办公软件接口自动化测试过程改进经验谈

杨琦
(中国软件评测中心 北京 100048)

【摘要】本文以笔者的一次办公软件接口测试经历为引,介绍了软件测试过程中不断提高测试自动化程度的经验方法,以及在此过程中获得的有益思考。

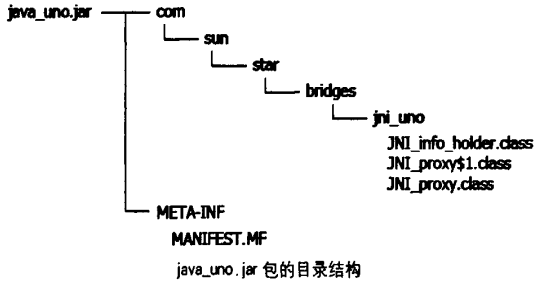
【关键词】软件测试;接口测试;自动化测试;Java;jar 包;Python

近期,笔者有幸参与了中国软件评测中心核高基实施部对国产办公软件的二次开发接口进行完整性测试的部分工作。虽然参与时间较短,涉及的内容也有限,但测试过程中仍然获得了一些经验和启示,值得记下来供读者参考。

1. 问题

目前流行的办公软件,如微软 Office、开源的 OpenOffice、国产的 WPS Office 等,通常会提供二次开发的编程接口,方便用户实现某些通过交互操作不易完成的任务。接口的提供方式,Windows 平台上的标准技术是微软的 COM 组件规范,常见的跨平台方案是使用 JavaBean 技术。我们的任务是根据厂商提供的 JavaBean 编程参考手册,测试某国产 Office 软件的二次开发接口的完整性。

Java 使用包(Package)的概念定义了适合发布和重用的软件单元 [1]。包的成员是一些相关联的类和接口,还包括可能用到的资源文件。在一个包中,嵌套的目录结构对应于命名空间,包和描述包的清单文件通常被压缩成一个 .jar 文件分发给客户 [2]。下图显示了 java_uno.jar 包的目录结构,除清单文件 MANIFEST.MF 外,共包含 3 个已编译的 Java .class 类文件。



java_uno.jar 包的目录结构

被测软件的开发手册罗列了支持的 Java 接口,完整性测试需要做的就是在提供的 jar 包中找到对应的类文件。我们知道,jar 包其实是压缩成 zip 格式,只要找到安装目录下的所有 jar 包,然后按 zip 格式打开,就可以去寻找其中包含的类文件。测试方法非常简单直观,但主要问题是,手工操作的效率非常低下。

1) 为了避免命名冲突,包的规范命名是以发布厂商的倒序网络域名作为前缀(如上图所示的 star.sun.com),再加上包自身的逻辑结构所添加的层次(如上图的 bridges.jni_uno),jar 包中的目录嵌套往往会很长。

2) 被测软件包含的诸多 jar 包文件分散在安装路径的各级子目录中。

另外,待测的其它几款国产 Office 软件也提供了类似的 Java 二次开发接口,因此需要一种高度自动化的方法来查找和匹配安装目录包含的 jar 包中的类文件。下文大体按照当时采取的几种尝试,介绍了改进测试自动化过程的一些手段。每次尝试我们关注的中心问题都是:影响测试效率的瓶颈是什么?

2. 直接的方案

Windows 系统是按后缀名辨别文件类型的。容易想到,可以先把安装目录下的所有 .jar 文件重命名为 .zip 文件,然后在 .zip 文件中搜索 Java 类名,这样总比全手工快一些。

为了不影响被测软件的正常使用,需要把所有 jar 包复制到一个空文件夹中再进行重命名。这利用 Windows 自带的搜索功能和简单的控制台命令很容易实现。如果直接在安装目录下进行重命名操作,还原时可能把原有的 .zip 文件重命名成 .jar 文件,导致被测软件运行异常。当然,假如安装目录

下没有或仅有少量 zip 文件，也可以记下它们的信息，直接在安装目录下重命名，但是，怎样把不同路径下的 .jar 文件原地重命名为 .zip 文件，我们并不熟悉相关的命令行操作。

Windows XP 自带对 zip 压缩格式的支持，可以把 zip 压缩包视同普通文件夹，直接搜索其中包含的文件。比如，我们直接搜索 JNI_proxy.class，就会得到包含它的 jar 包中的路径。但是我们发现，测试机的 XP 系统无法进入 zip 文件搜索。网上很容易找到答案，原来是支持 zip 文件夹访问的一个系统组件没有注册：

```
D:\Test> regsvr32.exe zipfldr.dll
```

这可能是测试机系统使用了某些“优化”软件的后果。

3. 迂回的方案

显然，以上方案的自动化程度并不算高。

1）需要手动搜索 jar 包，然后复制到另一个文件夹里并重命名。如果软件安装目录下的 jar 包非常多或非常大，这个复制操作就会成为测试过程的瓶颈。

2）每个 Java 类名必须手动输入 Windows 搜索框进行逐个搜索，无法自动化处理。如果需要测试的 Java 接口非常多，这种人工输入和等待就会成为测试过程的瓶颈。

本节给出的方案可以解决第一个问题，下一节的方案同时解决了上述两个问题。

注册 zipfldr 组件的过程让我们对 .zip 格式产生了兴趣：操作系统默认是把 .zip 当成什么类型的文件来看待的呢？

```
D:\Test> assoc .zip
.zip=CompressedFolder
```

这个 CompressedFolder 文件类型默认是怎样打开的呢？

```
D:\Test> ftype CompressedFolder
CompressedFolder=rundll32.exe zipfldr.dll,RouteTheCall %L
```

原来如此。Windows XP 通过 rundll32 程序调用 zipfldr.dll 中实现的 RouteTheCall 函数来打开类型注册为 CompressedFolder 的文件，这样的文件会被 zipfldr.dll 当作 zip 压缩文件处理。那么，我们只要把 .jar 后缀注册为 CompressedFolder 类型，Windows 资源管理器就会把 .jar 文件等同为 zip 压缩文件，能够直接对其中的内容进行搜索。这个过程可以在软件安装目录下执行，既不需要复制任何文件，也不必重命名任何文件（命令注释以 # 开头，下同）：

```
# 记下 .jar 文件的原始类型
D:\Office> assoc .jar
.jar=jarfile
# 重新注册 .jar 的文件类型
D:\Office> assoc .jar=CompressedFolder
.jar=CompressedFolder
# ... 直接搜索 Java 类文件
# 任务完成，改回原始类型
D:\Office> assoc .jar=jarfile
.jar=jarfile
```

（对 rundll32 程序感兴趣的读者可以参考微软知识库文章 [3]。）

4. 高效的方案

上述方案巧妙利用了 Windows 系统中的文件类型注册机制，省去了文件复制的开销，所有准备和收尾工作在被测软件之外进行，对系统和被测软件不造成任何影响，是一个比较优雅的改进。但是仍然存在两个问题。

1）如前所述，Java 类名必须手动输入。zipfldr 的搜索功能似乎只在 Windows 资源管理器中有效，无法在命令行下调用，限制了批处理的可能，自动化程度仍然不够。

2）类型重注册存在一个隐患：如果软件安装目录包含有 .zip 文件，在搜索时会将 .jar 文件和 .zip 文件一并处理。如果 .zip 文件过多或过大，就会成为测试过程的人为瓶颈。

第 2 个问题不难解决，可以先将 .zip 后缀临时注册为其它类型，待搜索结束后再改回来。剩下的问题是，怎样对大量需要测试的接口自动化处理呢？我们在网上搜索了一些工具，效果都不能令人满意。对于这样并不算复杂，但现成的测试工具偏偏难以处理的“定制”问题，与其多花精力去寻找刚好合适的工具，不如自己写程序实现。

Python [4] 是一种简洁实用的脚本型通用编程语言，自带功能丰富的标准库。其中，zipfile 库 [5] 提供了处理 zip 压缩文件的功能，利用它容易实现在 jar 包中对指定的文件进行搜索：

```
import zipfile  # 导入 zipfile 库

# find 函数：在 jarfile 中搜索 target 文件
def find(target, jarfile):
    # 将 jarfile 作为 zip 打开
    jar = zipfile.ZipFile(jarfile)
    # 获取内容信息列表
    ilist = jar.infolist()
```

```
    jar.close()
    # 遍历内容信息列表
    for info in ilist:
      # 如果包含指定的文件名
  if target in info.filename:
    # 打印具体路径
        print '%s --> %s/%s' % (
          target, jarfile, info.filename)
```

os 库 [6] 中的 walk 函数提供了强大的文件夹遍历功能，结合上面实现的 find 函数容易实现对所有开发接口的自动化搜索。以下代码假设所有待测接口被分行保存在文本文件 interfaces.txt 中：

```
import os  # 导入 os 库

# 读取所有接口
with open('interfaces.txt') as itf:
  targets = itf.read().splitlines()
# 遍历当前文件夹
for r, ds, fs in os.walk('.'):
  # 对其中每个文件
  for f in fs:
    # 如果是 .jar 文件
  if f.lower().endswith('.jar'):
    # 对每个接口
      for target in targets:
        # 进行搜索
        find(target, os.path.join(r, f))
```

以上短短十几行代码，就实现了我们希望的全自动化测试。所有接口的测试一次完成，对被测软件也不造成任何影响，甚至可以在被测软件运行的同时进行测试。这个方案的高效之处，不仅在于脚本程序不重不漏地实现了完成测试的必要步骤，也体现在代码简洁易懂，编写调试极为方便上。

Python 作为一种解释执行的动态高级语言，运行效率自然比不上 C/C++ 这样贴近系统层的编译型语言，但在我们的测试中，运行效率的瓶颈在于频繁的磁盘文件访问而不是内存中程序的执行，这样一来，使用 Python 和使用 C/C++ 完成同样的任务，执行时间也不会相差太多。

更值得注意的是 Python 程序在编写效率上的巨大优势。简洁精巧的语法规则和高度动态的类型系统结合，赋予了 Python 强大的表达能力，而丰富的“自给自足”（"Batteries included"）的标准库向用户呈现了较为统一的编程接口，降低了学习使用新库的门槛。所以，上面的程序可以说是一挥而就，也不费什么时间调试，唯一的额外操作就是在测试机上安装一个 Python 运行环境。

相比之下，囿于基本库的缺乏，编写相同功能的 C/C++ 程序无疑会麻烦很多。获取 zip 文件的内容信息列表可以借用开源的 zlib 库，遍历文件夹用 Windows API 实现比较繁琐，笔者想到的最佳方案是使用 Boost 库中的 Filesystem 子库。不过，两个库都必须先编译才能调用，而且分别使用了传统的 C 编程接口和高度模板化的 C++ 接口，用户不得不花费时间去理解两种迥异的抽象思路，实在得不偿失。

我们也可以把 Python 和较为“中庸”的 .NET 与 Java 平台略加比较。后二者也具备功能丰富的标准库，并且都有较为成熟的静态和动态语言的实现。这时 Python 的优势体现在，如果测试机不方便安装新程序，我们还可以把在其它计算机上编写的脚本程序和相关的库文件以及 Python 解释器 DLL 打包成一个独立的 exe 文件 [7]，大小通常为 5 兆字节左右，然后复制到测试机上作为“绿色”程序直接运行，实现了所谓的 xcopy 部署。

据笔者所知，.NET 和 Java 的标准做法都要求在被测机上安装完整的运行时库（Runtime）才能够运行程序，运行时库安装后的大小从数十兆到上百兆不等。因此，Python 运行环境的可裁剪性是其明显优势。

还应当补充说明一下上面使用的包含所有待测接口的文本文件是怎样生成的。被测软件厂商通过纸质编程参考手册提供了需要测试的所有接口，我们只能根据手册逐条抄写到一个文本文件。假如编程接口是很规整地排列打印在一起，可以考虑先扫描、再使用 OCR 软件识别、然后通过正则表达式匹配进行提取，虽然需要校对，一般仍比手工抄录快一些。

我们在实际测试时拿到的手册并不符合上述条件，但幸运的是，接口测试并不要求全部接口一次抄完再启动，完全可以边测抄好的部分边抄剩下的部分，即流水化作业，而抄写工作也可以分配到多人并行完成，使测试效率进一步提高。

5. 反思

我们测试时面对的实际情况远没有上面讨论的那么复杂，但借这个场景对可能影响测试效率的问题作一次扩展分析，也不失为一种有益的思考练习。过度优化容易弄巧成拙，使求得“最佳解”的代价超出直接采用简单方法的代价，不过从另一

个角度看，在日常测试工作中勤于思考和总结，习惯于把特定问题推广为一般问题和大规模问题，就会有备无患，遇到真正的困难也不致于手足无措。

测试一个软件，可以全手动操作，可以借助操作系统提供的功能，可以寻找和使用合适的测试工具，也可以自己编写程序来实现。一般而论，这几类方法的难度依次升高，局限性则依次降低。

测试人员究竟需不需要懂得编程？国外的代表性观点是，测试人员不仅要懂编程，而且水平还应比编程人员更高，否则难以发现软件中的缺陷。

国内的现状是，测试人员的编程能力普遍较弱，测试基本还是靠“选工具—配参数—跑用例—记结果”的简单套路，缺陷分析和协助调优的高附加值劳动只占很少比例，直接导致测试工程师的薪资水平远低于开发工程师、测试环节在软件开发流程中难以充分发挥质量保证提升的应有作用。

笔者认为，高效的开发和测试都要求具备“计算机化思维”，通过对计算机原理的深刻理解，较为准确地把握软硬件系统的运行状况，在遇到问题时就能比较迅速地发现症结并加以解决。这种思维的自然表达方式是驱动计算机去完成一些任务，使用的工具就是编程语言。

因此，测试人员在掌握专用的测试工具之外，应该学会使用至少一种通用编程语言，使“理解计算机”和“解决实际问题”的能力得到互相促进。过于依赖测试工具的后果是为工具所束缚，陷入“只会解决几类特定问题”的尴尬。

对于测试人员来说，通用型的脚本语言，如 Python、Ruby、Perl 等，都是很好的选择。它们共同的特点包括简洁灵活，方便快速开发调试，自带丰富的类库，可移植性强等。

值得强调，这些脚本语言既是通用型语言，可以直接用来构建复杂的应用，也适合作为“胶水”来粘合现有的工具，快速实现一些从零开发比较费时的功能。比如在上面的接口测试中，假设 Python 没有提供 zipfile 库，我们可以在脚本中调用某些能够输出 zip 内容列表的命令行工具，如开源的 unzip、7-Zip、JDK 包中的 jar.exe 等，读取它们的输出然后进行解析。这种能力无疑扩大了脚本语言在实际测试中的应用范围。

参考文献

[1] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha, The Java Language Specification (3rd Edition), Chapter 7. Package, http://java.sun.com/docs/books/jls/, Addison Wesley Professional, 2005.

[2] Oracle, Java SE Documentation — JAR File Specification, http://download.oracle.com/javase/6/docs/technotes/guides/jar/jar.html, 2010.

[3] Microsoft, KB 164787 — Windows Rundll 和 Rundll32 接口，http://support.microsoft.com/kb/164787.

[4] Python Programming Language Official Website, http://www.python.org/.

[5] zipfile — Work with ZIP archives, http://docs.python.org/library/zipfile.html.

[6] os — Miscellaneous operating system interfaces, http://docs.python.org/library/os.html.

[7] py2exe — build standalone Python executables for Windows, http://www.py2exe.org/.

作者简介

杨珈，男，1980 年生，毕业于中国科学技术大学电子工程与信息科学系，获博士学位。现任中国软件评测中心重大专项测试部软件研发测试工程师，关注网络安全领域。

【上接第 106 页】

及其安全测试自动化的研究[D]. 北京：北京交通大学，2007.

[5] 陈晨. 操作系统安全测评及安全测试自动化的研究[D]，北京：北京交通大学，2008.

[6] 牛晗晖. Linux 系统调用及其安全测试自动化的研究[D]，北京：北京交通大学，2009.

[7] 李耀东. Linux 操作系统存取访问控制机制的研究[D]，北京：北京交通大学，2008 年.

[8] 金怡. 安全操作系统自主访问控制与渗透测试[D]，北京：北京工业大学，2007.

基金项目："核高基"科技重大专项（the Core Electronic Components, High-end General Chips and Basic Software Foundation of China under Grant No. 2008ZX01045-001, 2009ZX01045-004, 2009ZX01045-005）；

电子信息产业发展基金（the Electronic Information Industry Development Foundation of China）。

作者简介：

蒋发群（1977-），男，博士，主要研究领域为软件和信息系统测试。

李艳波（1982-），男，硕士，主要研究领域为软件和信息系统测试。

办公软件接口自动化测试过程改进经验谈

作者：杨玚
作者单位：中国软件评测中心,北京,100048
刊名：信息安全与技术
英文刊名：INFORMATION SECURITY AND TECHNOLOGY
年，卷(期)：2010(9)

参考文献(7条)

1. py2exe-build standalone Python executables for Windows

2. os-Miscellaneous operating system interfaces

3. zipfile-Work with ZIP archives

4. Python Programming Language Official Website

5. Microsoft,KB 164787-Windows Rundll 和Rundll32 接口

6. Oracle,Java SE Documentation-JAR File Specification 2010

7. James Gosling;Bill Joy;Guy Steele;Gilad Bracha The Java Language Specification (3rd Edition),Chapter 7.Package 2005