

[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 200510035197.3

[45] 授权公告日 2008年7月9日

[11] 授权公告号 CN 100401264C

[22] 申请日 2005.6.6

[21] 申请号 200510035197.3

[73] 专利权人 华为技术有限公司

地址 518129 广东省深圳市龙岗区布吉坂
田华为总部办公楼

[72] 发明人 林培兴 吕学

[56] 参考文献

US2004/0128652A1 2004.7.1

CN1527509A 2004.9.8

US6601018B1 2003.7.29

CN1499374A 2004.5.26

US5784553A 1998.7.21

US2003-0056150A1 2003.3.20

数据驱动自动化测试方法研究. 金大海,
宫云战. 装甲兵工程学院院报, 第18卷第2期.
2004

审查员 郑宁

[74] 专利代理机构 深圳市顺天达专利商标代理有
限公司

代理人 郭伟刚 蔡晓红

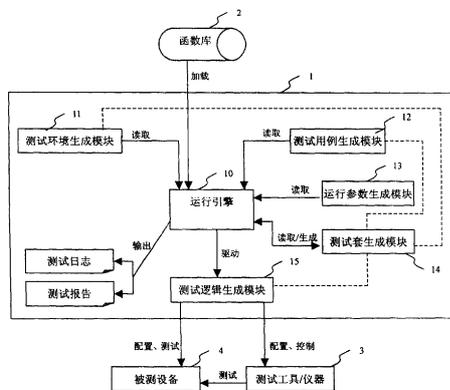
权利要求书3页 说明书10页 附图9页

[54] 发明名称

数据驱动的自动化测试系统及方法

[57] 摘要

一种数据驱动的自动化测试系统，其包括有测试工具和测试仪器、被测设备、测试平台及与该测试平台相连的函数库。函数库可包括产品函数库、测试工具函数库及测试仪器函数库；测试平台包括测试环境生成模块、测试用例生成模块、运行引擎、测试逻辑生成模块及测试套生成模块。本发明还提供一种数据驱动的自动化测试方法。本发明中所提供的系统及方法具备跨平台、跨工具平台移植；测试环境和测试逻辑可进行共享和重用；函数库的加载效率高；维护成本低等优点；且在本发明中利用测试套，提高了测试的效率。



1、一种数据驱动的自动化测试系统，其包括有测试工具和测试仪器及被测设备，其特征在于，进一步包括有用于驱动所述测试工具和测试仪器的测试平台，及与该测试平台相连、用于对所述测试工具和测试仪器及被测设备进行配置和控制的函数库；所述测试平台至少包括有一个运行引擎及与该运行引擎相连的测试环境生成模块、测试用例生成模块、测试逻辑生成模块及测试套生成模块；其中，

所述测试环境生成模块、测试逻辑生成模块、测试用例生成模块分别生成测试环境、测试逻辑、测试用例；

所述运行引擎从所述测试环境生成模块、测试逻辑生成模块及测试用例生成模块中读取测试环境、测试逻辑、测试用例，并加载所述函数库，根据选择的测试逻辑建立测试环境与测试逻辑的运行树，并按照所述测试环境与测试逻辑的运行树对被测试设备进行测试；

所述测试套生成模块根据测试用例的测试结果生成本次测试的测试套，所述测试套用于对测试用例进行分类，至少包含本次执行未通过的测试环境、测试逻辑、测试用例数据信息及测试脚本的运行信息。

2、如权利要求1所述的数据驱动的自动化测试系统，其特征在于：所述测试平台进一步包括有运行参数生成模块，用于生成提供测试日志输出条件及提供测试报告输出条件的运行参数；所述运行引擎在测试结束后根据所述运行参数生成模块所定义的输出条件输出测试日志和测试报告。

3、如权利要求2所述的数据驱动的自动化测试系统，其特征在于，所述测试报告用于表示本次测试中所有测试用例的通过情况统计，其包括测试数据统计、每一种测试结果的测试用例清单以及每一个测试用例的测试信息，该测试数据统计包括：测试持续时间、运行的测试逻辑脚本总数、运行的测试用例总数、执行结果分别为OK、POK、NG、NT的测试用例总数；该测试用例清单以及测试信息包括：测试逻辑编号、执行次数、自动化测试用例编号、系统测试

用例编号、执行时间、测试环境、自动化测试用例名称。

4、如权利要求 2 所述的数据驱动的自动化测试系统，其特征在于，所述测试日志用于记录本次测试过程中的测试用例的执行情况，其记录有本次测试的每个步骤、每个步骤的开始时间及用时信息，以及执行失败时的出错信息。

5、如权利要求 1 所述的数据驱动的自动化测试系统，其特征在于，所述函数库包括被测设备函数库、测试工具函数库及测试仪器函数库；

所述测试环境生成模块生成的测试环境用于建立或恢复本次测试的测试环境，所述测试环境定义有测试中所需的全局数据、测试环境下需要加载的函数库及测试环境的建立条件与恢复条件；

所述测试用例生成模块根据测试用例数据生成测试用例，该测试用例数据定义有测试逻辑所需的测试环境、测试逻辑运行所需加载的函数库及测试过程中测试逻辑所需的测试数据；

所述测试逻辑生成模块生成的测试逻辑用于定义测试流程，同步测试所用到的测试工具及测试仪器，并根据运行引擎提供的测试用例数据对被测设备进行测试。

6、如权利要求 5 所述的数据驱动的自动化测试系统，其特征在于，所述测试工具函数库及测试仪器函数库分别由所述测试工具和测试仪器提供，所述测试工具和测试仪器可被集成在测试平台中。

7、如权利要求 5 所述的数据驱动的自动化测试系统，其特征在于，所述被测设备函数库、测试工具函数库或测试仪器函数库根据测试环境及测试用例数据进行配置，由运行引擎在测试过程中加载，或由测试平台静态加载。

8、一种数据驱动的自动化测试方法，用于在测试平台中对被测设备进行自动化测试，其特征在于，包括如下步骤：

测试平台选择要运行的测试逻辑，并启动测试；

运行引擎根据运行参数生成模块所生成的运行参数进行初始化；

运行引擎根据选择的测试逻辑建立测试环境与测试逻辑的运行树；

运行引擎根据测试环境与测试逻辑的运行树执行本次测试；

其中，根据测试环境与测试逻辑的运行树执行本次测试进一步包括：

建立所述测试环境与测试逻辑的运行树中各层测试集所描述的环境以及最下层的测试逻辑的私有环境；

依次执行所述测试环境与测试逻辑的运行树中最下层的测试逻辑；

恢复所述最下层的测试逻辑的私有环境以及各层测试集所描述的环境。

9、如权利要求 8 所述的数据驱动的自动化测试方法，其特征在于，所述执行最下层的测试逻辑的步骤还包括：

加载所述测试逻辑所需的被测产品函数库；

加载所述测试逻辑所需的测试工具函数库；

取出每一条测试用例数据，执行所述测试逻辑；

启动执行所述测试逻辑的日志；

生成所述测试逻辑的测试用例执行报告。

10、如权利要求 8 所述的数据驱动的自动化测试方法，其特征在于，所述方法还包括：根据测试用例的测试结果生成本次测试未通过的测试用例组成的测试套。

11、如权利要求 8 所述的数据驱动的自动化测试方法，其特征在于，所述方法还包括：根据运行参数所定义的输出条件输出测试日志和测试报告。

数据驱动的自动化测试系统及方法

技术领域

本发明涉及自动化测试领域，更具体地，本发明涉及一种数据驱动的自动化测试系统及方法。

背景技术

在产品的研发过程中，当产品进入系统测试阶段后，为了保证产品的质量，需要对其在模拟实际的使用环境下的功能和性能进行全面的测试。根据在测试过程中所发现的产品的缺陷，开发人员对产品设计进行修正。为保证对原有错误的修改不会引入新的错误，需要多次对产品进行回归测试。产品的每次测试均需要由测试人员人工搭建环境来模拟实际的使用条件，我们称之为构造测试环境。为了保证测试结果的准确，还需要借助使用测试仪器/仪表来完成系统各项指标的测量。为了方便使用，大多数仪器/仪表提供有图形化操作界面，有些采用嵌入式操作系统，有些甚至提供可在计算机上安装的控制软件包。

而脚本语言（如TCL、PYTHON等）的诸如解释执行、可嵌入、可扩展的等的特性为实现自动化测试奠定了基础。由这些脚本语言所编写而成的产品（如电信设备或软件）不仅提供图形用户界面（Graphical User Interfaces, GUI），同时也提供命令行接口或人机接口，故可进行自动化测试。而在现有的自动化测试脚本设计中存在有诸多模式，如线性测试脚本设计模式、结构化测试脚本设计模式、共享测试脚本设计模式或关键字驱动测试脚本设计模式。

例如，基于Rational公司的Robot工具平台，就可以实现一种应用于GUI的自动化测试中的关键字驱动测试方法。该测试方法的整个过程所包含功能都是由关键字驱动，关键字控制了整个过程。请结合图1及图2所示，可更清楚地了解这种测试方法的实现机制和过程。在这种测试方法中，测试人员会首先使用类似Excel工作表的表格，以输入关键字（Key-Word）的方式建立测试用例。

如图1中，就是一种测试用例（如KeyWords_Web.xls）的示例，其由关键字及其参数组成，其中，第一列是关键字，说明要执行的动作，其包括诸如输入（ENTER）、动作（ACTION）及校验（VERIFY）等等；第二到第四列，是执行第一列的动作所必需的参数；而第五列标示了每一测试步骤是否通过。

测试用例在运行过程中，驱动脚本(Drive TSL script)运行初始化脚本(Logon TSL script)，为整个测试进行初始化工作，并载入控制文件（Run.txt），获取当前要运行的测试用例（*.xls）。

驱动脚本(Drive TSL script)在初始化工作完成后，启动控制脚本(Controller TSL script)，由控制脚本（Controller TSL script）来负责解析整个测试用例文件的关键字，根据这些关键字来调用与之对应的实现脚本（User Defined TSL script）。

实现脚本实际执行每个测试动作，并且使用其它列作为参数，执行完后将执行结果回传给控制脚本。当控制脚本执行到最后一行，整个测试用例就结束执行；直到控制文件（Run.txt）中所有用例文件（*.xls）运行完毕，本次测试过程结束。

但是现有这种关键字驱动的自动化测试方法存在有如下不足之处：

现有的技术主要应用于 GUI 自动化测试，并基于 Rational 公司的 Robot 工具平台，其跨平台（特指操作系统）及跨工具平台的移植性弱，不适合应用于通信设备类产品的自动化测试；测试数据和测试逻辑没有分离，且其对测试环境的建立和恢复的机制不成熟，测试数据一旦发生变化，将导致维护成本高；测试逻辑冗余，共享性较差；不能根据实际需要加载所需的产品函数库，移植成本高，灵活性较差；同时加载过多的产品函数库会导致整个自动化测试过程变得缓慢。

发明内容

本发明针对现有技术不足之处提供了一种数据驱动的自动化测试系统和方法，其测试用例可跨平台/跨工具平台移植；其测试环境可共享和重用；函数库

的加载效率高且具备配置/控制第三测试工具进行测试的能力。

为了解决上述问题，本发明采用的技术方案在于：提供一种数据驱动的自动化测试系统，其包括有测试工具和测试仪器及被测设备，其特征在于，进一步包括有用于驱动所述测试工具和测试仪器的测试平台，及与该测试平台相连、用于对所述测试工具和测试仪器及被测设备进行配置和控制的函数库；所述测试平台至少包括有一个运行引擎及与该运行引擎相连的测试环境生成模块、测试用例生成模块、测试逻辑生成模块及测试套生成模块；其中，所述测试环境生成模块、测试逻辑生成模块、测试用例生成模块分别生成测试环境、测试逻辑、测试用例；所述运行引擎从测试环境生成模块、测试逻辑生成模块、及测试用例生成模块中读取测试环境、测试逻辑、测试用例，并加载函数库，根据选择的测试逻辑建立测试环境与测试逻辑的运行树，并按照所述测试环境与测试逻辑的运行树对被测试设备进行测试；所述测试套生成模块根据测试用例的测试结果生成本次测试的测试套，所述测试套用于对测试用例进行分类，至少包含本次执行未通过的测试环境、测试逻辑、测试用例数据信息及测试脚本的运行信息。

其中，所述函数库包括有被测设备函数库、测试工具函数库及测试仪器函数库；所述测试环境生成模块生成的测试环境用于建立或恢复本次测试的测试环境，所述测试环境定义有测试中所需的全局数据、测试环境下需要加载的函数库及测试环境的建立条件与恢复条件；所述测试用例生成模块根据测试用例数据生成测试用例，该测试用例数据定义该测试逻辑所需的测试环境、该测试逻辑运行所需加载的函数库、及测试过程中测试逻辑所需的测试数据；所述测试逻辑生成模块用于生成本次测试的测试逻辑，该测试逻辑用于定义测试流程，同步测试所用到的测试工具及测试仪器，并根据运行引擎提供的测试用例数据，对被测设备进行测试；

其中，所述测试平台进一步包括有运行参数生成模块，用于生成提供测试日志输出条件及提供测试报告输出条件的运行参数；所述运行引擎在测试结束后根据运行参数生成模块所定义的输出条件输出测试日志和测试报告。

其中，所述测试报告用于表示本次测试中所有测试用例的通过情况统计，其可包括测试数据统计、每一种测试结果的测试用例清单以及每一个测试用例的测试信息，该测试数据统计包括：测试持续时间、运行的测试逻辑脚本总数、运行的测试用例总数、执行结果分别为 OK、POK、NG、NT 的测试用例总数；该测试用例清单以及测试信息包括：测试逻辑编号、执行次数、自动化测试用例编号、系统测试用例编号、执行时间、测试环境、自动化测试用例名称。

其中，所述测试日志用于记录本次测试过程中的测试用例的执行情况，其记录有本次测试的每个步骤、每个步骤的开始时间及用时信息，以及执行失败时的出错信息。

其中，所述测试工具函数库及测试仪器函数库分别由所述测试工具和测试仪器提供，所述测试工具和测试仪器可被集成在测试平台中。

其中，所述被测设备函数库、测试工具函数库或测试仪器函数库根据测试环境及测试用例数据进行配置，由运行引擎在测试过程中加载，或由测试平台静态加载。

本发明还提供一种数据驱动的自动化测试方法，用于在测试平台中对被测设备进行自动化测试，包括如下步骤：测试平台选择要运行的测试逻辑，并启动测试；运行引擎根据运行参数生成模块所生成的运行参数进行初始化；运行引擎根据选择的测试逻辑建立测试环境与测试逻辑的运行树；运行引擎根据测试环境与测试逻辑的运行树执行本次测试；其中，根据测试环境与测试逻辑的运行树执行本次测试进一步包括：建立所述测试环境与测试逻辑的运行树中各层测试集所描述的环境及最下层的测试逻辑的私有环境；依次执行最下层的测试逻辑；恢复最下层的测试逻辑的私有环境及各层测试集所描述的环境。

其中，执行最下层的测试逻辑的步骤还包括：加载测试逻辑所需的被测设备函数库；加载测试逻辑所需的测试工具函数库；取出每一条测试用例数据，执行测试逻辑；启动执行测试逻辑的日志；生成测试逻辑的测试用例执行报告。

其中，所述方法还包括：根据测试用例的测试结果生成本次测试未通过的测试用例组成的测试套。

其中，所述方法还包括：根据运行参数所定义的输出条件输出测试日志和测试报告。

本发明的有益效果在于：在本发明中，运行引擎通过诸如纯 TCL 脚本实现，具备跨平台、跨工具平台移植的能力；在自动化测试过程中，测试环境和测试逻辑可共享和重用，在不同的测试环境中可以驱动相同的测试逻辑；通过测试逻辑配置文件，可以指定要加载的测试工具/仪器/产品的函数库，使函数库的加载效率提高；测试逻辑与测试数据相分离，使得测试用例的维护成本更低；测试数据以表格形式表示，使得测试用例的扩展更为快捷；一次自动化执行完后，如果有未执行通过的测试用例，则由运行引擎生成一个测试套，包含本次执行未通过的测试环境、测试逻辑、测试用例数据等信息，这样下次执行时只需执行测试套，就可执行未通过的测试用例，提高了测试的效率。

附图说明

图 1 是现有技术中一种测试用例格式示意图。

图 2 是现有的一种关键字驱动的测试系统结构示意图。

图 3 是本发明数据驱动的自动化测试系统的结构示意图。

图 4 是本发明数据驱动的自动化测试系统的更详细的功能模块示意图。

图 5 是本发明数据驱动的自动化测试系统中测试报告格式的示意图。

图 6 是本发明数据驱动的自动化测试系统中测试日志格式示意图。

图 7 是本发明数据驱动的自动化测试方法的流程图。

图 8 是本发明数据驱动的自动化测试方法中测试环境及测试逻辑的运行树及运行说明示意图。

图 9 是图 8 中执行测试逻辑的流程图。

图 10 是本发明数据驱动的自动化测试系统的物理存储结构示意图。

具体实施方式

首先对本发明所涉及的技术名词说明如下：

测试集 (Test Cluster)，其是若干测试逻辑的相同测试环境配置和组网配置，

是测试环境在逻辑上的划分，是组成测试环境的最小实体。

测试逻辑 (Test Logic)，其是若干测试用例的相同的测试流程，即测试流程相同、测试数据不同的一组测试用例的脚本描述，测试逻辑可以在不同的测试环境下运行。

测试用例数据 (TestCaseData)，其由不同测试数据项的取值组成，测试数据项的取值的每一种组合，构成了不同测试用例的测试用例数据。

测试用例 (TestCase)，组成测试用例数据的测试数据项的取值的每一种组合，与测试用例数据对应的测试逻辑一起，称为一个测试用例。

测试套 (TestSuite)，是按产品特性、模块或其它 (基本功能、系统功能) 区分的，若干测试环境、测试逻辑、测试用例数据运行顺序、运行次数的集合。

工具命令语言 (tool command language, TCL)，一种解释执行的脚本语言。

如图 3 所示，是本发明中数据驱动的自动化测试系统的结构示意图。从中可以看出，本测试系统主要包括有测试平台 1、测试工具和测试仪器 3、被测设备 4。并包括有运行引擎 10 及与该运行引擎 10 连接的函数库 2。

其中，测试平台 1 可以是任意一个集成有脚本解释器 (如 TCL) 的测试平台，也可是脚本的 Shell (如 TCL Shell)，其可以管理、编辑测试脚本，并启动测试脚本开始测试。测试脚本包括测试环境、测试逻辑、测试用例数据。

测试工具和测试仪器 3 为第三方程序，其用于测试被测设备 4，该两者还可以集成到测试平台 1 中，接受测试平台 1 的集中管理。

运行引擎 10 在逻辑上属于测试平台 1，完成测试平台 1 与测试脚本之间的适配，根据测试脚本中的配置，加载所需函数库 2，并控制整个测试过程；

在测试过程中，测试脚本建立测试环境，并通过脚本控制第三方测试工具和测试仪器 3 来测试被测设备 4；在测试过程中，测试工具 3 之间、测试仪器 3 之间以及测试工具 3 与测试仪器 3 之间的协调通过测试脚本来同步。

如图 4 所示，是本发明中数据驱动的自动化测试系统的更详细的功能模块框图。其包括有测试平台 1、及分别与测试平台 1 相连的函数库 2、测试工具/仪器 3 及被测设备 4。其中测试平台 1 又进一步包括有运行引擎 10、测试环境

生成模块 11、测试用例生成模块 12、运行参数生成模块 13、测试套生成模块 14 及测试逻辑生成模块 15，其中，测试套生成模块 14 为可选模块。下面分别对每一功能模块进行说明：

函数库 2 可包括有多种函数库，如被测设备函数库、测试工具函数库及测试仪器函数库等。其中，被测设备函数库用于配置和控制被测设备；测试工具函数库用于配置和控制测试工具；测试仪器函数库用于配置和控制测试仪器。这些函数库根据测试环境及测试用例数据进行配置，由运行引擎在测试过程中加载，也可配置成由测试平台静态加载。

而测试工具和测试仪器 3，其可提供上述的测试工具函数库或测试仪器函数库，接受测试环境及测试逻辑的配置和控制，并完成对被测设备 4 的测试。

被测设备 4 为测试过程中被测试的设备，其接受测试环境的配置和控制，接受测试逻辑的配置、控制和测试。

测试环境生成模块 11 主要用于完成本次测试的测试环境的建立及恢复。该测试环境包括被测设备及周边设备 4、测试工具/测试仪器 3；其定义测试中所需的全局数据（变量）；定义测试环境下需要加载的函数库；定义测试环境的建立条件（如被测设备及周边设备、测试工具/测试仪器的初始化等）；定义测试环境的恢复条件（如被测设备及周边设备 4 资源的释放，测试工具/测试仪器 3 等公共资源的释放等）。

测试用例生成模块 12 主要用于根据测试用例数据生成测试用例，其中，该测试用例数据定义测试逻辑需要在哪些测试环境运行；定义测试逻辑运行所必须的特定测试环境；定义测试逻辑运行所必须加载的特定的函数库；定义在测试过程中，测试逻辑所需的测试数据；用于运行引擎 10 驱动测试逻辑完成测试。

运行参数生成模块 13 主要用于生成运行参数，该运行参数提供测试日志输出条件及提供测试报告输出条件。

测试逻辑生成模块 15 用于生成本次测试的测试逻辑，该测试逻辑用于定义测试流程；在测试过程中，同步测试所用到的测试工具及测试仪器 3；根据运行引擎 10 提供的测试用例数据，完成对被测设备 4 的测试。

测试套生成模块 14 用于生成本次测试的测试套。所述测试套包含本次执行未通过的测试环境、测试逻辑、测试用例数据等信息。测试套是测试脚本在逻辑上的划分，可归纳为基本功能测试套、特性测试套等等。在测试套中仅包含内容：要运行哪些测试脚本（测试环境、测试逻辑、测试用例数据），以及这些测试脚本（测试环境、测试逻辑、测试用例数据）的运行信息（如运行顺序、运行次数）。

运行引擎 10，其在测试平台 1 完成测试逻辑、测试环境、测试用例数据的编辑，并启动当前所选择的测试逻辑的测试之后，控制测试脚本的执行。运行引擎会从测试环境生成模块 11、测试逻辑生成模块 15、及测试用例生成模块 12 中读取测试环境、测试逻辑、测试用例数据（并在测试开始前，建立测试环境；测试结束后，恢复测试环境）；加载函数库，自动识别函数库的版本号，更新函数索引文件；测试过程中，根据测试用例数据，驱动测试逻辑对被测设备 4 进行测试；在测试过程中，根据运行参数，输出测试报告和测试日志；根据测试用例的测试结果，驱动测试套生成模块 14 生成本次测试未通过的测试用例组成的测试套，方便后续对未通过的测试用例的再次测试。

测试报告是在测试结束后，由运行引擎 10 提供。该测试报告用于表示本次测试结束后，本次测试中所有测试用例的通过情况的统计。如图 5 所示，是本发明中的一种测试报告的示例。从中可以看出，该测试报告列出了测试数据统计，该测试数据统计包括诸如：测试持续时间、运行的测试逻辑脚本总数、运行的测试用例总数、执行结果分别为 OK、部分 OK (partial OK, POK)、不好 (Not Good, NG)、未测试 (Not Test, NT) 等的测试用例总数。并列出了上述不同执行结果的详细信息。

在表示测试过程中，由运行引擎 10 记录当前运行的测试用例的执行情况；测试结束后，由运行引擎 10 输出完整的测试用例的执行情况（即测试日志）；如图 6 所示，是本发明中一种测试日志的格式示例，其详细地记录了本次测试的每个步骤，及每个步骤的开始时间及用时等信息。

如图 7 所示，是本发明数据驱动自动化测试的方法的流程图。在步骤 S70

中，测试平台选择要运行的测试逻辑，并启动测试。在步骤 S71 中，运行引擎根据运行参数生成模块所生成的运行参数完成初始化工作。在步骤 S72 中，运行引擎根据本次运行的测试逻辑，建立测试环境与测试逻辑的运行树，该测试环境及测试逻辑的运行树及运行说明可参见图 8 所示。在步骤 S74 中，运行引擎根据测试环境与测试逻辑的运行树开始执行本次测试。首先在步骤 S76 中，建立各层的测试集所描述的环境，参照图 8 所示的运行树，即依次：执行功能 1 建立环境 TestCluster1；执行功能 2，建立环境 TestCluster11；执行功能 3：建立环境 TestCluster111；执行功能 4：建立测试逻辑 TestLogic1 的私有环境。并在步骤 S78 中，依次执行最下层的测试逻辑，即执行 TestLogic1；执行功能 D：恢复测试逻辑 TestLogic1 的私有环境；同理，继续执行功能 5：建立测试逻辑 TestLogic2 的私有环境；执行 TestLogic2；执行功能 E：恢复测试逻辑 TestLogic2 的私有环境；执行功能 6：建立测试逻辑 TestLogic3 的私有环境并执行 TestLogic3；执行功能 F：恢复测试逻辑 TestLogic3 的私有环境。然后，在步骤 S79 中，反方向恢复各层测试集所描述的环境。即执行功能 C：恢复环境 TestCluster111；执行功能 7：建立环境 TestCluster112，并执行在测试集 TestCluster112 下运行的测试逻辑；执行功能 G：恢复环境 TestCluster112；执行功能 8：建立环境 TestCluster113，并执行在测试集 TestCluster113 下运行的测试逻辑；执行功能 H：恢复环境 TestCluster113；执行功能 B：恢复环境 TestCluster11；执行功能 9：建立环境 TestCluster12，并执行 TestCluster12 的测试逻辑；执行功能 I：恢复环境 TestCluster12。执行功能 A：恢复环境 TestCluster1。

如图 9 所示，是本发明图 8 中执行最下层测试逻辑的流程图。在本图 9 中，是以执行 TestLogic1 为例进行说明。在步骤 S780 中加载测试逻辑所需的被测设备函数库；在步骤 S782 中，加载测试逻辑所需的测试工具函数库；在步骤 S784 中，取出每一条测试用例数据，执行测试逻辑；在步骤 S786 中，启动执行测试逻辑的日志；在步骤 S788 中，生成测试逻辑的测试用例执行报告。最下层的其他测试逻辑（诸如 TestLogic2、TestLogic2.... TestLogic5）的过程均与此类似。

其中，从运行树中可以看出，测试逻辑 TestLogic2 在两个环境下执行，达

到测试过程中测试逻辑共享的目的，在不同环境下的不同的测试结果通过测试用例数据中相应的数据项来表示。

如图 10 所示，是本发明中数据驱动自动化测试的物理存储结构示意图。华为公司的数据驱动自动化测试的存储结构由五部分组成：函数库 (Lib)，测试脚本库 (Scripts)，测试套 (TestSuites)，测试报告 (Report)，测试日志 (Log)；

其中，函数库又可分为按测试工具、测试仪器划分的公共函数库及按被测设备划分的被测设备函数库，每个被测设备对应一个函数库。

测试脚本库也是按被测设备进行划分；该测试脚本库中存放有被测设备的测试环境、测试逻辑及测试用例数据；及存放有数据驱动自动化测试体系的环境参数配置文件。

测试套库也是按被测设备划分；存放有被测设备的测试套。

测试报告目录也是按被测设备划分；每启动一次测试过程，产生本次测试过程的测试报告，记录本次测试过程中运行的测试用例的通过情况。

测试日志目录也是按被测设备划分；每启动一次测试过程，产生本次测试过程的测试日志，记录本次测试过程中测试用例的运行过程。

在本发明中，运行引擎的实现除了 TCL 实现之外，也可以通过其它脚本语言（如 Python 等）或编译语言（如 C/C++，Pascal 等）实现；数据驱动自动化测试体系的物理存储结构也可通过数据库（如 Oracle、SQL Server 等）来实现。

在本发明中，运行引擎通过诸如纯 TCL 脚本实现，具备跨平台、跨工具平台移植的能力；自动化测试过程中，测试环境和测试逻辑可进行共享和重用，在不同的测试环境中可以驱动相同的测试逻辑；通过测试逻辑配置文件，可以指定要加载的测试工具/仪器/被测设备的函数库，使函数库的加载效率提高；测试逻辑与测试数据相分离，使得测试用例的维护成本更低；测试数据以表格形式表示，使得测试用例的扩展更为快捷；一次自动化执行完后，如果有未通过的测试用例，则由运行引擎生成一个测试套，包含本次执行未通过的测试环境、测试逻辑、测试用例数据等信息，这样下次执行时只需执行该测试套，就可执行未通过的测试用例，提高了测试的效率。

列1 关键字 (Key_word)	列2 域/屏幕名称 (Field/Screen Name)	列3 输入/校验数据 (Input/Verification Data)	列4 注释 (Comment)	列5 通过/失败 (Pass/Fail)
开始测试 (Start_Test)	屏幕 (Screen)	主菜单 (Main Menu)	校验开始指针 (Verify starting Point)	
输入 (Enter)	选项 (Selection)	3	选择支付项 (Select Payment Option)	
动作 (Action)	按键 (Press_Key)	F4	访问支付屏幕 (Access Payment Screen)	
校验 (Verify)	屏幕 (Screen)	支付邮寄 (Payment Posting)	校验支付授权 (Verify Payment Accessed)	
输入 (Enter)	支付额 (Payment Amount)	125.87	输入支付数据 (Enter Payment Data)	
	支付方式 (Payment Method)	支票 (Check)		
动作 (Action)	按键 (Press_Key)	F9	处理支付 (Process Payment)	
校验 (Verify)	屏幕 (Screen)	支付屏幕 (Payment Screen)	校验屏幕保留 (Verify screen Remains)	
校验数据 (Verify_Data)	支付额 (Payment Amount)	\$125.87	校验更新数据 (Verify updated Data)	
	当前结余 (Current Balance)	\$1,309.77		
	状态消息 (Status Message)	已过帐 (Payment Posted)		
动作 (Action)	按键 (Press_Key)	F12	返回主菜单 (Return Main Menu)	
校验 (Verify)	屏幕 (Screen)	主菜单 (Main Menu)	校验返回菜单 (Verify return to Menu)	

图 1

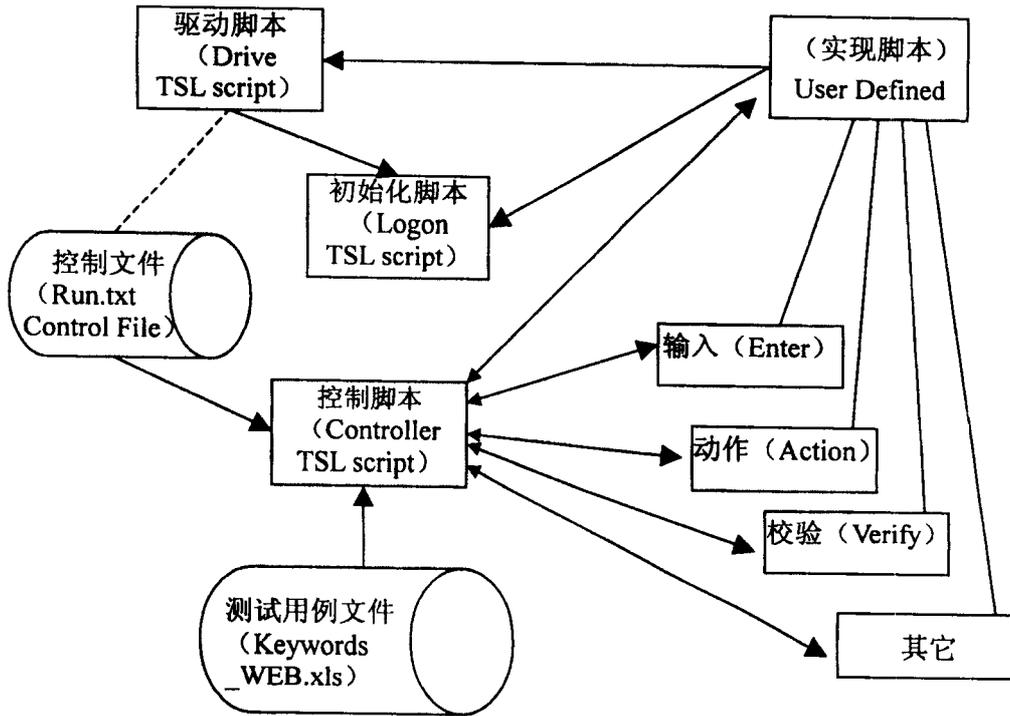


图 2

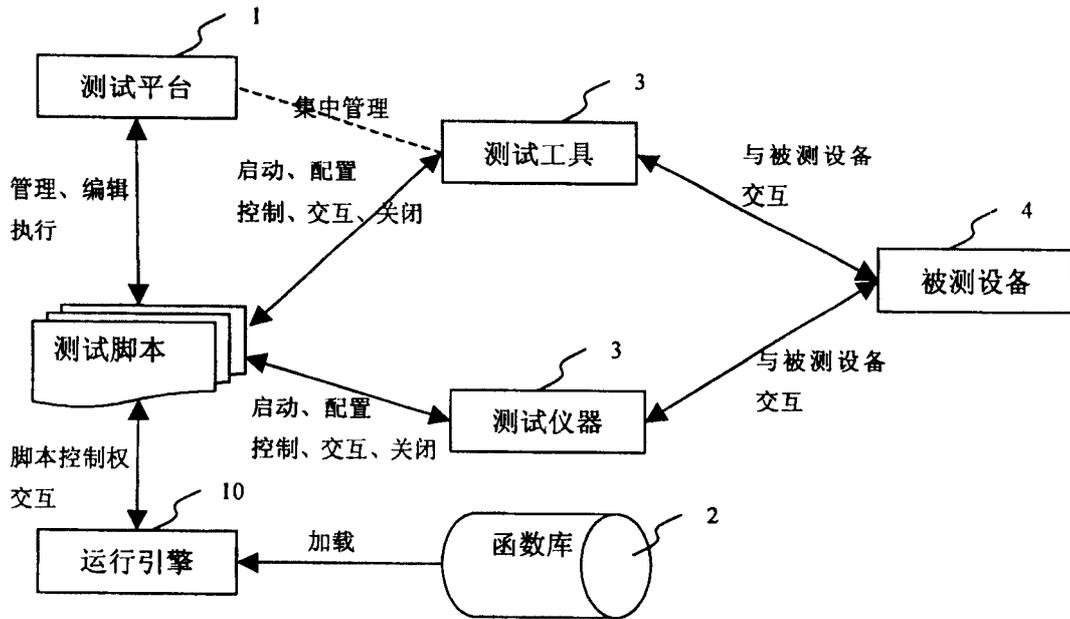


图 3

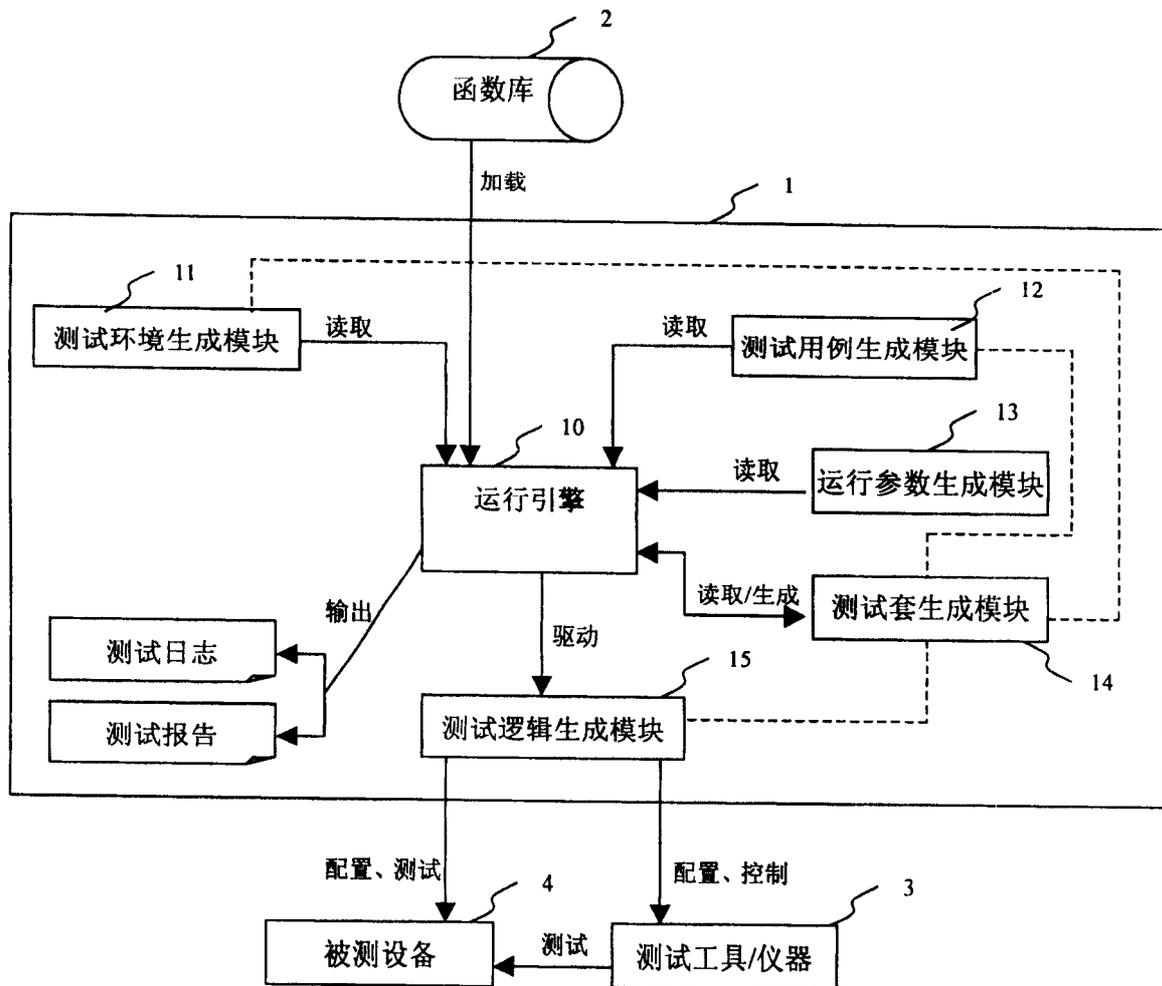


图 4

一、测试数据统计:

1. 测试持续时间: 22
2. 运行的测试逻辑脚本总数: 5
3. 运行的测试逻辑总数: 5
4. 运行的测试用例总数: 10
5. 执行结果为OK的测试用例总数: 2
6. 执行结果为POK的测试用例总数: 2
7. 执行结果为NG的测试用例总数: 5
8. 执行结果为NT的测试用例总数: 1

二、测试用例执行结果详细信息

1. 执行结果为OK的测试用例:

testLogicNo	times	testCaseNo	excuteTime (Seconds)	correspondTestCaseNo	testClusters	testCaseName
TestLogic_01_03	1	TestLogic_01_03_01	< 1		TemplateTc1/TemplateTc11	HelloWorld1
TestLogic_02_03	1	TestLogic_02_03_01	< 1		TemplateTc1/TemplateTc11	HelloWorld2

2. 执行结果为POK的测试用例:

testLogicNo	times	testCaseNo	excuteTime (Seconds)	correspondTestCaseNo	testClusters	testCaseName
TestLogic_01_03	1	TestLogic_01_03_02	< 1		TemplateTc1/TemplateTc11	HelloWorld3
TestLogic_02_03	1	TestLogic_02_03_02	< 1		TemplateTc1/TemplateTc11	HelloWorld4

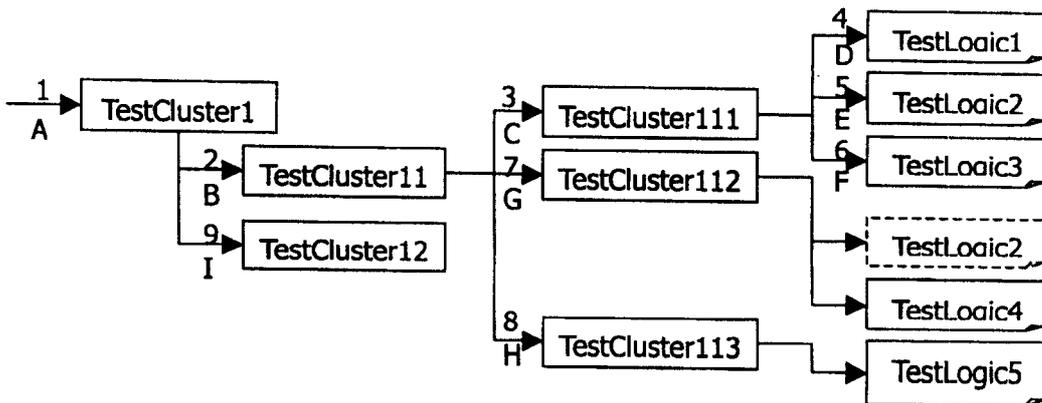
3. 执行结果为NG的测试用例:

testLogicNo	times	testCaseNo	excuteTime (Seconds)	correspondTestCaseNo	testClusters	testCaseName
TestLogic_01_03	1	TestLogic_01_03_03	< 1		TemplateTc1/TemplateTc11	HelloWorld5
TestLogic_02_01	1	TestLogic_02_01_01	0		TemplateTc1/TemplateTc11	HelloWorld6
TestLogic_02_03	1	TestLogic_02_03_03	< 1		TemplateTc1/TemplateTc11	HelloWorld7
TestLogic_01_02	1	TestLogic_01_02_01	< 1		TemplateTc2/TemplateTc21	HelloWorld8
TestLogic_02_02	1	TestLogic_02_02_01	< 1		TemplateTc2/TemplateTc21	HelloWorld9

4. 执行结果为NT的测试用例:

testLogicNo	times	testCaseNo	excuteTime	correspondTestCaseNo	testClusters	testCaseName
TestLogic_01_01	0					

图 5



- 1: 执行 TestCluster1 的环境建立配置命令、脚本;
 - 2: 执行 TestCluster11 的环境建立配置命令、脚本;
 - 3: 执行 TestCluster111 的环境建立配置命令、脚本;
 - 4: 执行 TestLogic1 的环境建立配置命令、脚本;
 - 5: 执行 TestLogic2 的环境建立配置命令、脚本;
 - 6: 执行 TestLogic3 的环境建立配置命令、脚本;
 - 7: 执行 TestCluster112 的环境建立配置命令、脚本;
 - 8: 执行 TestCluster113 的环境建立配置命令、脚本;
 - 9: 执行 TestCluster12 的环境建立配置命令、脚本;
- A: 执行 TestCluster1 的环境恢复配置命令、脚本;
 - B: 执行 TestCluster11 的环境恢复配置命令、脚本;
 - C: 执行 TestCluster111 的环境恢复配置命令、脚本;
 - D: 执行 TestLogic1 的环境恢复配置命令、脚本;
 - E: 执行 TestLogic2 的环境恢复配置命令、脚本;
 - F: 执行 TestLogic3 的环境恢复配置命令、脚本;
 - G: 执行 TestCluster112 的环境恢复配置命令、脚本;
 - H: 执行 TestCluster113 的环境恢复配置命令、脚本;
 - I: 执行 TestCluster12 的环境恢复配置命令、脚本;

- a: 执行 TestLogic1
- b: 执行 TestLogic2
- c: 执行 TestLogic3
- d: 执行 TestCluster112 的测试逻辑;
- e: 执行 TestCluster113 的测试逻辑;
- f: 执行 TestCluster12 的测试逻辑;

- i、加载 TestLogic 所需的产品函数库;
- ii、加载 TestLogic 所需的工具函数库;
- iii、取出每一条测试用例, 执行 TestLogic
- iv、记录执行 TestLogic 的日志
- v、统计 TestLogic 的测试用例执行报告

图 8

```

[+++ The D:\Studio.auto\autotest\ main .tcl begin to execute 2005-01-20 12:01:49 +++]
->TestApp: 载入测试包TestLib,版本号: 02.01.7
->2005-01-20 12:01:50-(TestLib-)加载函数库: TrReport,版本号: 1.5
->2005-01-20 12:01:51-(TestLib.RunEngine):请稍后...正在加载测试集
->2005-01-20 12:01:53-(TestLib):+++++++
->2005-01-20 12:01:53-(TestLib):建立测试集: TemplateTc2
->2005-01-20 12:01:53-(TestLib):正在加载测试集TemplateTc2所需的函数库
->2005-01-20 12:01:53-(TestLib):执行测试集TemplateTc2的环境TERM1的建立配置脚本
->2005-01-20 12:01:53-(TemplateTc2):execute TemplateTc2 TERM1 SetupScripts
->2005-01-20 12:01:53-(TestLib):+++++++
->2005-01-20 12:01:53-(TestLib):建立测试集: TemplateTc1
->2005-01-20 12:01:53-(TestLib):正在加载测试集TemplateTc1所需的函数库
->2005-01-20 12:01:53-(TestLib):执行测试集TemplateTc1的环境TERM1的建立配置脚本
->2005-01-20 12:01:53-(TemplateTc1):execute TemplateTc1 TERM1 SetupScripts
->2005-01-20 12:01:53-(TestLib):+++++++
->2005-01-20 12:01:53-(TestLib):建立测试集: TemplateTc22
->2005-01-20 12:01:53-(TestLib):正在加载测试集TemplateTc22所需的函数库
->2005-01-20 12:01:53-(TestLib):执行测试集TemplateTc22的环境TERM1的建立配置脚本
->2005-01-20 12:01:53-(TemplateTc22):execute TemplateTc22 TERM1 SetupScripts
->2005-01-20 12:01:53-(TestLib):加载本次测试逻辑TestLogic_02_02的测试逻辑配置文件.....请稍候.....
->2005-01-20 12:01:53-(TestLib):正在加载测试逻辑TestLogic_02_02运行所需的函数库...
->2005-01-20 12:01:53-(TestLib):加载函数库: D:/Studio.auto/autotest/Lib/Template
->2005-01-20 12:01:54-(TestLib):加载函数库: D:/Studio.auto/autotest/Lib/Template/Agent
->2005-01-20 12:01:54-(TestLib):加载函数库: D:/Studio.auto/autotest/Lib/Template/Common
->2005-01-20 12:01:54-(TestLib):加载函数库: D:/Studio.auto/autotest/Lib/Template/Specialty1
->2005-01-20 12:01:54-(TestLib):加载函数库: D:/Studio.auto/autotest/Lib/Template/Specialty2
->2005-01-20 12:01:54-(TestLib):加载函数库: D:/Studio.auto/autotest/Lib/Template/Specialty3
->2005-01-20 12:01:54-(TestLib):加载函数库: D:/Studio.auto/autotest/Lib/Template/Sys
->2005-01-20 12:01:54-(TestLib):加载函数库: D:/Studio.auto/autotest/Lib/Template/Sys/Common
->2005-01-20 12:01:55-(TestLib):建立本次测试逻辑:TestLogic_02_02预置条件
->2005-01-20 12:01:55-(TestLib):执行测试逻辑TestLogic_02_02的环境TERM1的建立配置脚本
->2005-01-20 12:01:55-(TestLib):=====
->2005-01-20 12:01:55-(TestLib):开始测试逻辑测试, 测试逻辑编号为TestLogic_02_02
->2005-01-20 12:01:55-(TestLib):-----
->2005-01-20 12:01:55-(TestLib):开始测试用例测试(第1选)-测试用例编号:TestLogic_02_02_01

->2005-01-20 12:01:55-(TestLib):          测试用例名称:HelloWorld
->2005-01-20 12:01:55-(TestLib):          测试逻辑编号:TestLogic_02_02
->2005-01-20 12:01:55-(TestLib):          关联系统测试用例编号:
->2005-01-20 12:01:55-(TestLogic_02_02):aTestCaseData(ipAddr):df\ a f,df\sd-10.1.1.14::gaEnv1(slot1)-$aEnv1(mySlot33)
->2005-01-20 12:01:55-(TestLogic_02_02):length:2
->2005-01-20 12:01:55-(TestLogic_02_02):t1111:df\af,df\sd
->2005-01-20 12:01:55-(TestLogic_02_02):vp1:\ab,c \
abc\ abc abc
->2005-01-20 12:01:55-(TestLogic_02_02):Env1(slot2):2
->2005-01-20 12:01:55-(TestLogic_02_02):Env1(slot2Str):aaa
->2005-01-20 12:01:55-(TestLogic_02_02):Env1(slot3):10.1.1.14::gaEnv1(slot1)
->2005-01-20 12:01:55-(TestLogic_02_02):Env1(mySlot3):10.1.1.14::gaEnv1(slot1)
->2005-01-20 12:01:55-(TestLogic_02_02):Env1(mySlot33):10.1.1.14::gaEnv1(slot1)
->2005-01-20 12:01:55-(Template):窗口名: 10.76.161.84, 申请资源: df\ a f,df\sd-10.1.1.14::gaEnv1(slot1)-$aEnv1(mySlot33)
->2005-01-20 12:01:55-(Template):窗口名: 10.76.161.84, 执行命令: NG
->2005-01-20 12:01:55-(TestLogic_02_02)::TestLogic_02_02:: TestLogic_02_02 -测试结果: NG
->2005-01-20 12:01:55-(TestLogic_02_02)::TestLogic_02_02:: TestLogic_02_02 -测试结果: NG
->2005-01-20 12:01:55-(TestLib):停止测试用例测试
->2005-01-20 12:01:56-(TestLib):结束测试逻辑测试, 测试逻辑编号为TestLogic_02_02
->2005-01-20 12:01:56-(TestLib):=====
->2005-01-20 12:01:56-(TestLib):恢复本次测试逻辑:TestLogic_02_02预置条件
->2005-01-20 12:01:56-(TestLib):-----
->2005-01-20 12:01:56-(TestLib):恢复测试集: TemplateTc22
->2005-01-20 12:01:56-(TestLib):执行测试集TemplateTc22的环境TERM1的恢复配置脚本
->2005-01-20 12:01:56-(TemplateTc22):execute TemplateTc22 TERM1 RestoreScripts
->2005-01-20 12:01:56-(TestLib):+++++++
->2005-01-20 12:01:56-(TestLib):恢复测试集: TemplateTc1
->2005-01-20 12:01:56-(TestLib):执行测试集TemplateTc1的环境TERM1的恢复配置脚本
->2005-01-20 12:01:56-(TemplateTc1):execute TemplateTc1 TERM1 RestoreScripts
->2005-01-20 12:01:57-(TestLib):+++++++
->2005-01-20 12:01:57-(TestLib):恢复测试集: TemplateTc2
->2005-01-20 12:01:57-(TestLib):执行测试集TemplateTc2的环境TERM1的恢复配置脚本
->2005-01-20 12:01:57-(TemplateTc2):execute TemplateTc2 TERM1 RestoreScripts

```

图 6

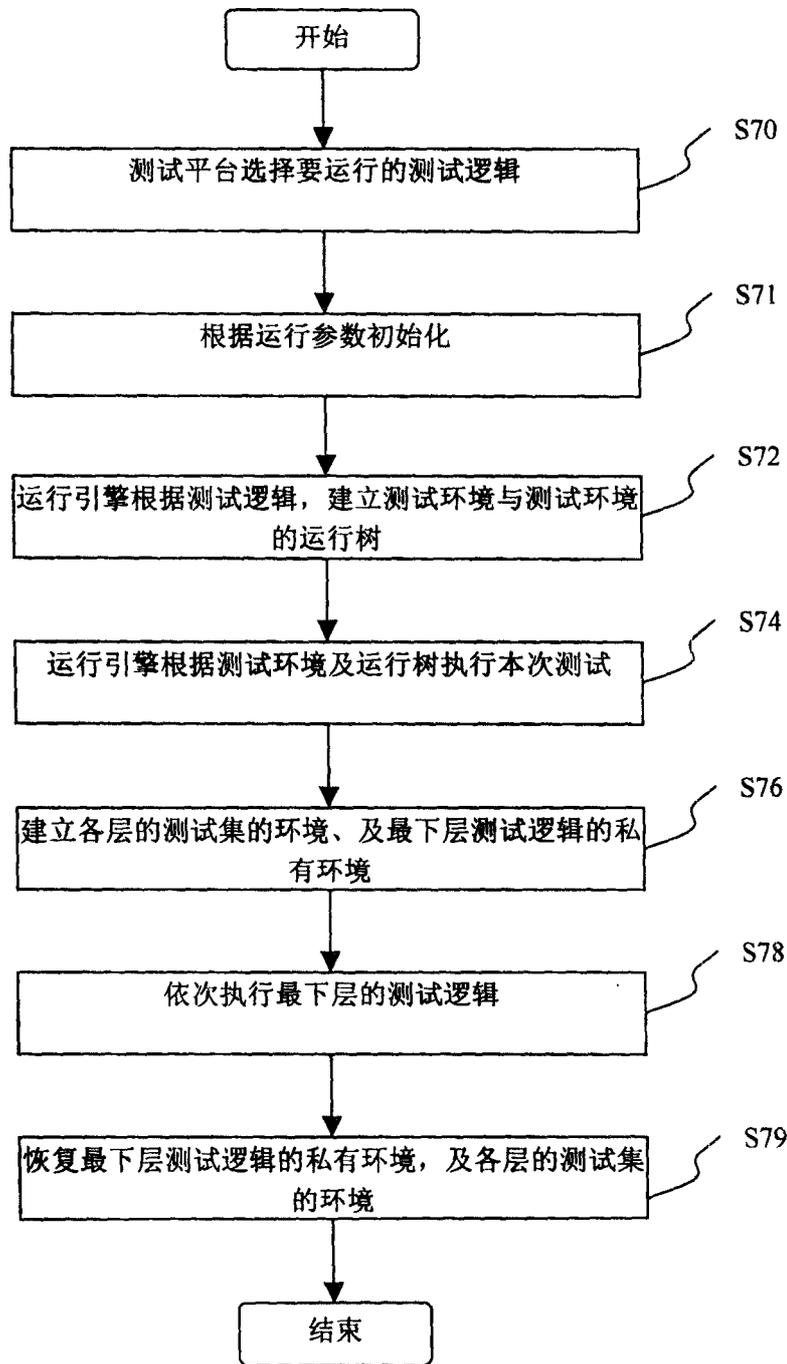


图 7

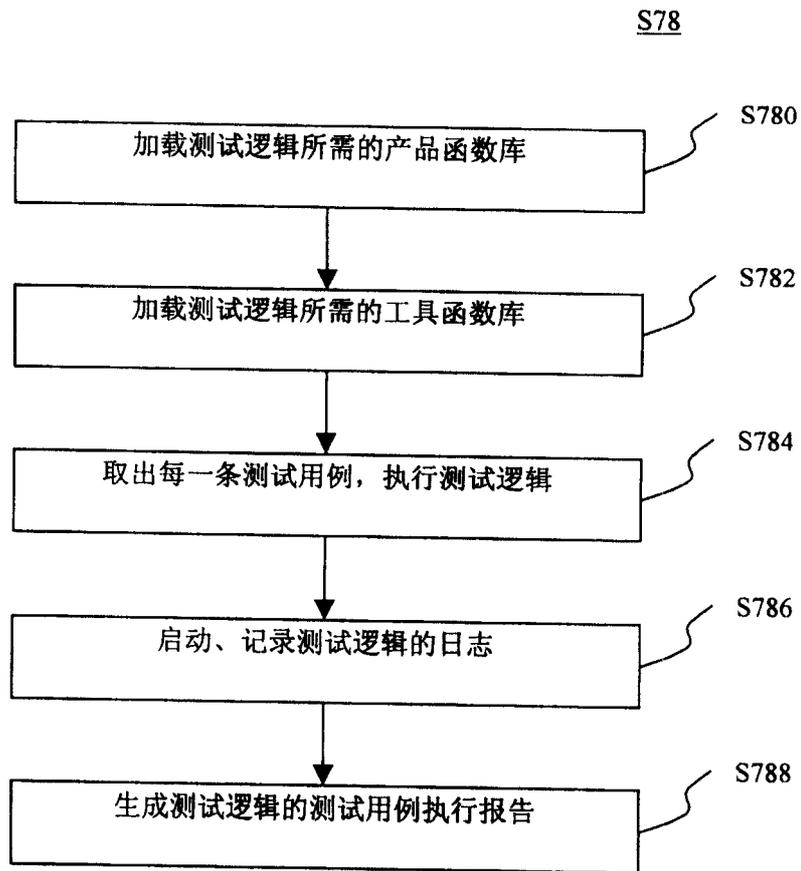


图 9

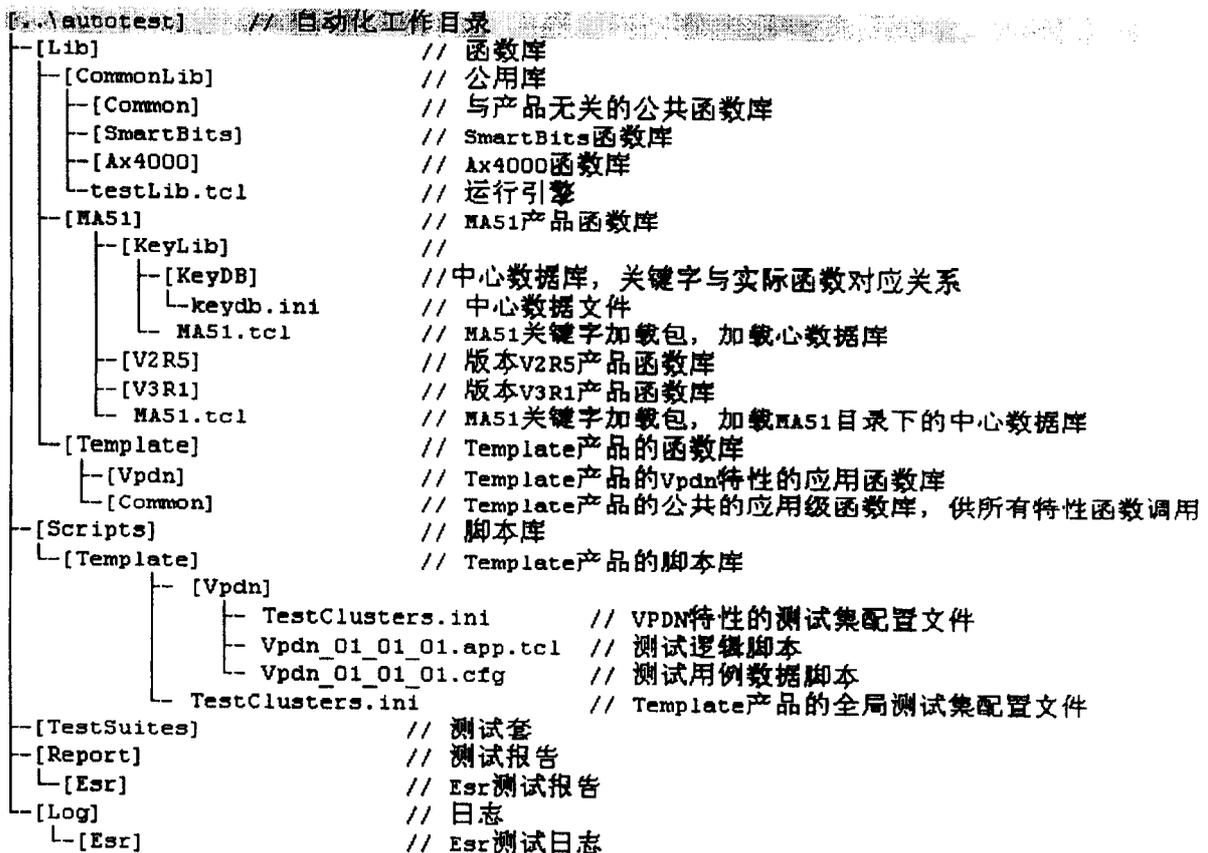


图 10