基于领域驱动的自动化测试框架研究与应用

胡继东,鞠炜刚
(中兴通讯南京研究所,江苏 南京 210012)

摘 要:随着软件产品的交付周期越来越短,自动化测试的应用范围更加广泛。为解决自动化测试用例编写维护复杂、效率低下的问题,在传统测试框架的基础上提出了一种基于领域驱动的自动化测试框架。用领域语言来描述测试用例,对被测领域进行领域建模,以组件化思想为指导,采用分层架构,用面向对象方法设计、开发了领域测试库,同时提出了领域驱动测试的实施流程,包括启动、建模、设计、开发、测试、版本管理和发布的过程、方法。通过采用领域驱动测试框架可以使得测试用例的组织、设计和开发更加有效,提高了测试用例的开发和维护效率,测试用例更加易于解性、清晰简洁,能够通过重构快速应对变化,在通信系统测试中得到了应用推广,取得了很好的效果。

关键词:领域驱动;自动化测试;领域建模;组件化;分层架构
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2016)04-0162-05
doi:10.3969/j.issn.1673-629X.2016.04.036

Research and Application of Automation Testing Framework Based on Field-driven

HU Ji-dong, JU Wei-gang
(ZTE Nanjing Institute, Nanjing 210012, China)

Abstract: As the delivery period of software products becomes shorter, the application scope of automation test becomes wider. In order to solve the problem of low efficiency and complexity in writing and maintaining automation test cases, an automation test framework is put forward based on domain-driven on the existing traditional test framework. The new framework describes test cases in domain language, creates model to the tested domain, guides design with the idea of modularization, adopts layered architecture, and designs and develops domain test library by using ODD. At the same time, the implementation procedure of domain-driven test is raised, which includes process and method of starting, modeling, design, development, test, version management and publishing. By adopting the domain-driven test framework, the system makes organization, design and development of test cases more efficient, and thus improves efficiency in development and maintenance of test cases. In this case, test cases become easy to understand, clear and concise, and satisfy quick changes requirements by using restructuring. Thus, the framework is widely deployed in telecommunication system tests and has achieved good results.

Key words: domain driven; automated test; domain modeling; modularization; layered architecture

0 引 言

软件测试在软件生命周期中占有十分重要的地位,是保障软件质量的重要手段,因此必须不断地对软件进行测试[1]。随着市场竞争的加剧,软件产品的交付周期越来越短,采用纯手工测试的方法越来越不能满足需要,因此通过自动化测试,提高软件测试的质量和效率,缩短软件的交付周期。但随着软件产品的规模和复杂度越来越高,测试脚本的编写越来越复杂,不能满足自动化测试的要求,需要采用新的技术和方法[2-3]。文中提出了一种基于领域驱动的自动化测试思想,在此基础上设计开发了自动化测试框架,以更好地完成对软件产品的自动化测试。

1 传统测试框架

随着自动化测试的应用范围越来越广,测试用例规模越来越大,复杂度越来越高,需要有相适应的自动化测试框架来支持。在传统测试框架中,测试用例脚本由复杂的流程和实现细节描述,直接提供给测试工

收稿日期:2015-07-12 **修回日期**:2015-10-16 **网络出版时间**:2016-03-22
基金项目:国家自然科学基金资助项目(61402482)
作者简介:胡继东(1979-),男,硕士研究生,工程师,研究方向为软件测试、敏捷测试。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20160322.1521.066.html

具执行,对被测系统进行测试[4-5]。其中测试工具是定制化的工具,测试用例脚本必须采用工具所能支持的格式来编写,不具有通用性。由于涉及具体的流程和测试实现细节,存在以下问题:

(1)测试用例脚本细节众多,编写速度慢,效率低;

(2)测试用例的编写有大量的拷贝粘贴,不符合编码规范;

(3)一旦接口实现变化就需要修改所有相关的测试用例脚本,可维护性差;

(4)耦合度高,用例整合耗费大量精力。

这些问题导致测试用例脚本编写维护的成本很高,速度慢,同时脚本不容易理解,给测试人员使用带来了极大的不便。一般的关键字驱动框架虽然解决了部分问题,但缺乏有效的方法从测试领域出发来解决关键字的设计、组织难题,需要有一种新的自动化测试框架和方法来解决这些问题。为此文中研究设计了基于领域驱动的自动化测试框架,并在通信系统产品的测试中进行了有效的应用。

2　领域驱动测试概述

2.1　领域驱动测试定义

领域驱动测试就是用一种高层的领域语言来描述测试用例,在一种通用的框架调度下驱动领域测试关键字库,直接驱动测试引擎发起测试的方法。

领域驱动测试是在关键字驱动测试的基础上发展而来的[6-7],其主要特点是从被测系统的领域模型出发,从测试角度进行领域建模,并通过分层设计对测试用例和领域关键字进行有效的分析、设计、开发和组织管理,从而可以有效地对被测系统进行测试。

2.2　领域驱动测试优点

领域驱动测试主要具备以下优点:

(1)测试用例用领域语言描述,更加清晰、简洁、易于理解;

(2)测试用例可以通过组合领域关键字和参数来设置;

(3)当实现方式发生变化时,可以通过更新升级领域驱动测试库来应对,测试用例不需要做任何修改;

(4)针对敏捷测试变化的需求,重构领域测试模型和用例,适应变化。

3　基于领域驱动的自动化测试框架

3.1　系统框架

改进后的基于领域驱动的自动化测试框架如图 1 所示。

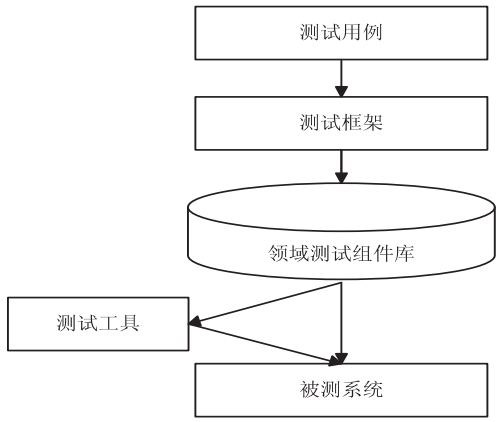

图 1　领域驱动测试系统框架

在基于领域驱动的自动化测试框架中,测试用例脚本由高层的领域语言描述,提供给一个通用的测试驱动框架。该框架进行调度执行,调用领域驱动测试库中的相关高层领域测试关键字,直接驱动测试工具引擎发起测试。采用高层领域语言描述用例的一个好处是测试用例脚本清晰、简洁、易于理解,并可以快速编写、修改和维护。从改进前后的测试框架原理来看,实质是采用了组件化的思想和分层架构,下面进行详细阐述。

3.2　组件化思想

基于领域驱动的自动化测试框架的测试用例脚本结构如图 2 所示。

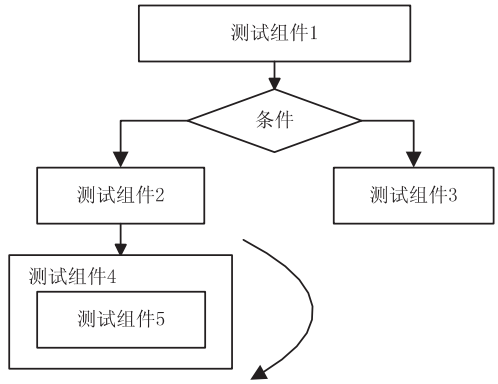

图 2　组件化的测试用例脚本结构

图 2 中的各测试组件为领域驱动测试库中的相应领域测试关键字。测试用例脚本由高层的领域语言和领域驱动测试库构成,具体的测试用例脚本通过测试库的领域测试关键字组合而成,可以有顺序、选择和循环三种组合关系。组件化带来的好处是增强了测试脚本代码的可复用性,减少了重复和冗余,同时测试人员使用领域测试关键字组装测试用例脚本,也易于理解,编写简单[8]。

3.3　分层架构

测试用例脚本由高层领域关键字组成,领域测试关键字库的设计采用了分层架构,以某产品的测试为例,如图 3 所示。

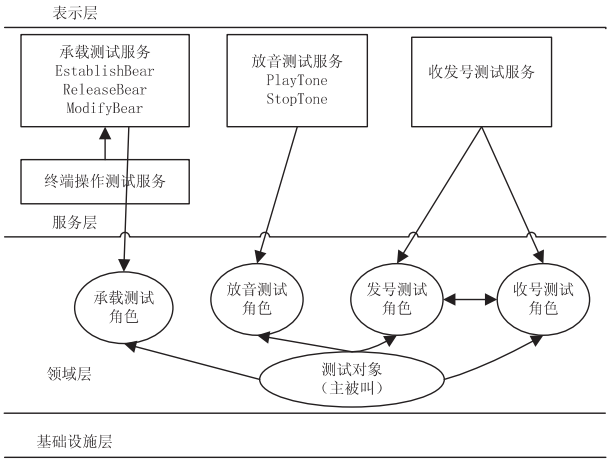

图 3 分层架构

测试人员在表示层编写测试用例,测试用例由服务层不同测试服务提供的高层领域关键字组合而成;领域驱动测试库则由服务层、领域层和基础设施层构成,层次清晰。

其中服务层对外提供高层领域测试接口,对内协调驱动领域层测试对象交互协作完成测试。服务层一般根据不同的业务范围划分为多个测试服务,每个测试服务提供一系列的高层领域测试关键字。例如,承载测试服务提供了建立承载、释放承载等测试接口。

领域层提供了一系列领域对象,主要包括各种测试角色和测试对象。其中测试角色实现具体的业务测试逻辑,通过相互交互协作完成测试;测试对象则在不同的测试场景中扮演不同的角色。

基础设施层提供了对被测设备控制以及文件、网络、异常等基础设施。

采用表示层、服务层、领域层和基础设施层的分层架构[9],测试用例脚本和测试库结构清晰,明显优于扁平化结构,更有利于测试用例脚本和领域关键字库的组织管理。

3.4 领域建模和设计

基于领域驱动的自动化测试框架的一个重要核心思想是采用领域建模和面向对象的设计。其主要特点是从被测系统的领域模型出发,从测试角度进行领域建模,并使用面向对象方法进行设计[10-11]。

其中,建立领域模型需要根据需求的变化和理解的深入不断进行重构。采用面向对象设计可以很有效地将领域模型映射为实现对象,同时可以采用面向对象设计的一些原则,甚至可以使用一些设计模式,来达到优化设计的目的[12-13]。

4 实施流程

领域驱动测试的实施流程分为启动、领域建模、设计、开发和测试、版本管理和发布这五大步骤,下面介绍每一步骤的实施方法。

4.1 启 动

在启动阶段主要评估领域驱动测试实施的必要性,如果至少具有以下因素之一,则因素越多采用基于领域驱动的自动化测试框架的必要性越大:

(1)测试用例编写太复杂,效率低下;

(2)测试用例的测试逻辑和参数需要补充丰富;

(3)新版本跨度较大,测试用例需要大量重新修改、整合、裁剪;

(4)协议升级导致测试用例大量细节要修改;

(5)被测业务领域复杂、规模较大,可预期的需求变化多。

以某产品测试为例,确定实施领域为某产品网元业务测试。由于某产品用例编写复杂,需要通过组合和参数丰富化用例,而且版本升级后用例大量需要重新修改、整合、裁剪,具有实施的必要性。

4.2 领域建模

领域建模主要采用以下方法进行:首先是确定范围,然后使用通用语言进行交流,对领域的业务、功能进行交流,从核心业务开始,逐步进行,通过进一步对信息进行组织和抽象,建立领域模型,需要不断进行重构。

以某产品为例,首先确定了实施领域为业务测试,确定了被测对象为 A 网元、发起测试的对象为 B 网元以及测试的业务范围。通过和领域专家的交流,从核心业务承载控制开始,逐步梳理了 SDP、终端操作等业务,发现了领域模型的关键性概念和元素,形成了通用语言。通过将 A 网元的领域知识进行组织,分而治之,划分为承载测试、终端操作测试、SDP 测试等几个主要的范围。划分的原则是高类聚、低偶合。通过领域建模建立了 A 网元的领域测试模型,如图 4 所示。

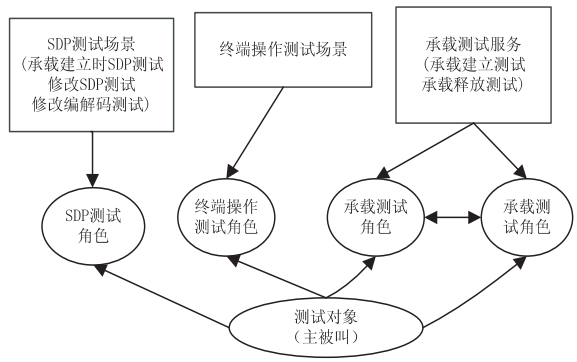

图 4 领域模型

模型中一共有 3 个测试服务,分别是承载测试服务、终端操作测试服务、SDP 测试服务。分别提供了各自的高层领域测试服务,如承载测试服务提供了承载建立和承载释放测试服务。

各测试服务由相应的测试角色交互协作提供测试

服务,图中测试角色用圆表示,例如承载测试服务由承载测试角色提供测试服务,测试对象是终端,图中用椭圆表示。测试对象在不同的测试场景中充当不同的测试角色。

4.3　面向对象设计

采用面向对象方法进行设计,将模型映射到对象。由于 SDP 测试场景类和终端操作测试场景类可以继承承载测试场景类,而 SDP 测试角色类和终端操作测试角色类可以继承承载测试角色类,通过简化设计,UML 类图如图 5 所示[14-15]。

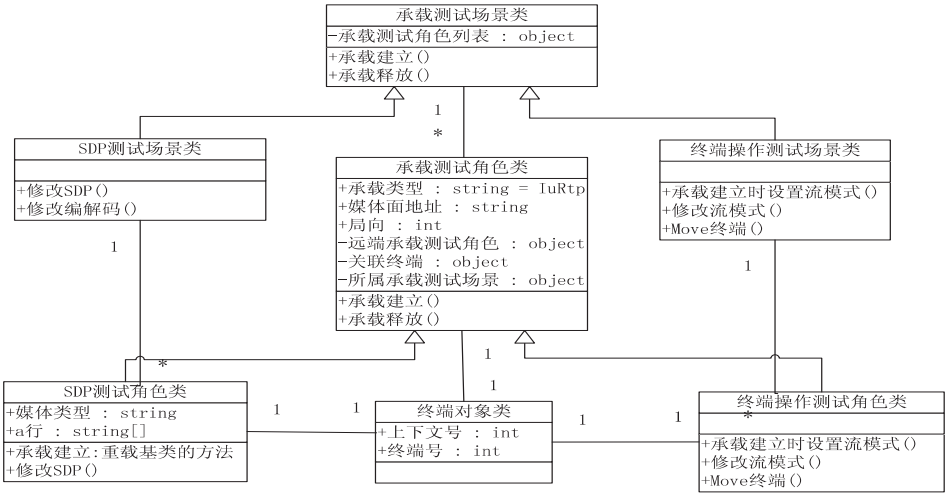

图 5　UML 类图

4.4　开发和测试

4.4.1　测试库平台

领域建模和面向对象设计完成后,用 python 对领域驱动测试库进行了开发,测试库平台结构如图 6 所示。

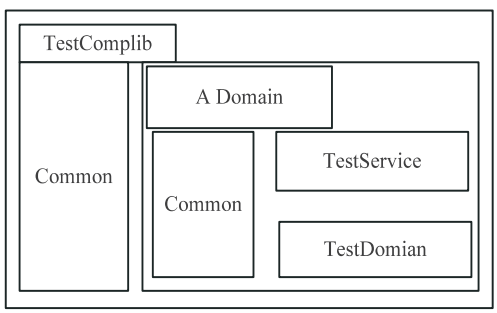

图 6　测试库平台结构

其中 TestCompLib 提供了一个可扩展的领域驱动测试库通用平台。其中的 common 部分则是整个测试库平台的公共基础部分,提供测试用例集、测试用例、测试方法、测试引擎、异常日志等基类接口和公共实现。A 是 TestCompLib 中的一个具体领域的测试库,代表一个被测领域 A。A 中的 TestService 包括了 A 测试领域的各种测试服务类的定义和实现,提供各种领域测试服务,例如实现了承载测试服务(class BearTestService)及其提供的承载建立和承载释放测试服务(def EstablishBear 和 def ReleaseBear)。A 中的 TestDomain 则包括了 A 测试领域的各种测试对象和测试角色类的定义和实现。A 中的 common 部分继承自 TestCompLib 中的 common,是 A 测试库的公共基础部分的具体实现。

4.4.2　TDD

领域驱动测试库的服务层是根据测试场景划分的,每个测试场景又提供高层领域测试服务接口,很自然地适用 TDD。为此,针对测试库的各测试场景设计编写了 TDD 测试驱动用例,并同步设计相应的领域测试服务接口,进行测试、开发的反复迭代,最终完成了领域测试服务的实现并通过了测试,有力保障了领域测试库的质量。

4.5　版本管理和发布

使用领域驱动测试后,测试用例采用代码的方式进行版本管理和发布,原则是测试库随项目大版本发布给测试用例编写和执行人员使用。主要采用如下方法:

(1)上层测试用例采用高层领域语言编写,一般不会变化,但为了便于管理,按项目版本拉分支;

(2)领域测试驱动库属于实现层,是对具体测试的封装,不同版本的具体测试实现可能会有差异,因此按照项目大版本拉分支;

(3)编写测试用例时,使用对应版本分支的测试库;

(4)对项目版本测试执行时,使用对应版本的测试用例和测试库组合。

5　应用实例

5.1　应用情况

在产品 A 测试中,给测试人员提供了一个 A 领域驱动测试库,并进行了实际应用,下面对应用方法和结

果进行介绍。

测试人员使用领域驱动测试库提供的领域测试关键字来描述编写测试用例脚本,一般按以下步骤进行:

(1)分析测试用例涉及的测试场景;

(2)对每个测试场景分析测试行为,可以分别由哪些领域测试服务提供;

(3)从领域测试库相应的场景库中取对应的领域测试关键字来描述;

(4)组合多种领域测试关键字,设置不同参数,完成测试用例脚本。

以下面测试用例为例子说明:主叫 IuRtp 承载,被叫 ARtp 承载,呼叫建立后对主叫播放 bcg/bwt 包音,停音后释放呼叫。

该用例涉及两个测试服务:承载测试服务、放音测试服务。其中承载测试服务包括承载建立、承载释放测试服务,分别对应领域测试关键字 EstablishBear、ReleaseBear;放音测试服务包括放音、停音测试,分别对应领域测试关键字 PlayTone、StopTone。对相关测试角色设置不同参数,组合多种领域测试关键字,完成后的测试用例脚本用高层领域关键字组合而成,如下所示:

```
主叫 IuRtp 承载,被叫 ARtp 承载,播放 bcg/bwt 包音
${termT1} SetTestTerm C1 T1
${termT2} SetTestTerm C1 T2
${BearTermT1} ActBearTestRole ${termT1} IuRtp ${office0} ${IpAddr}
${BearTermT2} ActBearTestRole ${termT2} ARtp ${office0} ${IpAddr}
${PlayToneT1} ActPlayToneRole ${termT1} ${office0} bcg-bwt
EstablishBear ${BearTermT1}
EstablishBear ${BearTermT2}
PlayTone ${PlayToneT1}
StopTone ${PlayToneT1}
ReleaseBear ${BearTermT1}
ReleaseBear ${BearTermT2}
```

5.2 效果评价

使用基于领域驱动的自动化测试框架取得了较好的效果,主要有以下几点:

(1)测试用例使用领域语言描述,更容易理解和编写;

(2)协议或版本升级只需要修改测试库,用例不需要任何修改,维护方便;

(3)测试用例可以通过组合关键字和设置参数轻松扩展。

使用基于领域驱动的自动化测试框架前后代码静态统计结果对比如表 1 所示。

表 1 代码静态统计结果

	文件数	代码行数	平均圈复杂度	函数平均行数	人力投入
使用前	1	31 056	8	817	8
使用后	22	2 551	3	31	4

6 结束语

基于领域驱动的自动化测试框架应用范围很广,在不同的行业领域都可以应用。在新产品研发过程中,在产品需求阶段就可以同步考虑对测试领域的建模,然后设计、开发相应的领域驱动测试库来提供测试服务,构建测试用例脚本,从而采用 ATTD 的方法来驱动产品的开发。这种理念可以用于新功能特性的自动化用例脚本的设计开发。

未来应该向测试领域建模方法和专项领域的测试框架和模型设计两个方向来继续探索研究和积累实际应用经验。

参考文献:

[1] Patton R. 软件测试[M]. 北京:机械工业出版社,2002.

[2] 吴显光. 软件自动化测试[J]. 中国新通信,2012,14(14):67-69.

[3] 龚 丹. 自动化测试之我见[J]. 计算机光盘软件与应用,2012(17):83-84.

[4] 崔红军,饶若楠,邵培南. 一种 API 自动化测试工具的设计与实现[J]. 计算机工程,2007,33(4):270-271.

[5] 夏 晶. 基于 QTP 的功能自动化测试框架的研究与应用[D]. 武汉:武汉科技大学,2010.

[6] 王 君,朱美正,李 欣. 关键字驱动测试框架的研究与实现[J]. 计算机工程与设计,2010,31(10):2246-2248.

[7] 候 勇. 关键字驱动的自动化测试系统的研究[D]. 西安:西安电子科技大学,2009.

[8] Sametinger J. Software enginering with reusable companents[M]. Berlin, Germany: Springer-Verlag, 1997.

[9] Evans E. 领域驱动设计[M]. 北京:人民邮电出版社,2010.

[10] 易利涛,周肆清,丁长松. 信息抽取中领域本体建模方法研究[J]. 计算机技术与发展,2011,21(10):23-27.

[11] Boggs W, Boggs M. Mastering rational XDE[M]. 邱仲潘,译. 北京:电子工业出版社,2003.

[12] 邵维忠,杨芙清. 面向对象的系统设计[M]. 北京:清华大学出版社,2003:160-174.

[13] Szysperski C. Component software: beyond object-oriented programming[M]. [s. l.]: Addison Wesley, 2002.

[14] 冀振燕. UML 系统分析设计与应用案例[M]. 北京:人民邮电出版社,2003:3-9.

[15] Roff J T. UML a beginner's guide[M]. 张 瑜,译. 北京:清华大学出版社,2003:9-13.