEMU 仿真器上。

5.2 测试用例

根据图 3 表示的每一个测试情况，生成测试用例。如表 2 的两个测试情况对应的测试用例的执行序列：

Task1：基本任务，ready，优先级 2，不可抢占

Task2：扩展任务，suspended，优先级 3，可抢占，拥有事件 Event2

Task3：扩展任务，suspended，优先级 4，可抢占，拥有事件 Event3

表 3　中断测试用例执行序列

运行级别	动作	返回值	测试情况
Task1	ActivateTask(Task2)	E_OK	
Task1	发生 ISR2 中断		
ISR2	中断返回		1
Task1	TerminateTask()		
Task2	发生 ISR2 中断		
ISR2	ActivateTask(Task3)	E_OK	
ISR2	中断返回		2
Task3	TerminateTask()		

5.3 测试结果

在 MiniOSEK 上运行中断模块的测试用例，并分析测试用例的输出，发现中断模块测试用例的执行顺序和 API 的返回值都与预期一致，这说明在测试范围内，MiniOSEK 的中断系统是符合 OSEK 规范的。

6　结束语

本文提出了一种基于分类树对 OSEK OS 进行一致性测试的方法，并详细说明了用该方法生成一致性测试用例的过程，最后通过对 MiniOSEK 中断系统的一致性验证，说明了该方法的有效性。通过对 OSEK OS 进行一致性测试，能够有效地检验 OSEK OS 的正确性，并提高应用软件的移植性和 OSEK OS 本身的移植性，使得 OSEK OS 应用到更广泛的领域。

参考文献

1 OSEK/VDX Operating System Specification Version 2.2.3. OSEKGroup, 2005.

2 ISO/IEC 14515-1-2000: Information technology--Portable Operating System Interface(POSIX)--Test methods for measuring conformance to POSIX--Part1: System interfaces.

3 Felipe Lalanne, Stephane Maag. From the IMS PoC service monitoring to its formal conformance testing. Proceedings of the 6th International Conference on Mobile Technology, Application & Systems, Nice,

(下转第 216 页)

(上接第 212 页)

France, 2009.

4 Ana Cavalli, Stephane Maag,Edgardo Montes de Oca. A passive conformance testing approach for a MANET routing protocol. Proceedings of the 2009 ACM symposium on Applied Computing, Honolulu, Hawaii, 2009:207—211.

5 Farchi E, Hartman A, Pinter SS. Using a model-based test generator to test for standard conformance. IBM Systems Journal, 2002,41(1):89—110.

6 Offutt J, Abdurazik A. Generating Tests from UML Specifications. Second International Conference on the Unified Modeling Language(UML99), Fort Collins, CO, 1999:416—429.

7 Henniger O, Ural H. Test Generation Based on Control and Data Dependencies within Multi-Process SDL Specifications. Proceedings of the 2nd Workshop of the SDL Forum Society on SDL and MSC(SAM2000), Grenoble, France, June 2000.

8 Grochtmann M, Grimm K. Classification Trees for Partition Testing. Journal of Software Testing, Verification, and Reliablity, 1993,3(2):63—82.