第34卷第3期 萍乡学院学报 2017年6月
Vol.34 NO.3 Journal of Pingxiang University Jun.2017

基于消息交互的测试用例编写方法的实现

颜 丽

（萍乡学院 信息与计算机工程学院，江西 萍乡 337000）

摘 要：在编写自动化测试用例时，消息交互流程的逻辑代码是测试中必不可少的关注点，同时消息接口及其上所携带的消息内容也是测试主要关注点。本文介绍了一种针对消息交互的测试用例编写实现方式，可以较好的改善同类用例的可读性，从而使得用例的编写和维护变得简单、明确。

关键词：交互流程；消息构造；接口测试

中图分类号：TP311.1 **文献标识码**：A **文章编号**：2095-9249（2017）03-0069-03

1 问题的提出

自动化测试是在指在无人工干预的情况下，自动的完成一系列测试，因此它能够极大的提高测试的工作效率，从而保证软件的代码质量[1]。根据测试的执行和校验方式，自动化测试分为系统测试，功能测试，单元测试三类[1]。一般情况下，功能测试工具[2]在编写测试用例时，面对消息交互流程的测试方法都是将消息构造逻辑、接口校验逻辑与消息统一检查，这样使得用例的可读性差，同时也在相似的消息处理上以及在一次消息交互流程内导致了很多冗余代码，使得自动化测试用例的开发维护成本较高，影响了自动化测试的高效性[4]。

2 解决思路

要解决背景介绍中提到的功能测试的用例编写问题，就需要使测试用例能清晰表现出每个消息流。

为了满足上述要求，首先必须要有明确的消息发送源和消息接收的目标，主要就是通过模拟外围交互的虚拟子系统构造消息原，并发给真实被测目标系统。然后确定功能测试用例的编写机制，本方法是提前将消息的处理进行注册，对每一步消息流程都注册一个流程执行接口用于匹配是否为期望消息和是否要继续执行，以及一个消息处理接口用于构造消息或校验消息），并使真实模块和虚拟模块间能够按照注册的消息流程和接口顺序执行。最后校验被测系统的消息输出是否正确来完成自动化测试，其流程如图1所示。

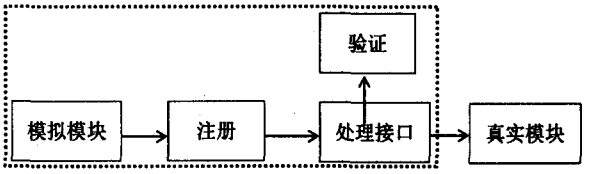

图1 基于消息交互的测试用例编写流程

该解决思路使得依赖用例的消息构造代码逻辑、接口校验逻辑与消息流程得到了有效分离，用例的目的就变得明确易读。

3 实现要点

本文所述方法的测试用例编写代码如下：

```
_TEST(Admit)
{
adopter<<submitter (expects(ADMIT_REQ), with(admitReq));
adopter>>submitter (expects(ADMIT_RSP), with(admitRsp));
adopter<<submitter (expects(ADMIT_ACK), with(admitAck));
}
```

收稿日期：2017-04-18

作者简介：颜丽（1983—），女，江西萍乡人，讲师，硕士，研究方向：计算机应用、软件工程、数据库。

此用例的功能是一个请求-响应的三次消息的交互：

（1）submitter 向 adopter（参考箭头的方向）发送了 ADMIT_REQ 请求消息。其中，消息接口的内容通过 admitReq 类来进行构造。

（2）submitter 期待 adopter 向 submitter 回应一个 ADMIT_RSP 响应消息。其中，消息接口内容的校验是通过 admitRsp 类来进行的。

（3）submitter 向 adopter 发送了 ADMIT_ACK 确认消息。其中，消息接口内容通过 admitAck 类来进行构造。

3.1 真实模块与虚拟模块

一般情况下，测试过程可以将被测子系统作为真实模块，而将其他外围交互的子系统都作为虚拟模块。这样只需画出真实模块与虚拟模块之间的消息交互流程，就可以完成测试用例的编写。

基于以上原则，用例中的真实模块与虚拟模块可以转化为如下代码：

```
SimuProcesssubmitter;
RealProcessadopter(EXPECTS_MSG);
```

3.2 消息处理的注册机制

为了在注册中将消息的流程执行和消息本身的构造与校验分离开来，本方法有效的将消息构造和校验封装成了一个 TestObj 对象，而将消息流程的执行封装成了 TestAct 对象。同时定义了一个的位置指示 pos 来记录当前消息流程执行到了哪一个环节。代码如下所示：

```
std::vector<TestObj*>object_container;
std::vector<TestAct *>action_container;
intpos;
```

3.3 构造消息的处理

一个消息构造的过程就是模拟虚拟模块给真实模块发送一个消息过程。注册消息构造的具体代码如下：

```
RealProcessconst operator<< (TestProcess&rhs)
{
TestObject *simuObj = InteractionManager::getInstance().getCurrentObject();
FtDbg("%s receive msg %s!", GetProcName(procType),
GetMsgNameByProc(procType, simuObj->getMsgId()));
InteractionManager::getInstance().appendAction(new MsgReply(procType));
return *this;
}
```

从以上代码可以看出，消息构造的核心就是注册了一个 MsgReply 的对象，该对象主要就是调用消息构造接口完成消息的构造，并将构造好的消息发送给指定的真实模块。由于消息构造是从虚拟模块发给真实模块，因此消息流程在执行时，不需要其他行为驱动就可以直接发送。

3.4 校验消息的处理

一个消息校验的过程就是当真实模块按测试流程处理完毕，并回应给虚拟模块一个消息时。注册消息校验的具体代码如下：

```
RealProcessconst operator>> (TestProcess&rhs)
{
TestObject *simuObj = InteractionManager::getInstance().getCurrentObject();
// 验证消息名和 ID 及对应处理流程
FtDbg("%s send msg %d!", GetProcName(procType), simuObj->getMsgId());// 发送消息构造 InteractionManager::getInstance().appendAction(new MsgReceive());
return *this;
}
```

从以上代码可以看出，消息校验的核心就是注册了一个 MsgReceive 的对象，该对象主要就是判断真实模块发送的消息是否为期待消息，并调用具体的消息校验接口校验消息内容的正确性。由于消息校验是从真实模块发给虚拟模块，因此消息流程在执行时，只有等到了期待的消息，才能驱动走到下个消息。

3.5 用例的消息执行

每个用例的实现主要是一个消息流程的注册处理过程，那就需要提前将消息流程的行为与接口按照用例要求注册好。若执行时出现了与注册预期不一致的情况，则用例失败；否则注册的消息流程都处理完后，则用例执行成功。除此之外，在进程开始时需要注册一个额外的操作处理接口。在用例执行时就会将操作处理接口动态映射到代理上，让代理负责管理用例的真正执行，管理过程如图 2 所示。

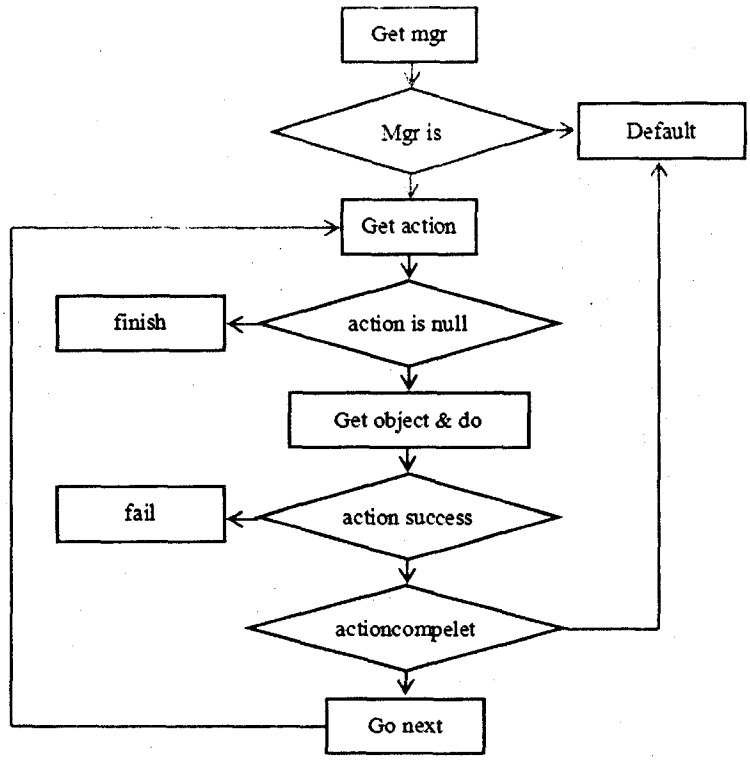

图 2 用例执行的管理过程

4 效果评价

本方法解决了功能测试工具在编写测试用例时将大量的消息构造代码逻辑、接口校验逻辑与消息流程混杂在一起的问题，使得用例的可读性和可维护性得到了较大提高，同时让用例的重用性也变得更简单，并且使得测试用例的开发和扩展变得更容易。对于消息流程复杂或消息流程场景很多的功能测试用例的开发场景，具有一定的参考性和借鉴性。

参 考 文 献

[1] 李明泽. 浅谈软件自动化测试技术[J]. 科教导刊，2016,（11）：171~172.

[2] 李华宝. 软件测试自动化关键技术分析[J]. 数字技术与应用，2016,（6）：234~234.

[3] 呼晓黎. 软件自动化测试技术的研究与实现[D]. 成都：西南交通大学，2007.

[4] 邓璐娟. 自动化测试框架技术及应用[J]. 计算机测量与控制，2016, 24,（9）：86~88.

〔责任编校：范延琛〕

Implementation of Test Case Writing Method Based on Message Interaction

Yan Li

(School of Information and Computer Engineering, Pingxiang University, Pingxiang Jiangxi 337000, China)

Abstract: When writing automated test cases, the logic code of the message exchange process is an indispensable concern in the test, and the message interface and the message content carried on it are also the primary focus of the test. The paper introduces a way to prepare the test case for message interaction, which can improve the readability of similar cases, and makes the writing and maintenance of the cases simple and clear.

Key words: interactive process; message construction; interface testing