一种面向状态 COM 组件的测试用例生成方法

吴 忠
(江西方兴科技有限公司 技术部,江西 南昌 330025)

摘 要:基于组件的开发方法提高了软件的可复用性和软件开发效率,但组件具有的封装和状态特性增加了组件和基于组件软件的测试难度。依据组件的状态特性,在组件中引入状态性概念,将组件分为非状态组件和状态组件。对具有不同特征的组件进行有针对性的测试。对于状态组件,除使用非状态组件的测试方法外,还定义了扩展邻接表和扩展约束对照表来存储状态信息,并结合基于流的测试方法来产生方法序列测试用例及脚本。在理论研究的基础上开发出 COM(Component Object Model)组件测试用例及脚本自动生成系统,验证了测试用例及脚本生成方法的有效性。

关键词:COM 组件;组件测试;状态组件;测试用例
中图分类号:TP311.51 **文献标识码**:B **文章编号**:1674-7720(2010)10-0070-04

A generating approach of test cases based on state COM component

WU Zhong
(Technology Department of Jiangxi Fangxing Science & Technology Co., Ltd., Nanchang 330025, China)

Abstract: The component-based development method improves the software reusability and software development efficiency. However, component packaging and status properties increase the testing difficulty of component-based software. According to the component state features, the components are divided into non-state components and state components. The pairs of components with different characteristics are tested respectively. For the state components, in addition to use the test methods of non-state component, we expand the definition of adjacency table and the constraints comparison table to store state information. Combined with flow-based test methods, the test cases and script are generated for testing method sequence. The automatically generating system of COM(component object model) component test cases and script is developed based on our theory research to verify the effectiveness of the generation method.

Key words: COM component; component testing; state component; test cases

目前基于组件的软件工程(CBSE)已成为软件工程领域的研究热点[1,2,4]。CBSE 是从面向对象的开发方法发展而来,其特点是能够实现组件的重用,使组件实现“即插即用”,缩短软件开发周期,降低开发维护成本。各种新型组件开发技术进一步提高了组件开发效率及组件性能,但组件的一系列问题始终没有得到较好的解决。状态组件是 COM 组件中另一类非常重要、使用非常广泛的组件。由于状态组件具有状态特性,对状态组件的测试要求也更高,难度也更大。

本文将在非状态组件分析的基础上,针对状态组件的特性进行研究。将详细分析状态组件与扩展有限状态机的共性,并结合状态组件的状态特性及方法调用序列要求,分析两者间的关联性。通过将状态组件转换为等价的扩展有限状态机,并将信息存储在自定义的扩展邻接表和扩展约束对照表中,再结合有向图的遍历算法来产生测试方法序列。方法序列中单个接口成员函数的测试用例的生成,采用非状态组件中使用的测试用例生成方法。

1 状态组件与状态机

由于状态组件与有限状态机存在状态特性上的相似性[1-3],因而在状态组件测试用例生成方法中引入了扩展有限状态机。

有限状态机 FSM(Finite State Machine),已经成为需求规格说明的一种相当于标准的表示方法[5,6]。在很多结构化分析中都使用了某种形式的有限自动机,而且在绝大多数的面向对象分析中广泛使用。

在现有的软件测试领域经常使用的状态机是在原始状态机的基础上经过改造的扩展状态机 EFSM(Extended FSM)[7]。EFSM 可定义为一个 5 元组：EFSM=<∑,S,T,Γ,s0>，其中∑是状态机接受的输入集合；S 是状态机的状态集合；Γ 是逻辑表达式集合；T 是转换集合，T=∑×S×S×Γ，表示状态机处于一个起始状态，s1∈S 时，接收到一个输入 e∈∑，而且满足条件 λ∈Γ，那么状态机的状态将变成 s2∈S；s0∈S 是状态机的初始状态集。

由状态组件的特性分析和有限状态机的介绍可知，状态组件与扩展有限状态机有很大的关联性，具体体现在以下几个方面：

(1)状态组件的接口成员方法一般都有输入参数，而对没有输入参数的接口成员函数，可以将输入参数置为空。这些输入参数与有限状态机中的输入集∑具有一致性，因此可以认为方法的输入参数对组成了状态组件的输入集∑。

(2)状态组件的接口成员函数组成的函数方法集合[8]，与有限状态机的状态相似，都是一类事物的集合，具有关联性。可以将状态组件的接口成员函数方法抽象成一种特定的扩展状态 S′。这些扩展状态组成的集合便可以构成一个扩展状态集 S。

(3)一些方法的后置断言有相应的后置断言条件，这些后置断言条件约束了方法可以转换到哪一个后序方法上。这与扩展状态机中的 Γ 是一致的。因此可以将这些后置断言条件组成的集合当作 Γ。

(4)当一个初始方法 M1 完成后，在满足一定的后置断言条件时，可以根据后置断言和后置断言方法的输入参数，转移到下一个方法上。这与扩展状态机中的 T 转换集合也是一致的。

(5)另外，状态组件有一些接口成员方法没有前置断言，即可以被直接调用而没有特定的要求。可以认为这些没有前置断言的方法是一种初始状态方法集，即对应于扩展状态机中的 s0。

由上述关联性分析可知，可以将状态组件的方法转换为有限状态机中的状态，而方法之间约束条件可以转换为有限状态机各状态之间的转移，其具体的对照形式如图 1 所示。

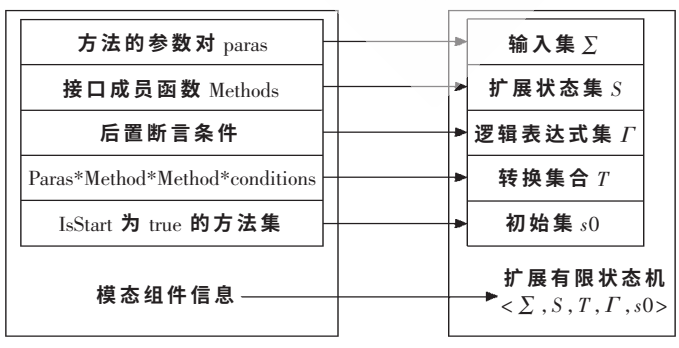

图 1 状态组件信息向扩展有限状态机转换时的对照图

综上所述，可以充分地认为状态组件可以转换为等效的扩展有限状态机模型。这样可以通过扩展用于存储扩展有限状态机信息的扩展邻接表来存储状态组件的信息。

2 状态组件测试用例生成

在将状态组件的信息转换为扩展有限状态机，并存入扩展邻接表和扩展约束对照表中后，可以通过有向图的相关性质来遍历状态组件所有方法转换，以此产生方法调用序列集。状态组件接口成员方法序列集的生成过程分为两个步骤：第一步是产生初始的方法序列集(算法 1)；由于第一步产生方法序列集中可能会存在冗余，因此第二步是去除初始方法序列集中的冗余方法序列，生成最终状态组件接口成员方法序列集(算法 2)。

算法 1：初始方法序列集的生成算法

输入：最大循环数 MaxCircle，状态组件的扩展邻接表 ExternAdjacencyTable，扩展约束对照表 ExternConditionTable

输出：初始方法序列集 SequencesList

```
{
  foreach(head ∈ ExternAdjacencyTable.ExternAdjacency-
                                                  Head)
    if(head is start method)
        CreateSequence(head,head.name);
}
CreateSequence(ExternAdjacencyHead head,string sequence)
{
  foreach(conn ∈ head.Connections)
  {
    TempSequence=sequence;
    TempHead=ExternAdjacencyHead[conn.MethodNum];
    if(TempHead.visittime<MaxCircle)
    {
        TempSequence+=conn.postConditionID+
                              TempHead.methodname;
        TempHead.visittime++;
        if(TempHead is end method)
            Add TempSequence into SequencesList;
        if(TempHead has connections)
            CreateSequence(TempHead,TempSequence);
    }
  }
}
```

算法的时间复杂度主要由初始方法的数量 M 以及方法的间转移的边数 E 决定。每一个初始方法都要完成一遍所有边的扫描，因此算法的时间复杂度为 O(M×E)。

算法 2：方法序列集约简算法

输入：初始方法序列集 SequencesList

输出：最终方法序列集 Sequences

```
{
    foreach(tempSequence ∈ SequencesList)
    {
      I=0;
      foreach(tempSequence2 ∈ SequencesList)
      {
            if(tempSequence2 contains tempSequence)
                I++;
            if(I>1)
                break;
      }
      if(I>1)
            continue;
      else
        Add tempSequence into Sequences;
    }
}
```

算法时间复杂度由两个 foreach 循环决定，两个循环都将扫描一遍初始方法序列集，因此算法时间复杂度 $O(n^2)$。

3 系统实现及分析

由前面的介绍可知，COM 组件测试用例及脚本生成子系统从组件的信息说明文档中提取基本信息，再针对状态组件和非状态组件的不同特性分别进行测试用例的设计。

对于非状态组件会将单个接口成员函数的参数按二元组合覆盖产生的测试用例直接用于生成测试脚本；而对于状态组件，二元组合覆盖产生的测试用例会用于填充接口成员函数方法序列中的相应函数，因此测试用例文档中会比非状态组件的测试用例多一个 TYPE 字段。在完成了测试用例和方法序列集的准备后，使用脚本生成模块来完成测试脚本的自动化生成及编译工作。COM 组件测试用例及脚本生成系统的结构如图 2 所示。

测试用例生成子系统关键实现技术主要有以下两项：

(1)信息存储与提取——XML 技术

可扩展标记语言 XML 近年来被许多行业广泛采用。组件信息、边界值信息以及测试数据等的存储都采用了 XML 文档的形式。使用 XML 来定义自己的数据信息存储格式，可以按照要求灵活地变更和修改。而且.NET 提供了丰富的操作接口，可以方便、灵活地使用 XML。

(2)测试脚本自动生成——反射机制和 CodeDOM 技术

反射机制(Reflection)是一种运行时机制，允许代码在运行时获得数据类型信息，即代码在运行时可以获得一个变量的数据类型。不仅如此，代码运行时还可以获得类的成员属性、方法、域和构造器等许多信息。测试用例生成子系统通过按照测试驱动器自定义的用户属性

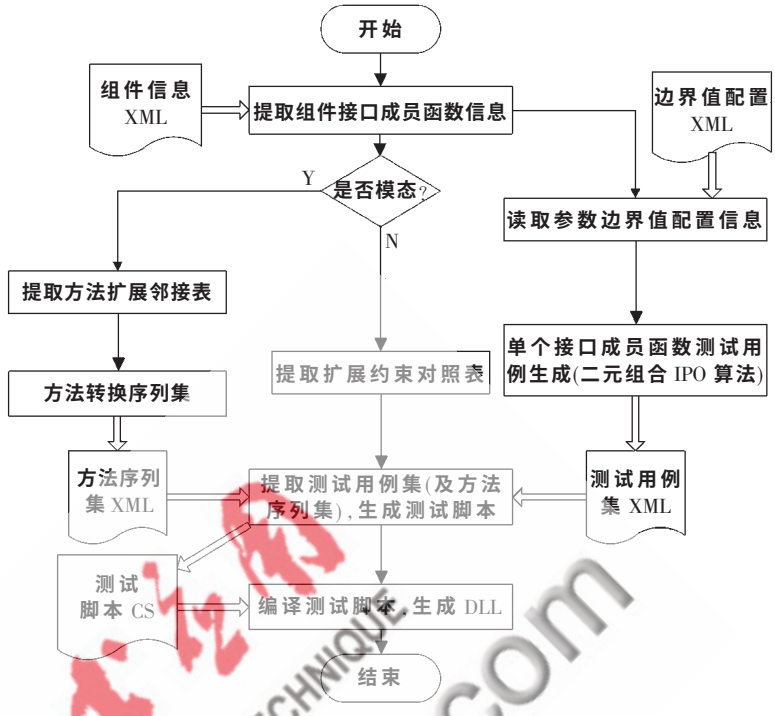

图 2 测试用例及测试脚本生成子系统结构图

来提供测试脚本，这包括类型属性[TestFixture]、开始函数属性[SetUp]、终止函数属性[TearDown]以及测试函数属性[Test]等。按要求生成的测试脚本在编译成测试程序集后，驱动器可以根据约定去查找具有相应属性的元素进行调度。这也是与动态监控和错误注入的一个接口约定。

CodeDOM 技术：.NET Framework 中用 CodeDOM 以一种语言中立的方式来表示源代码文档。通过 CodeDOM 创建对象图，然后使用特定于某种语言的 CodeDomProvider 类的派生类产生相应语言的测试脚本，最后编译生成测试用例集。COM 组件测试脚本到测试程序集的生成流程如图 3 所示。

对于状态组件，采用扩展邻接表来存储状态组件的状态信息，通过有向图的遍历算法自动产生方法序列，这样大大的提高了方法序列集生成的效率。同时在程序中加入了对循环的控制，并对有重复的方法序列进行了约简，大大的减少了测试序列集的数量，并保证了测试的有效性不会随之降低。表 1 展示了典型组件约简前后的方法序列数量的对照关系，可以看出约简后的方法序列数量较约简前有了显著的减少。

通过 IPO 算法生成的测试用例可以有效触发 COM 组件中的错误，例如针对 COM 组件 PDG2.dll 的接口成员函数 Register()生成测试用例及脚本，在编译成程序集后供驱动器使用，可以监测到错误异常。

状态组件是组件中经常使用的一类组件，状态特性的引入使得其测试更加的复杂化，测试用例生成的要求更高、代价也更大。本文通过分析状态组件的

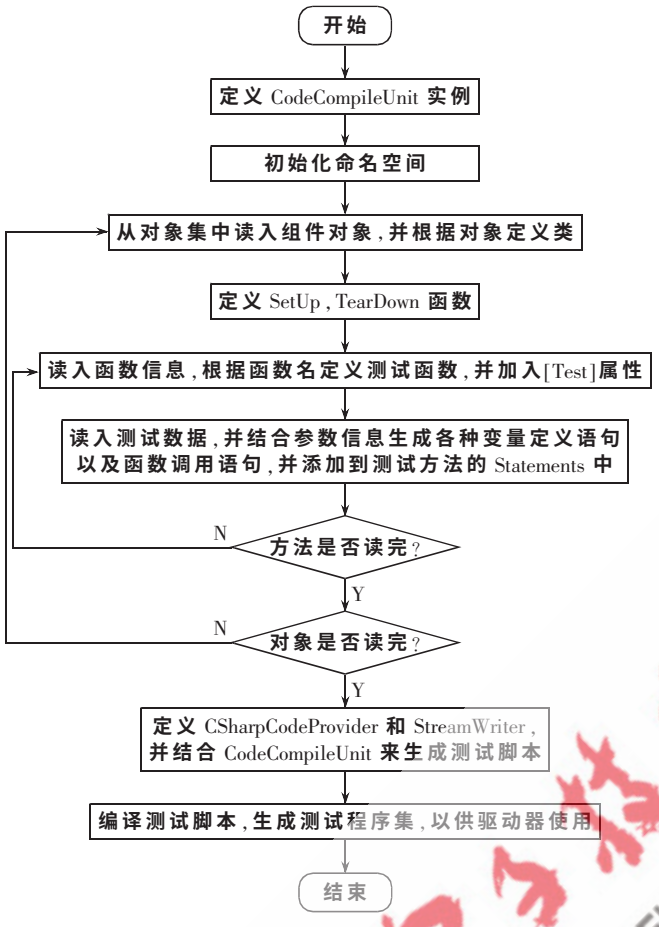

图 3 组件测试脚本及程序集的生成流程图

特性及其与扩展有限状态机的相关性介绍，结合自定义的扩展邻接表和扩展约束对照表等存储结构，较好地实现了状态组件测试方法序列用例及脚本的生成，也使得状态组件的测试代价大大降低、测试效率显著提高。

表 1 组件约简前后方法序列数量对照表

循环控制 N	2	3	4	5	6	7
约简前	18	68	250	922	3 430	12 868
约简后	6	20	70	252	924	3 432
减少为/%	33.3	29.4	28.0	27.3	26.9	26.7

参考文献

[1] ALI Y D, UYAR M U. A method enabling feasible conformance test sequence generation for EFSM models[J]. IEEE Transactions on Computers, 2004, 53(5): 614-627.

[2] 蒋凡，魏蓉，郜吉丰. 基于扩展有限状态机测试序列生成方法研究[J]. 计算机工程与应用，2007，43(7)：62-64.

[3] 战德臣，王忠杰，徐晓飞. 基于 XML 的组件标准化描述[J]. 计算机工程与应用，2003，39(4)：89-92.

[4] 刘永红. 构件及基于构件的软件测试研究[D]. 中科院成都计算机应用研究所，2006.

[5] 赵明华，陈榕，王小鸽. 基于元数据的构件自动测试技术研究[J]. 计算机工程与设计，2006，27(10)：1731-1736.

[6] CHANG Liu, RICHARDSON D. Software components with retrospectors[C]. In Proceedings of International Workshop on the Role of Software Architecture in Testing and Analysis, 1998: 63-68.

[7] RAKESH S, DAVID C, PAUL S. A passive test oracle using a Component's API[C]. Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05), 2005: 561-567.

[8] 杨建军，陈卫东，叶澄清，等. 面向组件的接口变异测试方法[J]. 浙江大学学报(工学版)，2003，37(2)：129-133.

(收稿日期：2009-12-26)

作者简介：

吴忠，男，1980 年生，本科，助理工程师，主要研究方向：软件集成与测试。