基于元数据的构件自动测试技术研究

赵明华[1]， 陈 榕[2]， 王小鸽[2]
(1.清华大学深圳研究生院软件工程中心，深圳 518055；
2.清华大学信息技术研究院操作系统与中间件技术研究中心，北京 100084)

摘 要：针对构件技术的特点，提出并采用新的测试模式实现构件的自动测试。首先分析了基于源代码的自动测试工具的局限性，并根据构件的自动测试的特点，提出了基于元数据的构件自动测试方法。接下来介绍了采用该方法设计的构件自动测试工具的原理、部件和功能，通过测试结果的比较并结合工程实践的应用说明了此种自动测试方法的可行性。

关键词：自动测试；构件技术；元数据；CAR 技术；基于构件的软件
中图分类号：TP311.5

Research on automated testing technique for component based on metadata

ZHAO Ming-hua[1], CHEN Rong[2], WANG Xiao-ge[2]
(1. Software Engineering Center, Shenzhen Graduate School, Tsinghua University, Shenzhen 518055, China;
2. Operating System and Middleware Technology Research Center, Tsinghua University, Beijing 100084, China)

Abstract: The limitations of source-code-based automated testing tools are investigated, the properties of automated component testing are discussed, and an approach based on component metadata is proposed. In particular, a testing tool is presented with its theory, design principles and main functions. The preliminary test results and comparison with other automated testing tools has shown that the method is practical from the view of the software engineering practice.

Key words: automated testing; component technique; metadata; CAR technology; component-based software

1 引 言

自动测试是软件测试环节的一部分，利用软件测试工具完成一些重复率高、运行频繁或手工输入困难的测试，可以节约工程成本、缩减开发周期。目前应用的软件测试工具种类很多，如企业级测试工具 WinRunner，性能负载测试工具 LoadRunner，单元测试工具 CPPUnit、JUnit 等，都具备一定的自动测试能力。但从测试原理来看，它们所进行的都是基于源代码的测试，测试人员如果使用这些工具编制自动测试程序，不但要熟悉测试对象的源代码，而且经常需要调试测试脚本，不可避免地造成了工时的延长和交流的冗余。

随着构件技术的发展，基于构件的软件设计方法正逐渐成为新的趋势。构件作为软件形态新的载体，应用日益普遍，构件的测试技术方面随之产生了一些新问题[1]。在这种条件下，传统的自动测试工具，由于其设计模式[2]的局限性，已经不能适用于构件的自动测试。

这些工具在测试构件产品时，仍然采用基于源代码的测试模式，要求待测的构件提供特定的测试接口，或者在实现代码中嵌入测试语句。对于已发布的构件成品来说，其接口不变性和源码隐藏的特点，将给自动测试工作带来很大的困难。

本文在研究构件测试技术的基础上，提出了一种适用于构件的自动测试方法，并完成了相关测试工具初级版本的开发。相比于其他的测试工具，其优势在于：

1. 采用基于元数据[3]的测试手段，直接使用构件作为操作对象，通过分析元数据信息，自动生成测试代码。
2. 动态生成的测试代码采用规范的格式书写，方便测试人员修改测试代码，形成新的测试程序。
3. 测试用例数据以配置文件的形式输入工程中，测试人员可以方便地定制测试数据。
4. 构件升级后，只要运行本工具，就能自动生成适合新构件的测试代码，测试人员无需做任何源码的修改。

2 源代码模式和元数据模式的比较

基于源代码的自动测试工具的测试模式是，一般由用户编制测试脚本，测试工具根据模块的实现源码进行语法分析，生成测试代码，或者由用户在被测模块源码的适当位置插入一些测试函数，参与模块代码的编译。在测试运行时，测试控制台监测并输出模块的测试数据，汇报模块的运行情况。

这种测试模式对于面向对象的模块或单元测试来说，是行之有效的，但如果运用于构件的测试，则无法体现构件技术的优势，存在着明显的局限性。将基于源代码的测试模式同基于元数据的测试模式做一比较，前者的不足之处主要体现在五个方面，如表 1 所示。

表 1 基于源代码和基于元数据的测试模式的比较

基于源代码的测试模式	基于元数据的测试模式
测试依赖于源码的获取，对构件的使用者来说，这一要求有时无法得到满足。	从构件直接获取元数据，生成测试代码，无需借助于构件的实现代码。
插点式的编程模式，导致测试时系统的代码量增大，编译时间延长，而且在一定程	测试代码动态生成，不会改变构件的实现机制。构件在异进程独立运行，效

度上降低了模块的运行效率。测试结果可能不真实。	率不会受到影响。测试结果具有可靠性。
对于测试计划中的任何项目模块，其中可能存在的某些未知错误会导致整个测试过程的中止，或者会影响其他项目的测试数据。	采用异进程运行机制，构件中可能存在的未知错误发生时，不会引起测试进程的崩溃，也不会影响其他项目的测试结果。
如果模块内部的函数名称、参数个数或参数类型有所改动，原有的测试就不再适用，需要重新产生或修改测试代码，重新编译。这是一件繁琐的工作。	构件升级时，能够从中获取升级之后的元数据，根据它们生成新的测试代码，具有接口一致性。保证了测试的可重用性。
模块在编译期进行链接，没有动态加载的机制，因此无法通过网络进行远程的自动测试。	构件能够动态加载、位置透明的特点，为远程的自动测试实现提供了可能。

可以断言，传统的基于源代码模式的自动测试工具已无法适应构件二进制封装、动态升级、动态加载的特点，不能满足构件的自动测试的需求。采用基于元数据的测试模式，才能更好地承担构件的自动测试工作。

3 构件技术与自动测试

3.1 构件技术的特点

随着软件技术的发展，软件的构成有着大型化、工厂化的趋势，“面向对象编程”模型的缺点日益突出，逐渐被“面向构件编程”的模型所取代。为适应不同软件开发商提供的构件模块能够可以相互操作的需求，构件之间的连接和调用要通过标准的协议来完成。构件化编程模型提供了动态构造部件模块的机制，构件在运行时是动态加载的。其代表是 COM 技术。[3]

构件技术本身的特点，使构件的自动测试相比于面向对象编程的软件的自动测试，有了更为突出的优点：

1. 测试构件只要经过元数据的提取，就能获得所测的类、接口、方法的信息，测试人员无需关心源代码的实现。这样就做到了开发与测试的分离，节约了开发成本。
2. 构件的标准接口保证系统可分解成多个功能独立的单元，为测试工作计划的制定、任务的划分提供了方便。
3. 构件独立于编程语言，只要接口符合构件的协议标准，都可作为测试对象。与此同时，测试代码也可以采用不同的语言来编写。

3.2 构件的自动测试工具设计目标

基于上述分析，在 CAR(Component Assembly Run-Time)构件技术[3]的支持下，作者设计了一个基于元数据的构件的自动测试工具 CARBAT(CAR Blind-Automated-Testing)。

CARBAT 的设计目的在于辅助完成构件的黑盒测试、单元测试、回归测试等自动测试工作。其系统目标为：

1. 自动分析构件。CARBAT 能够自动分析待测构件的元数据，无需借助于构件的实现代码。
2. 测试用例自动选取。测试用例以配置文件的形式存在，作为系统输入的一部分。CARBAT 根据构件的函数参数类型自动选取测试数据。
3. 动态生成测试程序。CARBAT 根据构件的元数据和测试用例动态生成测试程序，保证了构件升级时测试的可重用性。同时，允许用户修改定制测试程序。
4. 测试功能易于扩展。多线程测试、多进程测试、压力测试等测试方法可以很方便地添加到已有的系统中。
5. 灵活的功能开关。为每种测试方法设置功能开关，测试控制台根据开关状态决定具体执行的测试方法组合。
6. 全程自动化控制。测试用例的生成、测试代码的产生和编译运行、测试结果报告的输出，全过程由 CARBAT 自动完成。
7. 界面友好，易于操作。CARBAT 提供命令行和图形界面两种输入方式，操作简单。

3.3 CARBAT 与其他测试工具的比较

表 2 显示了 CARBAT 与 TestComplete[4]，RhiTech Code Testing Tool[5]三种可用于测试构件的工具的比较结果。

表 2　三种构件测试工具的比较

	RhiTech	TestComp-lete	CARBAT
测试模式	基于源代码编写测试代码	利用脚本语言编写测试代码	读取元数据生成测试代码
测试脚本	手工编写	辅助生成	自动生成，可定制
测试代码	手工编写	生成测试框架	自动生成，可定制
测试结果	自动生成图表和日志记录	可录制并回放测试过程	自动输出结果和测试报告
可测构件类型	COM	COM，.NET 和 CORBA 等	COM，CAR 等
构件升级时测试的复用性	不能复用	需要重新生成测试脚本代码	自动生成新的测试

可以看出，RhiTech 基于构件开发者的角度，侧重于源代码级的测试；TestComplete 基于构件测试者的角度，侧重于测试脚本代码的编写和过程录制。CARBAT 同样从构件测试者的角度出发，侧重于测试过程的自动化和测试的复用性。

4 开发构件的自动测试工具

CARBAT 工具已在“和欣”操作系统中初步开发完成。“和欣”操作系统是科泰世纪有限公司开发的基于 CAR 构件技术的嵌入式操作系统[3]，提供了 CAR 构件运行平台。

4.1 自动测试的原理

归纳来说，构件的自动测试的进行，在原理上分为三步：

1. 构件 dll 的解析，获取元数据信息；
2. 根据测试数据，生成测试程序；
3. 运行测试计划，输出测试结果和测试报告。

首先，CARBAT 需要对构件进行解析。在构件 dll 中，存在类的描述信息（Class Info），从中可以获得该构件所包含的类名、类的数目、接口名、接口数目、函数的名称、返回值类型、参数个数、参数传递类型和数据类型等详细信

息，这些是构造自动测试程序[6]的基础。类信息的结构如图1。

CARBAT 通过类信息的装载操作得到构件元数据的相关信息，建立相应的类、接口、函数关系表格，为测试代码的书写做好准备。

其次，CARBAT 进行测试程序的构建和书写。在测试程序的头文件中，自动加入待测函数个数、线程数目、测试级别等定义，并按照约定的命名规范对待测函数进行声明。在测试程序的源代码文件中，自动加入类的实例化代码，并且在头文件中声明的函数内部自动填入规范的测试代码。

最后，当测试程序书写完成之后，CARBAT 自动运行测试程序，输出接口函数的详细信息、测试数据的执行情况，函数的返回类型和返回值。

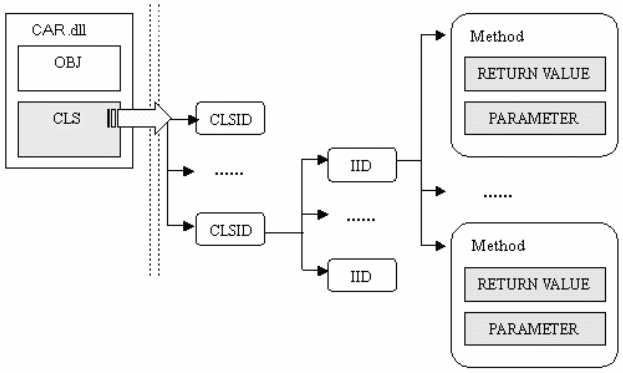

图 1　CAR 构件的元数据信息示意图

CARBAT 采用异进程（CTX_DIFF_PROCESS）实例化的方式，当某些输入数据导致构件运行发生异常甚至崩溃时，测试进程不受影响，并能够汇报错误发生地和错误代码，由此保证了 CARBAT 本身的稳定性，异常发生时测试计划仍然可以完成。

4.2 设计的主要部件

根据 4.1 节的原理分析，设计出 CARBAT 工具的工作流程图，如图 2 所示。

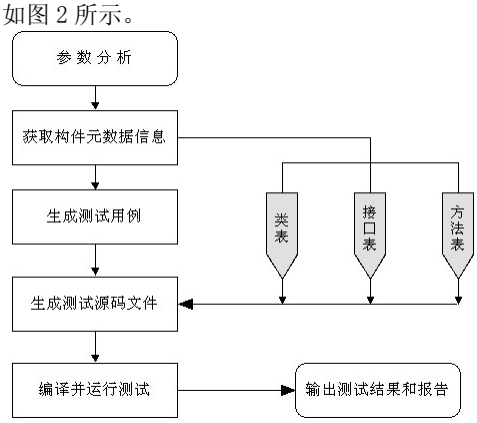

图 2　CARBAT 工具工作流程图

从图 2 可以看出，CARBAT 的体系结构，主要由两部分组成。第一部分是测试生成器，主要部件有：参数分析器、TLB 信息获取器和代码产生器，其中输入部分是待测构件和测试用例配置文件；输出部分是测试代码，包括头文件、源码文件和编译文件。第二部分是测试运行器，主要部件为自动编译运行测试器和测试结果报告生成器。其中输入部分是第一阶段产生的测试文件；输出部分是测试结果和测试报告文件。

测试用例采用配置文件的形式输入。在默认的配置文件中，根据不同的数据类型，设计了一些常用值和边界值作为输入条件，如 32 位整型数据，包括-2147483648，-1，0，1，2147483647 等。用户也可以根据自己的需要对配置文件进行修改，增删数据都很方便。

测试文件采用规范的格式书写，可读性良好，用户也可以根据自己的需要修改测试文件，重新编译运行，得出新的测试结果。

4.3 目前实现的功能

目前，经过在“和欣”操作系统环境中的实践，CARBAT 已经能够很好地完成自动分析目标构件、自动生成测试程序、自动执行生成的测试程序、自动生成测试报告等功能。

CARBAT 支持测试构件接口函数常用的参数类型。目前的可测参数类型包括：

1. 基本数据类型，如 char，int，double，float，short，long 等，以及无符号数据类型，如 unsigned int，unsigned long 等；
2. 扩展的构件技术相关的数据类型，如 HRESULT，GUID 等；
3. CAR 技术所特有的自描述数据类型，如 EzBuf，EzStr，EzPoint 等；
4. 枚举类型（enum），以及经过 typedef 定义的别名等特殊数据类型。

在运行参数上，CARBAT 专门设置了多线程开关，当开关打开时，工程自动执行构件对象的多线程测试。通过多线程的自动测试，能够进一步提高构件的实际应用价值。

由于构件的语言无关性，使用任何语言编译器开发的标准构件均可作为测试对象，这也扩展了 CARBAT 工程的应用领域。

4.4 一个应用的例子

CARBAT 实际应用的优点之一是能够发现某些基于源码的手工测试难于发现的 bug，下面仅举一例作为说明。

考虑下面的 C++程序段，这里 g_nCount 为整型全局变量，函数 InterlockedIncrement 执行原子操作，其功能是将其参数指针所对应的变量值加 1：

```
int g_nCount = 0;
int GetUniqueCount()
{
    InterlockedIncrement(&g_nCount);
    return g_nCount;
}
```

如果多线程执行这段代码，有一个潜在的 bug：由于线程调度的随机性，很可能造成某个线程刚执行完 InterlockedIncrement，在 return 之前另外一个线程又执行 InterlockedIncrement，结果对全局变量 g_nCount 而言，其值自加了两次，造成了返回结果的语义错误。

采用基于源代码的手工测试方式，跟踪每个线程得到的 InterlockedIncrement 结果都是正确的（g_nCount 自加 1），因此难于找到这类潜在的 bug。

而 CARBAT 通过设置线程数目，模拟构件运行在多量线程时的执行情况，提高了这类 bug 的重现几率。经过对 GetUniqueCount 返回值的跟踪，很容易发现和报告该 bug。

5 结 论

本文从自动测试的需求出发，针对构件可获取元数据的特点，提出了一种新的构件自动测试方法。该方法与传统的基于源代码的测试模式不同，直接以编译好的构件 dll 作为操作对象，经过元数据的提取、测试数据用例的加入、参数类型的处理，能够自动完成从生成测试代码、编译运行测试，到输出测试结果、打印测试报告的全过程，给构件软件的开发者和测试工作人员提供了很大的方便，在软件工程中的测试环节节约成本、提升效率等方面均有积极作用。

该法也易于推广到其他的构件平台之上，如 COM，CORBA，JAVA 等。

下阶段的工作目标是结合构件服务的注册、查找、激活机制，探索实现 CARBAT 对远程测试的支持。

参考文献：

[1] Adrita Bhor.Software Component Testing Strategies[EB/OL].http://www.ics.uci.edu/~abhor/ics221/comp_test.htm,2001.

[2] C.Szyperski.Component Software-Beyond Object Oriented Programming[M].Addison Wesley,1997.

[3] 科泰世纪科技有限公司.Elastos 资料大全[Z].2004.

[4] TestComplete-Automated Software Testing Tool[EB/OL].http://www.automatedqa.com/products/tc.asp,2004.

[5] RhiTech code testing tool documentation[EB/OL]. http://rhitech.com/Products/TestsEmulator/document.htm, 2004.

[6] 科泰世纪有限公司.基于构件定义语言自动生成 C++程序的方法[P].中国专利号:03100827.5,2003.

基金项目：国家“863”高技术研究发展计划项目（编号 2001AA113400；2003AA1Z2090）

作者简介：

赵明华(1980-)，男，河北省徐水县人，硕士研究生，研究方向为构件化文件系统、构件测试技术；

陈榕(1957-)，男，清华大学信息技术研究院操作系统与中间件技术研究中心副主任，上海科泰世纪有限公司 CEO、首席科学家，研究方向为网络操作系统，构件技术；

王小鸽(1957-)，女，博士，副教授，清华大学信息技术研究院操作系统与中间件技术研究中心副主任，研究方向为并行与分布式计算，软件开发方法。