Linux 下自动化测试执行管理工具的设计与实现①

韩　涛　高　静 (北京航空航天大学 计算机学院 北京 100191)

摘　要： 测试自动化是提高软件测试效率的重要途径。尽管各种测试工具具有强大的执行功能，但没有对测试过程和测试资源进行有效的管理。设计并实现了一个 Linux 下的自动化测试执行管理工具，侧重于测试执行管理和测试资源管理，并详细阐述了其总体构架、模块设计和关键技术。

关键词： 测试自动化；测试执行管理；测试执行过程；测试资源

Implementation and Design of Automated Test Execution Management Tool Under Linux Operating System

HAN Tao, GAO Jing

(School of Computer Science and Technology, Beihang University, Beijing 100191, China)

Abstract: Test automation is an important way to improve the efficiency of software testing. Various testing tools have excellent executive ability. However, they don't pay much attention to the management of test resources and test execution processes. Therefore, an automated test execution management tool is designed and implemented in this paper, which focuses on the management of test execution process and testing resources. The paper elaborates the general architecture, module design and the key technology.

Keywords: test automation; test execution management; test execution process; test resource

1　引言

软件测试所需时间已经占到整个软件开发周期的40%以上[1,2]。人们利用软件的自动化测试技术，发现正常测试中很难发现的缺陷(如内存方面的缺陷) ;执行手工测试困难或不可能做的测试(如多用户并发操) ;运行更多更频繁的测试(如重复输入相同的测试输入、回归测试)。总之，测试框架的一致性、可重复性和复用性能弥补手工测试的不足，获得更彻底的测试，从而提高测试质量。自动化测试工具大多是以 junit 为基础，以测试用例(TestCase)和测试套件(TestSuit)为主要运用，对所要测试的类和主要方法加进测试方法，然后作断言(assert)判断，如果与预期结果不符合，就抛出异常，且可以一次执行多个测试用例。这样能简化人工的干预，称之为自动化测试。与非分布式自动化测试工具相比，分布式自动化测试工具支持多台计算机之间的协同测试，具有以下优越性：①可以测试需要在分布式环境下运行的代码，完善了测试功能；②支持在分布式环境下进行测试工作，即将一个独立的测试程序集分解为多个测试部分在多台计算机上并发进行，提高了测试效率。

目前很多国外商业公司和开源组织在开发测试工具。Windows 系统下的自动化测试执行工具的研究起步早，发展快，已经形成一些比较成熟的自动化测试工具。其中有 Rational Robot、AdventNet QEngine、QA Run、WinRunner 和 Test Partner 等，他们分别针对 web 测试、Java 程序测试，是应用广泛的自动化软件测试工具。

随着 Linux 操作系统的普及和发展，Linux 下自动化测试执行管理工具越来越受到软件测试领域的重视，基于 Linux 操作系统的自动化测试也有一定的发展。较为成熟的工具有 DejaGnu、Test Environment Toolkit (TET3)和 Testify 等。

① 基金项目:国家高技术研究发展计划(863)(2009AA012406);2008 年度电子信息产业发展基金招标项目(中标序号:8)
收稿时间:2009-07-14;收到修改稿时间:2009-08-19

虽然这些工具在 Linux 系统上支持各种功能的自动化测试，但是存在的问题也是明显的，如 Deja-Gnu 支持远程测试但执行效率有明显缺陷；Test Environment Toolkit (TET3)支持分布式测试但操作过于复杂、没有友好的用户界面；Testify 是一个商业的测试框架，提供了 GUI 界面，但功能没有 TET3 强大，且代码不开放。另外，这些工具没有考虑到对整个测试过程和测试资源的管理，如测试项目和测试用例的管理和测试过程的完全自动化等。

同时，国内还没有自主研发的自动化测试工具。国内的大部分软件开发公司停留在使用国外开发的测试软件的程度，几乎没有成型的国产 Linux 下自动化测试工具。

综合国内外 Linux 下自动化测试执行管理工具的现状，本文设计实现的 Linux 下自动化测试执行管理工具，侧重于对测试资源和测试过程的管理，又研究了分布式测试调度的模型，实现了真正意义的自动化测试。

2 Linux下自动化测试执行管理工具的总体架构

2.1 总体构架

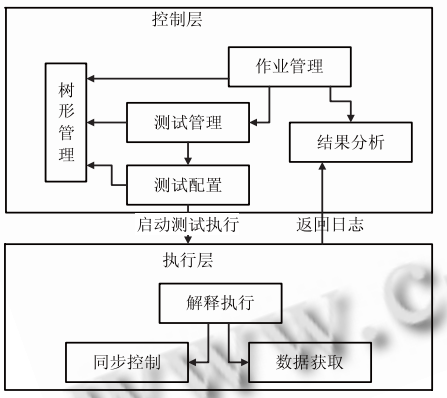

图 1 总体的分层体系结构

分层结构是目前复杂应用系统开发时普遍使用的模式，软件层之间的依赖关系约束较宽松，要求上层可以依赖于它的所有下层。因此，本框架采用分层模式，包含控制层和执行层，如图 1 所示，其中箭头表示通信和交互关系。执行层由解释执行模块、同步控制模块和数据捕获模块组成，负责分布式执行时作业调度、测试过程控制和数据的捕获；控制层由树形管理模块、作业管理模块、测试管理模块、测试配置模块和结果分析模块组成。控制层向执行层发送测试执行的命令；执行层执行、监控测试过程，测试过程结束后向控制层提交测试结果，结果分析模块分析测试结果得到报告。

2.2 Linux 下自动化测试执行管理工具的任务转换模型

用作业和任务来模拟具体的测试任务。把测试项目，如回归测试(regression test)，看作机群系统里的作业(job)，每个作业分成多个任务(task)来执行。每个作业有自己的资源定义文件，其定义了每个作业(job)的任务(task)以及最终执行测试的所需数据。

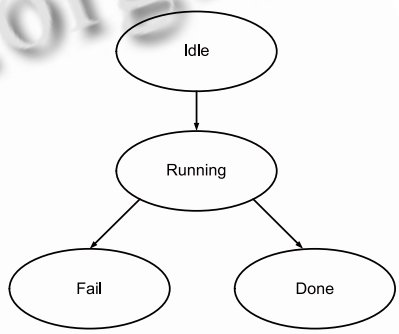

图 2 任务状态转换模型

本文定义一个任务共有四种状态： 空闲(Idle)，运行(Running)，完成(Done)， 失败(Fail)，如图 2 所示。记录这些状态的任务状态文件存放在控制主机上，所有执行机对其只有可读权限。由于测试的特殊性，该生命周期并不是一个闭合的系统。因为一个任务失败意味着该测试作业的失败，所以不需要重新调度该任务直到它成功，完成或失败就是一个任务的结束。结合任务的生命周期，该工具运行测试作业的通用步骤如下：

步骤 1：控制主机读取资源定义文件，解析该文件，得到一个作业的具体任务。此时该任务状态为 Idle。

步骤 2：执行机收到任务后，同样读取资源定义文件上的信息，启动执行层来执行测试，并把任务状态改为 Running。

步骤 3：执行完测试后，执行机会将该任务(task)的状态改为 Done 并退出。如果这个任务运行失败，则执行机将把状态改为 Fail。

步骤 4：控制主机等到任务(task)完成后，开始收集测试结果。

步骤 5：控制主机进入下一个任务的循环。当一个作业的所有任务都完成后，控制机把所有的测试结果汇总到一个文件。

这就是自动化测试框架的基本模型[3]。

3 Linux下自动化测试执行管理工具的设计与实现

3.1 控制层的设计实现

控制层是整个自动化测试执行管理工具的总控制台，它控制测试执行整个过程，采用 Eclipse 插件技术、RCP(Rich Client Platform，富客户端)技术等实现。这样开发出来的程序能方便地继承 eclipse 自身框架，不用花费大量精力设计用户界面(UI)且 RCP 程序是一个可以脱离 Eclipse 独立运行的插件——它完全屏蔽了 Eclipse 原有的内容，使用户界面(UI)更加符合用户要求，这也就克服了插件技术的最大缺点。在 eclipse 的插件式体系结构中，这些技术容易集成在一起，构成一个有机的整体[4-7]，并且具有良好的可扩展性和移植性。控制台表现为树形结构，包含以下功能：

3.1.1 树形管理

树形结构善于表现隶属关系或者层次关系，具有独特的扩展和折叠分支的能力，能够以较小的空间显示出大量的信息，一目了然地传达出数据之间的层次关系。灵活运用这种树形结构不但能增强软件本身的表现力，同时也使软件容易被用户所接受。

此树形结构组件的设计针对多个测试作业的管理，即有较多节点的树，需要实现如下主要功能：

(1) 测试作业实现分层、动态的加载方式，即在初始化树时，只加载第一级的节点(测试作业)，在用户点击某节点时，加载此节点的下一级节点。

(2) 自动判断是否有下一级节点。由于采用动态加载的方式，若能自动判断是否有下一级节点，用户就可以直观地看到是否有下一级信息。

(3) 测试用例查找功能。由于测试用例较多，信息量大，一棵树上可能有几百个用例，用例查找功能可以使测试人员快速查询到需要的信息，增强软件的易用性。

(4) 用例定位功能。在一棵具有较多用例的树中，具有相同显示标签的用例很多，在查找时，实现上下定位用例可以使测试人员方便地进行查找、统计。

(5) 实现用例显示标签的可配置。对于不同的作业需求，要求用例的显示标签可能不同，此功能的实现，增强了软件的可移植性。

(6) 可扩展性、灵活性。在任务变化时只需改动树的配置信息，就能生成相应的树形结构，无需对程序进行任何改动，易于扩展。

作业管理模块和测试管理模块调用树形管理模块，为用户提供了对测试作业、测试任务和测试脚本的新建、打开、导入、编辑、删除、重命名等常用功能和查找定位、显示配置等非常用功能。每一个测试作业所含的文件存储在以作业名为文件夹名的文件夹下，每个测试任务包含的文件存储在以测试任务名称为文件夹名的文件夹下；各文件夹在控制台上表现为树形图的分枝节点；各文件表现为叶子节点。如图 4 所示。

3.1.2 测试配置功能

本功能通过一个向导对话框完成对测试执行所必需参数的配置，比如对编译、运行和清除阶段的参数的配置。同时，让用户方便地在编辑区定制资源定义文件。资源定义文件指明测试用例的执行顺序和执行方式，例如顺序为并行，执行方式为分布式。

控制台实现的作业管理、测试管理和测试配置功能，使用户在统一的界面对测试作业进行统一的管理，可以提高工作效率。

3.1.3 结果分析功能

日志文件由执行层返回，记录了运行用例的情况。通过对日志的统计分析，得到定制的分析结果，即为报告。本功为用户提供多种的日志分析方式，例如统计分析、与以前的日志的比较分析等。

3.2 执行层设计与实现

解释执行模块、同步控制模块和数据捕获模块是执行层的三个功能模块，提供了测试执行的执行控制、作业调度和结果捕获的功能。解释执行模块实现解释资源定义文件、获取测试指令，且在现有系统的条件下，优化整合资源，对任务进行调度分配的功能。同步控制模块实现各个参与系统的通信，使得每个任务执行完成后，各个参与主机进行一次通信。捕获数据模块实现在测试用例执行时，把相关的信息写到日志文件中的功能。

解释执行模块是执行层的核心模块。编译执行模块的体系结构框架图如图 3 所示。其中，000 系统是主系统，负责执行本地测试任务并管理所有参与系统的工作；系统 001、系统 002 是参与系统中运行分布式测试的系统，负责执行本机上的测试任务，受控于 000 系统。

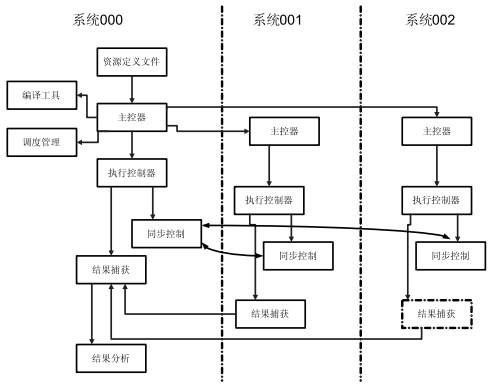

图 3 解释执行模块体系结构

具体解释如下：

(1) 资源定义文件记录所有的测试任务且描述测试用例的执行流程，它决定了主控器如何处理测试程序。

(2) 编译工具把测试程序编译成可执行文件。

(3) 主控器的功能是读取、编译资源定义文件、获取指令，控制参与系统上的测试用例控制器。

(4) 调度管理根据资源文件和系统中主机资源现状对测试任务进行分配。

(5) 执行控制器等待主系统启动，接收主控器传送的测试任务，控制同步和数据获取，执行测试任务中的测试用例。

(6) 同步控制工具负责分布式测试中在不同的机器上执行测试程序的同步。

(7) 结果捕获工具收集分布在不同机器上的测试执行结果，并在整理后写入执行结果日志文件。

(8) 日志文件记录测试结果，被送到控制层的结果分析模块进行进一步的数据处理。

在非分布式测试中，主控器直接启动执行控制器，执行控制器接收主控器传送的测试任务，执行测试任务中的测试用例，并直接将测试执行结果写入日志文件。在分布式测试执行中，有并行、远程两种执行方式。不同执行方式的执行情况不同：在并行方式下，系统 000 的主控器根据调度管理控制系统 001 和系统 002 的主控器，各个系统的主控器控制各自的测试执行过程，在固定的时间内进行同步，直到测试用例运行完成；在远程方式下，系统 000 的主控器控制系统 001 和系统 002 的主控器进行测试用例的运行，而系统 000 本身不运行测试用例。所有的结果分析功能在 000 系统上完成。

3.3 实例

该工具已经在实验室中得到了应用，以测试作业 myproject 为例，其运行情况如图 4 所示：

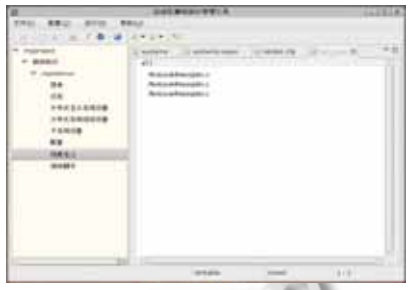

图 4 运行截图

图中左侧是树形控制台，右侧是编辑区。编辑区编辑的是资源文件的定义，其中 testcase 文件夹下包含 c 语言测试用例 example1、example2、example3 ，这三个测试用例是串行执行的，每个测试用例只顺序执行一次。

4 结语

本文实现的 Linux 下自动化测试执行管理工具，具有测试脚本自动化执行、测试结果收集与分析、测试运行管理、测试工程管理、测试配置等功能，允许串行、并行、重复、远程和分布式等多种测试执行顺序，提供详细的测试日志和分析报告。最重要的是，该框架实现了真正意义上的自动化测试，从作业调度到报告的生成都是工具自动运行，并且支持多个测试作业重复、并行分布式执行。目前该工具已经投入日常使用，实验结果证明其大大缩短了测试时间，提高了测试效率。

参考文献

1 Ron Patton.软件测试.北京:机械工业出版社, 2002. 1－50.
2 Mosley DJ, Posey BA. 软件测试自动化. 北京:机械工业出版社, 2003. 100－226.
3 刘锷,苗克坚,刘震.基于分布式系统的 GUI 自动化测试框架.计算机仿真, 2007,10:261－263.
4 Gamma E, Beck K. Contributing to Eclipse: Principles, Patterns, and Plug-Ins Addison-Wesley Professional, 2003. 1－20.
5 Clayberg E. ECLIPSE 插件开发.北京:人民邮电出版社, 2006. 20－22.
6 陈刚. Eclipse 从入门到精通.北京:清华大学出版社, 2005. 218－221.
7 Gallardo D, Burnette Ed, McGovern R. Eclipse in Action: A Guide for Java Developers. Manning Publications, 2003. 1－50.