文章编号：2095-2163(2023)01-0149-05 中图分类号：TP319 文献标志码：A

金融分布式接口自动化测试工具设计

高冬梅[1]，梅新奎[2]，宿文玲[1]，宋笑兵[1]
（1 黑龙江财经学院 财经信息工程学院，哈尔滨 150080；2 哈尔滨工程大学 智能科学与工程学院，哈尔滨 150080）

摘 要：在金融分布式系统中，系统提供的功能被封装为服务、消息监听和定时任务，以接口的方式提供应用。接口的正确性和可靠性如何通过接口测试得到高效的保障，是亟待解决的问题。为此，本文提出接口自动化测试工具。首先，基于某金融系统设计工具的体系结构；其次，建立接口测试模型，为测试用例产生数据驱动和结果验证；最后，为了提高测试效率，工具实现测试代码生成、模型生成和期望结果生成。通过某金融公司 10 个迭代，完成 20 个服务接口、2 个消息监听接口和 4 个定时任务接口的测试任务共 600 个用例。结果表明，该工具为金融系统提供质量保障，能够对接口返回结果、数据库表、异常和异步消息事件进行精细全面的验证。

关键词：自动化测试；测试工具；接口；测试代码生成；精细验证

Design of automatic test tool for financial distributed interface

GAO Dongmei[1], MEI Xinkui[2], SU Wenling[1], SONG Xiaobing[1]
(1 School of Finance and Information Engineering, Heilongjiang University of Finance and Economics, Harbin 150080, China;
2 School of intelligent science and engineering, Harbin Engineering University, Harbin 150080, China)

【Abstract】 In the financial distributed systems, the functions provided by the system are encapsulated as services, message monitoring and timing tasks, and applications are provided in the form of interfaces. How to ensure the correctness and reliability of the interface efficiently through the interface test is an urgent problem to be solved. To this end, an interface automated testing tool is proposed. First, the architecture of the tool is designed based on a financial system. Secondly, an interface test model is established, based on which data-driven and result verification are generated for test cases. Finally, in order to improve test efficiency, the tool implements test code generation, model generation and expected result generation. Through ten iterations of a financial company, a total of six hundred test tasks of twenty service interfaces, two message monitoring interfaces and four timed task interfaces were completed. The results show that the tool provides quality assurance for the financial system and fine and comprehensive verification of interface return results, database tables, exceptions and asynchronous message events.

【Key words】 automated testing; testing tools; interfaces; test code generation; fine-grained verification

0 引 言

目前，研究人员对接口自动化测试工具展开了深入研究。研究主要集中在接口自动化测试框架设计与实现、算法生成测试用例、模型驱动测试等方面。对于接口自动化测试框架设计与实现方面，文献[1]应用 Jenkins 平台和 TestNG 技术，设计并实现了移动端接口测试框架；文献[2]应用 HttpClient 和 TestNG 技术，设计与实现了 Web 的 http 接口测试框架；文献[3]应用 Web 开发技术，设计与实现了轻量级接口测试框架；文献[4]应用 Python 语言分析线上日志，对流量进行回放，设计与实现了服务端接口测试框架。在算法生成测试用例方面，文献[5]应用蚁群算法生成测试用例；文献[6]应用正交实验设计算法，生成了大批测试用例。在模型驱动测试方面，文献[7]建立接口语义模型，对接口语义进行逻辑推理、建立规则，生成测试用例；文献[8]建立 UML 时序图的元模型，生成测试代码和用例；文献[9]通过系统变更前后的 UML 用例图、类图和活动图变化，生成回归用例；文献[10]通过状态图生成中间模型图，选择相应覆盖率标准，覆盖图中路径，

基金项目：黑龙江财经学院重点项目(XJZD202213)；黑龙江省教育科学“十四五”规划 2021 年度重点课题(GJB1421557)。

作者简介：高冬梅(1982-)，女，硕士，讲师，主要研究方向：企业与服务智能计算；梅新奎(1989-)，男，博士研究生，主要研究方向：控制科学与工程；宿文玲(1983-)，女，硕士，副教授，主要研究方向：电子信息工程；宋笑兵(1976-)，男，学士，讲师，主要研究方向：大数据。

通讯作者：高冬梅 Email：896149540@qq.com

收稿日期：2022-04-04

生成测试用例并分析覆盖率。

然而,目前已有的自动化测试框架或工具仍然存在以下问题:

(1)只能对被测系统发布的服务或 http 接口进行测试,对不属于服务的接口无法测试。例如,定时任务和异步消息事件。

(2)接口输出验证不完整,只能对接口返回结果的部分参数做简单的断言验证,被测接口的数据库表变化、发送的异步消息事件和异常无法得到验证。

(3)生成的测试用例,依赖算法或模型,表达能力有限。

为此,针对质量要求严格的金融分布式系统,本文提出接口自动化测试工具,对服务接口和不属于服务接口实现测试,接口返回结果实现精准验证。测试工具基于 IntelliJ IDEA 插件开发和实现,插件支持生成测试代码、生成测试结果、生成数据库模型、生成类模型和编写用例,并提供图形化界面。

1 体系结构设计

测试工具体系结构的实现,必须从被测金融分布式系统开发的体系结构入手。如图 1 所示,该系统提供开户、借款、还款等金融类 SOA 服务;接收上游系统传递的异步消息;可进行结息、出账和到期扣款等系统定时任务处理。这些接口作为金融类系统重要的功能和服务,都应该从接口测试层面去保障。在原有被测接口系统体系结构中引入测试工具,测试工具和被测接口隶属于同一个系统,系统以多模块的方式组织在一起。这种组织方式有利于迭代过程中开发和测试使用同样的配置管理方式,将其部署到实际生产环境中,可以对测试模块进行打包排除。其次,这种组织方式有利于对不属于服务的异步消息、定时任务等进行接口测试,对质量要求严格的金融类系统提供全面的接口测试保障。

测试工具包括生成测试代码、生成类模型、生成数据库模型、生成测试结果、数据驱动和流程模板。生成测试代码,根据接口生成测试代码基本结构。生成类模型,根据接口输入参数、返回结果生成类模型文件。生成数据库模型,根据数据库配置生成数据库表模型。生成测试结果,根据开发接口生成的结果进行收集和验证,接口生成的结果包括接口返回结果、数据库变更、发送的消息和可能的异常,测试框架对这些结果进行收集后和期望的结果进行对比验证。数据驱动的设计是为准备调用开发模块所需要的数据,并完成开发接口的调用。流程模板的设计是为每个接口测试提供流程的模板,所有接口测试的流程都包括初始化、获取数据驱动、执行测试和生成结果与验证。

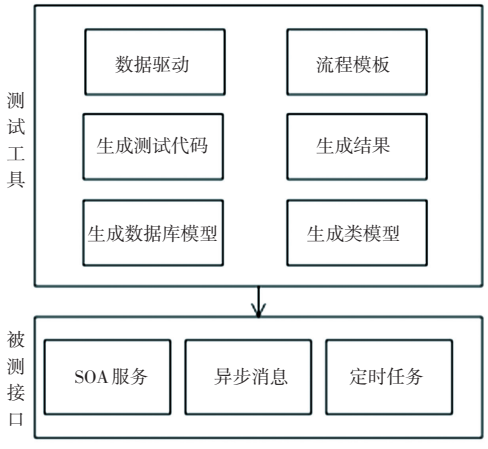

图 1 系统体系结构

Fig. 1 System architecture

2 关键技术实现

2.1 接口测试模型

文献[6]从数据模型、服务模型和计划模型 3 方面建立接口测试模型;文献[7]从服务输入、服务输出、服务运行参数和服务间依赖关系 4 方面建立服务接口契约模型。文献[8-10]从模型驱动测试角度,建立 UML 时序图的元模型、类图、用例图、活动图和状态图,生成测试代码。文献[11]从测试接口、测试用例、测试套件、输入输出和前置条件定义测试模型及关系。

在以上文献研究的基础上,金融分布式系统接口测试模型,从基本对象模型和数据准备模型两个方面抽象。其中,基本对象模型,用于描述接口测试中包含的对象;数据准备模型,用于描述数据准备时需要的要素。

(1)基本对象模型定义包括:表、异步消息事件和对象。

VirtualTable = < tableName, tableBaseDesc, flags tableData >

VirtualEventObject = < eventObject, eventCode, topicId>

VirtualObject = < description, objClass, object, objBaseName, objBaseDesc, flags>

其中,tableName 为表名;tableBaseDesc 为表描述;flags 为验证标识符;tableData 为表中数据;eventObject 为事件对象;eventCode 为事件编码;

topicId 为事件 ID；description 为对象的描述信息；objClass 为对象对应类；object 为对象信息；objBaseName 为基类名；objBaseDesc 为基类描述。

（2）数据准备模型定义包括：数据库表、上下文参数、输入参数、输出结果、异步消息事件集合、异常和测试单元。

VirtualDataSet = <{ virtualTable$_i$ }>：数据库表集合；

VirtualParams = <{ param$_i$ }>：上下文参数集合；

VirtualArgs = <{ inputArg$_i$ }>：输入参数集合；

VirtualResult = <result>：返回结果对象；

VirtualEventSet = <{ virtualEventObject$_i$ }>：异步事件对象集合；

VirtualException = <expectException>：期望异常对象；

TestUnit = <description>：数据库表等 6 种类描述信息的抽象。

数据驱动是以文件的形式提供接口测试所需要的数据，常见的数据交换格式有 XML[12]、JSON、YAML[13]和 CSV。本文选择 YAML 格式文件，避开了 XML 文件中的各种引号和括号等，上手较快并且文件可读性较高。YAML 文件由模型中定义的输入参数、准备的数据集、期望的数据集、期望的消息事件集、期望的输出结果和期望的异常组成。其中，输入参数负责组装接口所需要的所有参数；准备数据集负责组装接口所需要的所有数据库数据；期望数据集，负责被测接口数据库表的变化期望值；期望消息事件集，负责被测接口异步消息事件的期望值；期望结果，负责被测接口返回结果的期望值；期望异常，负责被测接口产生异常的期望值。在 YAML 文件中，每个测试用例都准备好这 6 类元组后，将每个元组的内容转换为数据准备模型中的对象，调用被测接口，实现文件方式的数据驱动。

结果验证是以期望数据和接口返回数据进行对比，如果对比结果全部相等则用例成功，否则返回用例失败。结果验证的内容包括模型中数据库表的数据集、消息事件集、异常和接口返回结果。验证内容的每个属性都要进行对比，通过定义标记 flag 实现是否需要验证某个属性内容。flag 由 6 元组组成，其中，Y 表示需要验证该属性；N 表示不需要验证该属性；C 表示该属性作为查询条件时，返回大于 0 条记录；CN 表示该属性作为查询条件时，返回 0 条记录；R 表示用正则表达式匹配该属性；DS 表示对时间字段的验证，后面可以加毫秒数如 DS500，表示和当前时间差在 500 毫秒内，满足该范围则验证通过。

2.2　流程模板

某金融系统要实现其接口测试流程，先要分析出所有接口测试通用的测试流程模板基类，所有测试类继承该流程模板基类，流程模板提供通用的测试流程框架。其次，在通用的测试流程中抽取流程模板，在该模板中实现部分通用方法，所有测试类可以根据实际情况重写流程模板中方法。根据以上分析，流程模板中接口测试流程定义如下：

（1）初始化。在测试类执行前，根据配置文件读取数据库配置，扫描并获得定义的测试注解（如：测试前后清理注解、验证前后注解和验证注解等），获取被测试接口的接口名称、方法和参数。

（2）获取数据驱动。以文件的形式提供接口测试所需要的数据准备。其中包括输入参数、数据库准备的数据、期望数据和接口需要的上下文数据等。

（3）执行测试。执行测试包含测试前准备、准备测试数据、测试执行、检查结果数据、清理测试数据和测试后执行 6 个步骤。

①测试前准备是抽象方法，每个用例可根据实际需要选择是否重写该方法。

②测试数据是对该用例执行提供数据准备，每个用例都需提前为接口运行提供必要的数据（请求参数数据和数据库数据等）。例如：测试借款接口，要准备借款接口参数数据和开户的数据库数据。

③测试执行是根据反射机制调用被测接口、传递请求参数数据和捕获被测接口返回结果。

④检查结果数据是对接口所有输出和期望输出进行对比，若所有对比都成功则用例执行成功，否则用例执行失败。接口输出包括接口的返回数据、数据库变更数据、发送的异步消息数据和返回的异常，所有这些输出都要进行对比，保障每个用例进行全面的验证。

⑤清理测试数据是对测试准备的数据库数据和接口运行后产生的数据库数据进行清理，避免用例多次重复运行而产生数据之间干扰而导致用例失败。

⑥测试后执行是抽象方法，每个用例可以根据实际情况决定是否需要执行测试后的清理工作。

（4）结果收集验证。在程序运行结束后，自动收集运行结果进行对比验证（参见 2.3 节）。

2.3　测试生成

测试生成以减少手工重复劳动为目的，在开发工具 IntelliJ IDEA 中开发插件，实现生成测试代码、生成类模型、生成数据库模型和生成期望结果。这

4 个功能可直接在开发工具右键菜单中直接使用。

生成测试代码及文件。在 Velocity 模板文件中设置测试代码模板,抽取每个接口测试不同的参数,作为替换的变量。按照插件开发规范继承相关的 ACTION,当选中某个接口时获取接口名称、包名等信息,替换模板中的变量,生成测试代码的类文件、相关包和测试驱动文件。

生成某个类的 CSV 文件格式。当选中某个类时,获取类的名称、属性等相关内容,在指定包中生成类的 CSV 文件。在 CSV 文件中可以设置该类每个属性多列值,作为多个用例的输入参数、输出参数和验证结果值。

生成数据库表的 CSV 格式。在配置文件中配置数据源相关信息,插件获取配置信息,在界面中用户选择要生成的数据库和相关表,在指定包中生成表信息的 CSV 文件。在 CSV 文件中可以设置表中的多条记录,作为多个用例的数据库准备数据和数据库期望数据。

生成接口测试产生结果自动收集,拦截运行接口时返回的结果、产生的 SQL、异常和发送的异步消息事件,将所有结果存储在临时文件中。当用例运行后执行一次,将临时文件中的数据存储到用例 YMAL 文件的期望结果中,进行人工检查期望数据的正确性。该方式避免了所有期望数据都进行手工填写,提高了测试效率。后续再运行用例时,自动对比结果正确性。

3 工具应用与分析

在某金融公司借款系统研发迭代中使用测试工具,在 10 个迭代中完成 20 个 SOA 服务接口、2 个消息监听接口和 4 个定时任务接口的测试任务共 600 个用例,从测试效率、检错性、回归保障 3 方面分析实验结果。

测试效率:10 个迭代完成 600 个用例,平均每个迭代 60 个用例。由于分布式系统链路较长,若这些用例通过功能测试手工去完成,平均每天执行 20 个用例则需要 3 人。此外,用例执行受环境稳定性影响较大,需要从链路的发起端完成业务,若被测系统处于中下游,设计的异常测试用例很难模拟或无法测试。采用接口测试工具完成,自动生成测试代码、生成输入对象和生成结果,使测试人员关注在用例的设计、数据的准备和结果的核对上,完成 60 个用例只需 4 小时。

检错性:接口自动化测试工具 AutoTest[6]、SOAPUI[14] 和 JMeter[15] 等,只能对接口返回结果的部分属性验证。与其相比,本文提出的接口测试工具可以对 SOA 服务和不属于服务的异步监听消息和定时任务等进行测试,检查的输出结果全面,可以对接口的所有输出验证,包括接口返回结果、数据库表数据、发送的异步消息事件和异常,对象的每个属性精细化验证。发现的问题更全面,对质量要求严格的金融类系统提供了可靠质量保障。在 10 个迭代中发现的缺陷类型主要有接口输入验证错误 24 个、接口内部逻辑错误 15 个、接口返回结果错误 21 个、数据库表字段内容错误 30 个和消息内容错误 10 个。这些缺陷按严重程度分类,其中严重缺陷 24 个、一般缺陷 35 个、轻微缺陷 40 个。

回归保障:随着迭代的进行,自动化接口测试用例增加,可回归用例数量也同时增加,600 个用例的运行时间只需 1 分钟并且验证全面,在代码重构或回归业务时提供有利质量保障。

4 结束语

本文设计的自动化测试工具,很好的解决了接口测试数据验证不全面及不精细问题,实现了接口输入参数和准备的数据库数据转换为数据驱动文件、接口测试代码生成、编写测试用例可视化和接口输出结果自动收集等功能。工具在某金融公司得到了很好的应用和推广,为多个金融业务系统提供可靠质量保障。

参考文献

[1] 赵红芳. 基于 TestNG 的接口测试框架的设计与实现[D]. 成都:西南交通大学,2017.
[2] 蒋灵仙. 基于 Testng 的 Web 接口测试的自动化框架设计与实现[D]. 杭州:浙江工业大学,2016.
[3] 孙立哲. 轻量级接口自动化测试框架设计与实践[J]. 计算机应用与软件,2020,37(1):27.
[4] 王少康. 基于用户日志分析的访问流量回放自动化测试框架的设计与实现[D]. 北京:北京邮电大学,2020.
[5] 劳天. 基于蚁群算法的软件接口测试用例生成[J]. 计算机工程与设计,2018,39(1):79.
[6] 卓欣欣. 服务接口测试自动化工具的研究[J]. 计算机研究与发展,2018,55(2):358.
[7] 王博. 基于接口语义自动机的嵌入式软件构件与时序测试研究[D]. 北京:清华大学,2016.
[8] Meryem Elallaoui, Khalid Nafil. Automated Model Driven Testing Using AndroMDA and UML2 Testing Profile in Scrum Process [J]. Procedia Computer Science,2016,83:221.
[9] Pardeep Kumar Arora, Rajesh Bhatia. Agent-Based Regression Test Case Generation using Class Diagram, Use cases and Activity Diagram[J]. Procedia Computer Science,2018,125:747.

(下转第 157 页)