第33卷 第14期 计 算 机 工 程 2007年7月
Vol.33 No.14 Computer Engineering July 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)14—0275—02 文献标识码：A 中图分类号：TP306.2

基于 WinRunner 的金融系统自动化测试方案

郭 琼，李秀斌
(中国科学院研究生院，北京 100039)

摘 要：结合金融系统的特点，该文提出了一种基于 WinRunner、测试库、数据驱动、数据库备份的自动化测试方案，介绍了金融系统的开发环境，分析了测试工具的选用、自动化测试库的构建、数据驱动的自动化测试、数据库备份等，给出了外围设备仿真(键盘、屏幕、刷卡机、密码键盘、打印机)的实现方案。

关键词：自动化测试；WinRunner；传统终端；仿真终端

Automated Test Method of Finance System Based on WinRunner

GUO Qiong, LI Xiubin
(Graduate School of Chinese Academy of Sciences, Beijing 100039)

【Abstract】 In consideration of characteristics of financial system, this paper proposes an automated test proposal based on WinRunner, test database, data drive and backup of database. It introduces developing environment of financial system, selection of test tools, construction of automated test database, automated test of data drive and backup of database etc. It gives emulation mode of the peripheral equipments. These peripheral equipments include keyboard, screen, card reader, code keyboard and printer.

【Key words】 automated test; WinRunner; traditional terminal; emulational terminal

“软件测试自动化”已成为软件工程领域中的一个重要课题。软件测试工作走向成熟化、标准化的必经之路是“自动化测试的实施”。不是任何项目、任何软件系统都适宜自动化测试的，金融系统作为一个大型行业软件系统，具有自身的特点。在分析金融系统的优势、劣势之后，针对金融系统自身的特点，本文给出了一种解决方案。

1 金融系统环境的介绍

以民生银行项目组为例，国内大多金融系统(主要指银行系统)的开发环境，分为前台和后台开发，即客户端和服务器端，前台(客户端)使用传统终端机，一般为 SCO 系统，常用的开发平台有长天集团 EUCP、中联 VHOST；后台(服务器端)一般为 IBM AIX400 系统、Informix 数据库系统，使用 C 或 Java 语言开发。后台程序以交易为单位，每个交易为一个事务。

测试人员通常使用 NetTerm 仿真终端连接前台，在本机上进行测试，也可直接在终端机上进行测试。以交易为单位，一个交易为一个测试单元，测试人员针对每个交易组织测试案例，编写测试文档。交易又分为

(1)联机交易：“柜面柜员”在白天直接为客户办理业务时的交易，如通常的取款和存款交易等；

(2)批处理交易：“批处理柜员”在夜间“跑批”时的交易，如利息计算和自动扣息等交易；

(3)中台交易：涉及自动提款机等“非柜面柜员”的交易；

(4)Java 新平台交易：“个贷系统”独立于整个核心系统、使用 Java 语言编写的交易。

本文提出的方案只讨论交易(1)的情况。

2 自动化测试部署

2.1 测试工具的选用

WinRunner是MI(mercury interctive)的产品，作为一种企业级的功能测试工具[1]，WinRunner具有如下特征：

(1)支持录入功能，可通过录入功能直接产生测试脚本，使用方便简单、容易入手。

(2)支持数据检测，可验证数据库中数值，确保交易的正确性。

(3)支持数据驱动。

(4)支持多种平台：Windows95，Windows98，WindowsNT，Windows2000。

(5)支持多种测试环境。

(6)支持结果分析，测试运行结束后，WinRunner 的互动式报告工具会列出测试中发现的错误的出错的位置，提供详细易读的报告。

(7)支持与测试管理工具 TestDirector 的交互，可通过 TestDirector 检查测试运行结果，方便测试案例的管理。

2.2 自动化测试库的构建

为了提高测试的可扩展性、脚本的可读性、易编写性，本文提出基于测试库的自动化测试[2,3]，将测试脚本中常用的代码写成函数，单独成为 1 个模块。本方案将测试库分为 5 个模块：

(1)民生测试系统初始化模块(InitMinShengTestSys)；

(2)民生测试系统启动模块(StartMinShengSys)；

(3)民生测试功能库(MinShengTest_FUNCLIB)；

(4)民生测试业务库(MinShengTest_TRANLIB)；

(5)民生测试系统测试套控制模块(MinShengTranCtl)。

2.3 数据驱动的自动化测试

在金融系统的测试组中，会有一定比例的缺乏测试经验

作者简介：郭 琼(1977－)，女，硕士研究生，主研方向：智能处理，多智能体，网络优化设计；李秀斌，硕士研究生

收稿日期：2006-08-25 **E-mail：**zjkgq@sina.com

的业务专家参与测试，并在测试中起着很重要的作用，本文提出了数据驱动的自动化测试[5,6]。传统的方法将测试数据写到简单表格中，通过测试库和上层的少量开发语言写一个解析器，解释表格中的数据。考虑到金融系统测试环境稳定、测试目的明确，将测试数据(环境配置数据、输入数据、库表检测数据)分别按格式写到文件中，建立一个以交易为单位的文件系统，只在测试库中，对文件格式做出辨别，分离出想要的测试数据。这样的架构方式使数据与测试脚本分离，一个脚本可以对应多个输入文件和库表检测文件，增强了脚本的复用性，也使脚本向结构化、格式化方向发展，易于理解、开发。

2.4 数据库备份和案例回放

在测试过程中存在一个问题，数据的有效性，即期望值的有效性，数据用过就作废，期望值会随着案例的回放而改变。例如在取款交易中，假定用户现在的余额是 200 元，本文要做一个取 100 元的取款交易案例回放，有效的期望值应该是 100 元，这样做案例回放是没有问题的，如果没有其他因素的影响，案例将顺利通过，但第 2 次做这个案例回放时，原来 100 元的期望值已过期无效，这时可以修改库表检测文件，使案例仍能顺利进行，对于大量的案例来说，修改库表文件也是一个复杂而繁琐的工作。本文给出了基于数据库备份的测试方案。在脚本入库前，需要做一个数据库的备份。如图 1 所示，数据库呈阶梯状增长，在每个状态之间，可以做新项目的手工测试。当需要做该案例回放时，可以直接恢复到相应的数据库状态，进行案例回放。测试脚本入库应以项目为单位按批入库。

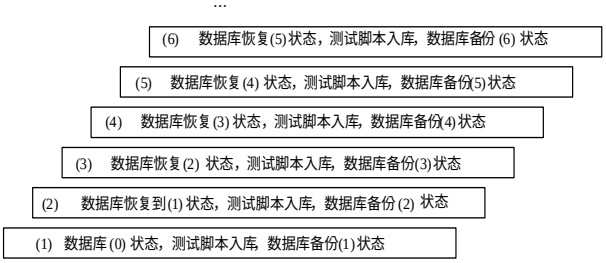

图 1 数据库状态转换

3 自动化测试仿真的实现

现阶段实施的自动化测试与手工测试相比较，就是采用程序模拟手工测试的过程。在自动化测试过程中，原来由手工控制的外设，现在由程序来控制，不再进行手工干预，自动化测试仿真(即模拟手工测试)的实现，成为了自动化测试实现的关键。

自动化测试不在真正终端机上进行，使用 NetTerm 仿真终端，在本机(PC 机)上进行。测试过程中，终端(NetTerm 仿真终端)需要仿真的外设包括键盘、屏幕、刷卡机、密码键盘、打印机(并口)和串口。

3.1 键盘、屏幕的仿真

使用 NetTerm 仿真终端，其仿真键盘等同真正终端机上的键盘。屏幕可通过 NetTerm 进行远程连接来仿真，显示效果同真正终端。但在自动化测试过程中，原来由眼睛来判断正误，现在必须由程序来判断，必须在屏幕上抓取相应的标志给程序作出判断，而自动化测试工具 WinRunner 是基于 Windows 的自动化测试工具，它提供基于对象的辨识方式、基于像素的辨识方式。后者对程序的精度要求较高，屏幕位置稍有变化就会导致脚本的不可复用。本文选用基于对象的辨别方式，而在该方式中，WinRuner 只能辨别到“对象 1 级”，即 NetTerm 终端，而由它连接显示的屏幕内容在 WinRunner 看来，只是字符流，必须编写一系列的功能函数，实现对 NetTerm 屏幕内容的辨识。为了使辨识效果达到最佳状态，必须对 NetTerm 进行预设置。

3.2 刷卡机的仿真

现有终端上对测试影响最大的外设为读卡设备。该设备由于需要串口供电，终端的串口 1 脚带有 5V 电压，而普通 PC 上的串口不带电，因此不能直接应用于 PC 机。但可单独对其 1 脚供电，其余引脚连于 PC 机进行通信。该方法可以实现终端外设读卡设备与 PC 机的通信，但仍需要人工刷卡。为取消人工干预，需通过修改程序，对读卡设备进行仿真。

在给出刷卡的仿真方案之前，先介绍一下刷卡过程。首先在当前光标跳到需刷卡的 ddu 时，执行该 ddu 的域前动作，在域前动作中，先对该 ddu 清空，然后调用 pub_readtrack 读卡公共构件，获得卡的二磁道及三磁道信息，并将其赋给该 ddu。在 pub_readtrack 公共构件中，等待用户的刷卡动作，将刷卡获得的二磁道信息赋给一个公共的 ddu:trk2，若等待刷卡的动作超时，则会提示是否重刷，若选择是则继续等待刷卡动作，选择否则可以将卡号手输入 ddu 中，但在不允许手输的交易中，会在该 ddu 的域后动作中，通过判断公共 ddu:trk2 是否为空，来得知是否是刷卡获得的卡号，若此时为空，则会报错，然后又将光标跳到该 ddu，继续执行该 ddu 的域前动作。

若想取消刷卡动作，有 2 个方案：(1)修改需要刷卡 ddu 的域后动作，在等待刷卡超时后允许手工输入卡号，这样刷卡动作可以等同为其他 ddu 的操作；(2)修改读卡的公共构件，直接将卡号付给 ddu:trk2 字段，起到一个瞒天过海的作用。方案(1)修改量大且会损毁源代码，被否定。在方案(2)中，通过修改公共刷卡构件来实现刷卡仿真，可在读卡程序前加一条判断语句，当卡号字段输入为特定数据时，程序从指定位置读取卡的信息，而不使用刷卡器刷卡，这样既简便又不影响手工测试的源流程，几乎没有恶性影响(如：卡号字段输入“58381301001”，前 4 位“5838”表示不刷卡从文件读取卡号信息，中间 4 位“1301”表示正在测试的程序，最后 1 位“001”表示该卡信息的顺序号，后几位组合起来标志卡信息存储的文件名)。本文最初忽略了在某些交易的域前动作中，会将该 ddu 清空的情况，导致期待的刷卡超时输入到 ddu 中的值(比如 58381301001)在第 2 次调用刷卡构件时无法获得。这样可通过 2 次读文件的方法来直接获得卡信息，即第 1 次先读 1 个固定文件(可命名为 readcard.txt)，将原来要超时输入的内容保存在此文件中，根据文件内容找到卡信息的存储文件，然后第 2 次读文件，获得卡信息。注意第 1 个文件用完后即被删除。这样一个过程使其达到了刷卡真正意义上的仿真，又不影响原来流程，因为第 1 个文件用后即会被删除掉。但这一方案会涉及 TSL 脚本与前台 EUCP 的通话、本机和前台机的通信，在 1 次自动化测试中可能会进行多个交易的测试，这时，可能有 2 个以上的交易需要刷卡，且卡信息不同，可以在 TSL 脚本中写文件，在 EUCP 中即读卡构件中读该文件，这一过程有 4 个实现方案：(1)将文件保存在前台机，TSL 脚本远程写文件；(2)将文件保存在本机，读卡构件远程打开文件读内容；(3)TSL 脚本先在本机写文件，写好后传到前台机；(4)TSL 脚本在本机写好文件保存在本机，由读卡构件从本机取到前台机后再读。 (下转第 279 页)