一种嵌入式软件的类状态测试框架设计

Designing the State Testing of Class Framework of an Embedded Software system

马 可　陈蜀宇　石振明　（重庆大学计算机系 重庆 400044）

摘　要： 在介绍UML状态图的基础上，以嵌入式软件的类作为基本测试单元，研究了基于UML状态图的类状态测试技术，进一步设计出基于UML状态图的嵌入式软件类状态测试框架。该测试框架的系统体系结构由测试初始化配置、测试用例生成、测试脚本生成、测试执行和结果分析五个模块组成。

关键词： UML　状态图　类状态测试　测试框架

1　引言

近年来，随着嵌入式软件开发方法的广泛应用，嵌入式软件测试已成为软件工程领域的一个研究热点。嵌入式软件测试是嵌入式软件开发不可缺少的一环，是保证软件质量、提高软件可靠性的关键。由于嵌入式程序本身所具有的封装性、继承性、多态性、动态绑定等特性，使得嵌入式软件测试的策略和内容有很大不同。测试的视角扩大到包括复审分析和设计模型，测试的焦点从模块转向了类。在嵌入式程序测试中，最小的可测试单元是类和类的实例，最小的可测试单位是封装的类或对象。

UML（Unified Modeling Language）作为一种面向对象的建模语言，提供了各种图形从不同的角度和抽象层次上描述软件系统。近年来，UML在嵌入式软件分析与设计中得到了广泛应用，随之而来的是给嵌入式软件测试提出了新的课题。目前，国内外许多学者在基于UML的嵌入式软件测试方面开展了一系列的研究工作。本文在介绍UML状态图技术的基础上，以嵌入式软件的类作为基本测试单元，研究了一种基于UML状态图的类状态测试技术，讨论了基于状态说明的测试用例设计覆盖准则，并进一步设计出基于UML状态图的嵌入式软件类状态测试框架。

2　UML及其状态图

UML的定义包括UML语义和UML表示法两个部分。UML语义描述基于UML的精确元模型定义。元模型为UML的所有元素在语法和语义上提供了简单、一致和通用的定义性说明，使开发者能在语义上取得一致，消除了因人而异的表达方法所造成的影响。UML表示法定义了UML符号的表示法，为开发者或开发工具使用这些图形符号和文本语法为系统建模提供了标准。UML定义了5类、共10种模型图：①用例图，从用户角度描述系统功能，并指出各功能的操作者。②静态图，包括类图、对象图和包图。③行为图，描述系统的动态模型和组成对象间的交互关系，包括状态图和活动图。④交互图，描述对象间的交互关系，包括时序图和合作图。⑤实现图，包括组件图和配置图。

其中，UML状态图描述类的对象所有可能的状态以及事件发生时状态的转移条件，从而说明了对象的所有可能的动态行为。每个对象被视为是一个孤立实体，它通过检测事件然后对事件进行响应来与外界进行通讯。UML状态图通常包括状态和转移。状态是一个抽象的元类，定义对象在其生命周期中的条件或状况。转移包括事件和动作。事件是发生在时间空间上的一点值得注意的事情，表达了对象可以检测到的变动，任何影响对象的事物都可以被描述成事件。动作是原子性的，它通常表示一个简短的计算处理过程。在UML中，图形上每一个状态图都有一个初始状态（实心圆），用来表示状态机的开始，还有一个终止状态（半实心圆），用来表示状态机的终止，其他的状态用一个圆角的矩形表示。转换表示状态间可能的路径，用箭头表示，事件写在由它们触发引起的转换上。具体状态图如图1所示。

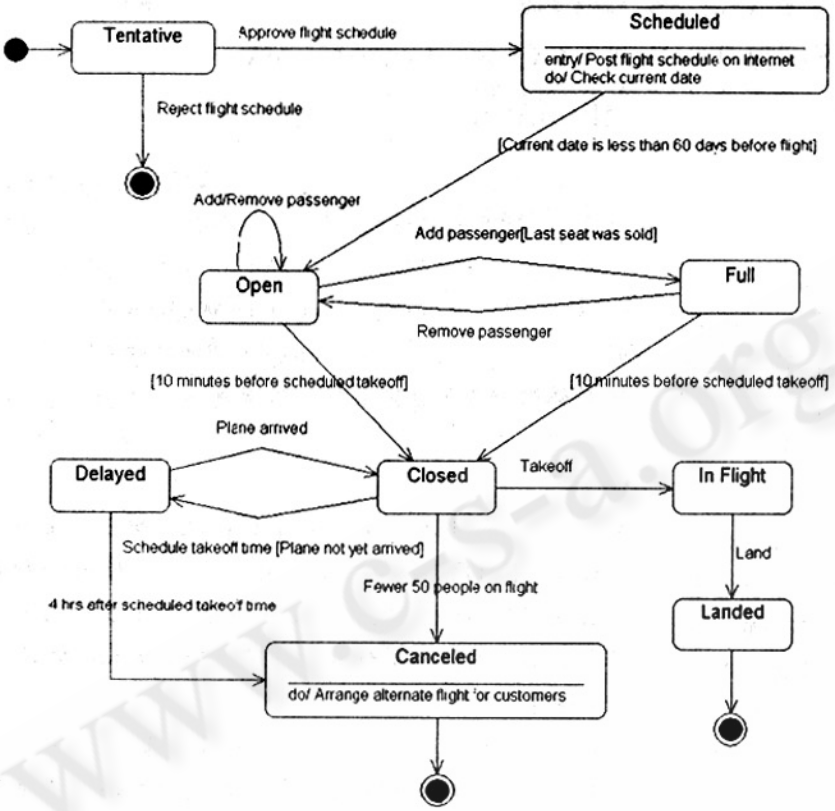

图 1　状态图

3　基于 UML 状态图的类状态测试

3.1　类状态机

状态是值的集合，一个类的状态由它的变量的所有的域定义。对象的成员变量的当前值确定了对象的当前状态，而对象当前变量的值是由以前的消息序列决定的，根据当前的变量的值又限制了可接受的消息序列和输入参数的值域。

在一般情况下，一个类的实例对象处在某一状态时，它只能接受某些消息（这些消息可能是外部的一个信号消息，也可能是对其方法的调用），并根据当前的状态对消息参数的值域进行限制，然后对这些消息做出响应，产生应有的外部输出或改变对象的内部成员变量的值，从而到达另外一个状态。类的这种属性可以用状态机模型来表示，一个状态机是事件（输入），状态和动作（输出）组成的抽象。状态机有 4 个构成元素：①状态：概括了与过去输入有关的信息的抽象，它在决定系统收到后续输入的行为时是必需的；②转换：一个允许的两个状态的序列。一个转换是由一个事件引发的，它必须说明一个接受状态和结果状态；③事件（消息）：一个输入或时间间隔；④动作：结果或跟随一个事件的输出。

3.2　类状态测试步骤

基于状态的测试的一般过程如下：

（1）根据测试需求生成测试用例。测试需求也就是测试的覆盖标准，对基于状态的测试一般要求覆盖所有的状态以及状态转换。测试用例由消息序列和测试数据组成，每个测试用例都要求完成一次由初始状态到特定状态的转换，然后返回初始状态。

（2）运行测试用例（测试驱动）。

（3）验证被测类是否到达正确的状态并做出了正确的响应。

类状态测试主要是检查状态变化和行为而不是类内部的逻辑，所以数据错误可能会遗漏。当用基于状态的测试来确认类时，那些没有定义成为对象状态的数据成员通常会被忽略。因此，基于状态的类测试需要用白盒测试、功能测试来弥补这些缺陷。

3.3　测试覆盖标准

对于类的基于状态的测试就是要求每个公有的方法（服务）都至少被调用一次。在 UML 状态图基础上有以下几种测试覆盖标准：(1) 状态覆盖：产生的测试用例能够测试每一个状态；(2) 转换覆盖：用以度量测试用例对转换的覆盖程度，分为 0 — 长度（0 - Switch）覆盖：要求覆盖单个转换；1 — 长度覆盖：要求测试各转换的两两组合；n — 长度覆盖：要求覆盖 n + 1 个转换；(3) 状态 — 事件对覆盖：测试时组合每个状态和事件；(4) 状态 — 转换覆盖：要求测试用例满足：①每个状态都要被测试至少一次；②每个转换都要被

执行最少一次。

状态 — 转换覆盖标准是目前基于状态的测试中最常用的覆盖标准，大多数的测试用例生成算法都是以此为目标，它能生成相对较短的测试用例，并且具有很高的揭示故障的效率，因此本文采用此标准进行嵌入式软件类状态测试框架的设计。

4 测试框架的设计

本测试框架的系统体系结构由测试初始化配置、测试用例生成、测试脚本生成、测试执行和结果分析五个模块组成。测试初始化配置模块用来配置系统的相关信息；测试用例生成模块读取 XML 文件表示的 UML 状态图模型，并在此基础上从状态图中得到类状态的满足状态 — 转换覆盖标准的测试用例；测试脚本生成模块用来把测试用例翻译到相应的实现语言，生成测试代码；测试执行模块用来编译和执行测试脚本；结果分析模块用来分析测试执行模块记录的信息，并给出相应形式的结果报告。具体系统框架结构图如图 2。

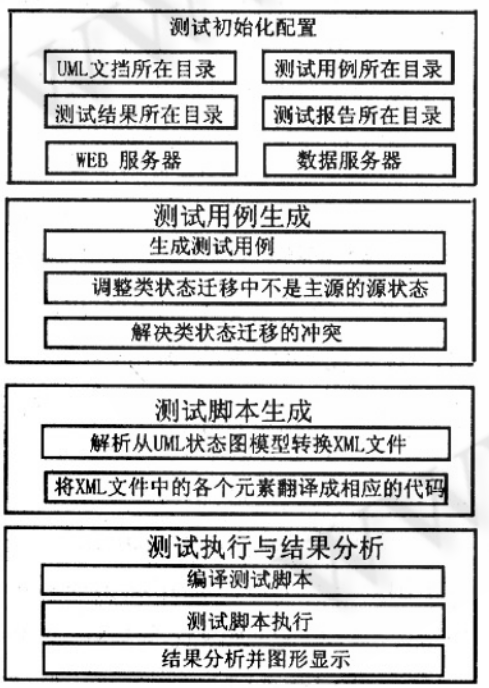

图 2 测试框架结构图

4.1 测试初始化配置模块

测试初始化配置模块用来配置系统相关信息，这些信息包括系统服务和服务信息。系统属性包括 UML 所在文档所在目录，测试用例所在目录，测试结果所在目录，中间结果存放目录，测试报告所在目录等信息。运行测试也需要服务资源，这可能包括 WEB 服务器，数据库服务器等，因此我们需要配置这些服务启动所需的信息等。

4.2 测试用例生成模块

4.2.1 生成测试用例

从 UML 状态图模型视图中生成的类状态的满足状态 — 转换覆盖标准的测试用例将使用 XML 格式的文件来表示，即保存在一个 XML 文件中，然后生成一个类状态自动机 CSM(Class State Machine)。

4.2.2 调整类状态迁移中不是主源的源状态

使用深度优先搜索算法搜索类状态自动机，得出转换路径集，并调整类状态迁移中不是主源的源状态。因为不是所有路径的迁移关系的源状态都是该迁移关系的主源，因此执行测试用例肯定不会产生正确的测试结果，所以对于测试用例中的迁移关系，如果其主源不是该迁移的源状态，则需要修复，即在该主源的状态机中找到一条从其初始状态开始并到达被测试迁移的源状态的测试序列。

4.2.3 解决类状态迁移的冲突

因为在关于类状态的满足状态 — 转换覆盖标准的测试中，测试用例的代码覆盖率和自动化程度要求较高，需要对于类状态的每一条转换迁移路径至少得到一个测试用例，但这容易生成大量的不可执行路径和造成类状态迁移的冲突，最终造成代码覆盖率和自动化程度的下降，因此解决类状态迁移的冲突是本框架关注的一个重点。

4.3 测试脚本生成模块

从 UML 状态图模型中生成基于类状态的满足状态 — 转换覆盖标准的测试用例后，我们就要把测试用例翻译到相应实现语言的测试脚本。这部分工作包括解析从 UML 状态图模型转换的类状态的满足状态 — 转换覆盖标准的 XML 文件以及把 XML 文件中的各个元素翻译成相应的代码。整个工作由三个核心部分组成：描述约束的翻译、测试控制式样的设计和测试脚本样式设计。

4.4 测试执行与结果分析模块

测试执行模块用来执行测试脚本，结果分析模块则对执行结果进行分析和图形化显示。它所做的工作包括：

(1) 编译测试脚本。这是测试执行的第一步，非常关键，测试脚本生成模块生成的文件是源代码文件，测

试执行模块读取从 UML 状态图模型转换的类状态的满足状态 — 转换覆盖标准的 XML 文件，然后进行编译工作。

（2）测试执行。即加载和执行测试用例，将输出信息的分类、格式化及永久保存，包括测试通过或者不通过的信息。

（3）结果分析。使用 Log4j 日志技术来记录程序的其它信息，通过设置 Log4j 的配置文件我们可以把日志信息以 XML 的格式记录到日志文件中，最后的测试结果分析将以文本或图形的形式展现出来。

4.5 测试框架的优点

与现有的类状态测试框架相比，该框架具有较高的测试效率和准确率。如基于状态的类测试工具 Class tester，由预处理模块、状态变迁图绘制模块、测试用例生成模块和测试程序自动生成模块组成，其自动化程度较高，能根据被测类的状态转换图自动生成各种覆盖类型的测试用例；并自动生成相应的类测试驱动器；自动进行测试的执行和测试结果的比较。而本文所描述的测试框架体系结构不仅能达到 Class tester 的自动化程度，还进一步调整类状态迁移中不是主源的源状态，使用深度优先搜索算法搜索类状态自动机，得出转换路径集，从而提高测试代码生成效率和测试准确性。

另一种基于 UML Statecharts 的测试框架则利用现有的 UML 工具画出 UML Statecharts 图，将其转换成 FREE 模型图，接着根据全路径覆盖准则产生复合迁移序列的算法，最后以 FREE 模型为基础生成类的测试用例。本文所提出的类状态测试框架与其相比，具有更充分的测试覆盖准则：状态 — 转换覆盖标准，能生成相对较短的测试用例，并且具有更高的揭示故障的效率。

其次，还有一种基于 UML 的面向对象软件测试框架，将测试过程分成系统测试、类族测试和类测试，生成的测试用例分别对应于统一软件过程中的初始、细化和构造，结合了 UML 和统一软件开发过程，具有实际工程应用意义。但其对测试用例生成方法的效率和覆盖程度并未做深入讨论，所采用的代码覆盖标准会生成大量的不可执行路径，最终造成代码覆盖率和自动化程度的下降。而本文中的类状态测试框架虽然没有对测试对象进行分层测试，只在类测试阶段进行了深入探讨。但也正因为如此，其重点研究的深度优先搜索算法关注于解决类状态迁移的冲突问题，测试用例的代码覆盖率和自动化程度明显提高。

5 总结与进一步工作

在上述测试框架研究的基础上，目前已经实现了基础框架的构建，并且通过对基础框架的适配与修改完成了基于 UML 状态图的类状态测试环境的基本实现。在基础框架中实现了测试初始化配置、测试类的管理与状态调度、测试包类的样式结构和测试驱动程序开发等框架类，可以完成对测试脚本和 UML 状态图的预处理，实现类状态的满足状态 — 转换覆盖标准的简单覆盖测试，以及对测试结果数据的收集和显示等主要功能。对于嵌入式软件类状态测试的应用开发来说，开发者只需要在此框架内，把每个模块的功能实例化，把框架各个模块之间的交互关联体现到具体应用中，就能够节省测试环境的开发和准备时间，同时提高测试的效率。

从框架研究的本身来看，本框架的研究并没有到此结束，还需要近一步的改进和优化，使其实现最佳的设计复用。在现有基础上，后续将重点研究解决类状态迁移的冲突和提高测试精度，在进一步的工作中，将综合考虑控制流和数据流特性，并结合启发式算法，以提高代码的覆盖率和生成测试用例的自动化程度。

参考文献

1 张庆、雷航，一种基于 UML 状态图的面向对象类级测试模型 [J]，计算机工程，2005，21(11).

2 张雪萍、庄雷、范艳峰，基于状态的类测试技术研究 [J]，小型微型计算机系统，2002，23(9).

3 Craig Larman. UML 和模式应用：面向对象分析与设计导论 [M]，北京：机械工业出版社，2001.

4 KIMY G, HONGH S, BAED H, etal, Test Cases Generation from UML State Diagrams[J]. IEE Proceeding on Software, 1999, 46(4).

5 Latella D, Massink M, A formal testing framework for UML state diagrams behaviors: from theory to automatic Verification. http://portal. acm. org/ citation. cfm.

6 Cheij D, A software architecture for building interchangeable test systems, Proceedings of 2001 IEEE AUTOTESTCON C. Philadelphia: IEEE, 2001.