

自动化回归测试的技术和实现

李刚毅¹, 金蓓弘²

(1. 中国科学院 研究生院, 北京 100049; 2. 中国科学院 软件研究所, 北京 100080)

摘 要: 提出了两种提高回归测试自动化程度的技术。其中一种技术采用数据驱动的方式, 使得测试脚本成为可以驱动所有类似测试用例组的通用脚本, 同时, 实现了测试执行和测试逻辑的分离, 使得测试用例的修改和维护更加容易。介绍的另一种技术使用附加的动态链接库来恢复被测软件的图形界面状态, 使得软件图形界面的自动测试不易受到被测软件状态改变的影响, 提高了整个自动测试系统的健壮性。

关键词: 软件测试; 回归测试; Silktest

中图法分类号: TP311 文献标识码: A 文章编号: 1001-3695(2006)02-0186-03

Techniques and Realization of Automated Regression Testing

LI Gang-yi¹, JIN Bei-hong²

(1. Graduate School, Chinese Academy of Sciences, Beijing 100049, China; 2. Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)

Abstract: Two techniques of improving automated regression testing are presented. One of which adopts the means of data driven, such that the testing scripts become general scripts, which can drive all similar test groups. Meanwhile, the operation and logic of the test are separated. As a result, it is easy to maintain and modify the test cases. The other technique discussed uses additional Dynamic Link Libraries to recover the GUI state of AUT, such that the automated GUI testing is not easy to be affected by the changes of AUT's states, and the robustness of the entire automated testing system is improved.

Key words: Software Testing; Regression Testing; Silktest

软件测试是关系到软件开发和维护成本的重要环节^[1]。任何软件产品在正式发布之前都必须经过严格的测试。随着计算机技术的迅速发展, 软件的结构越来越复杂, 同业竞争越来越激烈。为了保证软件产品的高度可靠性和竞争力, 很多软件开发机构都将其主要的研制力量投入到软件测试之中^[2]。

根据不同的测试目的和测试方法, 软件测试可以分成不同的测试类型, 回归测试 (Regression Test) 是其中的一种。回归测试是在软件开发过程中为确保软件质量而进行的一种常用的验证测试方法^[3]。回归测试需要在软件出现发展性的改变 (即, 规范和代码发生变化) 和修正性改变 (只有代码发生变化) 时运行^[4], 用于确保软件在增加了新功能或者软件修改后, 现有功能仍然继续可用。由于软件的新增代码中可能包含错误代码, 或者新增代码或经过修改的代码可能对软件的其他部分产生影响, 从而使软件不能正常工作, 所以有必要在软件修改之后对软件进行测试, 以验证软件的现有功能是否仍然可用。

回归测试是软件测试中的重要组成部分, 占有很大的比重。大约 30% 左右的错误 (Bug) 是通过回归测试发现的^[5]。而面向对象的软件产品的开发过程通常是渐进式或迭代式的, 这需要在开发过程中不断测试代码改变部分对未改变部分的影响, 即不断进行回归测试, 因此, 面向对象的软件产品的大部分测试预算会使用在回归测试上^[6]。

上述回归测试的执行特点, 决定了实施回归测试, 是一项工作量大、烦琐的工作。因此实现自动化的回归测试可以提高

测试效率和保证测试的可靠性。现在可供选择的第三方的自动测试工具很多, 如 WinRunner, SilkTest, 等。这些测试工具通常可以对软件的图形用户界面 (GUI) 进行冒烟测试 (Smoke Test) 和回归测试。

SilkTest 是美国 Segue 公司开发的一种软件 GUI 的自动测试工具。使用 SilkTest 软件可以将被测试软件 (Application Under Test, AUT) 的 GUI 对象 (例如, 窗口、按钮、菜单、文本等) 和在其上进行的各种鼠标或键盘操作录制下来, 转换成 Segue 公司开发的一种面向对象的第四代编程语言 (4Test) 的脚本。SilkTest 通过重新运行所记录的操作, 实现自动测试的目的。

1 数据驱动的回归测试方法

1.1 测试实施环境的特点

在开发面向基于应用服务器的分布式应用的 IDE 工具软件的过程中, 往往需要对服务器和 IDE 工具之间的协同工作能力进行测试, 以保证使用 IDE 工具软件开发出的应用程序最终可以正确地安装到应用服务器上并正常工作。由于软件产品通常都采用渐进式或迭代式的开发策略, 因此需要频繁地运行涉及协同工作能力的回归测试。

这种回归测试的特点是:

- (1) 需要在软件的集成开发环境 (IDE) 中操作;
- (2) 通常将测试案例的运行结果输出到文本文件中, 作为测试通过与否的标准或参考;
- (3) 随着软件版本的升级, 测试用例应具有一定的可扩展

性, 并易于维护。

涉及协同工作能力的分布式回归测试的典型步骤如图 1 所示。首先, 在 IDE 中将服务器端组件部署到应用服务器上。这需要在 IDE 中进行一系列的手工操作。当服务器端组件部署到应用服务器上之后, 需要在 IDE 中生成或配置与服务器端组件相对应的客户端组件。最后, 从 IDE 中(或者通过生成可执行文件的方式) 运行客户端组件, 进行客户端与服务器端的交互操作测试, 并产生测试结果。

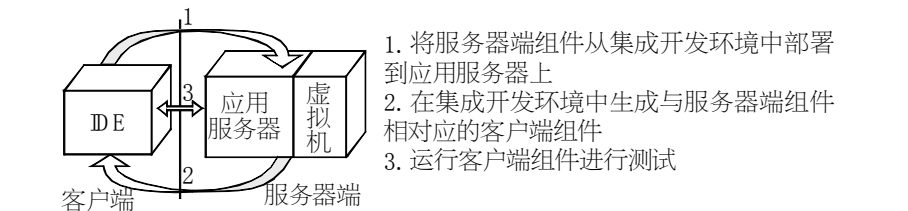


图 1 对开发分布式应用的 IDE 工具软件进行测试的典型步骤

在设计测试用例时通常将相关的用例设计成一组。每组用例都要执行一次图 1 所示的各个步骤。对于复杂的应用软件来说, 这种协同工作能力的回归测试可能包含很多组测试用例, 如果手工运行这些测试, 就需要一遍又一遍地重复执行类似的操作, 这是令人乏味的, 同时也耗费了大量的人力和时间。尽管创建自动测试所需要的准备时间是手工测试的三至十倍^[5], 但是由于上述原因, 这样做还是值得的。

1.2 回归测试的步骤

考虑到上述测试的部署、配置和运行的操作主要在集成开发环境中进行, 因此可以通过使用 SilkTest 录制操作来实现测试的自动化。SilkTest 将所记录的图形用户界面对象转换成 4Test 语言对象, 将所记录的各种鼠标或键盘操作转换成 4Test 语言的方法。由于 4Test 语言是面向对象的, 所以可以对 4Test 语言脚本进行各种修改, 也可以用变量代替录制测试步骤过程中使用的实际值, 从而实现数据驱动的自动测试。在测试 IDE 工具软件时, 不同测试用例组的操作之间具有极大的相似性, 因此存在采用数据驱动的方式来进一步优化用 SilkTest 录制的测试操作的可能性。也就是说, 录制一个典型的测试流程, 然后通过对 SilkTest 录制的脚本进行编程修改, 将其改造成可以驱动全部测试用例的通用驱动程序。

图 2 显示了用 SilkTest 实现数据驱动的回归测试的流程。

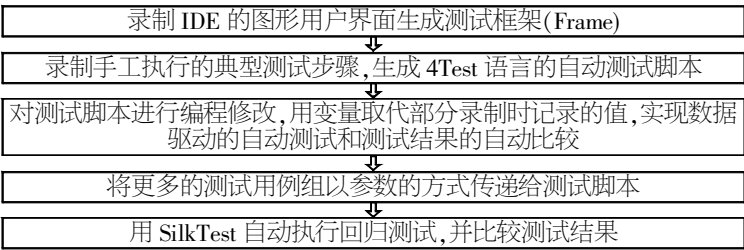


图 2 用 SilkTest 实现数据驱动的回归测试的流程

下面将具体地介绍如何用 SilkTest 实现数据驱动的回归测试。

根据 SilkTest 的要求, 在用 SilkTest 录制测试步骤之前, 应当首先录制与测试过程中的鼠标或键盘事件相关的 AUT 的各个 GUI 控件, 例如窗口、菜单、按钮等, 也就是所谓的“对象声明”过程。由于在本文所说的回归测试中, 鼠标或键盘事件主要与软件的 GUI 控件相关, 因此需要首先对软件的 GUI 控件进行记录。用 SilkTest 记录 AUT 的 GUI 控件时, 只需要将鼠标依次在 GUI 控件上划过即可。SilkTest 会将这些控件转换成相应的 4Test 语言对象。SilkTest 在记录控件的同时, 也记录了它

们与父控件之间的隶属关系。所记录的 GUI 控件对象及其隶属关系称为测试框架(Frame)。SilkTest 会自动为所记录 GUI 控件命名, 但是这些名称通常是由控件名和序号组成, 不易于理解, 因此需要将所记录的控件用易于理解和记忆的名称重新命名, 以便于今后的开发和维护。

在测试框架录制完成之后, 需要录制手工执行的典型测试步骤。SilkTest 会将记录下来的手工操作步骤转换成 4Test 语言的脚本。下面的代码就是记录配置服务器连接操作的一段 4Test 语言代码示例:

```
ConfigureHost ( )
DeployDialog. HostName. SetText ( " AppServer1" )
DeployDialog. PortNumber. SetText ( " 9000" )
DeployDialog. Login. SetText ( " Admin" )
DeployDialog. PassWord. SetText ( " abc123" )
DeployDialog. ConnectButton. Click ( )
```

在上述代码中, 用 4Test 语言记录了在部署配置对话框中输入服务器名、端口号、登录用户名和密码等内容, 并点击“连接”按钮与服务器进行连接。应当注意的是, 代码中使用录制时输入的具体值作为各个方法的参数, SilkTest 运行测试时会自动将这些值填充在部署配置对话框的相应位置上。

为了实现数据驱动, 让测试具有更大的灵活性, 我们需要对录制好的 4Test 语言脚本进行修改, 用参数替代录制过程中使用的实际值, 然后通过某种方式为这些参数赋值。对于上面的例子, 可以定义一个包含各个输入项的数据的数据类型:

```
type HOSTINFO is record
STRING sHost / 主机名
STRING sPort //端口号
STRING sLogin //用户名
STRING sPassword //密码
```

然后把这个数据类型传递给 ConfigureHost 方法:

```
ConfigureHost ( HOSTINFO data)
DeployDialog. HostName. SetText ( data. sHost)
DeployDialog. PortNumber. SetText ( data. sPort)
DeployDialog. Login. SetText ( data. sLogin)
DeployDialog. PassWord. SetText ( data. sPassword)
DeployDialog. ConnectButton. Click ( )
```

在测试时使用的有关服务器配置的实际值可以存放在某个数据文件中, 并在程序的适当部分将其读入。

对于不同的测试用例组, 它们之间的差别可能还包括组件名称和文件存储位置等内容, 那么只需要用上面所说的方法将这些存在差异的地方的具体值用参数代替, 并在运行特定的测试用例组的时候, 从相应的数据文件中读取相应的值, 就可以把专用的测试步骤变成通用的测试步骤。因此, 仅仅需要用 SilkTest 录制一个典型测试用例的运行步骤, 再经过一些相应的修改后, 就可以用 SilkTest 自动运行这类回归测试中的所有测试用例。另外, 由于实现了数据驱动的测试, 因此对测试项目的增加和修改也是非常容易的。

我们通常使用 SilkTest 来验证 AUT 的 GUI 的正确性。在这种情况下, SilkTest 依次执行 4Test 语言脚本中的各个操作, 并判断每步操作是否会导致预期的显示结果。如果每一步的显示结果都与所预期的一致, 那么 SilkTest 就认为测试通过了, 否则, 就认为测试失败, 并在自动生成的测试结果文件中显示运行结果与预期结果之间的差异。

但是, 本文所述的方法使用 SilkTest 实现了涉及软件之间协同工作能力的数据驱动的回归测试, 其中 SilkTest 自动执行

的是运行测试用例组的步骤。也就是说 SilkTest 只是实现了测试用例组的执行过程,而不涉及测试用例组和测试用例本身的逻辑。所以,即使所有的测试步骤都顺利地执行了,并不代表测试也已经顺利地通过了。

为了利用 SilkTest 自动对测试结果进行判断,可以在测试用例中加入输出语句,这样在运行测试的时候,所有测试用例的运行结果都会输出到指定的日志文件中。当 SilkTest 运行完一个或一组测试用例之后,将包含测试结果的日志文件与一个基准文件(Benchmark File)进行比较。一般来说,基准文件是在一个或一组测试用例全部正确运行的情况下得到的测试输出日志,它通常是在测试用例设计完成的时候就确定的。如果包含测试结果的日志文件与基准文件完全匹配,那么就可以认为测试通过了。如果它们之间存在差别,那么就认为测试没有通过,并在 SilkTest 的测试结果文件中打印两个文件之间的差别。

2 自动恢复软件 GUI 测试环境

2.1 测试实施环境的特点

图形用户界面(GUI)已经成了与软件系统交互的最常用的手段^[7],也是分布式应用程序常用的界面形式。为了满足个性化的需求和方便用户使用,很多软件的 GUI 都允许用户进行定制和配置,或者根据当前操作的不同动态地显示菜单和工具栏。这在手工进行 GUI 测试的时候并不会产生什么问题,然而,现在的自动测试工具软件都采用记录-回放技术^[8],在使用它们回放测试步骤时,却往往因为测试时的 GUI 状态与录制时不一样而使得测试软件不能正确识别 AUT 的 GUI,从而导致测试失败。

此外,当自动运行一组测试用例的时候,前面的测试用例对 AUT 的 GUI 的操作也可能会改变其状态,或者由于前面的测试用例意外中断,而不能将 GUI 引入下一个测试用例所需要的状态,这都会导致后面的测试用例无法继续执行下去。

由于上述定置和使用上的原因,在使用自动测试工具进行测试时 AUT 的 GUI 状态往往与录制测试用例时的状态存在着差别,从而导致因测试工具不能正确地识别 AUT 而使得测试失败。因此,需要一种能够实现自动恢复软件 GUI 测试环境的方法。

2.2 自动配置测试环境

SilkTest 软件除了可以录制对 AUT 的 GUI 的操作外,还可以录制 GUI 的各种状态。SilkTest 软件的恢复系统可以在测试用例自动运行前或测试用例意外中断执行时,将 AUT 的 GUI 重新恢复到测试所需要的状态,从而让后面的测试用例得以继续执行。这样,就减少了在测试过程中对出现测试失败的情况进行手工干预的需要,使得自动测试系统更加健壮。

SilkTest 软件的恢复系统提供诸如 DefaultBaseState, DefaultScriptEnter, DefaultScriptExit, DefaultTestcaseEnter 和 DefaultTestcaseExit 等多种缺省的自动恢复方法,测试开发人员可以通过重新定义这些方法来实现自己的自动恢复功能。

当运行测试用例的时候, SilkTest 软件的恢复系统首先执行 DefaultTestcaseEnter 方法载入 AUT 的缺省基本状态。基本状态是期望在每个测试用例开始执行之前 GUI 所处的状态。

缺省情况下,当满足下列四个条件的时候, SilkTest 软件就认为 AUT 处于基本状态: AUT 的 GUI 应用程序正在运行; AUT 的 GUI 应用程序没有最小化; AUT 的 GUI 应用程序是激活的; 除了 AUT 的 GUI 应用程序的主窗口(Main Window)外,该应用程序的其他窗口都没有打开。

在载入缺省基本状态的时候, SilkTest 软件还会调用 AUT 主窗口类的 BaseState 方法,载入用户定义的基本状态。然后,依次载入测试用例所需要的应用程序状态。在 AUT 进入测试用例所需要的应用程序状态的时候,就可以执行测试用例了。如果测试用例执行完成或者测试用例因故中断,那么 SilkTest 软件的恢复系统就执行 DefaultTestcaseExit 方法,这个方法会调用 SetBaseState 方法来载入 AUT 的缺省基本状态和用户定义的基本状态。这样,运行自动测试的时候就基本上不需要人为的干涉了。

不过,通过录制的方式依次记录 AUT 的基本状态和运行测试用例所需要的各个状态是比较耗费时间的,也会使软件测试的恢复系统方法过于复杂而不便于维护,另一方面, GUI 应用程序往往是很复杂的。为了方便用户的使用,具有复杂的 GUI 的应用程序通常都允许用户进行个性化定置,并记录诸如最近编辑过的文件等信息。这些设置和记录的信息会保存在注册表或者配置文件中,它们可能正是自动测试所需要的,也可能会成为导致测试失败的因素。虽然通过替换注册表信息和配置文件可以很容易使 AUT 达到测试状态,然而,这些需求已经超出了 SilkTest 软件的处理能力,因此,需要基于 SilkTest 提供一种更灵活有效的办法来实现系统恢复。我们通过编写动态链接库,并结合 SilkTest 软件的 4Test 脚本语言,解决了上述问题。

SilkTest 软件在初次访问自动测试的脚本文件时调用 DefaultScriptEnter 方法。由于上述替换注册表和 AUT 配置文件的操作应当在启动 AUT 之前执行,因此应当在 DefaultScriptEnter 方法中实现动态链接库的调用。

下面给出利用动态链接库初始化自动测试环境的步骤(图3)。在这个过程中,需要 4Test 语言脚本调用附加的动态链接库完成下述操作。首先,为了保证测试中使用的数据库内容的一致性,先进行测试数据库的填充或初始化。然后,为了保证测试结束后可以将 AUT 的状态恢复到当前的状态,要对 AUT 的注册表信息和配置文件进行备份。最后,将包含测试用例所需要的 AUT 状态信息的注册表信息导入注册表中,并用满足测试需求的配置文件替换原有的配置文件。现在, AUT 已经按照测试的需求进行了配置,接下来就可以结束 DefaultScriptEnter 方法,并启动 AUT, 让软件进入测试的基本状态。

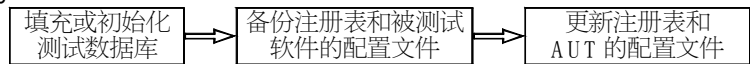


图3 初始化自动测试环境的步骤

同样,可以在 DefaultScriptExit 方法中定义测试结束以后的一些操作。例如,可以在测试结束后将测试前备份的注册表信息重新导入注册表、将测试前备份的配置文件重新替换回来。如果需要,还可以进行相应的数据库恢复操作。这样,自动测试的运行受到因 AUT 的状态改变的影响就大大减轻了,从而使得自动测试系统更加健壮。

(下转第 207 页)