

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 11/36 (2006.01)



[12] 发明专利说明书

专利号 ZL 200410086041.3

[45] 授权公告日 2007 年 7 月 11 日

[11] 授权公告号 CN 1326044C

[22] 申请日 2004.10.22

[21] 申请号 200410086041.3

[73] 专利权人 中国工商银行股份有限公司

地址 100031 北京市西城区复兴门内大街
55 号

[72] 发明人 李旭风 刘家桦 丘嘉宜 李从虎
张宇明 江炜斌

[56] 参考文献

CN1484790A 2004.3.24

US6668340B1 2003.12.23

US5634098A 1997.5.27

CN1227643A 1999.9.1

US5594892A 1997.1.14

US5513315A 1996.4.30

审查员 毛习文

[74] 专利代理机构 北京三友知识产权代理有限公司

代理人 任默闻

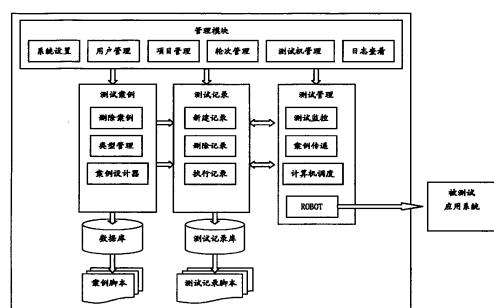
权利要求书 7 页 说明书 30 页 附图 16 页

[54] 发明名称

一种基于脚本解释工具的自动化软件测试系统

[57] 摘要

本发明为一种基于脚本解释工具的自动化软件测试系统，包括测试案例库，测试记录库，测试案例处理单元，测试记录处理单元，测试案例管理单元，测试总控单元；测试总控单元通过人机界面接受用户的操作指令信息，并控制测试案例处理单元从测试案例库中读取测试案例的初始数据，经用户的编辑操作之后，保存测试案例脚本，并将其路径保存在测试案例库中；测试案例经过测试记录处理单元生成测试记录，保存在测试记录库中，并保存测试记录脚本；运行测试记录，测试记录脚本通过测试案例管理单元传送到指定测试机中的 ROBOT 运行；测试数据通过运行 ROBOT 脚本传送到各应用程序进行处理，将运行结果经测试案例管理单元保存到测试记录库。



1. 一种基于脚本解释工具的自动化软件测试系统，其中包括，

测试案例库：用于存储测试案例信息、测试案例脚本；

测试记录库：用于存储测试记录信息、测试记录脚本；

测试案例处理单元：用于对测试案例的生成、删除、重用进行处理；

测试记录处理单元：用于对测试记录的生成、删除、复制、运行状态统计进行处理；

测试案例管理单元：由脚本解释工具构成，用于管理测试记录运行期间的所有操作；

测试总控单元：用于对系统资源和硬件资源进行控制；

所述的测试总控单元通过人机界面接受用户的操作指令信息，并控制所述的测试案例处理单元从所述的测试案例库中读取测试案例的初始数据，经用户的编辑操作之后，以文本形式保存测试案例脚本，并将其路径保存在测试案例库中；所述的测试案例经过所述的测试记录处理单元生成测试记录，保存在所述的测试记录库中，同时以文本的形式保存测试记录脚本；运行所述的测试记录，测试记录脚本通过所述的测试案例管理单元将测试数据传递到各应用程序进行处理，得到运行结果，将运行结果经所述的测试案例管理单元保存到所述的测试记录库。

2. 根据权利要求1所述的系统，其特征在于，在所述的测试案例库中，测试案例信息与测试案例脚本一一对应，并且所述的测试案例信息存储于数据库中，所述的测试案例脚本以文本的形式存放；通过对数据库中测试案例信息的管理实现对所述的测试案例脚本的管理。

3. 根据权利要求1所述的系统，其特征在于，所述的测试案例处理单元包括：删除案例模块、案例类型模块、以及案例设计器。

4. 根据权利要求3所述的系统，其特征在于，所述的案例设计器是一个可视化案例设计模块，其通过图形界面，添加交易代码，修改交易的输入项

和对应的动作，生成测试案例脚本。

5. 根据权利要求 3 所述的系统，其特征在于，所述的案例设计器能够记忆用户曾经测试过的各个功能模块的输入输出数据。

6. 根据权利要求 3 所述的系统，其特征在于，所述的案例设计器使用的文件包括：测试案例库中的测试案例脚本；使用的表包括：

案例表：纪录已经生成的测试案例的信息；

交易输入项表：记载测试原始数据流，保存一次测试的所有输入项的名称和提示信息；

案例类型表：管理所有案例类型信息；其中，
生成案例类型的树型结构：

读取所述的案例类型表；由于系统默认第一层节点的上一层节点代码全部为 0，故扫描所有上一层类型代码为 0 的类型代码为第一层的节点，扫描所有上一层类型代码为第一层代码的类型代码为第二层的节点，以此类推。

7. 根据权利要求 4 所述的系统，其特征在于，所述的案例设计器包括自动生成界面测试案例单元，该自动生成界面测试案例单元包括：

交易代码管理模块：用于进行交易代码搜索、交易代码新增、交易代码修改、交易代码删除；

输入项管理模块：用于进行输入项新增、输入项修改、输入项删除；

检查项管理模块：用于检查项新增、检查项修改、检查项删除；

自动生成测试数据模块：用于交易输入场检查、测试案例数据生成；

测试规范导入导出模块：用于测试规范导入、测试规范导出。

8. 根据权利要求 7 所述的系统，其特征在于，所述的交易是指：各种被测应用软件。

9. 根据权利要求 1 所述的系统，其特征在于，所述的测试记录处理单元用于引用已经生成的测试案例，对测试案例中的变量进行赋值，生成可以运行的测试记录。

10. 根据权利要求 9 所述的系统，其特征在于，所述的测试记录处理单元使用的文件包括：测试案例库中的案例脚本和测试记录库中的测试记录脚本；使用的表包括，

案例表：纪录已经生成的测试案例的信息；

测试记录表：纪录已经生成的测试记录的信息；

项目名称对照表：纪录测试库中的项目和对应的项目编号；

轮次名称对照表：纪录测试库中的轮次和对应的轮次编号；其中：

新建测试记录的测试案例树由四层组成；前三层是案例类型的三个层次；第四层是用户名；以三个层次得到案例类型代码，搜索案例表，得到所有不重复测试案例的提交者名称；然后用提交者名称搜索用户管理表的 realname 字段，得到提交者的真实名称，显示为案例树的第四层；在案例脚本中增加赋值语句生成测试记录脚本。

11. 根据权利要求 1 所述的系统，其特征在于，所述的测试记录处理单元还包括测试记录重用模块，用于从已经生成的测试记录中生成新的测试记录，并保留原测试记录的所有变量值，直接运行测试或者修改某几个变量值后进行不同功能点的测试。

12. 根据权利要求 11 所述的系统，其特征在于，所述的测试记录重用模块包括，单条复制：用于把一个测试记录复制到指定的某个目录下，复制中不改动任何变量值；

批量复制：用于把同一用户目录下同一案例编号的多条测试记录复制到指定的目录下，复制中可以指定要修改哪些变量值，并统一修改。

13. 根据权利要求 12 所述的系统，其特征在于，所述的批量复制所使用的文件包括：测试案例库中的测试记录脚本；

使用的表包括，

测试记录表：纪录已经生成的测试记录的信息；

项目名称对照表：纪录测试库中的项目和对应的项目编号；

轮次名称对照表：纪录测试库中的轮次和对应的轮次编号；

用户管理表；

其中，

批量复制主要由两个循环组成，一个大循环是逐个复制选中的测试记录，第二个是复制每个测试记录的时候，循环读原记录脚本，生成新的记录脚本；

批量复制中的批量修改可变值是在复制前，列出所有可变值，让用户指定要修改的变量名称；复制过程中，碰到这些变量的赋值时，就用用户指定的值来复制；

赋值区后的复制操作：把赋值区后的内容从源文件复制到目标文件，然后把目标文件上传到案例库服务器；

数据库操作：复制完一个测试记录后，就把该测试记录的项目编号，轮次，案例编号，记录数等组成一条记录插到测试记录表中。

14. 根据权利要求 1 所述的系统，其特征在于，所述的测试案例管理单元包括：测试监控模块、测试案例传递模块、计算机调度模块。

15. 根据权利要求 1 所述的系统，其特征在于，所述的测试总控单元包括：系统设置模块、用户管理模块、修改密码模块、测试机管理模块、日志查看模块、轮次管理模块、项目管理模块。

16. 根据权利要求 1 至 15 任意一项所述的系统，其特征在于还包括：

至少一台客户端计算机：执行所述的测试总控单元的功能；

至少一台测试机：载有测试终端，脚本解释工具，由测试案例处理单元、测试记录处理单元、测试案例管理单元构成的自动化测试服务端、案例数据库；

案例库服务器：存储有测试案例库、测试记录库；

其中，

所述的客户端计算机、测试机、案例库服务器相互耦合，且所述的测试机与运行被测应用软件的应用服务器连接。

17. 根据权利要求 16 所述的系统，其特征在于，所述的客户端计算机、

测试机、案例库服务器之间采用 TCP 协议连接；所述的测试机与应用服务器之间采用相应协议的连接，所述的相应协议包括：TCP 协议、HTTP 协议或 TELNET 协议。

18. 根据权利要求 16 所述的系统，其特征在于，在所述的测试机起一个服务充当 Server；所述的客户端计算机充当 Client，并通过 TCP 连接到所述的测试机；在所述的客户端计算机端起一个定时器，每隔一段时间向测试机请求一次截屏，把截屏信息在客户端计算机显示出来，实现对所述测试机屏幕的监视。

19. 根据权利要求 18 所述的系统，其特征在于，在所述的 Client 和 Server 端自定义一些用于通讯的 TCP 通讯命令，该 TCP 通讯命令如下：

以“PLAY”字符串开头的执行指令；

“stop1”开头的停止单条测试记录执行的指令；

“stopAll”停止一次提交的所有测试记录的执行的指令；

“disconnect”为断开连接指令。

20. 根据权利要求 18 所述的系统，其特征在于，测试机操作系统启动时，自动启动自动化测试服务端；自动化测试服务端会自动把计算机运行管理表中的 status 字段置为 0，表明这台测试机可用；

当测试记录执行时选定这台测试机，会把它的 status 字段置为 1，保证同一时间只能有一个用户在使用这台测试机；当测试机已经全部被占用时，自动化测试排队管理服务器不停地发 test 指令给测试服务端，检测测试机的状态，如果发现测试机状态异常，且计算机运行管理表中的 status=0，就调用 shutdown 程序远程重启测试机，同时置计算机运行管理表中当前测试机的状态变为 2；如果发现有空闲的测试机，取得其 IP 后就开始检测 queuecontrol 表，一旦发现有 flag=0 的 ID 号，就按 ID 号升序排列，取出第一条记录的 ID 号查询 execute 表，把所有符合条件的测试记录通过 TCP 发“PLAY”指令到空闲的测试机，同时将计算机运行管理表中的 status 字段置为 1；测试机还有

一个 status 为 3 的维护状态，当手动设置 status=3 时，排队管理服务器不再检测此测试机的状态，方便技术人员作更新程序维护操作。

21. 根据权利要求 19 所述的系统，其特征在于，测试机接到“PLAY”指令就开始把测试记录对应的脚本文本通过 ftp 下载到 robot 的脚本目录，同时生成与脚本名相对应的扩展名为 script.rtxml 文件，使 robot 能够识别和能够调用此脚本，并在 Access 数据库表 casedb 中插入需要执行的测试记录，其中 ID 字段对应其提交时产生的 ID，status=0，name=脚本名。

22. 根据权利要求 19 所述的系统，其特征在于，Robot 在服务端启动时被启动，执行一个不断循环检测 Access 数据库表 casedb 的大脚本，按 sequence 升序选择出 casedb 中 status=0 的记录，按顺序调用其对应的测试记录脚本；执行时将 casedb 表和 oracle 数据库中的 execute 表中当前执行记录的 status 字段的值置为 1，执行完毕后将 status 置为 2。

23. 根据权利要求 19 所述的系统，其特征在于，执行过程中，用户是可以取消已经提交的正在处理或者未处理的测试记录的，用户通过自动化测试平台的一个排队监控窗口可以看到自己提交的所有 ID 对应的测试记录以及每个 ID，每个测试记录当前处于的状态，可以手动刷新和自动刷新；当用户选中一个 ID 时，如果这个 ID 是已经提交的未处理或者处理中的，用户可以通过按“停止执行”按钮发送一个“stopAll”指令来取消一个 ID 对应的所有处理中的测试记录的执行；当用户选中一个测试记录时，如果这个测试记录是已经提交的未处理或者处理中的，用户按“停止执行”时，会发送一个以“stop1”为首的指令到测试服务端，停止一个未处理或者处理中的测试记录的执行；服务端接收到“stop1”停止指令后，会把 casedb 和 oracle 数据库中 execute 表中对应的测试记录的 status 字段置为 4；收到“stopAll”指令后，会把 casedb 和 oracle 数据库 execute 表中的所有未处理或者正在处理的测试记录状态置为 4，同时把 queuecontrol 表中对应 ID 的记录的 flag 字段也置为 4；Deadloop 脚本每次调用一个测试记录脚本时都会先检测准备执行的

测试记录的 status 值，如果发现值为 4，则跳过此测试记录脚本的执行。

24. 根据权利要求 19 所述的系统，其特征在于，当脚本解释工具运行时，客户端起一个定时器调用 ClientMonDLL.dll 通过 tcp 连接到测试终端，获取测试终端做交易时的画面，实现实时屏幕监控；

当自动化排队管理服务器发送完 PLAY 指令后，或者客户端发送完“stop1”和“stopall”指令后，服务端都会发送一个“disconnect”指令通知自动化排队管理服务器和客户端断开与服务器端的连接。

一种基于脚本解释工具的自动化软件测试系统

技术领域

本发明涉及网络和计算机软件技术，特别涉及计算机应用软件的测试，具体的讲是一种基于脚本解释工具的自动化软件测试系统。

背景技术

软件测试是一件非常重要的工作，特别是对于金融业、工业等复杂且庞大的应用软件测试而言更是如此，因为软件的运行质量将直接影响多方的利益。然而，现有技术中的软件测试方法大多以某种脚本解释工具为基础，采用单机运行、手工操作的方法对应用软件进行测试。

现有的脚本解释工具很多，Robot 工具便是其中之一。Robot 是 Rational 公司开发的用于进行功能测试和性能测试的工具。使用 Rational Robot 可以开发三种类型的脚本：用于功能测试的 GUI 脚本、用于性能测试的 VU 和 VB 脚本。

Robot 提供了以下功能：

- 1) 完成所有的功能测试。记录并回放应用完成的过程，并且通过检查点来检测监控对象的状态。
- 2) 完成所有的性能测试。结合使用 Robot 与 TestManager 来记录脚本，有助于判断一个多客户端的系统的性能是否满足既定的标准。
- 3) 使用 SQABasic、VB 和 VU 脚本运行环境来创建和编辑脚本。在脚本开发中 Robot 编辑器为复杂的编写提供了方便的编辑环境。
- 4) 应用于 Visual Studio.NET、Visial Basic、Oracle 表格、PowerBuilder、HTML 和 Java 的开发。甚至在应用界面中不可见的对象都可以作为测试对象。
- 5) 在脚本回放过程中收集应用的诊断信息。Robot 已经与 Rational Putify、Quantify、PureCoverage 整合起来。在诊断工具下运行脚本之后，

可以在日志中查看诊断结果。

Robot 当中面向对象的记录技术允许仅仅通过简单地运行测试记录就可以快速产生出测试脚本。Robot 使用面向对象的记录方法，通过对对象内部的对象名来识别对象，而不是通过屏幕上的坐标来识别。如果对象改变了位置或文本，Robot 仍然可以在回放中找到它们。Robot 中的对象测试技术允许测试一个测试项目中的任何对象，包括该对象的属性和数据。不论在界面中是否可见，都可以测试标准 Windows 对象和 IDE 对象。可见 Robot 为用户编写的测试记录脚本提供了一个解释和执行的系统，脚本的识别和运行都是通过 Robot 来完成的。

Robot 工具虽然可以完成软件测试的工作，但是现有技术中对 Robot 工具的使用却十分原始，致使测试效率低下、浪费人力，其具体表现为：

- 1) 不能对测试案例进行有效的管理，因此无法解决案例存储和重用的问题；
- 2) 测试在单机或者前台机运行，直接影响前台用户的其他操作；
- 3) 测试流程人工操作，大量占用人力；
- 4) 运行 Robot 工具的单机互不联系，无法实现测试资源的均衡管理；
- 5) 运行 Robot 工具的单机仅针对某一具体软件的测试而工作，因此毫无灵活性可言，更不能满足多样的业务需求。

综上所述，虽然现有技术中的脚本解释工具很多，且这些脚本解释工具（如 Robot 工具）也能够帮助用户完成软件测试的工作，但是基于软件测试工具的自动化的、灵活性强的、兼容性好的软件测试系统目前还没有出现。

发明内容

本发明的目的在于，提供一种基于脚本解释工具的自动化软件测试系统。用以对测试案例进行有效的管理，解决案例重用的问题。测试通过后台执行，不影响前台用户的其他操作。测试流程自动化，无需人工操作。实现测试资源的均衡管理。并且使整个系统具有较强的灵活性，能满足多样的业务需求，

可兼容多种协议。

本发明的技术方案为：一种基于脚本解释工具的自动化软件测试系统，其中包括，

测试案例库：用于存储测试案例信息、测试案例脚本；

测试记录库：用于存储测试记录信息、测试记录脚本；

测试案例处理单元：用于对测试案例的生成、删除、重用进行处理；

测试记录处理单元：用于对测试记录的生成、删除、复制、运行状态统计进行处理；

测试案例管理单元：由脚本解释工具构成，在计算机的调度和测试监控下管理测试记录运行期间的所有操作；

测试总控单元：用于对系统资源和硬件资源进行控制；

所述的测试总控单元通过人机界面接受用户的操作指令信息，并控制所述的测试案例处理单元从所述的测试案例库中读取测试案例的初始数据，经用户的编辑操作之后，以文本形式保存测试案例脚本，并将其路径保存在测试案例库中；所述的测试案例经过所述的测试记录处理单元生成测试记录，保存在所述的测试记录库中，同时以文本的形式保存测试记录脚本；运行所述的测试记录，测试记录脚本通过所述的测试案例管理单元将测试数据传送到各应用程序进行处理，将运行结果经所述的测试案例管理单元保存到所述的测试记录库。

在所述的测试案例库中，测试案例信息与测试案例脚本一一对应，并且所述的测试案例信息存储于数据库中，所述的测试案例脚本以文本的形式存放；通过对数据库中测试案例信息的管理实现对所述的测试案例脚本的管理。

所述的测试案例处理单元包括：删除案例模块、案例类型模块、以及案例设计器。

所述的案例设计器是一个可视化案例设计模块，其通过图形界面，添加交易代码，修改交易的输入项和对应的动作，生成测试案例脚本。所述的案

例设计器能够记忆用户曾经测试过的各个功能模块的输入输出数据。

所述的案例设计器使用的文件包括：测试案例库中的测试案例脚本；使用的表包括：

案例表 (TC)：纪录已经生成的测试案例的信息；

交易输入项表 (TRANSITEM)：记载测试原始数据流，保存一次测试的所有输入项的名称和提示信息；

案例类型表 (TT)：管理所有案例类型信息；

其中，

生成案例类型的树型结构：

读取所述的案例类型表。由于系统默认第一层节点的上一层节点代码全部为 0，故扫描所有上一层类型代码为 0 的类型代码为第一层的节点，扫描所有上一层类型代码为第一层代码的类型代码为第二层的节点，以此类推。

所述的案例设计器包括自动生成界面测试案例单元，该自动生成界面测试案例单元包括：

交易代码管理模块：用于进行交易代码搜索、交易代码新增、交易代码修改、交易代码删除；

输入项管理模块：用于进行输入项新增、输入项修改、输入项删除；

检查项管理模块：用于检查项新增、检查项修改、检查项删除；

自动生成测试数据模块：用于交易输入场检查、测试案例数据生成；

测试规范导入导出模块：用于测试规范导入、测试规范导出。

所述的交易是指：各种被测应用软件。

所述的测试记录处理单元用于引用已经生成的测试案例，对测试案例中的变量进行赋值，生成可以运行的测试记录。

所述的测试记录处理单元使用的文件包括：测试案例库中的案例脚本 和 测试记录库中的测试记录脚本；

使用的表包括：

案例表 (TC): 纪录已经生成的测试案例的信息;

测试记录表 (TD): 纪录已经生成的测试记录的信息;

项目名称对照表 (PNC): 纪录测试库中的项目和对应的项目编号;

轮次名称对照表 (TN): 纪录测试库中的轮次和对应的轮次编号;

其中，新建测试记录的测试案例树由四层组成；前三层是案例类型的三个层次；第四层是用户名；以三个层次得到案例类型代码，搜索案例表，得到所有不重复测试案例的提交者名称；然后用提交者名称搜索用户管理表(UM)的 realname 字段，得到提交者的真实名称，显示为案例树的第四层；在案例脚本中增加赋值语句生成测试记录脚本。

所述的测试记录处理单元还包括测试记录重用模块，用于从已经生成的测试记录中生成新的测试记录，并保留原测试记录的所有变量值，直接运行测试或者修改某几个变量值后进行不同功能点的测试。

所述的测试记录重用模块包括，单条复制：用于把一个测试记录复制到指定的某个目录下，复制中不改动任何变量值；

批量复制：用于把同一用户目录下同一案例编号的多条测试记录复制到指定的目录下，复制中可以指定要修改哪些变量值，并统一修改。

所述的批量复制所使用的文件包括：测试案例库中的测试记录脚本；

使用的表包括，

测试记录表 (TD): 纪录已经生成的测试记录的信息;

项目名称对照表 (PNC): 纪录测试库中的项目和对应的项目编号;

轮次名称对照表 (TN): 纪录测试库中的轮次和对应的轮次编号；

用户管理表 (UM);

其中，批量复制主要由两个循环组成，一个大循环是逐个复制选中的测试记录，第二个是复制每个测试记录的时候，循环读原记录脚本，生成新的记录脚本；

批量复制中的批量修改可变值是在复制前，列出所有可变值，让用户指

定要修改的变量名称；复制过程中，碰到这些变量的赋值时，就用用户指定的值来复制；

赋值区后的复制操作：把赋值区后的内容从源文件复制到目标文件，然后把目标文件上传到案例库服务器；

数据库操作：复制完一个测试记录后，就把该测试记录的项目编号，轮次，案例编号，记录数等组成一条记录插到测试记录表（TD 表）中。

所述的测试案例管理单元包括：测试监控模块、测试案例传递模块、计算机调度模块。

所述的测试总控单元包括：系统设置模块、用户管理模块、修改密码模块、测试机管理模块、日志查看模块、轮次管理模块、项目管理模块。

本发明所述的系统还包括：

至少一台客户端计算机：执行所述的测试总控单元的功能；

至少一台测试机：载有测试终端，脚本解释工具，由测试案例处理单元、测试记录处理单元、测试案例管理单元构成的自动化测试服务端、案例数据库；

案例库服务器：存储有测试案例库、测试记录库；

其中，所述的客户端计算机、测试机、案例库服务器相互耦合，且所述的测试机与运行被测应用软件的应用服务器连接。

所述的客户端计算机、测试机、案例库服务器之间采用 TCP 协议连接；所述的测试机与应用服务器之间采用相应协议的连接，所述的相应协议包括：TCP 协议、HTTP 协议或 TELNET 协议。

在所述的测试机起一个服务充当 Server；所述的客户端计算机充当 Client，并通过 TCP 连接到所述的测试机；在所述的客户端计算机端起一个定时器，每隔一段时间向测试机请求一次截屏，把截屏信息在客户端计算机显示出来，实现对所述测试机屏幕的监视。

在所述的 Client 和 Server 端自定义一些用于通讯的 TCP 通讯命令，该 TCP 通讯命令如下：

以“PLAY”字符串开头的执行指令；

以“stop1”开头的停止单条测试记录执行的指令；

“stopAll”停止一次提交的所有测试记录的执行指令；

“disconnect”为断开连接指令；

自动化测试平台每一次提交测试系统都会生成一个全系统唯一的标识 ID，并以 ID 为关键字在 queuecontrol 表中插入一条记录，所有这次提交的记录都按提交时指定顺序插入 execute 表。

自动化测试排队管理服务器不停地发 test 指令给测试服务端，检测测试机的状态，如果发现测试机状态异常，且计算机运行管理（PCM）表中的 status=0，就调用 shutdown 程序远程重启测试机，同时置 PCM 表中当前测试机的状态变为 2（正在重启）；如果发现有空闲的测试机，取得其 IP 后就开始检测 queuecontrol 表，一旦发现有 flag=0（未处理）的 ID 号，就按 ID 号升序排列，取出第一条记录的 ID 号查询 execute 表，把所有符合条件的测试记录通过 TCP 发“PLAY”指令到空闲的测试机，同时将 PCM 表中的 status 字段置为 1（忙）；测试机还有一个 status 为 3 的维护状态，当手动设置 status=3 时，排队管理服务器不再检测此测试机的状态，方便技术人员作更新程序等维护操作。

测试机接到“PLAY”指令就开始把测试记录对应的脚本文本通过 ftp 下载到 robot 的脚本目录，同时生成与脚本名相对应的扩展名为.script.rtxml 文件，使 robot 能够识别和能够调用此脚本。并在 Access 数据库表 casedb 中插入需要执行的测试记录，其中 ID 字段对应其提交时产生的 ID，status=0，name=脚本名。

Robot 在服务端启动时被启动，执行一个不断循环检测 Access 数据库表 casedb 的大脚本，按 sequence 升序选择出 casedb 中 status=0 的记录，按顺序调用其对应的测试记录脚本；执行时将 casedb 表和 oracle 数据库中的 execute 表中当前执行记录的 status 字段的值置为 1（正在运行），执行完毕

后将 status 置为 2 (运行完毕)。

执行过程中，用户是可以取消已经提交的正在处理或者未处理的测试记录的，用户通过自动化测试平台的一个排队监控窗口可以看到自己提交的所有 ID 对应的测试记录以及每个 ID，每个测试记录当前处于的状态，可以手动刷新和自动刷新。当用户选中一个 ID 时，如果这个 ID 是未处理或者处理中的，用户可以通过按“停止执行”按钮发送一个“stopAll”指令来取消一个 ID 对应的所有正在处理或者处理中的测试记录的执行；当用户选中一个测试记录时，如果这个测试记录是未处理或者处理中的，用户按“停止执行”时，会发送一个以“stop1”为首的指令到测试服务端，停止一个未处理或者处理中的测试记录的执行。服务端接收到“stop1”停止指令后，会把 casedb 和 oracle 数据库中 execute 表中对应的测试记录的 status 字段置为 4(已取消)。收到“stopAll”指令后，会把 casedb 和 oracle 数据库 execute 表中的所有未处理或者正在处理的测试记录状态置为 4 (已取消)，同时把 queuecontrol 表中对应 ID 的记录的 flag 字段也置为 4 (已取消)；Deadloop 脚本每次调用一个测试记录脚本时都会先检测准备执行的测试记录的 status 值，如果发现值为 4，则跳过此测试记录脚本的执行。

监控：当脚本解释工具运行时，客户端起一个定时器调用 ClientMonDLL.dll 通过 tcp 连接到测试终端，获取测试终端做交易时的画面，实现实时屏幕监控；

当自动化排队管理服务器发送完 PLAY 指令后，或者客户端发送完“stop1”和“stopall”指令后，服务端都会发送一个“disconnect”指令通知自动化排队管理服务器和客户端断开与服务器端的连接。

本发明的有益效果在于，1)建立测试案例库对测试案例进行有效的管理，解决了案例重用的问题。2) 测试通过后台执行，不影响用户的前台操作。3) 测试流程自动化操作，无需人工操作。4) 一台测试机一次只能运行一个系统应用程序，自动化测试系统在数据库里建立了测试机管理表，对每一台测试

机的忙闲状态进行判断，来决定测试案例在哪一台测试机上执行。实现了测试资源的均衡管理。5) 自动化测试系统的案例使用脚本解释工具 (Rational

测试工具等)的脚本语言(一种类 C++ 编程语言)进行编写和解释执行,因此,具有极高的灵活性和适应性,能满足各种各样的业务逻辑需要。具有较强的灵活性,能满足多样的业务需求。6)通过 ROBOT 可以对在 WINDOWS 系统下运行的应用程序进行测试:包括单机板的应用软件、C/S 网络应用系统、B/S 浏览器应用系统以及各种仿真终端软件等。

附图说明

- 图 1 为本发明系统网络结构图;
- 图 2 为本发明逻辑结构框图;
- 图 3 为本发明数据流程图;
- 图 4 为本发明数据结构图;
- 图 5 为本发明测试管理模块实现流程图;
- 图 6 为案例设计器的程序处理流程图;
- 图 7 为生成测试记录程序处理流程图;
- 图 8 为测试记录重用程序处理流程图;
- 图 9 为生成测试案例的处理流程图;
- 图 10 为测试案例重用流程图;
- 图 11 为具体实施方式的人机界面图;
- 图 12 为选择项目和轮次界面图;
- 图 13 为对所选项目和轮次进行确认的界面图;
- 图 14 为运行测试记录流程图;
- 图 15 为本发明系统运行 TELNET 协议的测试流程图;
- 图 16 为本发明系统运行 TELNET 协议的测试案例生成流程图;
- 图 17 为本发明系统运行 TELNET 协议的测试案例重用流程图;
- 图 18 为本发明系统运行 TELNET 协议的测试测试记录运行流程图。

具体实施方式

下面结合附图说明本发明的具体实施方式。本发明为一种基于脚本解释

工具的自动化软件测试系统，其中包括，测试案例库：用于存储测试案例信息、测试案例脚本；测试记录库：用于存储测试记录信息、测试记录脚本；测试案例处理单元：用于对测试案例的生成、删除、重用进行处理；测试记录处理单元：用于对测试记录的生成、删除、复制、运行状态统计进行处理；测试案例管理单元：由脚本解释工具构成，在计算机的调度和测试监控下管理测试记录运行期间的所有操作；测试总控单元：用于对系统资源和硬件资源进行控制；所述的测试总控单元通过人机界面接受用户的操作指令信息，并控制所述的测试案例处理单元从所述的测试案例库中读取测试案例的初始数据，经用户的编辑操作之后，以文本形式保存测试案例脚本，并将其路径保存在测试案例库中；所述的测试案例经过所述的测试记录处理单元生成测试记录，保存在所述的测试记录库中，同时以文本的形式保存测试记录脚本；运行所述的测试记录，测试记录脚本通过所述的测试案例管理单元将测试数据传送到各应用程序进行处理，得到运行结果，将运行结果经所述的测试案例管理单元保存到所述的测试记录库。

所述的测试案例是指：针对一定的测试目标，带有固定的业务逻辑和环境变量的测试脚本。

所述的测试记录是指：记录某次实际测试操作的数据的脚本。与测试案例的区别在于，测试案例针对某一个测试目标，但由于测试环境的不确定而带有未确定的属性。而测试记录针对的是为了实现某一个测试目标的一次具体测试，是测试案例的具体化。由于测试环境已经明确，所以测试记录把测试案例的属性具体化成数据了。

所述的测试机是指：安装了自动化测试服务端、Rational Robot 的机器，用于执行测试记录的脚本、实施具体的测试。

如图 1 所示，为本发明系统的实施例，其示出了一自动化测试系统网络结构图。该自动化测试系统基于 C/S 架构，其网络结构主要包括前台客户端、测试案例库、测试机。

前台客户端：作用是接受用户的各种指令，并将处理结果以可视化界面的形式展现给用户。

测试机：作用是处理指令以及完成各个功能。

测试案例库：包括测试案例库和测试记录库，是数据存储的核心部分。

如图 2 所示，自动化测试系统包括管理模块、测试案例模块、测试记录模块、测试管理模块、案例脚本、测试记录脚本、案例库、测试记录库这几个模块。其中：

管理模块：针对自动化测试系统的系统以及硬件资源进行管理，包括系统设计、用户管理、修改密码、测试机管理、日志查看、轮次管理、项目管理等子模块。根据用户的不同权限，拥有不同的子模块功能。

测试案例模块：对测试案例的生成、删除和重用进行管理，包括删除案例、案例类型管理、案例设计器等子模块。

测试记录模块：对测试记录的生成、删除和复制以及按照项目的分类进行测试记录运行状态的统计，包括新建测试记录、删除测试记录、执行测试记录、编辑代码等子模块。

测试管理模块：管理测试记录运行期间的所有操作，包括如测试监控、案例传递、计算机调度等子模块。

案例库：存放案例。

测试记录库：存放测试记录以及其相关测试结果信息。

如图 3 所示，用户的操作指令信息传送到案例处理模块，首先从案例库中读取案例的初始数据，经用户的编辑等操作之后，以文本形式保存案例脚本，并将其路径保存在案例库中。测试案例经过测试记录处理模块生成测试记录，保存在测试记录库中，同时以文本的形式保存测试记录脚本。运行测试记录时，测试记录脚本通过测试管理模块将测试数据传送到各应用程序进行处理后得到运行结果，再将运行结果经测试管理模块保存到测试记录库。

如图 4 所示，为本发明的数据结构设计。主要数据库表及其主要字段的

说明见表1、表2、表3、表4、表5、表6、表7、表8、表9、表10、表11、表12和表13所示。

表1 测试案例表 (TC)

字段名	数据类型	长度	键值	说明
FUPO	VARCHAR2	20	Y	测试案例编号
SCRIPT	VARCHAR2	8	N	脚本程序名
TCTYPE	VARCHAR2	6		种类
DPT	VARCHAR2	3027	N	描述
USERNAME	VARCHAR2	20	N	用户名
TCTIMES	NUMBER	10		案例被引用次数

表2 测试记录表 (TD)

字段名	数据类型	长度	键值	说明
PROJECTNO	VARCHAR2	8	Y	项目编号
TNO	VARCHAR2	1	Y	轮次
FUPO	VARCHAR2	10	Y	测试案例编号
RECNUM	VARCHAR2	2	Y	测试案例记录数
CASENUM	VARCHAR2	2	Y	交易步骤
SCRIPT	VARCHAR2	40	N	脚本程序名
TDTYPE	VARCHAR2	6	N	种类
DPT	VARCHAR2	3072	N	描述
STATUS	VARCHAR2	1	N	运行状态
RESULT	VARCHAR2	1	N	测试记录执行结果
USERNAME	VARCHAR2	20	N	用户名

表 3 测试报告表 (TR)

字段名	数据类型	长度	键值	说明
PROJECTNO	VARCHAR2	8	Y	项目编号
TNO	VARCHAR2	1	Y	轮次
FUPO	VARCHAR2	10	Y	测试案例编号
RECNUM	VARCHAR2	2	Y	测试案例记录数
CASENUM	VARCHAR2	2	Y	交易步骤
TRANS	VARCHAR2	6	N	交易号
ADDOPR	VARCHAR2	20	N	测试案例提交者
ENDPIC	VARCHAR2	40	N	测试结果截图存放路径
RESULT	VARCHAR2	1	N	测试结果 0: 失败; 1: 成功

表 4 用户管理表 (UM)

字段名	数据类型	长度	键值	说明
USERNAME	VARCHAR2	20	Y	用户名
REALNAME	VARCHAR2	20	N	真实姓名
PRIVE	VARCHAR2	20	N	用户权限串
TESTPCIP	VARCHAR2	15	N	占用的测试机 IP
NOTESID	VARCHAR2	50	N	用户的 NotesID

表 5 计算机运行管理表 (PCM)

字段名	数据类型	长度	键值	说明
IP	VARCHAR2	20	Y	测试机 IP
PCNAME	VARCHAR2	20	N	测试机的机器名
STATUS	VARCHAR2	20	N	测试机的状态
NOTESID	VARCHAR2	15	N	用户的 NotesID

表 6 交易输入项表 (TRANSITEM)

字段名	数据类型	长度	键值	说明
TRANS	VARCHAR2	10	Y	交易代码
FLAG	INT	10	N	类型标志: 0-提示信息, 1-输入项
STREAM	VARCHAR2	4000	N	数据结构: 各种类型的数据, 以逗号为分隔符存放

表 7 类型表 (TT)

字段名	数据类型	长度	键值	说明
TYPEID	VARCHAR2	20	N	类型代码
TYPENAME	VARCHAR2	40	N	类型名称
FTYPE	VARCHAR2	20	N	上层类型代码

表 8 数据流表 (DATASTREAM)

字段名	数据类型	长度	键值	说明
TRANS	VARCHAR2	6	Y	交易名
CREATEDATE	VARCHAR2	10	N	修改日期
STREAM	VARCHAR2	200	N	描述
SCRIPT	VARCHAR2	20	N	脚本名

表 9 轮次名称对照表 (TN)

字段名	数据类型	长度	键值	说明
TNO	VARCHAR2	2	Y	轮次号
TNAME	VARCHAR2	8	N	轮次中文名

表 10 项目名称对照表 (PNC)

字段名	数据类型	长度	键值	说明
PRONO	VARCHAR2	10	Y	项目编号
PRONAME	VARCHAR2	20	N	项目中文名称

建立在测试机上的数据库表为表 11:

表 11: CASEDB

字段名	数据类型	长度	键值	说明
ID	LONG	自动编号	Y	编号
NAME	CHAR	50	N	测试记录名
STATUS	INT	10	N	状态

表 12: QUEUECONTROL

字段名	数据类型	长度	键值	说明
ID	NUMBER	10	Y	编号
USERNAME	CHAR	20	N	用户名
FLAG	CHAR	1	N	状态
DATETIME	DATE		N	日期

表 13: EXECUTE

字段名	数据类型	长度	键值	说明
ID	NUMBER	10	Y	编号
SEQUENCE	CHAR	20	N	用户名
STATUS	CHAR	1	N	状态

在本发明中，测试管理模块作为测试监控和执行的主要模块，其实现主要是围绕以下三个问题的解决办法而展开的：

1) 实现对测试机屏幕的监视。在测试机起一个服务充当 Server，客户端充当 Client，通过 Tcp 连接到测试机，在 Client 端起一个定时器，每隔一段时间向测试机请求一次截屏，把截屏信息在客户端显示出来。

2) 实现对测试机的一些必须的控制。需要在 Client 和 Server 端协定一些用于通讯的指令。自定义的 TCP 通讯命令如下：

以“PLAY”字符串开头的执行指令；

以“stop1”开头的停止单条测试记录执行的指令；

“stopAll”停止一次提交的所有测试记录的执行的指令；

“disconnect”为断开连接指令；

3) 实现对测试机的调度。

自动化测试平台每一次提交测试系统都会生成一个全系统唯一的标识 ID，并以 ID 为关键字在 queuecontrol 表中插入一条记录，所有这次提交的记录都按提交时指定顺序插入 execute 表。

自动化测试排队管理服务器不停地发 test 指令给测试服务端，检测测试机的状态，如果发现测试机状态异常，且 PCM 表中的 status=0；就调用 shutdown 程序远程重启测试机，同时置 PCM 表中当前测试机的状态变为 2(正在重启)；如果发现有空闲的测试机，取得其 IP 后就开始检测 queuecontrol 表，一旦发现有 flag=0(未处理) 的 ID 号，就按 ID 号升序排列，取出第一条记录的 ID 号查询 execute 表，把所有符合条件的测试记录通过 TCP 发“PLAY”指令到空闲的测试机，同时将 PCM 表中的 status 字段置为 1(忙)；测试机还有一个 status 为 3 的维护状态，当手动设置 status=3 时，排队管理服务器不再检测此测试机的状态，方便技术人员作更新程序等维护操作。

测试机接到“PLAY”指令就开始把测试记录对应的脚本文本通过 ftp 下载到 robot 的脚本目录，同时生成与脚本名相对应的扩展名为 script.rtxml 文

件，使 robot 能够识别和能够调用此脚本；并在 Access 数据库表 casedb 中插入需要执行的测试记录，其中 ID 字段对应其提交时产生的 ID，status=0，name=脚本名。

Robot 在服务端启动时被启动，执行一个不断循环检测 Access 数据库表 casedb 的大脚本，按 sequence 升序 select 出 casedb 中 status=0 的记录，按顺序调用其对应的测试记录脚本；执行时将 casedb 表和 oracle 数据库中的 execute 表中当前执行记录的 status 字段的值置为 1（正在运行），执行完毕后将 status 置为 2（运行完毕）。

执行过程中，用户是可以取消已经提交的正在处理或者未处理的测试记录的，用户通过自动化测试平台的一个排队监控窗口可以看到自己提交的所有 ID 对应的测试记录以及每个 ID，每个测试记录当前处于的状态，可以手动刷新和自动刷新。当用户选中一个 ID 时，如果这个 ID 是未处理或者处理中的，用户可以通过按“停止执行”按钮发送一个“stopAll”指令来取消一个 ID 对应的所有正在处理或者处理中的测试记录的执行；当用户选中一个测试记录时，如果这个测试记录是未处理或者处理中的，用户按“停止执行”时，会发送一个以“stop1”为首的指令到测试服务端，停止一个未处理或者处理中的测试记录的执行。服务端接收到“stop1”停止指令后，会把 casedb 和 oracle 数据库中 execute 表中对应的测试记录的 status 字段置为 4（已取消）；收到“stopAll”指令后，会把 casedb 和 oracle 数据库 execute 表中的所有未处理或者正在处理的测试记录状态置为 4（已取消），同时把 queuecontrol 表中对应 ID 的记录的 flag 字段也置为 4（已取消）；Deadloop 脚本每次调用一个测试记录脚本时都会先检测准备执行的测试记录的 status 值，如果发现值为 4，则跳过此测试记录脚本的执行。

监控：当脚本解释工具运行时，客户端起一个定时器调用 ClientMonDLL.dll 通过 tcp 连接到测试终端，获取测试终端做交易时的画面，实现实时屏幕监控；

当自动化排队管理服务器发送完 PLAY 指令后,或者客户端发送完“stop1”和“stopall”指令后,服务端都会发送一个“disconnect”指令通知自动化排队管理服务器和客户端断开与服务器段的连接。

案例设计器的实现:

测试案例设计器是针对用户的使用习惯开发的一个可视化案例设计模块,其功能是通过图形界面,添加交易代码,修改交易的输入项和对应的动作,生成测试案例脚本。测试案例设计器可记忆用户曾经测试的各个功能模块的输入,有效的帮助用户在设计测试案例时,根据以往的工作,简化当前工作的流程。例如,用户曾经对某系统的部分模块进行测试,测试案例设计器会记住测试该模块时的所有输入输出数据,在以后的案例设计中,当用户再对该模块进行测试设计时,设计器会把该模块的曾经测试过的信息调出来提供给用户参考,方便了用户的设计的工作。

使用的文件: 案例库服务器上的案例文件(案例脚本)

使用的表: TC, TRANSITEM, DATASTREAM, TT

TC(案例表): 纪录已经生成的测试案例的信息,包括: 案例编号,案例描述,提交者,对应的脚本路径。字段内容详见数据库设计部分。

TRANSITEM: 是一个记载测试原始数据流的表,保存一次测试的所有输入项的名称和提示信息等。字段内容详见数据库设计部分。

TT(案例类型表): 用来管理自动化测试案例类型的表,在案例设计器模块中的作用是存放所有测试案例的类型。字段内容详见数据库设计部分。

如图 6 所示,显示案例设计器的第一个界面,生成案例类型的树型结构:读 TT 表,由于系统默认第一层节点的上一层节点代码全部为 0,故扫描所有上一层类型代码为 0 的类型代码为第一层的节点,扫描所有上一层类型代码为第一层代码的类型代码为第二层的节点,以此类推。在案例文本中,对应的被测模块有三个输入流: inputstr、outputstr、actionstr,分别以表格形式显示在界面上。

在 `transitem` 中, `flag` 有 4 个值: 0, 2 表示对应字段 `stream` 中的内容为 `inputstr` 的内容 (测试的输入项), 其中 0 表示选输项, 2 表示必输项; 1 对应 `actionstr` 的内容; 3 对应交易的出错信息。

每个被测模块的数据流由一系列输入项名称和输入值组成。其中输入值有以下几种:

可变值: 案例中不固定, 生成测试记录的时候再赋值。

固定值: 在案例中固定的一些数字, 字符等。

参数传递的流程是: 将前面的被测模块的输入或输出信息放到 `outputstr` 中, 以供后面的被测模块使用。

`inputstr`, `outputstr`, `actionstr` 的格式是”名称=值 | 名称=值 |”

测试案例生成测试记录

该模块的功能是引用已经生成的测试案例, 对测试案例中的变量进行赋值, 生成可以运行的测试记录。

使用的文件: 案例库服务器上的案例文件 (案例脚本) 和测试记录文件 (测试记录脚本)

使用的表: TC, TD, PNC, TN

TC (案例表): 纪录已经生成的测试案例的信息, 包括: 案例编号、案例描述、提交者、对应的脚本路径。字段内容详见数据库设计部分。

TD (测试记录表): 纪录已经生成的测试记录的信息, 包括: 项目编号、轮次、案例编号、记录数、案例描述、提交者、对应的脚本路径。字段内容详见数据库设计部分。

PNC (项目名称对照表): 纪录测试库中的项目和对应的项目编号。

TN (轮次名称对照表): 纪录测试库中的轮次和对应的轮次编号。

如图 7 所示, 新建测试记录的测试案例树由四层组成。前三层是案例类型的三个层次, 这个在前面的案例设计器中已经说明。第四层是用户名: 以三个层次得到案例类型代码, 搜索 TC 表, 得到所有不重复测试案例的提交者

名称。然后用提交者名称搜索用户管理表（UM）的 `realname` 字段，得到提交者的真实名称，显示为案例树的第四层。

在案例脚本中增加赋值语句生成测试记录脚本：在案例脚本中的数据流的一般格式是”项=值 | ...”，对于”值”为可变值的，用”`edit(i)`”表示。这里”`i`”表示该变量是案例脚本中的第几个变量。所以添加赋值语句的时候，根据界面上变量名的次序分别增加”`edit(i)=xxx`”等赋值语句。

测试记录重用的实现：

该模块的功能是从已经生成的测试记录“快速”生成新的测试记录，并保留原测试记录的所有变量值，可以直接运行测试，也可以修改某几个变量值后进行不同功能点的测试。

测试记录的重用包括两部分：单条复制和批量复制

单条复制：把一个测试记录复制到指定的某个目录下。复制中不改动任何变量值。

批量复制：把同一用户目录下同一案例编号的多条测试记录复制到指定的目录下。复制中可以指定要修改那些变量值，并同一修改。

由于批量复制包含单条复制的实现，所以以下介绍批量复制的实现。

使用的文件：案例库服务器上的测试记录文件

使用的表：TD, PNC, TN, UM

如图 8 所示：

1) 批量复制主要由两个循环组成，一个大循环是逐个复制选中的测试记录。第二个是复制每个测试记录的时候，循环读原记录脚本，生成新的记录脚本。

2) 批量复制中的批量修改可变值是在复制前，列出所有可变值，让用户指定要修改的变量名称。复制过程中，碰到这些变量的赋值时，就用用户指定的值来复制。

3) 赋值区后的复制操作：把赋值区后的内容从源文件复制到目标文件。

然后把目标文件上传到案例库服务器。

4) 数据库操作：复制完一个测试记录后，就把该测试记录的项目编号，轮次，案例编号，记录数等组成一条记录插到 TD 表中。

测试范围说明：

自动化测试系统可以对在 WINDOWS 系统下运行的应用程序进行测试：包括单机板的应用软件、C/S 网络应用系统、B/S 浏览器应用系统以及各种仿真终端软件等。

其它说明：

1) 客户端开发及运行环境

开发工具：Delphi；

数据库客户端：Oracle 9i 关系数据库的客户端；

运行系统：Windows 2000 或 Windows XP；

2) 服务器端开发及运行环境

开发工具：Delphi、C++；

数据库服务器：Oracle 9i；

脚本执行系统：IBM Rational 的软件产品 Robot；

运行系统：Windows 2000 或 Windows XP；

在本发明系统的具体实施中，还需进行安全设计，其中包括：

1) 用户管理系统

自动化测试系统拥有自己单独的用户管理；并且通过权限设置，对各个用户的操作进行限定。用户管理还包括自动化测试系统用户的新增和删除以及用户密码和权限的修改。

2) 运行日志

自动化测试系统会将用户登陆和更改数据库等操作以及相应时间记录下来，可供在测试发生问题时查询自动化测试系统的使用状态，方便地查找系统故障或问题的所在。

以下为本发明的具体实施方式的各个处理流程：

一、测试案例生成向导

如图 9 所示，生成测试案例的处理流程为：

1、自动化测试系统根据客户端输入的信息，发出搜索数据库的请求；

2、自动化测试系统从数据库中搜索符合客户端请求搜索条件的记录；

3、自动化测试系统将数据库搜索结果返回给客户端；

4、自动化测试系统在客户端上显示数据库搜索结果；

5、添加交易功能：从数据库中取出该交易相关的输入项、输出项和交互项以及其相关的赋值信息，显示在功能测试向导界面上。使用户快捷地获得该交易的相关信息，并且可以在原有交易的基础上，进行交易的修改；节约其对交易各项定义的时间；

添加处理块功能：选定相关交易，然后添加处理快，调出处理

恢复默认值功能：可以将指定交易的各个输入项的值恢复到上一次定义该交易各个输入项的值（从数据库中获取），使用户在修改该交易各个输入项值的过程中可以快捷、简单地重新获得上一次块的编辑框；可以在编辑框中添加处理流程代码。使功能测试案例能够适用于更多的测试情况；

删除交易功能：可以在操作界面上的案例交易流中将选定的交易以及其处理块删除；

删除处理块功能：可以在操作界面上删除选定的交易处理块；定义该交易各个输入项的值，从而保证交易的可用性；

清除数据功能：可以将指定交易的各个输入项的赋值全部清除掉，使用户不受以前输入项定义的影响，方便其重新定义输入项的值；

清空数据功能：可以将指定交易输入项的名称和值全部都清除，用户可以重新定义交易；

保存数据功能，可以将指定交易本次生成的交易输入项名称以及其赋值

保存到数据库中，当下次使用该交易时，可以从数据库中将此次保存的交易输入项以及其赋值显示出来；

- 6、客户端发出功能测试案例生成请求；
- 7、自动化测试系统按照该功能测试案例交易的生成过程生成功能测试案例；
- 8、自动化测试系统将生成的功能测试案例分别保存到数据库和测试案例库中；
- 9、自动化测试系统将生成的功能测试案例编号返回给客户端并且显示在客户端操作界面上。

二、编辑测试案例脚本

处理流程说明为：当用户选择该项时，自动化测试系统调出选择测试案例对话框。

当打开搜索树的每一层时，自动化测试系统读取数据库中测试案例分类信息；到最后一层时，自动化测试系统从数据库中搜索属于该测试类型的案例并且显示在界面中；选择需要编辑的测试案例进行编辑。

自动化测试系统调用编辑处理块，弹出编辑对话框，对测试案例脚本进行直接修改；然后保存。自动化系统将保存后测试案例脚本保存到测试案例库中。

三、编辑测试案例

处理流程说明：当用户选择该项时，自动化测试系统调出选择测试案例对话框。

当打开搜索树的每一层时，自动化测试系统读取数据库中测试案例分类信息；到最后一层时，自动化测试系统从数据库中搜索属于该测试类型的案例并且显示在界面中；选择需要编辑的测试案例进行编辑。

自动化测试系统调出测试案例的对话框。测试案例进行可视化修改：针对现有案例的测试交易流程进行调整，其功能描述见：测试案例设计。

四、建立新的测试记录

如图 10 所述，测试案例重用流程为：

- 1、自动化测试系统接受客户端输入的数据库搜索请求信息；
- 2、自动化测试系统从数据库中搜索符合客户端请求搜索条件的记录；
- 3、自动化测试系统将数据库搜索结果返回给客户端；
- 4、自动化测试系统在客户端上显示数据库搜索结果；
- 5、自动化测试系统接受客户端选择信息：测试案例所在生成的路径和选定的测试案例以及其属性定义，在数据库中添加一条新的测试记录；将数据库成功添加信息返回给客户端；并且显示在客户端界面上通知测试人员新的测试记录添加成功。

五、测试记录的重用

如图 5 所示，测试记录的重用处理流程：在图纸那个界面上，选定测试记录并且修改测试记录的属性。

点击“生成测试记录”按钮，弹出设置测试记录所属项目和轮次的对话框。选定生成测试记录的项目和轮次以及添加对其的描述。

如图 12 所示，点击“生成实例”按钮，生成指定项目和轮次下的新测试记录。实现测试记录可以跨越测试项目使用。

如图 13 所示，选择“Yes”或“No”。

六、运行测试记录

如图 14 所示，运行测试记录处理流程：

- 1、客户端发起运行测试案例的请求；
- 2、自动化测试系统从数据库设备管理表中搜索是否还有空闲的后台服务器供用户运行测试脚本；
- 3、如果有空闲的机器，则将用户的信息存放到设备管理表中；
- 4、如果有空闲的机器，则运行测试记录，否则等待空闲的测试机器；
- 5、自动化测试系统通过 FTP 方式从测试案例库读取与指定运行测试记录对应的测试脚本；
- 6、自动化测试系统将该测试脚本通过 FTP 方式传送到后台服务器上运行；

7、自动化测试系统将在自动化测试系统的用户使用端启动用户端程序和运行服务器上运行一个服务端程序对运行测试终端画面进行拷贝并且立即将其传送到用户使用端；

8、使用户可以通过自动化测试系统界面下端的对话框对后台服务器运行情况进行实时监控。当然，用户也可以选择不监控测试记录的运行情况，甚至可以把关闭自动化测试系统。当用户再次打开自动化测试系统时，自动化测试系统先搜索数据库的设备管理表中是否有用户的信息；如果有，则测试脚本还在运行当中，用户可以再次监控后台服务器的测试脚本运行情况；

9、如果没有，则表示测试脚本已经运行完毕，自动化测试系统将运行完毕的信息发送到客户端并且发送邮件通知用户测试案例运行完毕的信息以及详细运行情况；

10、客户端接受到运行完毕的信息提示，在界面通知用户可以在界面查询测试脚本运行结果。

以下为本发明的一应用实例：使用自动化测试系统进行 TELNET 协议的自动化测试。（如图 15 所示）

1、当测试人员需要进行测试时，可以在客户端搜索符合当前测试要求的测试案例；简化测试案例的准备过程，特别是当测试案例足够多时，测试人员基本上可以使用测试案例库中编写的测试案例或者测试记录，减少测试人员编写测试案例的时间；

2、自动化测试系统从数据库中搜索符合搜索条件的测试案例；

3、自动化测试系统将数据库搜索结果返回客户端；

4、如果返回没有符合搜索条件的测试案例，测试人员需要使用测试案例设计器生成界面测试案例或者功能测试案例。自动生成界面测试案例：简化了测试人员编写详细测试案例的时间，测试人员只需设定交易输入项需要的检查规则，就可以通过自动化测试系统自动生成界面测试案例。同样，功能测试案例生成向导提供友好的界面供测试人员编写功能测试案例。如果有符

合搜索条件的测试案例，则重用该测试案例或者测试记录；以下详细介绍：自动生成界面测试案例过程、功能测试案例生成过程、测试案例重用过程和测试记录重用过程；

5、测试人员在客户端选定需要运行的测试案例并且对其属性进行赋值，然后生成测试记录，接着提交需要运行的测试记录（可以是多个测试记录一起提交）；

6、测试人员可以选择实时监控运行服务器的测试记录运行情况或者进行其它案例的编写并且可以方便地查询到以前和当前测试记录运行的结果，从而提高测试人员的工作效率。以下详细介绍测试记录运行流程。

一、测试案例生成过程（如图 16 所示）

1、自动化测试系统根据客户端输入的信息，发出搜索数据库的请求；

2、自动化测试系统从数据库中搜索符合客户端请求搜索条件的记录；

3、自动化测试系统将数据库搜索结果返回给客户端；

4、自动化测试系统在客户端上显示数据库搜索结果；

5、添加交易功能：从数据库中取出该交易相关的输入项、输出项和交互项以及其相关的赋值信息，显示在功能测试向导界面上。使用户快捷地获得该交易的相关信息，并且可以在原有交易的基础上，进行交易的修改；节约其对交易各项定义的时间；

添加处理块功能：选定相关交易，然后添加处理快，调出处理块的编辑框；可以在编辑框中添加处理流程代码。使功能测试案例能够适用于更多的测试情况；

删除交易功能：可以在操作界面上的案例交易流中将选定的交易以及其处理块删除；

删除处理块功能：可以在操作界面上删除选定的交易处理块；

恢复默认值功能：可以将指定交易的各个输入项的值恢复到上一次定义该交易各个输入项的值（从数据库中获取），使用户在修改该交易各个输入项值的过程中可以快捷、简单地重新获得上一次定义该交易各个输入项的值，

从而保证交易的可用性；

清除数据功能：可以将指定交易的各个输入项的赋值全部清除掉，使用
户不受以前输入项定义的影响，方便其重新定义输入项的值；

清空数据功能：可以将指定交易输入项的名称和值全部都清除，用户可
以重新定义交易；

保存数据功能，可以将指定交易本次生成的交易输入项名称以及其赋值
保存到数据库中，当下次使用该交易时，可以从数据库中将此次保存的交易
输入项以及其赋值显示出来；

- 6、客户端发出功能测试案例生成请求；
- 7、自动化测试系统按照该功能测试案例交易的生成过程生成功能测试案例；
- 8、自动化测试系统将生成的功能测试案例分别保存到数据库和测试案例库中；
- 9、自动化测试系统将生成的功能测试案例编号返回给客户端并且显示在
客户端操作界面上。

二、测试案例重用流程（如图 17 所示）

- 1、自动化测试系统接受客户端输入的数据库搜索请求信息；
- 2、自动化测试系统从数据库中搜索符合客户端请求搜索条件的记录；
- 3、自动化测试系统将数据库搜索结果返回给客户端；
- 4、自动化测试系统在客户端上显示数据库搜索结果；
- 5、自动化测试系统接受客户端选择信息：测试案例所在生成的路径和选
定的测试案例以及其属性定义，在数据库中添加一条新的测试记录；将数据
库成功添加信息返回给客户端；并且显示在客户端界面上通知测试人员新的
测试记录添加成功。

在测试记录重用过程中，

- 1、在主界面上，选定测试记录并且修改测试记录的属性。（如图 11）
- 2、点击“生成测试记录”按钮，弹出设置测试记录所属项目和轮次的对
话框。选定生成测试记录的项目和轮次以及添加对其的描述。（如图 12）

3、点击“生成实例”按钮，生成指定项目和轮次下的新测试记录。实现测试记录可以跨越测试项目使用。(如图 13)

三、自动化系统处理测试记录运行流程(如图 18 所示)

- 1、客户端发起运行测试案例的请求；
- 2、自动化测试系统从数据库设备管理表中搜索是否还有空闲的后台服务器供用户运行测试脚本；
- 3、如果有空闲的机器，则将用户的信息存放到设备管理表中；
- 4、如果有空闲的机器，则运行测试记录；否则等待空闲的测试机器；
- 5、自动化测试系统通过 FTP 方式从测试案例库读取与指定运行测试记录对应的测试脚本；
- 6、自动化测试系统将该测试脚本通过 FTP 方式传送到后台服务器上运行；
- 7、自动化测试系统将在自动化测试系统的用户使用端启动用户端程序和运行服务器上运行一个服务端程序对运行测试终端画面进行拷贝并且立即将其传送到用户使用端；
- 8、使用户可以通过自动化测试系统界面下端的对话框对后台服务器运行情况进行实时监控。当然，用户也可以选择不监控测试记录的运行情况，甚至可以把关闭自动化测试系统。当用户再次打开自动化测试系统时，自动化测试系统先搜索数据库的设备管理表中是否有用户的信息；如果有，则测试脚本还在运行当中，用户可以再次监控后台服务器的测试脚本运行情况；
- 9、如果没有，则表示测试脚本已经运行完毕，自动化测试系统将运行完毕的信息发送到客户端并且发送邮件通知用户测试案例运行完毕的信息以及详细运行情况；
- 10、客户端接受到运行完毕的信息提示，在界面通知用户可以在界面查询测试脚本运行结果。

使用自动化测试系统进行 HTTP 和 TCP/IP 协议的自动化测试其使用过程与字符终端的自动化测试大致相同，只是需要不同的测试脚本基础模板。测

试案例是基于该模板而产生的。由于基于 HTTP 协议和 TCP/IP 协议的测试，通常是 B/S 结构或者 C/S 结构，不但需要对交易画面中信息的变化进行测试，而且需要对浏览器中控件的属性进行检查。所以进行 HTTP 和 TCP/IP 的自动化测试需要不同的测试脚本基础模版。

本发明的有益效果在于，1)建立测试案例库对测试案例进行有效的管理，解决了案例重用的问题。2) 测试通过后台执行，不影响用户的前台操作。3) 测试流程自动化操作，无需人工操作。4) 一台测试机一次只能运行一个系统应用程序，自动化测试系统在数据库里建立了测试机管理表，对每一台测试机的忙闲状态进行判断，来决定测试案例在哪一台测试机上执行。实现了测试资源的均衡管理。5) 自动化测试系统的案例使用脚本解释工具 (Rational 测试工具等) 的脚本语言 (一种类 C++ 编程语言) 进行编写和解释执行，因此，具有极高的灵活性和适应性，能满足各种各样的业务逻辑需要。具有较强的灵活性，能满足多样的业务需求。6) 通过 ROBOT 可以对在 WINDOWS 系统下运行的应用程序进行测试：包括单机板的应用软件、C/S 网络应用系统、B/S 浏览器应用系统以及各种仿真终端软件等。

仅通过以上具体实施例详细说明本发明，并非用于限定本发明。

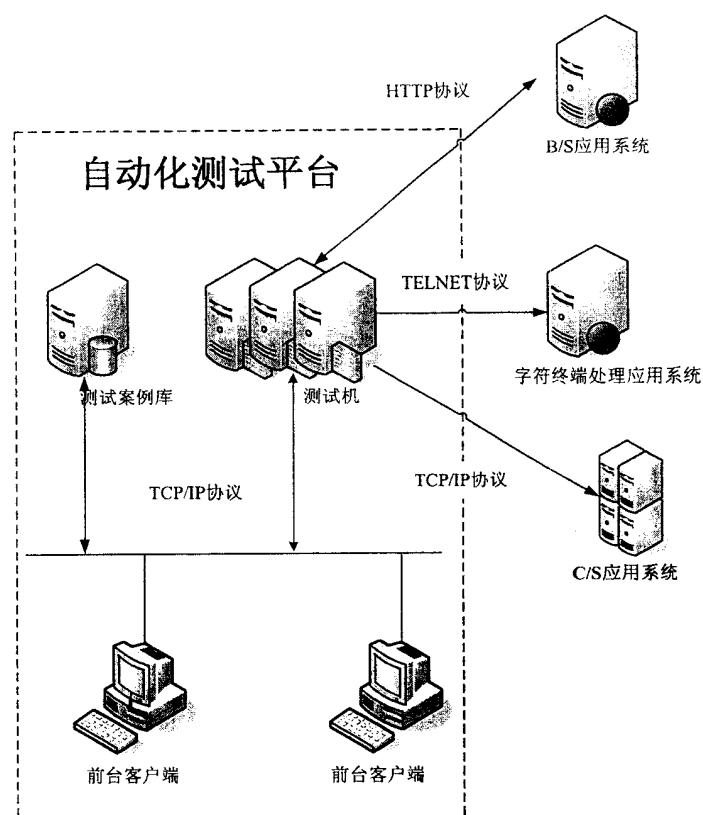


图 1

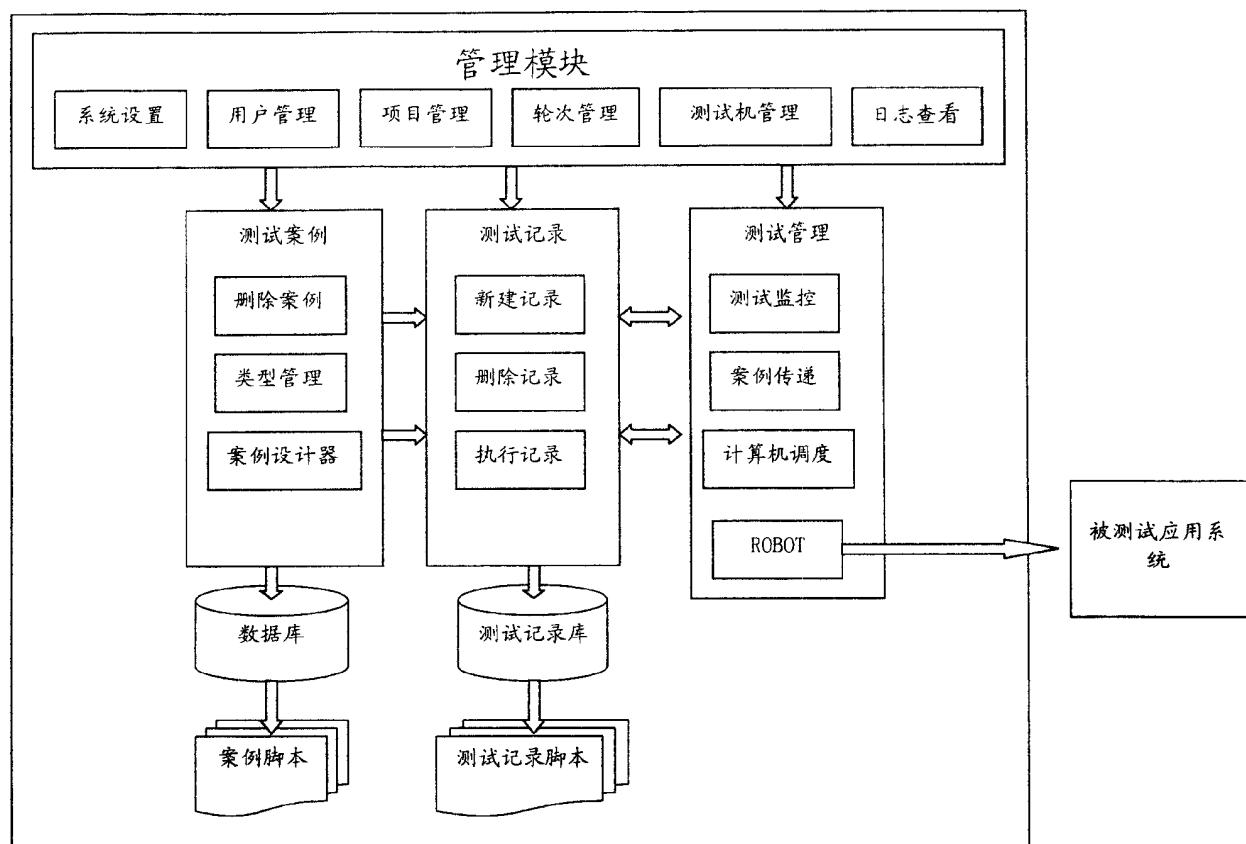


图 2

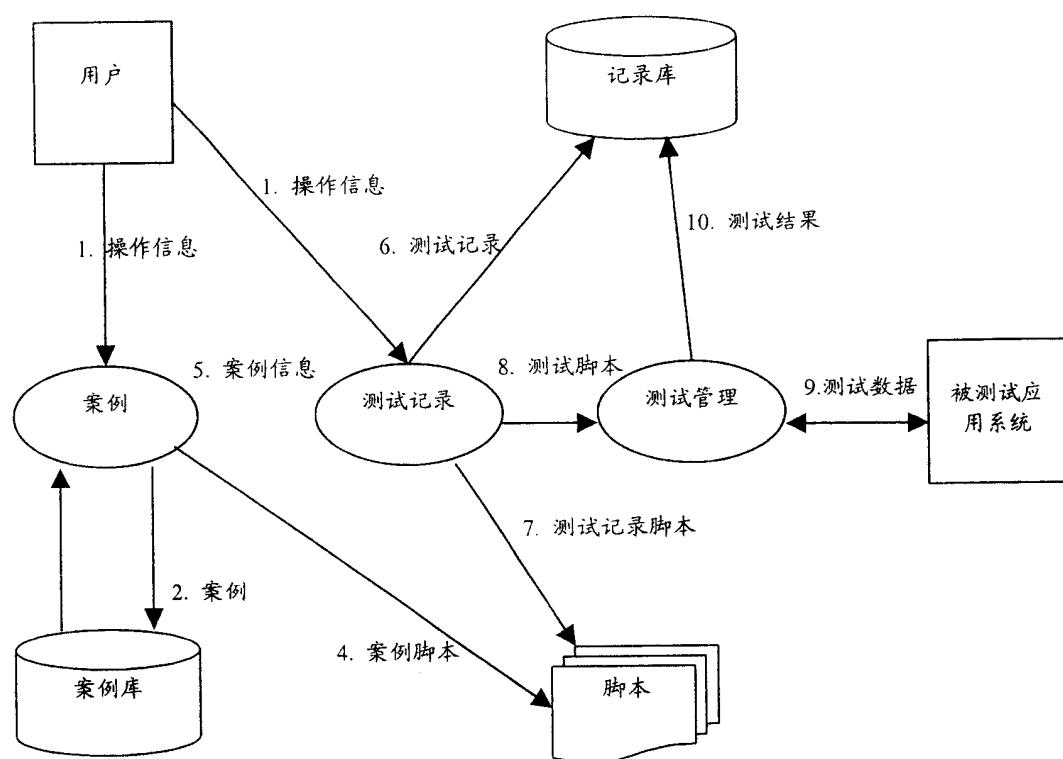


图 3

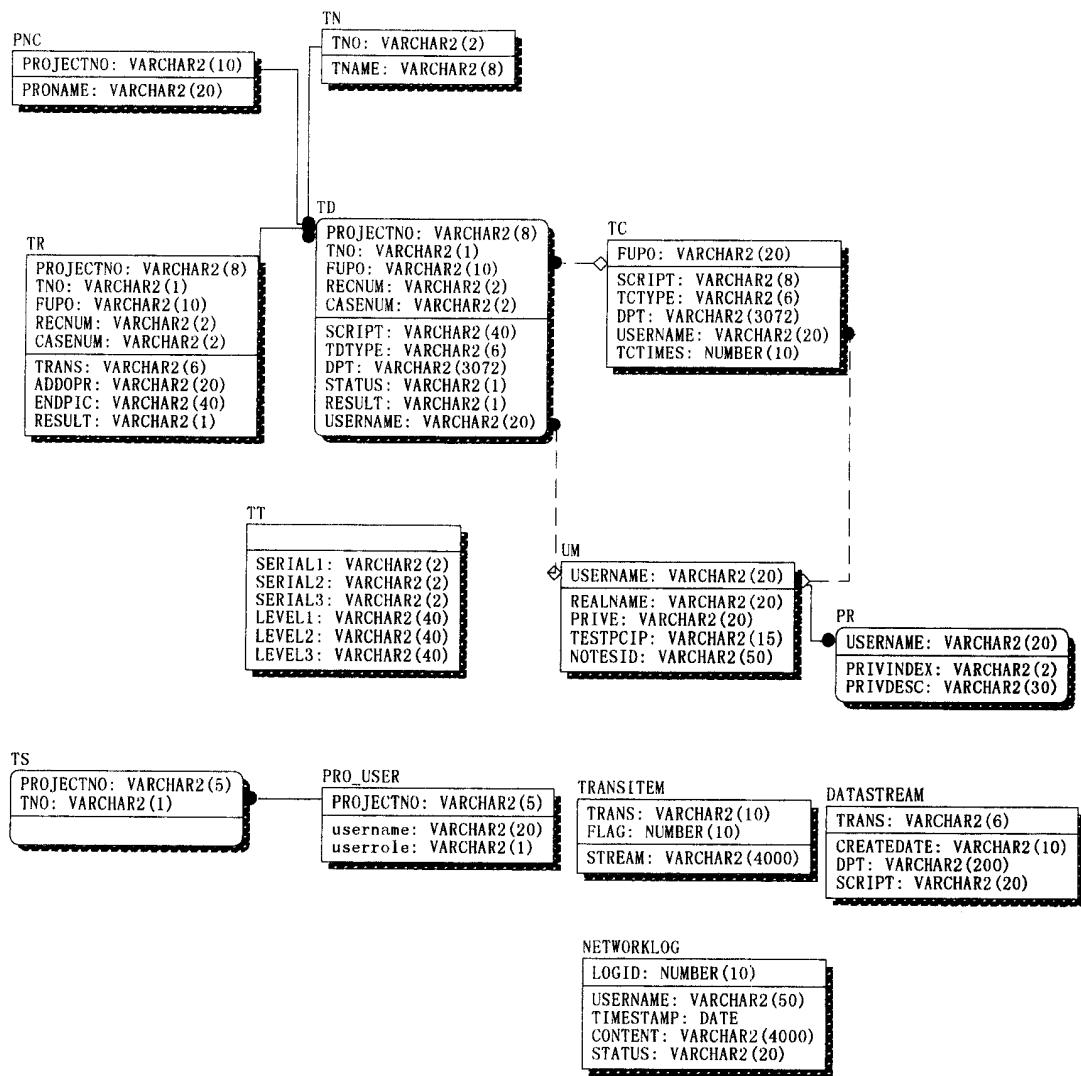


图 4

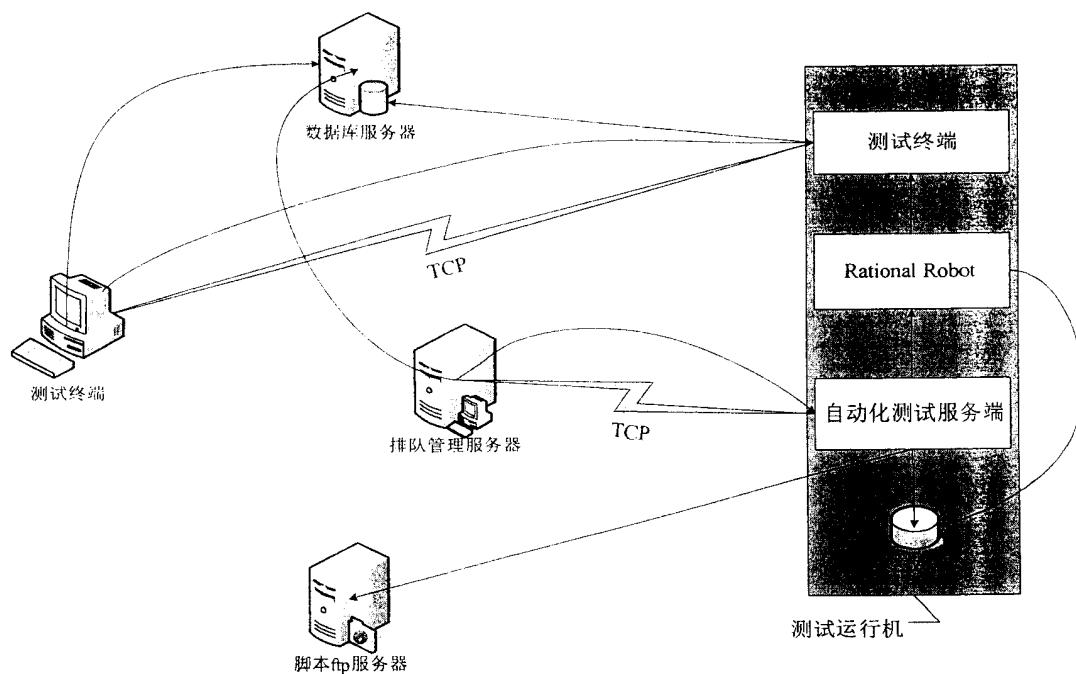


图 5

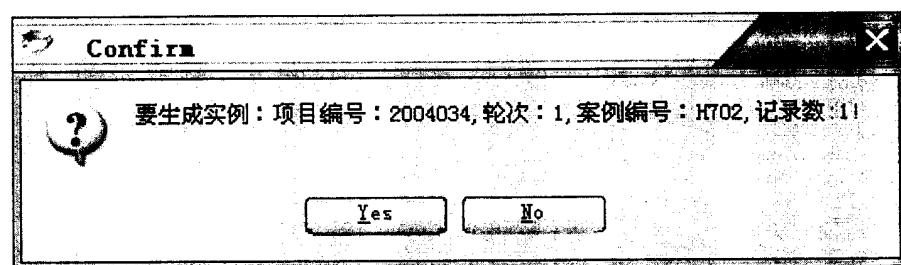


图 13

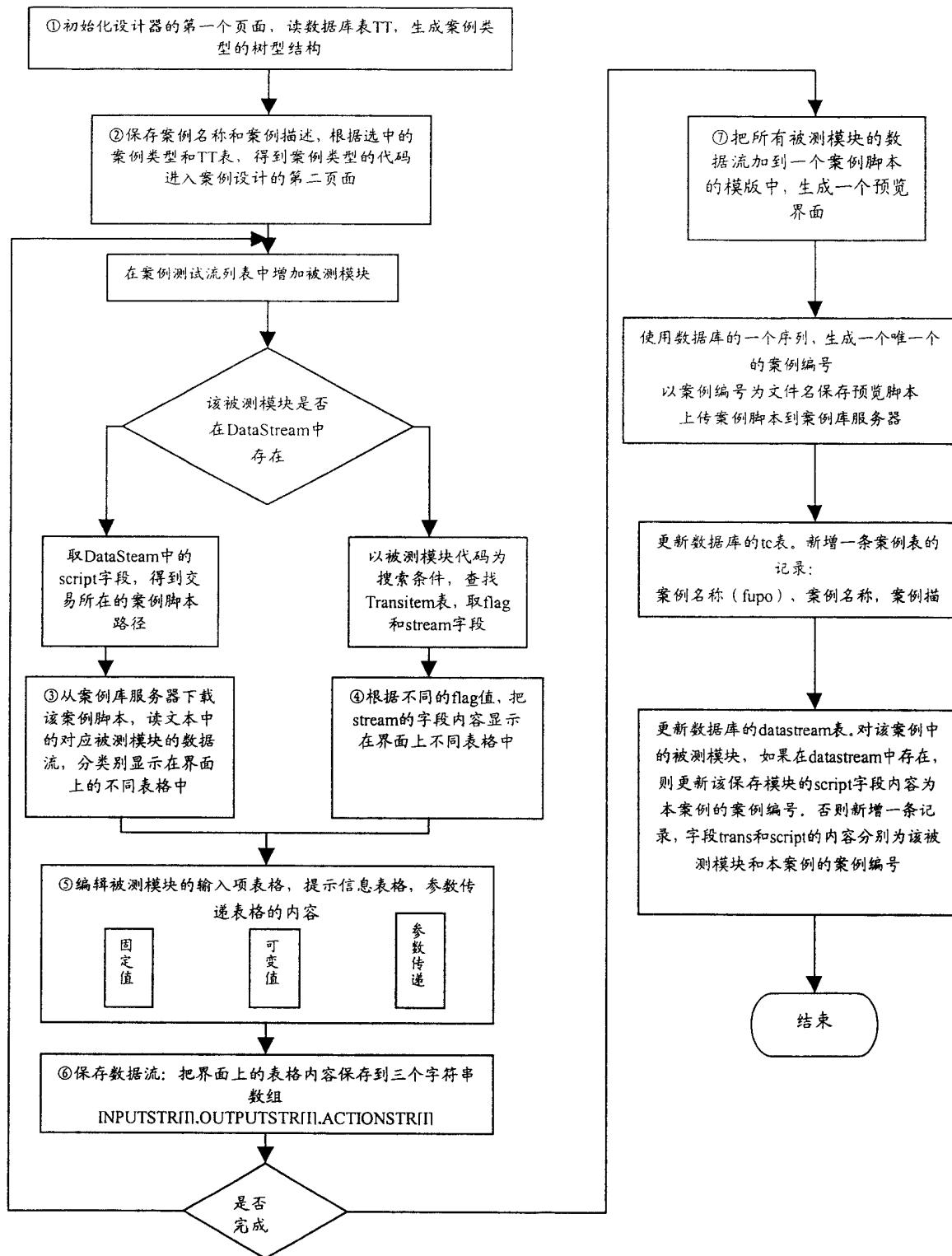


图 6

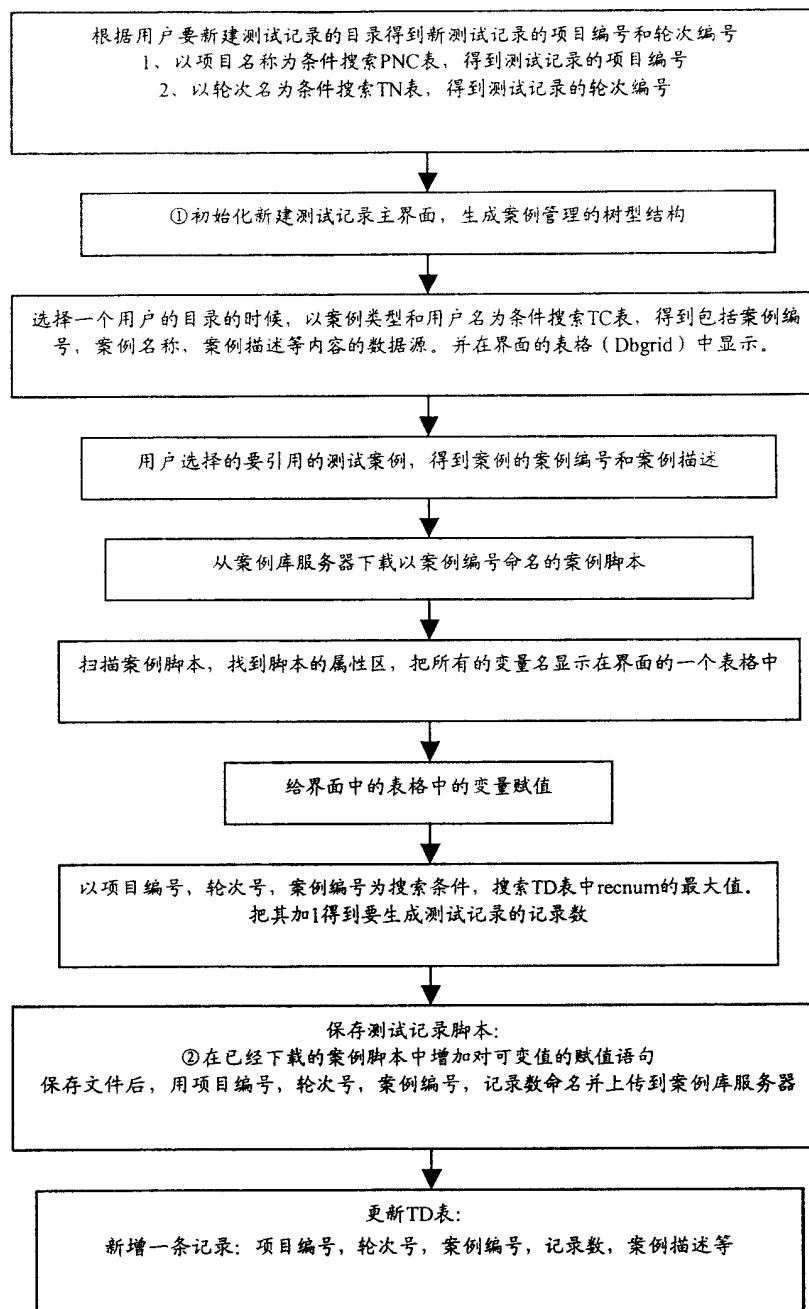


图 7

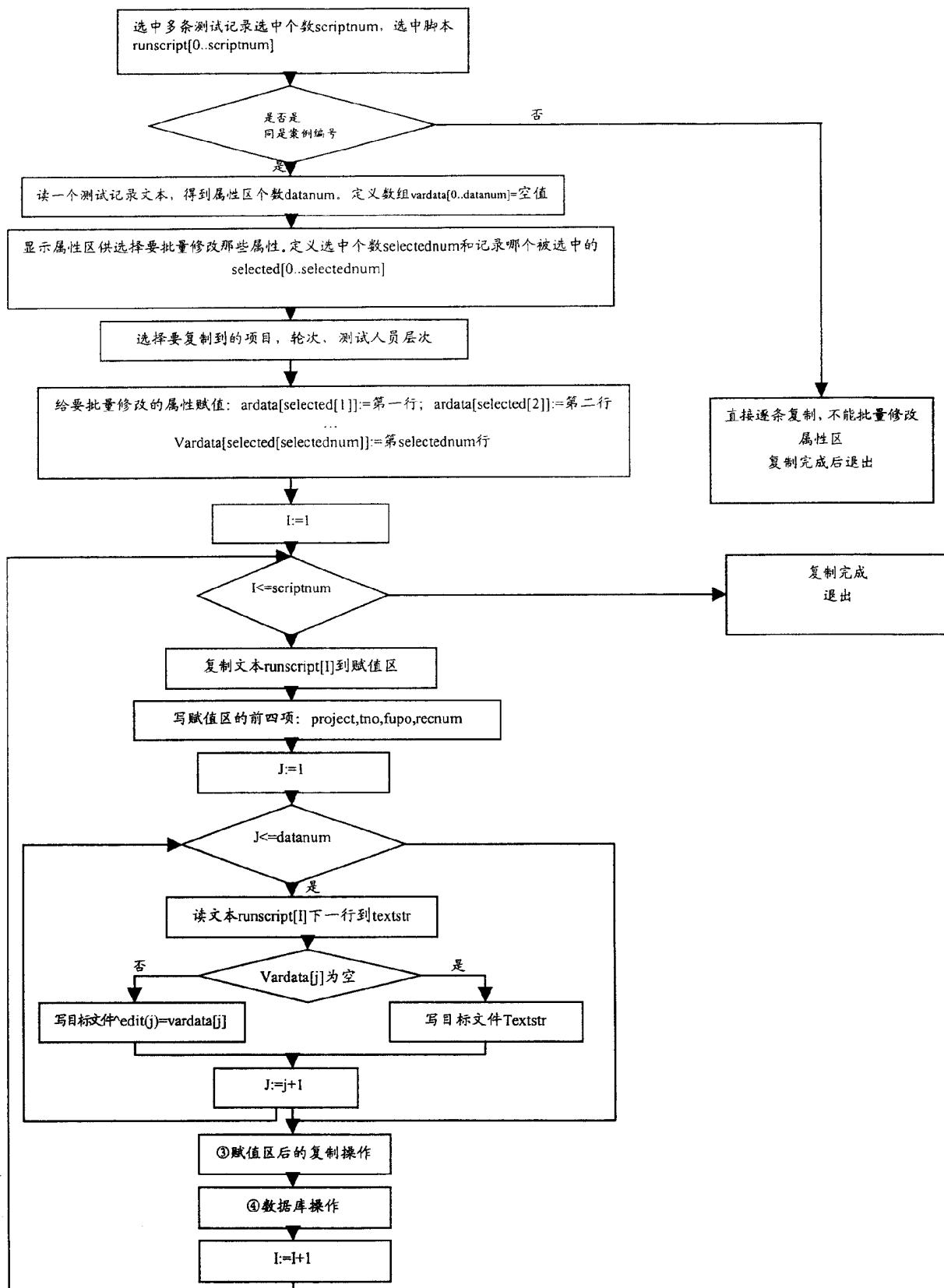


图 8

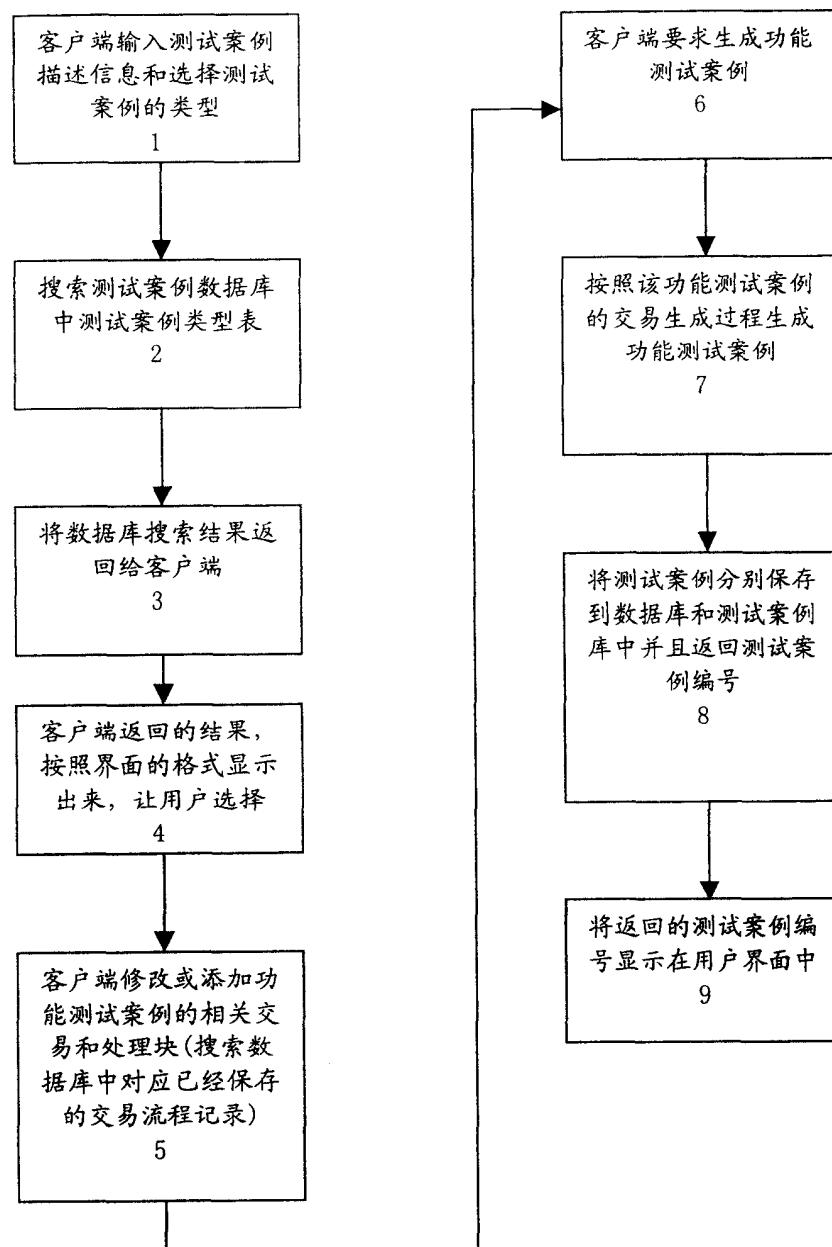


图 9

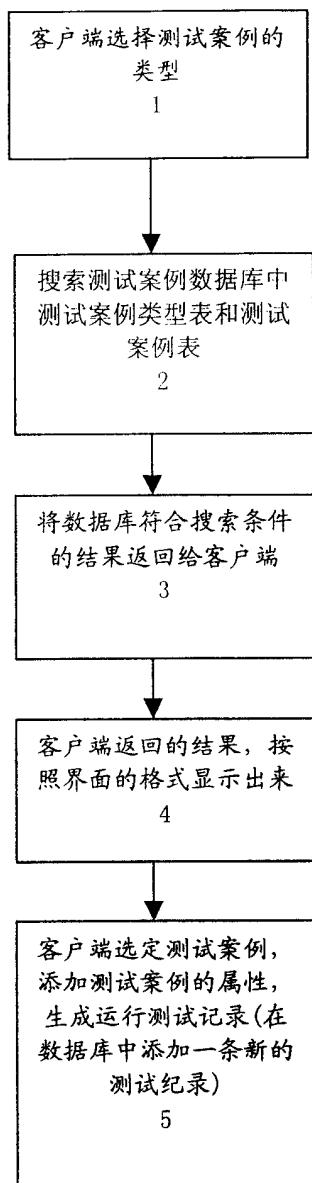


图 10

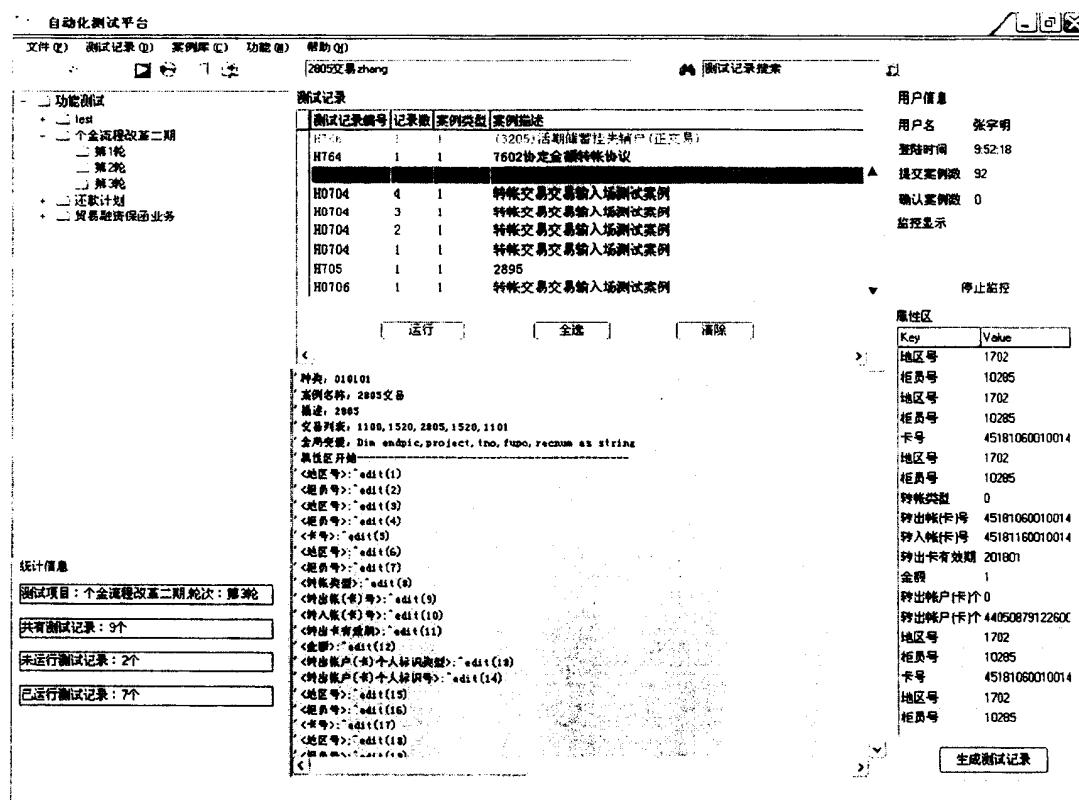


图 11

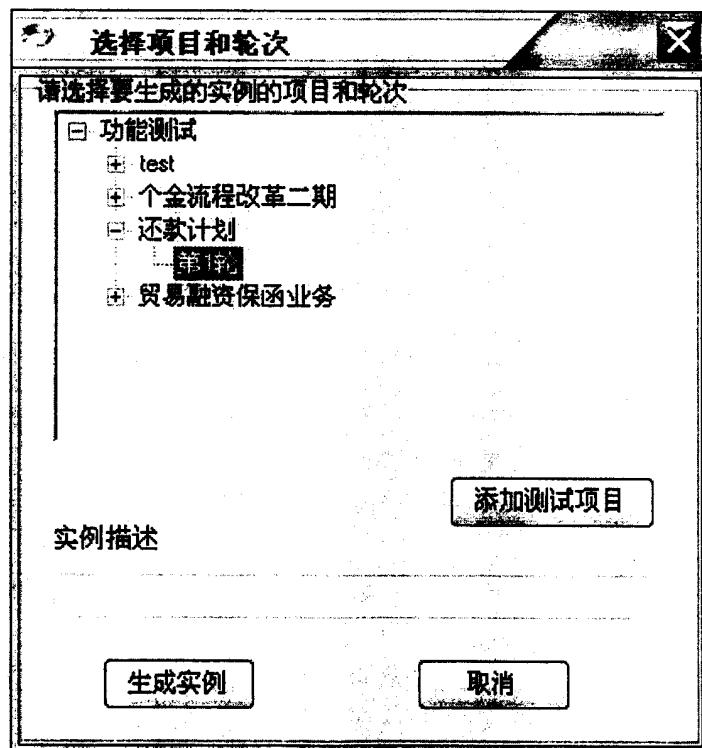


图 12

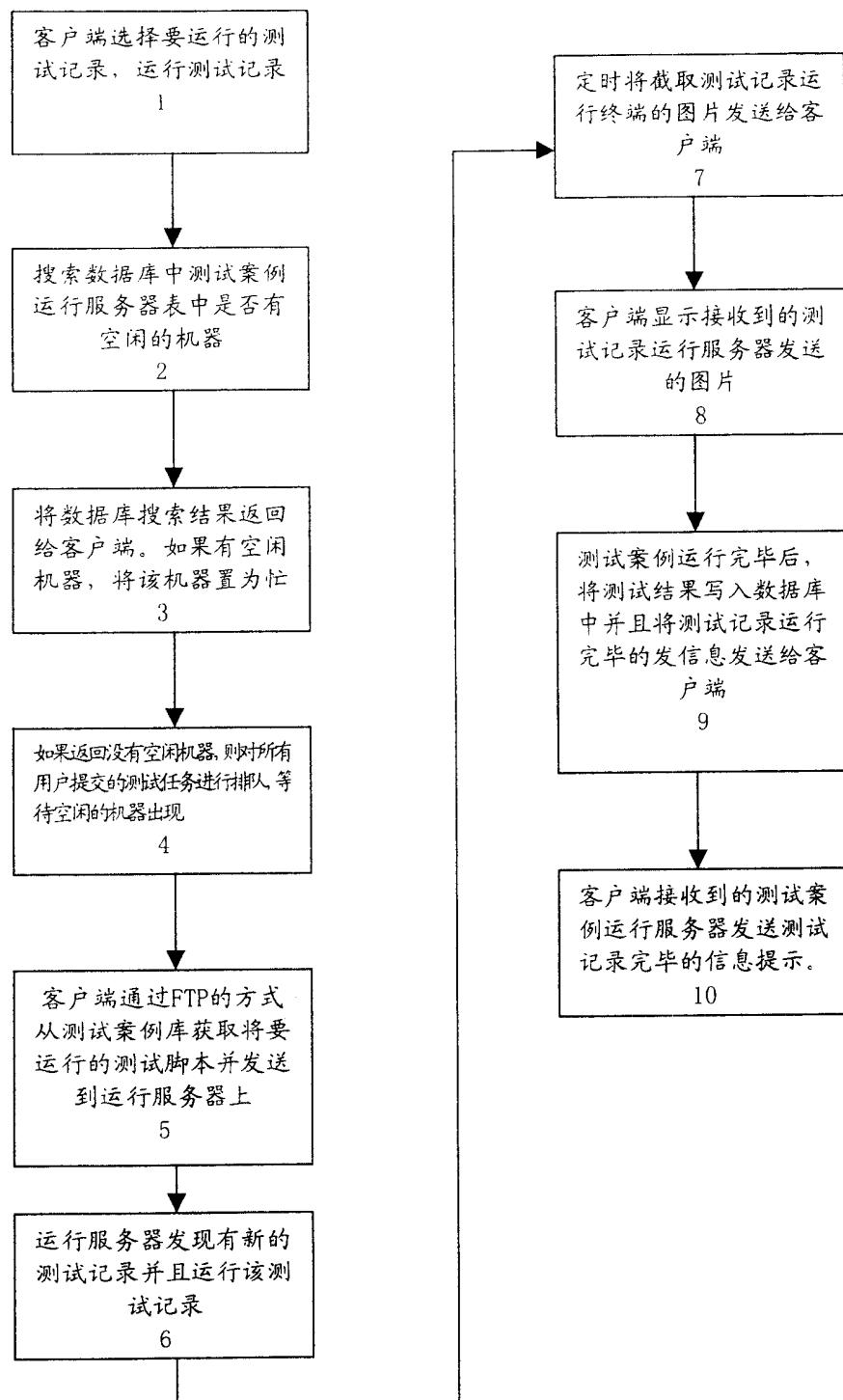


图 14

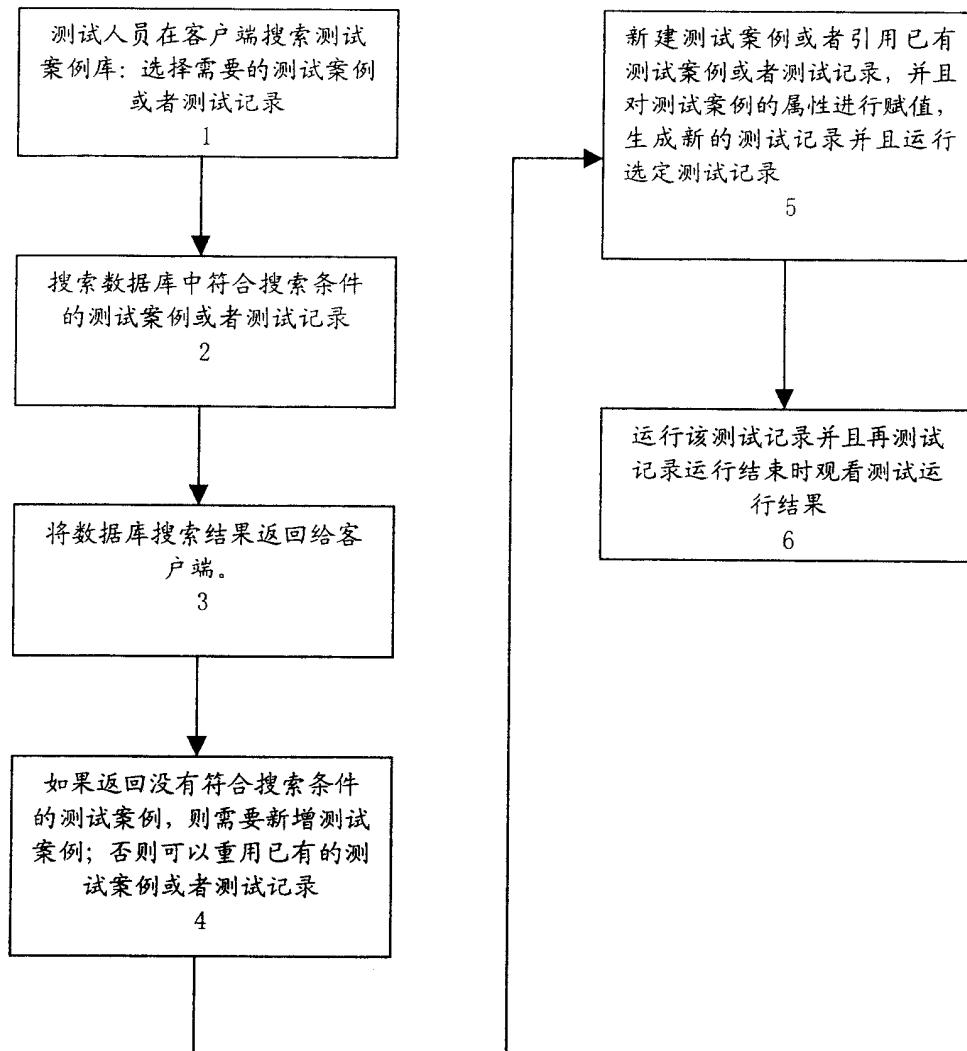


图 15

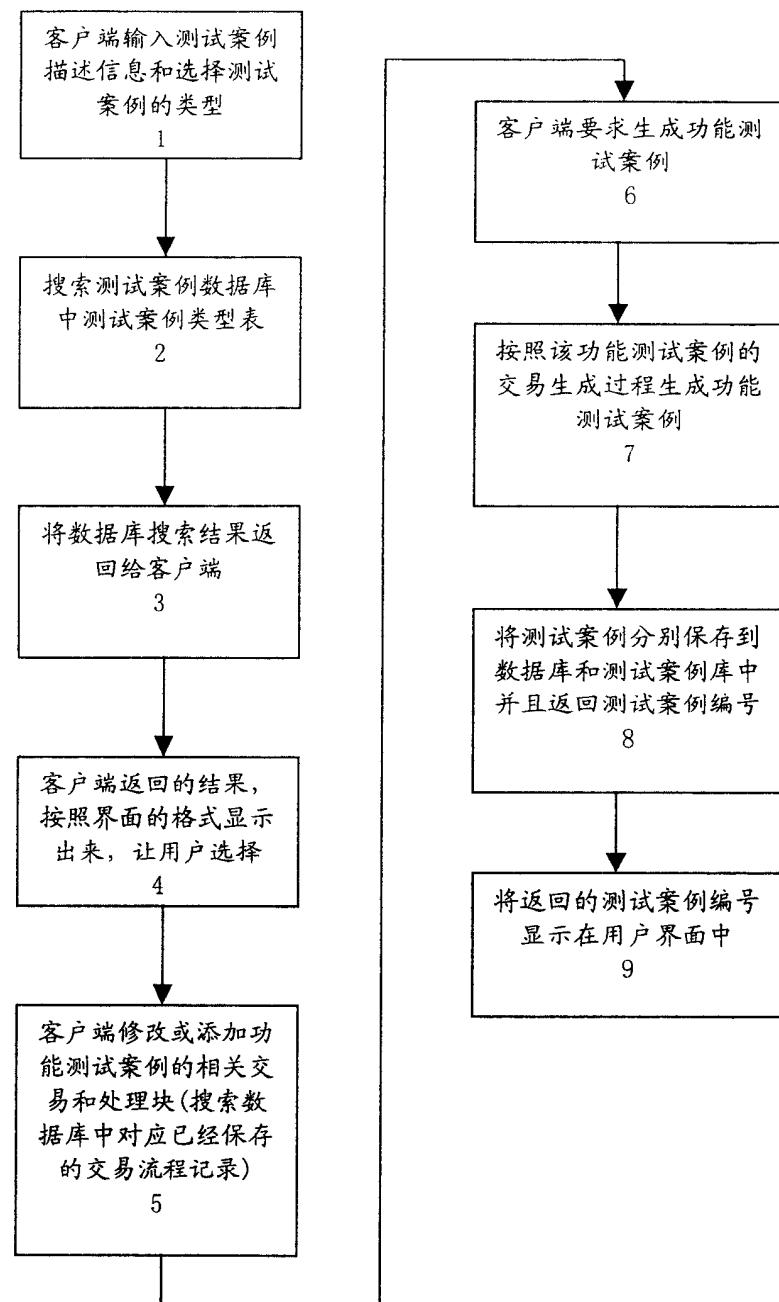


图 16

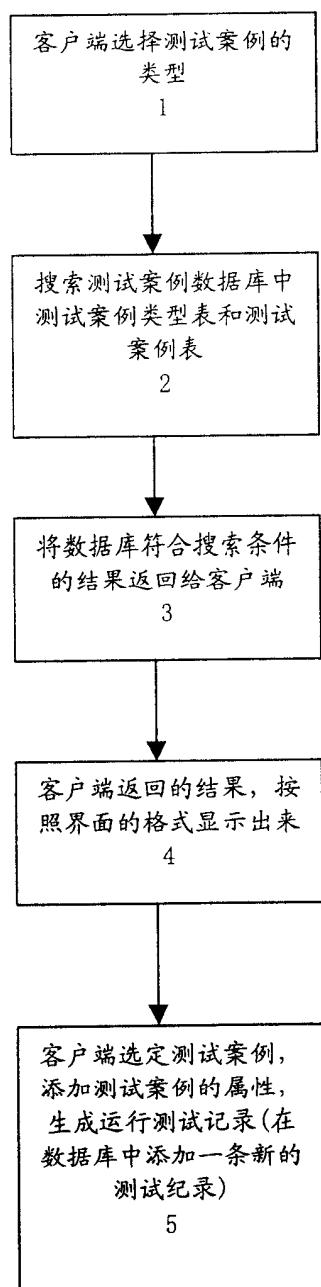


图 17

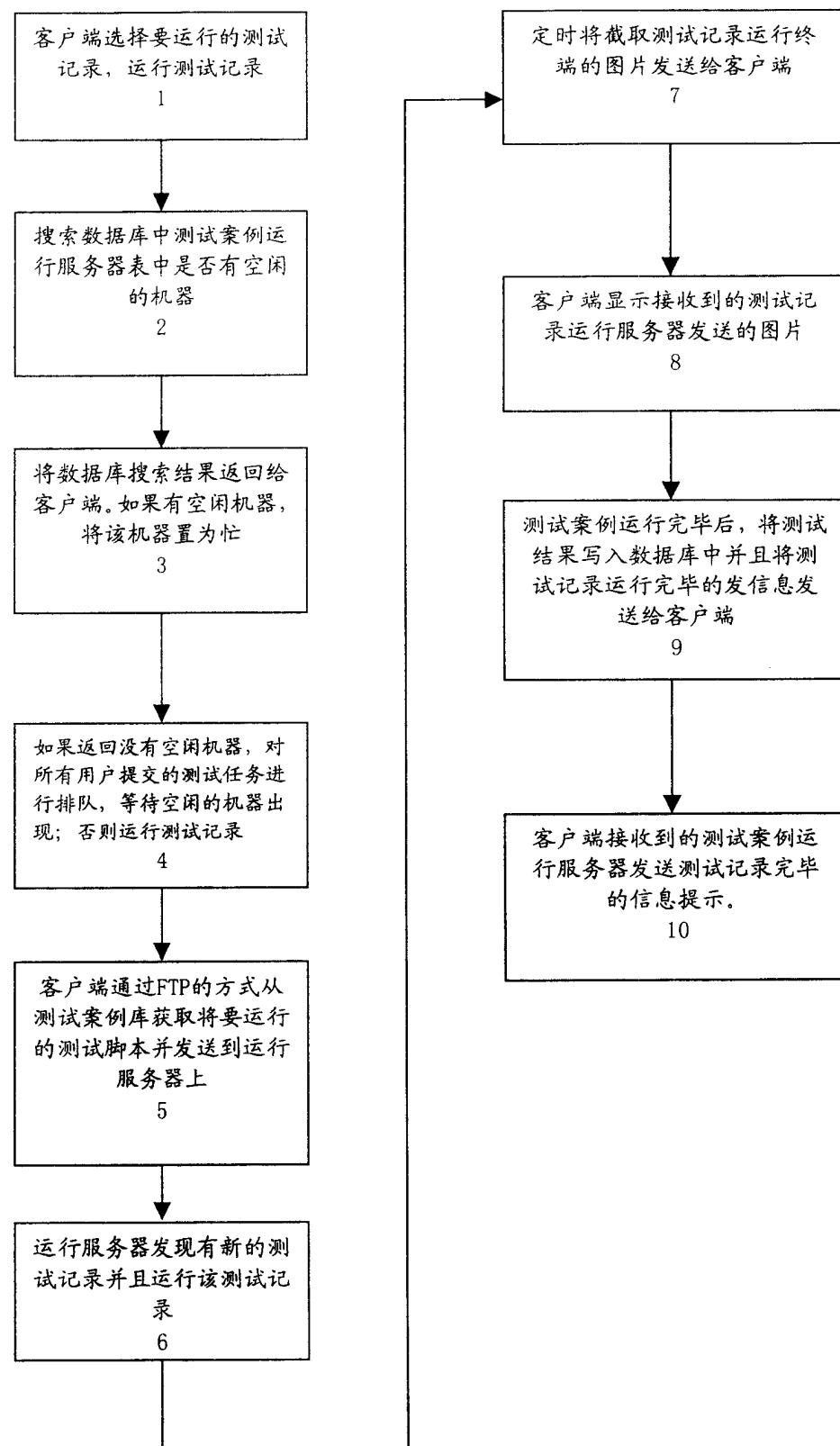


图 18