第 33 卷　第 4 期　　计　算　机　工　程　　2007 年 2 月
Vol.33　No.4　　Computer Engineering　　February 2007
·开发研究与设计技术·　文章编号：1000—3428(2007)04—0270—02　文献标识码：A　中图分类号：TP311.56

# 一种API自动化测试工具的设计与实现

崔红军[1,2]，饶若楠[1]，邵培南[2]
(1. 上海交通大学计算机科学与工程系，上海 200030；2. 华东计算技术研究所，上海 200233)

**摘　要：**给出一种 API 自动化测试工具的设计和实现方案，实现了被测 API 信息的自动提取、API 测试数据和测试用例辅助生成以及测试执行过程驱动与监控的自动化。介绍了使用该工具对一个实际的被测程序进行测试的过程和结果。
**关键词：**软件测试自动化；API 测试；软件测试驱动；软件测试用例生成

# Design and Implementation of An Automated API Test Tool

CUI Hongjun[1,2], RAO Ruonan[1], SHAO Peinan[2]
(1. Dept. of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200030;
2. East China Research Institute of Computing Technology, Shanghai 200233)

**【Abstract】** This paper intends to explore the ways to realize the automatic test tool for retrieving the tested API, generating the test data based on API together with the test case generated and the driving of the execution of test and monitor automatically. In the end, the results and procedures for testing an exact program with the tool are presented.
**【Key words】** Automatization of software test; API test; Driving of software test; Generation of software test case

软件测试作为提高软件质量的主要手段越来越受到人们的重视。软件测试阶段分单元测试、部件测试、配置项测试及系统测试，单元测试在编码阶段进行，该阶段测试质量越高越能减少软件后期测试所发现的错误数量，从而降低纠正错误的成本。单元测试中使用较多的是应用编程接口(Application Program Interface, API)测试。API 测试需要大量的编程工作，具有数量多、耗时长的特点，手工测试效率低，因此迫切需要研制针对 API 的自动化测试工具。这方面的工具国外已有较成熟的产品出现，典型代表如 ParaSoft 公司的 C++Test 和开源的 JUnit 等。

## 1　工具框架设计

本文提出的 API 自动化测试工具框架针对 C/C++语言，主要由源代码分析模块、测试用例设计以及测试用例执行模块构成，该工具的主要框架如图 1 所示。

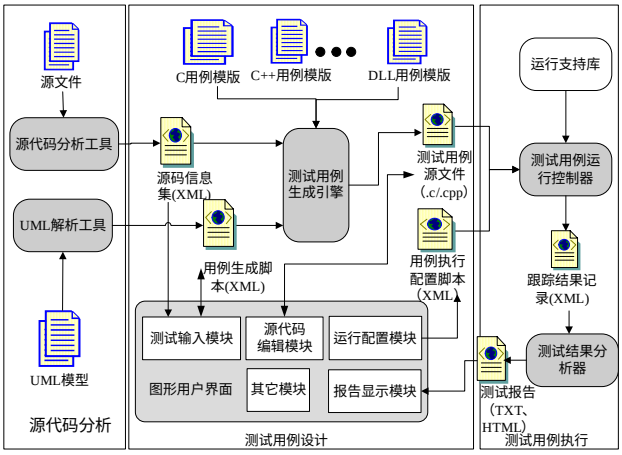


**图 1 API 自动化测试工具主要框架**

从图 1 中可以看出，该工具能够自动分析 C/C++头文件和源文件以及统一建模语言(Unified Modeling Language, UML)模型来产生 API 信息。测试用例设计模块利用这些 API 信息，结合用户输入的测试用例数据而自动生成测试用例源文件。测试用例执行模块根据用户的运行配置生成执行配置脚本，并自动完成所有程序的编译过程，得到测试用例的可执行文件，在用例运行过程中进行调度和控制，最终得到测试结果和测试报告。

考虑到今后工具的可移植性，本工具的开发语言采用 Java。该框架中构成系统工具的各个分模块之间尽量通过文件作为连接接口，这样可以大大降低系统的耦合度，以保证分工具的更新对系统其它工具的影响降到最低。

模块间的传输文件以 XML(Extensible Markup Language)的格式文档进行保存并供其它模块使用。常见的信息保存方式有纯文本、数据库和 XML 文档等方式。对于本工具而言，纯文本方式对复杂数据力不从心，而使用数据库方式又有点浪费。因为 XML 在数据交换和处理方面有独特的优势，再加上具有可定制等特征，所以成了本工具的首选。

## 2　系统实现

### 2.1　源代码分析模块

源代码分析模块主要是对被测 API 的头文件和源文件进行预处理、词法和语法分析，得到包含头文件信息、宏信息、用户自定义等信息，而且能提取出函数信息，包括函数的参数，类、类间关系，全局/静态变量等，并将分析的结果进行分类保存在 XML 文档中。

该模块的主要设计思想是使用 Lex 和 Yacc 工具构造功能不同的词法、语法分析器对软件源代码的扫描、分析，产

**作者简介：**崔红军(1974－)，男，硕士，主研方向：软件工程，软件测试；饶若楠，副教授；邵培南，研究员
**收稿日期：**2006-05-08　**E-mail：**cuihongjun@ecict.com.cn

生分析结果。Lex 工具是一种词法分析程序生成器，它可以根据词法规则说明书的要求来生成单词识别程序，由该程序识别出输入文本中的各个单词。yacc 工具是一种语法分析程序生成器，它可以将有关某种语言的语法说明书转换成相应的语法分析程序，由该程序完成对相应语言中语句的语法分析工作。关于这方面成熟的商用及开源产品很多，本工具引用了本单位以前的研究成果，本文不再展开。

考虑到目前 UML 在软件开发和设计中的日益普及，因此本工具提供一个 UML 模型文件(如 Rational Rose 生成的 mdl 文件)分析工具作为 API 信息获得的一个辅助手段。UML 模型文件中不仅包含着 API 信息，还包含被测 API 的状态转换信息、前置条件和后置条件等信息，更重要的是这些信息对后面的用例设计非常有用。Rose 模型的文件格式是固定的，在 UML 规范中有明确的格式定义，因此对这些信息的提取也非常容易。

### 2.2 测试用例设计模块

测试用例生成模块是本测试工具的核心模块，图 1 中源代码信息集(XML)就是源代码分析工具产生的被测 API 信息，用例生成脚本(XML)为测试人员输入的测试用例而被工具保存的信息，测试用例生成引擎使用生成算法根据用例模板自动生成测试用例的源文件(.c/.cpp)，以供执行模块使用。

(1)用例源文件的组织

实际应用中一个项目存在多个针对不同 API 的测试用例，为更高效地管理所有测试用例，必须引入一定的分层组织机制，我们将一定数量的用例组成一个测试序列(TQ，Test Sequence)，测试序列主要达到一个对测试用例逻辑分组的目的，该序列下的测试用例按照用户指定的顺序执行。一个测试源文件中可包含多个 TQ。

每个测试用例在源文件中就是一个函数的形式(称之为 TP)，这样既有利于程序的自动化生成，也方便用户更改生成的源文件。在函数体中至少包括初始化、被测函数调用、结果输出检查等内容。测试序列对其下所有测试用例函数的控制，以一个函数指针数组的形式来实现。定义一个 TQ 使用的结构体：

```
struct TQlist {
void (*TPfunc)(void);
int TQpos; };
struct TQlist testlist[];
```

TPfunc 是函数指针，指向各测试用例函数源代码文件中定义的地址。TQpos 说明该函数在 TQ 中的顺序。这样通过初始化 testlist 数组的过程来完成各测试函数的注册。

为增强生成代码的效率，在源文件开头和结尾部分可包括两个函数 StartUp()和 CleanUp()。StartUp()用来在执行用例程序前申请测试所需的公共系统资源，例如内存的分配等。CleanUp()负责在全部测试结束后的收尾工作，如释放原先申请的系统资源。

(2)用例生成的其它策略

C/C++测试用例源文件的生成是很复杂的。有些信息可能必须由用户手工干预才能完成，如全局变量的初始化，一个类存在多个构造函数时使用哪个生成本次测试的对象等。

每个测试用例相关数据(参数的赋值、预期的结果等)的输入可结合手工输入、基于等价类划分和边界值分析方法工具自动产生的方法来提供。

简单类型的结果输出检查可定义一个返回值接收变量_api_var_return，在每个 TP 内使用 assertion(ExpReturn, _api_var_return)来完成检查。复杂情况的结果输出检查就需要人工添加代码来完成，如返回值是一个对象的情况。

用例源文件生成的规则很多，这里无法一一展开。所有规则在工具中可封装成一个 jar 文件的算法包，工具启动时根据该算法包的定义完成源文件的生成。如进行系统算法更新只需替换该文件即可。

### 2.3 测试用例执行模块

用例执行模块负责将生成的源文件编译链接生成可执行文件，再进行相关执行调度。编译工具使用测试机器上安装的第三方编译工具就能满足需要，如 Microsoft 的 Visual C++编译器、GNU 的 GCC 编译工具，本工具只需根据用户指定的运行配置自动生成 makefile 文件即可。生成机制和用例文件的生成有点类似，也需要参考预先定义好 makefile 的生成模板，根据用户设置的编译信息(使用的编译链接器、预处理宏、依赖文件等)生成 makefile 文件。

目标文件生成后，根据工具提供的运行库支持来完成用例文件的执行。可以使用 Open Group 的 TETware 来辅助工具的运行。TETware 作为一种开源的、命令行的产品而提供了一种能将本地、远程、分布式和实时测试程序组合起来的多平台运行框架。

## 3 实例应用

由于篇幅限制，因此本文不可能将所有工具细节用一二个例子来完全包括。下面就以一个很简单的例子来说明该工具的使用过程及产生的重要文件，有些文件只保留了关键部分。被测函数如下：

```
int Summary(int i,int j)
{ int sum;
  sum=i+j;
  return sum; }
```

对该 API 自动分析获取的源代码信息集的文件部分内容如下：

```
…
<functionList>
 <function startLine="1" endLine="5" isImplement="true">
    <returnType>int</returnType>
    <funcName>Summary</funcName>
    <argList>
      <argument isAtomic="false">
        <argType>int</argType>
        <argName>i</argName>
        <argValue></argValue>
        <argOrder>1</argOrder>
      </argument>
        <argument isAtomic="false">
        <argType>int</argType>
        <argName>j</argName>
        <argValue></argValue>
        <argOrder>2</argOrder>
      </argument>
    </argList>
    </function>
 </functionList>
 …
```

创建一个测试序列 sq1，并在其下创建 2 个用例：第 1

(下转第 274 页)