文章编号:1671-4598(2016)06-0143-04 DOI:10.16526/j.cnki.11—4762/tp.2016.06.039 中图分类号:TP311.5 文献标识码:A

基于模型的测试在工作流应用程序中的研究

汪祎然，张树东
（首都师范大学 信息工程学院，北京 100048）

摘要：近年来，工作流技术越来越多的应用于软件程序中，与之相关的新工作流语言以及工作流引擎也得到了快速的发展；但是，目前辅助工作流应用软件的测试方法仍显不足，特别是工作流引擎测试方面还存在着严重的局限性；为此，提出了基于模型测试的方法对工作流引擎进行测试，此外，还引入了抽象测试框架的概念，将其应用到工作流的测试上，它能够为工作流引擎构建测试环境和测试套件；最后，在 Cumbia 平台上搭建的一个工作流引擎上进行应用，说明了基于模型的测试方法和抽象测试框架可以有效应用于工作流引擎的测试。

关键词：基于模型的测试；工作流测试；基于跟踪执行分析；模型驱动工程

Research of Model Based Testing for Workflow Application

Wang Yiran，Zhang Shudong
（College of Information Engineering，Capital Normal University，Beijing 100048，China）

Abstract：In recent years，more and more workflow applications applied in the software program. The new workflow language associated with the new workflow and workflow engine also got rapid development. However，the current test methods of the development of the auxiliary workflow application is still inadequate，especially the workflow engine test also there are serious limitations. To this end，this paper presents a method based on model based testing to carry out on the workflow engine. In addition，it introduces the concept of the abstract test framework. Applying it to the workflow of the test，it can build the test environment and test suites for the workflow engine. At the end，a workflow engine built on Cumbia platform shows that the using of method based on model testing and abstract test framework can be effectively applied to the workflow engine testing.

Keywords：model based testing；workflow testing；trace based execution analysis；model driven engineering

0 引言

工作流应用程序凭借它自动进行业务流转、大大减少人力的优势，已经渗透到了日常生活的各个领域，如何保证工作流应用程序的可靠性和质量成为了关键的问题。测试是提高软件可靠性和保证质量的最基本手段，目前工作流应用程序的测试还没有比较系统的方法，许多工作流在没有得到充分测试的情况下投入生成环境中，质量难以得到保证。因此，迫切需要对工作流应用软件进行有针对性的测试。

基于模型的语言、模型驱动技术和以测试为中心的软件开发技术逐步成熟，使基于模型的软件测试方法与技术在近几年得到了广泛的应用。该种模式带来的好处，如灵活性和效率的提高，增加了对业务流程的控制和可视性。目前，国内外有一些相关的研究正在开展。例如，为了更好的对 Web 端应用程序进行测试，Lsakowitz 等人将关系管理方法论用于描述 Web 应用的设计[1]；Coda 等人给出了面向对象模型 WOOM[2]，用高等抽象的原始实体来描述 Web 应用开发；Gellersen 等人提出了一种 Web 复合方法来结构化 Web 的开发过程[3]，扩展 UML 对 Web 应用的体系结构进行建模，以此建立 Web 应用的抽象测试模型。

为了使工作流应用软件的测试更有针对性和实效性，本文提出了两个研究成果：第一，提出了一种基于模型的工作流测试方案，并论述了基于 Cumbia 平台的工作流测试过程。Cumbia 是一个用来构建基于可执行模型的工作流应用程序[4]。第二，提出了抽象测试框架技术。抽象测试框架是一种用来开发基于模型的测试环境和特定工作流应用程序的测试套件。通过在 Cumbia 平台上的应用，证明了基于模型的测试方案和抽象测试框架可以有效应用于工作流引擎的测试。

1 工作流中基于模型的测试

1.1 基于模型的测试

基于模型的测试（Model-Based Testing，MBT）是一个通过比较被测系统以及被测系统模型行为和输出结果来测试的方法[5]。基于模型的测试基本步骤如下：

首先需要创建一个机器可读的模型，该模型表述了需求所表述的所有可能行为。当模型成型以后，基于模型的测试工具就能够通过分析模型自动生成测试用例。测试用例一旦生成，就可以在一个标准的单元测试框架中独立于模型运行。这些测试用例提供了测试序列去控制待测试系统，同时观察待测试系统的返回值，并与生成预期值进行比较，然后做出判定测试是通过还是失败，如图 1 所示。

其中对于测试用例的生成的环节，我们可以借鉴 Chourouk Bourhfir 提出的基于扩展有限状态机（Extended Finite State Machine，EFMS）测试的理论[6]以及 Diego Latella 提出的 UML 状态图测试框架来自动的生成测试用例[7]：

1）将实际的业务流程首先转化为状态图，再将具有层次

收稿日期:2015-12-22； **修回日期**:2016-03-08。

基金项目:国家科技支撑计划项目(2013BAH19F01)；高可靠嵌入式系统技术北京市工程研究中心，电子系统可靠性技术北京市重点实验室，北京市属高等学校创新团队建设与教师职业发展计划项目。

作者简介:汪祎然(1990-)，女，硕士研究生，主要从事软件工程、软件测试方向的研究。

张树东(1969-)，男，博士，教授，主要从事计算机网络、分布式计算方向的研究。

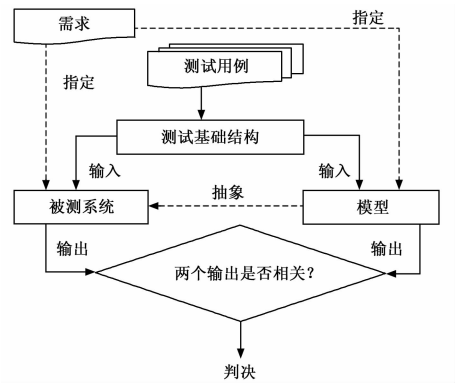

图 1 基于模型的测试

结构的状态图展开为 EFSM。

2）生成测试消息系列，对 EFSM 进行遍历，有向图的遍历可分为深度、广度两种遍历方式。为使生成的测试消息序列最短，同时运用两种方法生成不同的测试树，然后为每个状态选取从初始节点达到该状态的最短路径作为测试消息序列。

3）在 EFSM 的基础上为每个状态找到最短的唯一输入输出序列（Unique Input Output Sequence，UIOS），只要 EFSM 是连接的就可以保证为每个状态找到最短的 UIOS。

4）把每个状态的测试消息序列和 UIOS 结合起来组成测试用例。

因此，由于测试消息序列和 UIOS 都是路径最短的，所以测试用例也能保证是路径最短的。

1.2 测试工作流引擎

一般测试工作流引擎必须考虑到特殊的需求[8]，比如测试工作流引擎首先是验证它是否遵循工作流语言语义。但是仅仅对于实现其正确性的输出检查是不够的，重点是验证中间结果和涉及其中的每个元素是否正确执行，因此有必要具备一套完整的测试套件和更简易建成和发展的测试环境，以适应新的需求。工作流引擎的关键是支持流程的多个并发实例[9]，拥有高度并发性，为此，我们提出了抽象测试框架来提供特殊的测试环境。

2 抽象测试框架中的定义和执行测试用例

抽象测试框架（Abstract Test Framework，ATF）是用于开发基于模型的测试环境，为应用工作流创建测试套件的一个框架[10]。每一个使用抽象测试框架的环境可以共享该框架提供的两个事物。第一，它指定测试用例的结构，并提供基本机制执行它们。第二，它指定一个机制分析测试用例的执行，并提供工具执行这些分析。

2.1 测试用例的结构

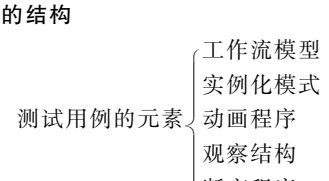

图 2 测试用例的元素

抽象测试框架的测试用例由 5 个元素组成，如图 2 所示。工作流模型是在测试用例运行时执行的流程。实例化模式是指如何对测试用例中的每个工作流模型举例和有多少个实例被创建。动画程序是由动画语言编写的，控制工作流的执行。观察结构用来实现执行中的信息收集。断言程序用于分析由观察结构收集来的数据。

2.2 执行测试用例

一旦工作流测试用例的元素被定义，测试用例就可以在抽象测试框架中执行[11]。工作流测试用例的执行涉及 4 个阶段，如图 3 中所示。

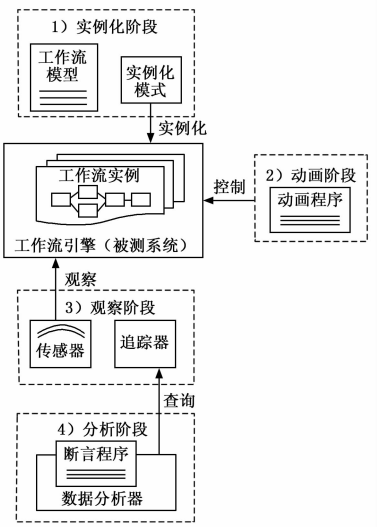

图 3 抽象测试框架中的工作流测试用例执行

在实例化阶段，一个实例化模式被用作创建工作流模型实例，实例在这个阶段创建观察结构。接下来在动画阶段，动画程序被执行。同时观察阶段也开始。一旦流程执行完成，分析阶段开始，断言程序就被执行[12]。在这一过程中，程序通过比较预先定义的预期结果与聚集在路径的相应信息，完成对一组断言的验证，数据分析器用于分析断言程序，从路径获得详细的信息。

2.3 抽象测试框架的元素

抽象测试框架规定了 3 个主要的部分：测试装载器、测试行为和测试运行器，如图 4 所示。

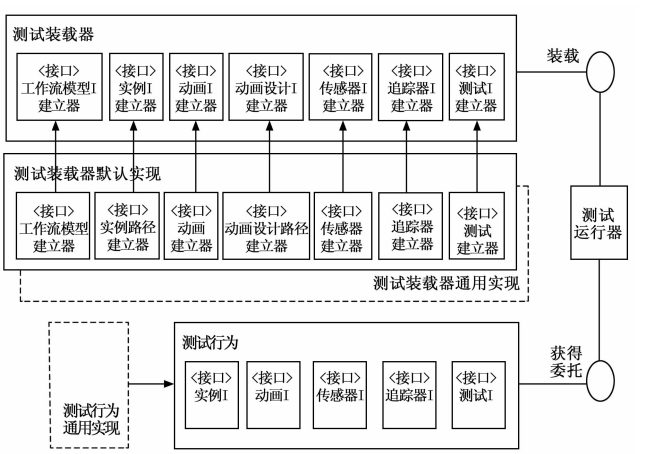

图 4 抽象测试框架的组成

测试装载器是负责装载测试用例的定义，其需要对装载的每个元素展示接口，使得它可以自定义不同测试用例来进行表示。测试行为是负责实施元素的行为，测试用例的执行。测试运行器使这两个部分的执行相协调。

由于测试行为对于每个使用抽象测试用例的工作流引擎均

需定义，这个部分就不能包括一个默认接口。反之，它只能向那些在用例执行中动态使用的部分展示接口。第一个必须实施的接口是实例Ⅰ，它的两个主要职责是初始化工作流引擎和创建所给工作流模型的实例。工作流模型的信息和其它接口需要的信息一样，使用测试装载器从测试用例的定义装载。一旦实例Ⅰ被定义，动画设计Ⅰ开始执行。这个接口的职责是翻译和执行一个动画程序。和动画设计Ⅰ一起，追踪器Ⅰ和传感器Ⅰ两个接口必须同时执行。它们的职责包括在动画执行中对事件做出反应、使事件和路径联合和处理那些事件的路径。这些接口一旦执行，测试Ⅰ接口就开始执行。这个接口是利用一组数据分析器来翻译和执行断言程序的，他们的职责是查询聚集在元素中的信息，使追踪器Ⅰ的接口执行。最后，测试运行器动态装载一个工作流测试用例中必要的元素，并且使用这些元素执行已给出的工作流测试用例定义。

3　测试开发路径

测试开发路径描述了构建一个已给出的工作流引擎的测试套步骤，包括一个基于抽象测试框架的测试基础架构的建构模型[13]。这个路径是由两个独立部分组成的，如图 5 所示。

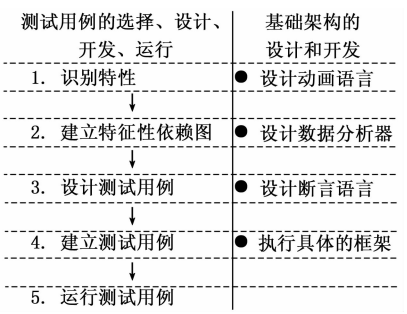

图 5　测试开发路径图

左边是设计、建造和运行测试用例的操作。第一步，识别特性，工作流语言中所有可测试的特性均需要被识别。共同特性包括控制结构，例如联结、分裂和顺序，还有不同种类的数据管理。第二步，建立特性依赖图，将识别出的特性组成有向图，在两个节点直接用弧线显示特性之间的依赖关系。第三步，设计测试用例，将所有需要为引擎创建一个全面的测试套件的活动进行分组。为了在第一步中识别出的每个特性，至少要设计一个测试用例。第四步，建立测试用例，使测试用例的说明具体化。最后一步，运行测试用例。这步使用一个来自特性依赖图的测试脚本，使测试执行自动化进行，在日益增加的复杂性中按一定的次序促进问题的检测。

右边是构建基础设施的操作，用来支持结构和建构和执行那些测试用例，这些步骤不必以特定的顺序执行。设计动画语言把需要的活动分组，设计用来编写动画程序的语法和语言语义；设计数据分析器把必要的任务分组，定义数据分析器所需的测试用例以及他们中的每一个如何收集必要的信息；设计断言语言把需要的活动分组，设计一个可以被用于编写断言程序的语言。最后步骤是执行具体的框架。

4　测试 J-Cumbia：一个基于抽象测试框架和测试开发路径的案例研究

4.1　XPM 引擎及 J-Cumbia 测试框架

Cumbia-XPM（eXtensible Process Metamodel）是一个用于描述工作流过程的元模型[14]。这个元模型由一套开放对象的特殊化元素组成。这些开放对象其中包括一个实体、一个状态机和一些与状态机的转换相关联的动作。J-Cumbia 测试框架是一个用来测试 J-Cumbia 引擎的专业化抽象测试框架，其工作流模型是 Cumbia-XPM 模型。

4.2　遵循路线图

下面就开始描述如何遵循测试开发路径步骤，使用测试基础步骤来测试 J-Cumbia，具体的测试用例如下：

第一步，识别特征。在分析 Cumbia-XPM 之后，根据 Cumbia-XPM 的主要特征识别出的特征总结。

第二步，建立特征依赖图。为确定 Cumbia-XPM 的特征组织为特征依赖图，如图 6。每一个测试用例在第三步的设计中，将与特征依赖图中的一个节点相关联。

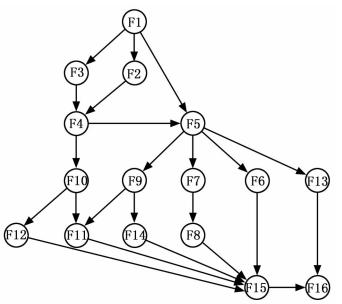

图 6　Cumbia-XPM 特征依赖图

第三步，设计测试用例。要详细说明一个测试用例的设计[15]。所谓的“进程 1”包含 3 个活动，即活动 1、活动 2 和活动 3。当这些活动执行，他们的工作区可以被测试用例的动画程序所控制。根据 XPM 的语义学和进程的结构，活动 3 只能在活动 1 和活动 2 执行完成后被执行。这个测试用例将检车这个规则是否遵守。图 7 在括号内显示数据在进程中的活动之间流动。

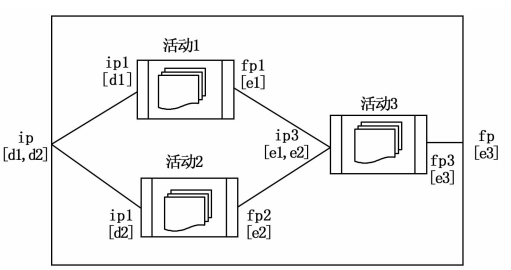

图 7　F8 的测试场景

对这个测试用例中的每一个执行，我们只要有一个进程的实例。因此，我们只要有一个动画程序，便可以指定数据初始化进程，指定命令控制活动 1、活动 2、活动 3 的工作区执行。

下面则为测试用例设计的动画程序：

```
animation{
init{
port(process1:pi){
var d1="data1";
var d2="data2";}
}
workspaces{
workspace(process1:ACT1:ws){
1:var e1=input(d1);
output(e1);}
```

```
workspace(process1:ACT2:ws){
1:var e2=input(d2);
output(e2);}
workspace(process1:ACT3:ws){
1:var t1=input(e1);
var t2=input(e2);
var e3=concat(t1,t2);
output(e3);}
}
}
```

这个程序的第一部分指定了进程将使用一对数据来初始化：data1 的值赋给 d1，data2 的值赋给 d2。活动 1 和活动 2 的工作区的行为是相似的：他们第一次被执行，他们获取了 d1 和 d2 的值，并用 e1 和 e2 给输出命名。当活动 3 被执行，它的工作区串联 e1 和 e2 的值，输出结果作为 e3。为了对这个测试用例描述一个成功的执行，使用断言语言编写了 4 个断言。

```
//断言 1:活动 3 的输入端就只接收一次数据。
assertion("1"){
let pwf=timeFull(process1.ip3) in
equal(pfw,1)}
//断言 2:活动 1、活动 2、活动 3 每一个就只被激活一次。
assertion("2"){
let act1=timesActivated(process1.ACT1) &&
let act2=timesActivated(process1.ACT2) &&
let act3=timesActivated(process1.ACT3) in
equal(act1,1) &&
equal(act2,1) &&
equal(act3,1)}
//断言 3:活动 3 的输入数据应该为"data1"和"data2"。
assertion("3"){
let inputE1= activityInput(process1.ACT3,1,e1) &&
let inputE2= activityInput(process1.ACT3,1,e2) in
equal(inputE1,"data1") &&
equal(inputE2,"data2")}
//断言 4:活动 3 需要在活动和活动 2 完成后被激活。
assertion("4"){
activityFollows(process1.ACT3,process1.ACT1) &&
activityFollows(process1.ACT3,process1.ACT2)}
```

上面提出的这 4 个断言为观察结构提出了需求。下面的事件和信息与检验断言相关：活动 3 的输入端获取完整（断言 1）；活动 1、活动 2 和活动 3 被激活（断言 2）；当活动 3 被激活后，输入数据被获取（断言 3）；活动 1 和活动 2 失效，活动 3 被激活（断言 4）。此信息可以通过传感器获得，通知活动的激活（在状态机中转换“激活”）和端口的数据接收（在状态机中转换“包”）。

第四步，建立测试用例。此步骤可依据上文中的自动生成测试用例方法来实现。

第五步，运行测试用例。我们提供了两种方法运行测试用例。第一种，使用测试脚本自动地执行所有测试用例，结果会以文本的方式告知用户。第二种，使用一个 Eclipse plugin 开发的图形化工具运行测试用例[16]。这个工具叫做 Test Viewer，以图形化的方式，呈现测试用例的执行和结果。为了显示结果，使用一个类似 JUnit 的 Eclipse 视图[17]，断言程序的结果可以很容易的被查看。

5 结语

本文详细阐明了一种基于模型的测试工作流应用程序的方法。为了配合这种特殊的测试方法，文中提出了抽象测试框架和测试开发路径，以此来搭建测试的环境以及创建相关套件，最终在 J-Cumbia 环境下应用，证明了基于模型的测试方案和抽象测试框架可以有效应用于工作流引擎的测试。文中使用了一个 XPM 进程引擎建立在 J-Cumbia 平台之上，模拟抽象测试框架与测试开发路径的广泛适用性。尽管最初抽象测试框架和测试开发路径是基于 Cumbia 引擎开发的，但他们同样可以应用在其他不相关的工作流引擎上。随着当前软件规模的越来越大，测试的规模也必将逐渐扩展，这就使得纯粹的基于程序的测试十分困难。而基于模型的测试不仅可以有效地提高效率，还提高测试用例生成的自动化程度，进行测试失效辨别，也有利于测试结果的评估与分析。

虽然文中提到的方法已在 J-Cumbia 平台上实现，但自动化测试还需要能够对测试用例的结果进行自动分析，在该方面还留有不足，将来也将进一步改进完善。

参考文献：

[1] Torsel A M. A testing tool for web applications using a domain-specific modelling language and the NuSMV model checker [J/OL]. IEEEXplore.

[2] 张 玲. Web 应用自动化测试框架的研究和应用 [D]. 上海：华东理工大学，2014.

[3] Zhu B B，Guo R，Feng M，et al. Component-oriented architecture for web mashups [P]. US：US9009657，2015.

[4] 冯 洋，张冬冬，刘 群. 层次短语翻译模型的介词短语调序 [J]. 中文信息学报，2012，26 (1)：31－36.

[5] 张 清. 基于模型的自动化测试工具的实现 [D]. 北京：北京交通大学，2013.

[6] 陆公正. 基于 EFSM 模型的测试用例优化生成及实例化 [D]. 上海：上海大学，2014.

[7] Aichernig B K，Lorber F，Tiran S. Formal test-driven development with verified test cases [A]. Special Session on Model-based Analysis & Testing of Embedded Systems [C]. 2014：626－635.

[8] 王双成，刘念祖，王小玲. 基于局部依赖分析的特征子集选择 [J]. 计算机研究与发展，2014 (44)：329－333.

[9] 冯 韵. 基于标记语言的工作流过程描述语言 [J]. 电脑知识与技术，2012 (8)：43－45.

[10] 郭 曦，张焕国. 基于谓词抽象的测试用例约简生成方法 [J]. 通信学报，2012，33 (3)：35－43.

[11] 张孟颖. 基于工作流的软件测试管理系统设计与实现 [D]. 南京：南京理工大学，2013.

[12] 王志才. 一种基于断言图的模型抽象技术的研究 [D]. 成都：电子科技大学，2013.

[13] 周婷婷，费树岷. 基于工作流的办公自动化系统研究与设计 [J]. 工业控制计算机，2013，26 (4)：102－104.

[14] 王 平. 基于组件的工作流定义工具的研究与实现 [D]. 沈阳：沈阳工业大学，2014.

[15] 张 娟. 软件测试中测试用例复用的研究 [D]. 上海：上海大学，2012.

[16] 张 娟，余童兰，吴取劲，等. 基本路径生成算法的图形化设计与实现 [J]. 南华大学学报（自然科学版），2014 (1)：88－90.

[17] 杨 鹏. 基于 Feed4JUnit 架构的单元测试技术研究与应用 [J]. 软件工程师，2014 (7)：25－27.