第36卷 第15期 计 算 机 工 程 2010年8月
Vol.36 No.15 Computer Engineering August 2010
·软件技术与数据库· 文章编号：1000—3428(2010)15—0089—03 文献标识码：A 中图分类号：TP311.5

基于场景模式的嵌入式软件测试用例设计

杨广华，齐 璇，施寅生
(北京系统工程研究所，北京 100101)

摘 要：将场景和模式方法用于嵌入式软件测试用例的设计与生成，通过对被测软件系统需求进行分析建模，将建立的场景模型划分到不同的场景模式中，依据场景模式构建测试场景的状态图，遍历场景状态图以获取测试执行路径，确定相关的测试数据，设计并生成测试用例。实验结果证明，该方法可以较快地设计和生成测试用例，提高测试用例的可重用性，实现对嵌入式软件的有效测试。

关键词：嵌入式软件；测试用例；场景模式

Design of Embedded Software Test Cases Based on Scenario Pattern

YANG Guang-hua, QI Xuan, SHI Yin-sheng
(Beijing Institution of System Engineering, Beijing 100101)

【Abstract】 This paper applies scenario patterns to design and generate embedded software test cases. By analyzing and modeling the requirements of the system under test, the scenario models constructed are classified into different patterns. The state diagrams of test scenarios are created according to scenario patterns. Test executing paths are derived by traversing the state diagrams, and test cases are designed and generated from test executing paths and test data. Experimental result proves that the method can design and generate test cases rapidly, enhance the reusability of test cases, and realize efficient test for embedded software.

【Key words】 embedded software; test case; scenario pattern

1 概述

随着计算机硬件和软件技术的飞速发展，嵌入式系统的软件规模和复杂性日益提高，对软件质量的要求越来越高。软件测试是现阶段保证软件质量的重要手段。对于嵌入式软件测试，被测软件一般具有实时性、并发性等特点，在测试用例设计时需要考虑这些特点，有针对性地进行设计。基于场景的测试方法逐渐成为软件测试领域的研究热点。文献[1]提出了基于场景模型的面向对象测试框架。文献[2]提出了基于场景的软件验证和测试方法SCENT。文献[3]提出了基于形式化规范和使用剖面的测试场景生成方法。文献[4]提出了基于UML和场景技术的测试用例生成方法。

本文提出基于场景模式的嵌入式软件测试用例设计方法，将场景模式应用于测试用例的设计与生成，为嵌入式软件测试提供支持。

2 场景模型

测试用例是为验证程序是否符合特定系统需求而开发的测试输入、执行条件和预期结果的集合。系统需求通常采用自然语言描述，需要以更加规范的方式表达。大部分嵌入式软件系统为反应式系统，适合采用基于状态的模型进行描述。

ACDATE模型比较适合对基于状态的反应式系统进行场景建模，使用Actor、Condition、Data、Action、Timing、Event这6个模型元素描述系统需求。其中，Actor表示场景中的物理实体或概念实体；Condition表示场景中的状态或判定条件，包括pre-condition(前置条件/状态)和post-condition (后置条件/状态)；Data表示场景中计算或处理过程的变量或常量信息；Action表示场景中状态改变的操作过程；Timing用于描述场景中其他元素的时间属性；Event用于描述场景中由系统对象产生或影响而发生的事件。

在场景集合中，常见的场景执行关系有以下4种：

(1)独立：若场景在运行过程中不依赖任何其他场景中的数据、控制信息等参数，则场景为独立场景。

(2)触发：一个场景的执行触发了另一个场景的执行。

(3)互斥：如果2个场景不可能同时发生，则它们是互斥的。

(4)依赖：如果一个场景必须发生在另一个场景之后，则称场景之间存在依赖关系。

系统需求发生变化时需要对测试用例进行修改，场景的依赖性分析有助于确定需求变化可能影响到的测试用例。场景模型中的依赖关系主要有数据依赖、条件依赖、控制依赖和功能依赖。

3 常见的场景模式

随着嵌入式软件系统规模的不断扩大，系统需求对应的场景数量会大幅增加。通过引入场景模式，将相似的场景归类到相应的场景模式中，依据场景模式的模板可以有效地指导测试用例的设计与生成。根据文献[5]的统计，嵌入式软件系统中常见的4种场景模式对应系统的需求占到总需求比例的70%以上。

4种场景模式包括：

(1)基本场景模式：当前置条件A有效，后置条件B在时间周期T内为真。

如图1所示，在嵌入式实时系统中经常使用前置条件pre-condition和后置条件post-condition表示因果关系或逻辑事件序列。

作者简介：杨广华(1981－)，男，硕士，主研方向：程序分析，软件测试；齐 璇，博士；施寅生，硕士

收稿日期：2010-01-28 **E-mail**：waterorsky@163.com

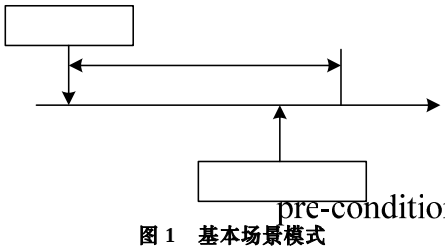


图 1　基本场景模式

(2)关键事件驱动场景模式：当关键事件发生，事件 R1, R2,…,R*n* 会在时间 t1 之前发生。

如图 2 所示，在关键事件驱动场景模式中，当关键事件发生后通常会期待有一系列的结果出现，而且对这些结果的出现一般会有时间的限制。

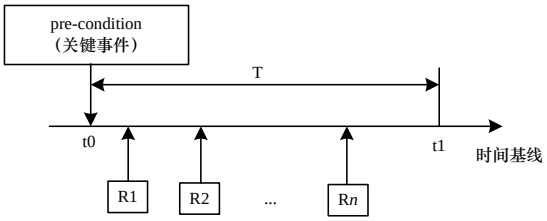


图 2　关键事件驱动场景模式

(3)定时事件驱动场景模式：关键事件之后的 t1 时间周期内，如果 P 事件发生，则 R 事件会在时间 t2 之前发生。

如图 3 所示，定时事件驱动场景模式强调在关键事件发生之后有一个时间周期，在此时间周期内发生的事件 P 和关键事件一起作为判断条件。

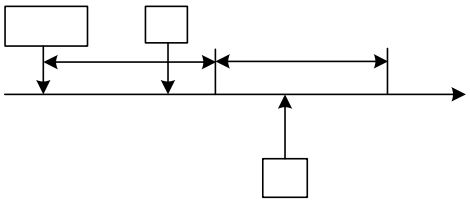


图 3　定时事件驱动场景模式

(4)时间片事件驱动场景模式：当 pre-condition 为真，如果事件 R1, R2, …, R*n* 分别在时间 t1, t2, …, t*n* 之前发生，则 post-condition 为真。

如图 4 所示，在时间片事件驱动场景模式中，依次分别在各个时间间隔内验证相应事件的发生。

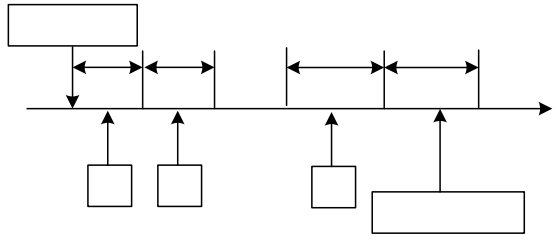


图 4　时间片事件驱动场景模式

4　测试用例设计与生成

测试用例的设计与生成会对测试效率、测试结果产生很大影响。测试用例的描述应该准确、规范、清晰、可维护性好。测试用例通常包括一个特定的测试场景及其对应的测试数据用以描述系统的特定执行流程。规范化的测试用例描述有利于测试用例的自动生成，表 1 给出一种基于场景的测试用例描述方式。

表 1　基于场景的测试用例描述

要素	属性	说明
概要	(1)ID	标识号
	(2)名称	用例名称
	(3)描述	功能描述
测试策略	(1)测试类型	功能、性能、强度等
	(2)测试准则	条件覆盖、路径覆盖、逻辑覆盖等
输入输出	(1)测试输入	测试输入
	(2)预期输出	预期输出
测试执行	(1)模块	包含的模块
	(2)接口	包含的接口
	(3)数据	包含的数据
	(4)执行路径	执行路径信息
条件	(1)前置条件	执行前需满足的条件
	(2)后置条件	执行后的预期状态
关联	(1)需求	对应的需求信息
	(2)测试脚本	测试脚本信息
其他	(1)状态	状态信息
	(2)进度	进度信息
	(3)风险	风险信息

基于场景模式的嵌入式测试用例设计与生成过程如下：

(1)被测系统需求分析，通过对被测系统文档进行分析，对被测系统进行需求建模并构建场景树。

(2)依次分析各个场景，将其划分到不同的场景模式类别中，构建测试场景的状态图。

(3)通过遍历场景状态图获取测试执行路径，确定相关的输入数据，生成测试用例。

4.1　场景建模

应用 ACDATE 模型对被测软件系统需求进行分析建模。对被测系统需求分解的结果可以形成场景树，场景树的结构采用分级的方式进行管理。把被测系统作为场景树的根节点，叶子节点是原子场景，每个分支看成是对被测系统需求的进一步分解，每个分支代表被测系统的一部分相关需求(如功能相关、数据相关或时间特性相关)。原子场景代表了系统一个最小的功能点，原子场景是不可再分和独立可测的。原子场景可以通过顺序、条件、并发或循环等各种控制结构组成复合场景。

4.2　场景状态图构建

对构建的场景进行分析，将其划分到不同的场景模式中。参照相应的场景状态图可以较快地为每个场景生成对应的状态图。对于无法划分到已有模式中的场景，需要单独构建场景状态图，然后可以考虑将此场景作为一种新的场景模式。图 5~图 8 分别是基本场景模式、关键事件驱动场景模式、定时事件驱动场景模式、时间片事件驱动场景模式所对应的场景状态图。

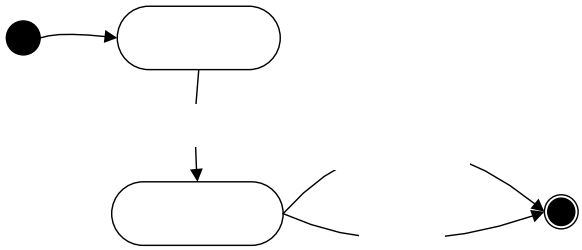


图 5　基本场景模式状态图

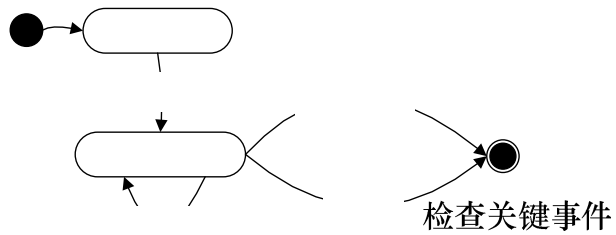


图 6　关键事件驱动场景模式状态图

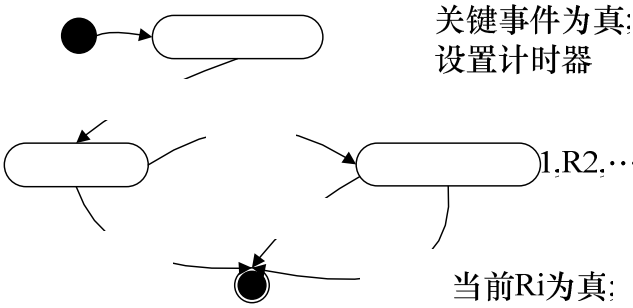


图 7　定时事件驱动场景模式状态图

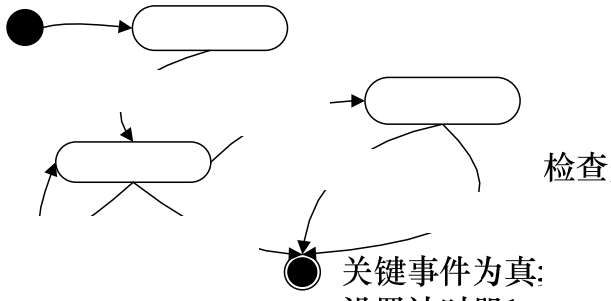


图 8　时间片事件驱动场景模式状态图

4.3　测试用例生成

基于场景模式的嵌入式软件测试在设计和生成测试用例时应满足节点覆盖准则、迁移覆盖准则、逻辑路径覆盖准则等，以保证测试用例的覆盖。节点覆盖准则要求每个节点至少执行一次。迁移覆盖准则要求每个迁移至少激活一次。逻辑路径覆盖准则要求所有分支路径至少执行一次，覆盖从初始节点到终止节点的每条独立路径。对迁移覆盖准则考虑执行 0 次和 1 次的情况，从而保证测试的准确性，同时避免较大的冗余。结合覆盖准则，对场景状态图中的全部路径进行遍历，可以获取相应执行过程中的控制流、数据流、分支节点的条件，确定每条路径所需的输入数据和状态条件。

对于测试用例来说，每次测试执行对应场景的一条基本路径。通过遍历场景状态图中的全部基本路径可以获得所有的测试执行路径，然后确定相关的输入数据。

测试执行路径生成方法如下：

(1)查找场景状态图中的开始节点，将开始节点放入一个堆栈结构中，记录开始节点为当前节点。

(2)查找当前节点的直接后继节点，按照深度优先的次序进行遍历，如当前节点有多个直接后继节点，即有多条输出路径，选择一条尚未访问的路径，将该路径及目标节点加入堆栈结构并将该路径标记为已访问，将该目标节点记录为当前节点；如当前节点只有一个直接后继节点，即只有一条输出路径，则直接将该路径及目标节点加入堆栈结构，将目标节点记录为当前节点。

(3)重复步骤(2)，直到当前节点没有后继节点。此时，从堆栈结构的栈底到栈顶的所有节点和路径信息就是一个测试场景的路径信息。

(4)从堆栈结构中依次弹出节点和路径，如当前栈顶对象是节点且有尚未访问的输出路径，则转到步骤(2)、步骤(3)，直到弹出堆栈结构中的所有对象为止，这样就完成了所有测试场景路径的遍历。

图 9 是 ATM 系统“密码验证”场景状态图，应用上述步骤，可以得到测试路径序列：{t1, t6}，{t1, t2, t3, t7}，{t1, t2, t5, t8}，{t1, t2, t4, t2, t3, t7}。

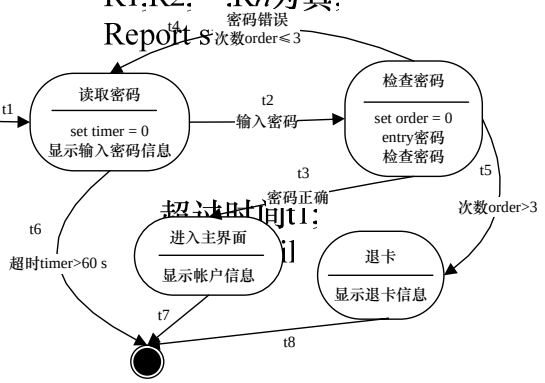


图 9　ATM 系统密码验证场景状态图

确定测试执行路径之后还需要确定相关的输入数据，对于软件测试用例来说，输入数据类型可能包括 int、bool、long 等简单类型和结构体、共用体等复合类型，简单类型数据的取值可以采用等价类划分、边界值、随机等方法生成，复合类型数据的取值通过遍历复合数据结构中的各数据项确定测试输入数据。为了较好地达到测试目的，可能需要将多种数据生成方法结合起来使用。

5　结束语

本文提出一种基于场景模式的嵌入式软件用例设计方法，将可以重用的场景模式应用于测试用例设计，节省了设计、生成和修改测试用例的时间和工作量。在设计和生成测试用例时不需要软件设计阶段的模型信息，是一种面向需求规范的黑盒测试方法。进一步的工作主要是对测试路径选择、测试数据生成及自动化的测试工具进行更深入的研究。

参考文献

[1] Tsai W T, Na Yinghui, Paul R J, et al. Adaptive Scenario-based Object-oriented Test Frameworks for Testing Embedded Systems[C]//Proc. of Conference on Computer Software and Applications. Oxford, UK: [s. n.], 2002.

[2] Ryser J, Glinz M. Using Dependency Charts to Improve Scenario-based Testing[C]//Proc. of International Conference on Testing Computer Software. Washington D. C., USA: [s. n.], 2000.

[3] Chang Kai, Liao Shih-Sung, Chapman R, et al. Test Scenario Generation Based on Formal Specification and Usage Profile[J]. International Journal of Software Engineering and Knowledge Engineering, 2008, 10(2): 185-201.

[4] 殷永峰, 刘 谦, 姜同敏. 基于 UML 的嵌入式软件测试用例生成方法研究[J]. 计算机应用研究, 2008, 25(10): 3018-3021.

[5] Deng Zhiyao, Sang Nan. Testing Result Statistics-based Rapid Testing Method for Safety-critical System[J]. Journal of Electronic Science and Technology of China, 2008, 6(3): 274-277.