基于RTRT单元测试

51Testing软件测试网

资深专家：周先生

专题提纲

- 单元测试介绍
- RTRT工具介绍
- 用RTRT搭建单元测试环境的基本过程
- 简单的演示一个单元测试例子

单元测试介绍

- 单元测试概念
- 单元测试目的
- 单元测试对象
- 单元测试环境
- 单元测试策略
- 单元测试活动
- 单元测试工具

单元测试概念

- 软件开发过程中要进行的最低级别的测试活动。
- 单元测试是对软件基本组成单元进行的测试。
- 基本单元不一定是指一个具体的函数或者一个类的方法。
- 具体实现时，也可能对应的是多个程序文件中的一组函数。

单元测试目的

单元测试的目的在于发现各模块内部可能存在的各种错误。

- 验证代码是否与设计相符合；
- 发现设计和需求中存在的错误；
- 发现在编码过程中引入的错误；
- 使得代码重构成为可能。

单元测试对象

单元测试对象一般对应详细设计中所描述的基本单元。

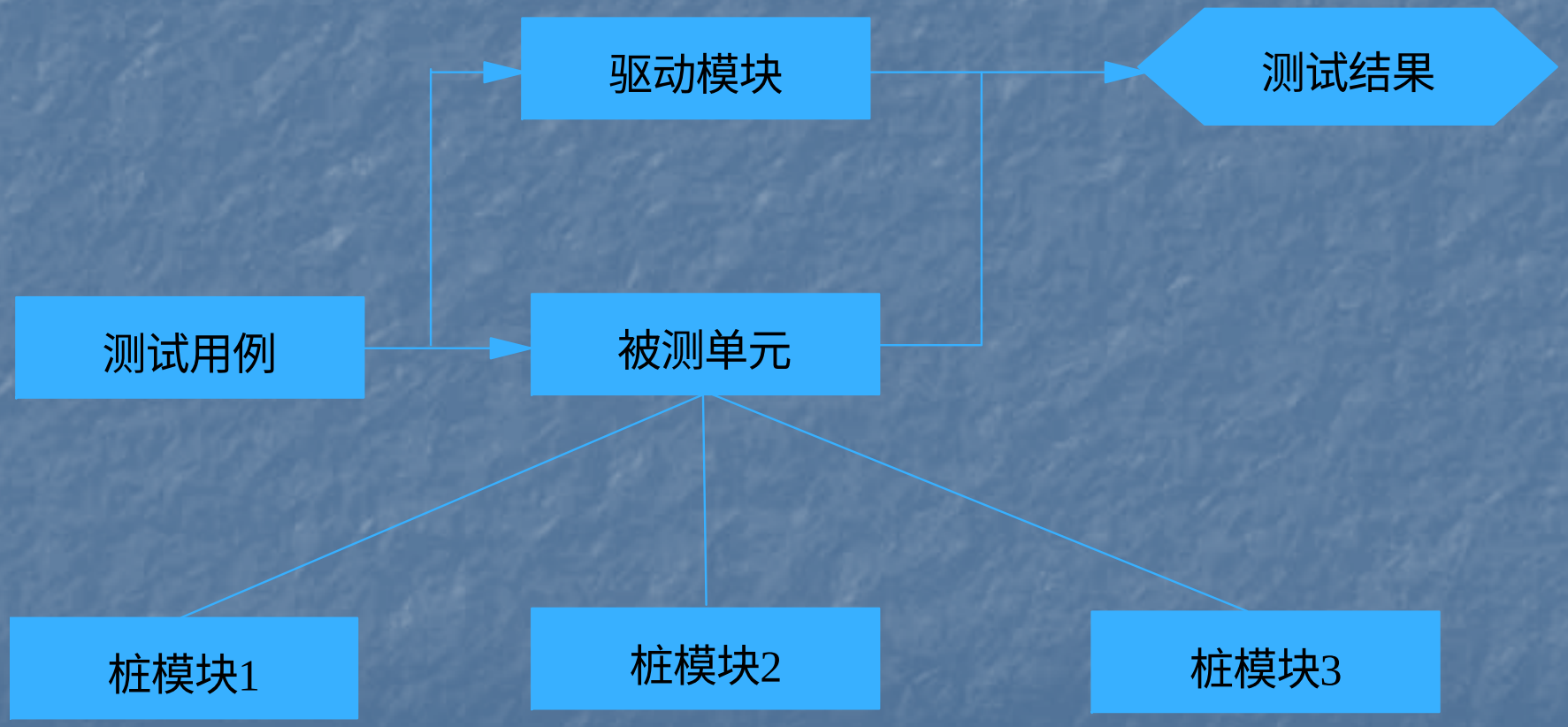
- 结构化编程语言，如C，单元测试对象是函数或者子过程。
- 面向对象语言，如C++，单元测试对象是类或者类的方法。

单元测试环境(一)

基本单元本身不是一个独立的程序，所以必须为每个基本单元开发驱动模块或桩模块。

- 驱动模块(Driver)：被测基本单元的主程序，它接收测试数据，并把数据传送给被测单元，最后输出实测结果。
- 桩模块(Stub)：用来代替被测基本单元调用的其他基本单元。

单元测试环境(二)



单元测试策略

- 自顶向下的单元测试策略
- 自底向上的单元测试策略
- 孤立单元测试策略

自顶向下的单元测试策略

- 方法：先对最顶层的基本单元进行测试，把所有调用的单元做成桩模块。然后再对第二层的基本单元进行测试，使用上面已测试的单元做驱动模块。依此类推直到测试完所有基本单元。
- 优点：在集成测试前提供早期的集成途径。在执行上和详细设计的顺序一致。不需要开发驱动模块。
- 缺点：随着测试的进行，测试过程越来越复杂，开发和维护成本增加。
- 总结：比孤立单元测试的成本高很多，不是单元测试的一个好的选择。

自底向上的单元测试策略

- 方法：先对最底层的基本单元进行测试，模拟调用该单元的单元做驱动模块。然后再对上面一层进行测试，用下面已被测试过的单元做桩模块。依此类推，直到测试完所有单元。
- 优点：在集成测试前提供系统早期的集成途径。不需要开发桩模块。
- 缺点：随着测试的进行，测试过程越来越复杂。
- 总结：比较合理的单元测试策略，但测试周期较长。

孤立单元测试策略

- 方法：不考虑每个单元与其它单元之间的关系，为每个单元设计桩模块或驱动模块。每个模块进行独立的单元测试。
- 优点：简单、容易操作，可达到高的结构覆盖率。
- 缺点：不提供一种系统早期的集成途径。
- 总结：最好的单元测试策略。

单元测试活动

- 单元测试计划
- 单元测试设计
- 单元测试实现
- 单元测试执行
- 单元测试总结

单元测试工具

常见的单元测试工具有：

- 代码静态分析工具：Logiscope，McCabe QA，CodeTest等;
- 代码检查工具：PC-LINT，CodeChk，Logiscopedeng等;
- 测试脚本工具：TCL、Python、Perl等；
- 覆盖率检测工具：Logiscope，PureCoverage，TrueCoverage ，McCabe Test，CodeTest等;
- 内存检测工具：Purify，BoundsCheck，CodeTest等;
- 专为单元测试设计的工具：RTRT，Cantata，AdaTest等

专题提纲

- 单元测试介绍
- RTRT工具介绍
- 用RTRT搭建单元测试环境的基本过程
- 简单的演示一个单元测试例子

RTRT工具介绍

- RTRT简介
- RTRT特点
- TDP基本概念介绍
- RTRT支持的主机系统
- RTRT支持的目标板系统
- RTRT单元测试基本过程

RTRT简介

Rational Test RealTime 软件包是Rational 公司的自动化测试工具集，支持从单元测试到集成测试，到系统的确认测试，从实时的嵌入式系统测试到分布式应用的测试，从标准 C 的测试到基于OO 的C++的测试，它包括：

- Unitest，单元测试；
- Coverage，结构覆盖；
- SystemTest，集成测试和系统测试；
- Trace，时序分析；
- Object Testing，面向对象测试；
- PurifyLT，内存访问分析；
- QuantifyLT，性能分析。

RTRT Unitest简介

自动化单元测试，最大的特点是自动化：

- 自动生成测试用例模板
- 自动生成测试程序
- 自动运行测试程序
- 自动生成测试报告
- 自动调用调试工具

RTRT Coverage简介

提供了9种不同水平的覆盖率测试

覆盖率水平	语言			
块覆盖率	C	Ada	C++	Java
调用覆盖率	C	Ada		
条件覆盖率	C	Ada		
函数、单元或方法覆盖率	C	Ada	C++	Java
链接文件		Ada		
模板			C++	
附加语句	C	Ada	C++	Java

RTRT特点

- 代码静态分析，功能测试和运行时分析相集成 ；
- 代码编辑、测试和调试相集成 ；
- Test RealTime通过分析源代码，自动生成测试驱动（Test Driver）和桩（Test Stub）模版。开发人员只需要在该测试脚本的基础上指定测试输入数据、期望输出数据以及打桩函数的逻辑；
- 测试执行后自动生成测试报告和各种运行时报告。测试报告展示通过或失败的测试用例，而运行时分析报告包括代码覆盖分析报告，内存分析报告、性能分析报告和执行追踪报告；
- 通过Target Deployment Port技术同时支持开发机和目标机的测试。

TDP基本概念介绍

- TDP：Target Deployment Port，目标适配技术。
- 利用TDP技术，使得测试与编译器、连接器、调试器以及目标结构无关，实现了跨多开发环境、多目标结构。

cvisual6.xdp - Rational Test RealTime Target Deployment Port Editor

File Edit Help

- Visual 6.0
  - Basic Settings
    - For C
    - For C++
  - Build Settings
    - Compilation function
    - Preprocessing function
    - Link function
    - Execution function
    - Debug mode function
    - postGen.pl
    - attolcc.exe
    - cl.exe
    - link.exe
  - Library Settings
  - Parser Settings

Debug mode function

Description:

This Perl function should launch the target system debugger. The goal is to enable the user to interactively debug the test program or the instrumented application.

This function is optional, and it is usually very difficult, unless in a Standard-Mode, to generate the test report while debugging interactively at the same time. Indeed, if the method used to retrieve the value of atl_buffer is based on a debugger script executed in an automated way, user-defined breakpoints may break this automation.

This is not a problem as long as the differences between an interactive debugging session and a batch execution of a test program or of an instrumented application are clear.

Scope: All features

Prototype:

atl_execdbg($$$)

{

```
sub atl_execdbg($$$)
{
        my ($exe,$out,$parameters) = @_;

        unlink($out);
        SystemP("msdev.exe \"$exe\"");

}
1;
```

Comment for "Debug mode function" node:

Ready

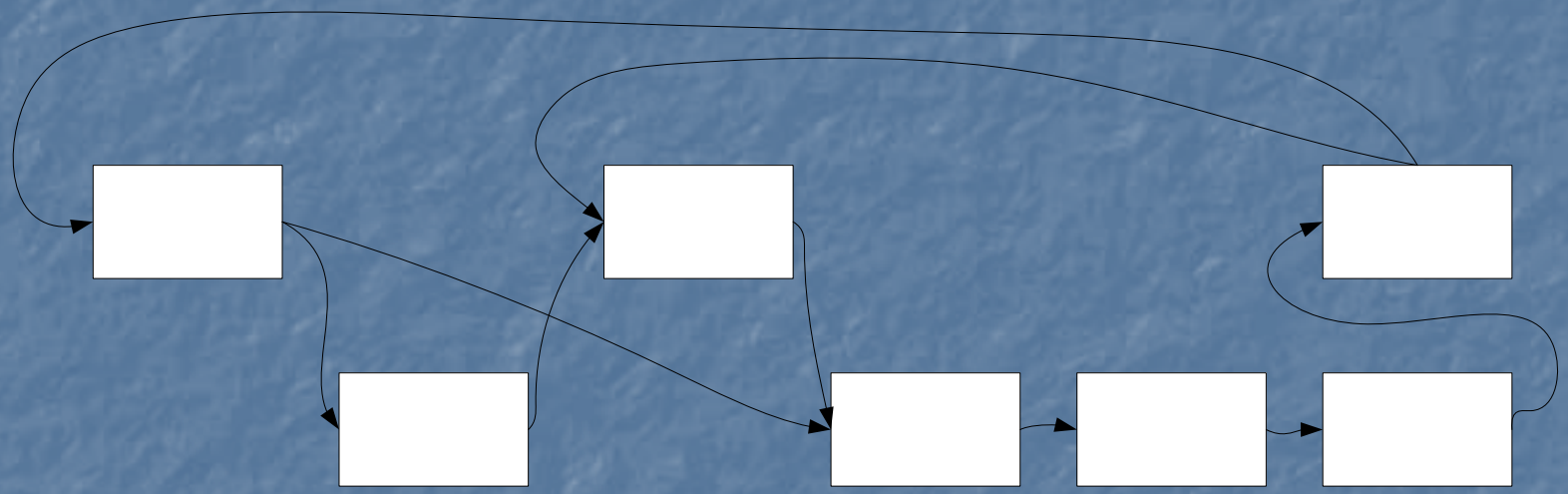
RTRT支持的主机系统

- Solaris 2.5, 2.6, and 2.7
- Windows 98, ME, NT 4.0, 2000
- HPUX 10.20
- SuSe Linux
- Red Hat Linux

RTRT支持的目标板系统

支持的超过200 种的交叉开发环境包括：ARM，Cosmic，Enea OSE，GreenHills，Hitachi，Keil，LynuxWorks，Montavista，Mentor Graphics，NEC，Nohau，Sun Microsystems，Tasking，Texas Instruments，WindRiver，Lauterbach Trace32等。

RTRT单元测试基本过程(一)



RTRT单元测试基本过程(二)

- 编码：开发人员在Test RealTime提供的C/C++语言编辑器中进行代码编写。
- 测试脚本模版自动生成：在被测源代码编译通过后，Test RealTime将通过对源代码进行分析，形成测试脚本模板。
- 增强测试脚本：开发人员根据设计的测试用例，在测试脚本模板的基础上增加和修改测试用例。
- 生成测试程序：Test RealTime将根据测试脚本生成C语言测试程序。
- 执行测试：Test Realtime编译测试程序、被测程序、连接并执行可执行程序。

RTRT单元测试基本过程(三)

- 生成测试报告：Test RealTime将根据测试执行产生的日志文件生成测试报告。
- 测试结果分析：开发人员根据测试报告判断被测程序质量或测试完备性。
- 解决错误：如果发现测试用例未通过，来定位错误位置，并修改错误。Test RealTime可以和开发环境的调试器（如Visual C 6.0的msdev.exe）集成，提高错误定位速度。
- 增强测试用例：增强测试用例来覆盖前次测试执行没覆盖的代码分支。

专题提纲

- 单元测试介绍
- RTRT工具介绍
- 用RTRT搭建单元测试环境的基本过程
- 简单的演示一个单元测试例子

RTRT搭建单元测试环境的基本过程

- 建立目录结构
- 安装配置VC和RTRT
- 创建工程并配置

建立目录结构

可考虑建立以下目录：

- 源文件目录：用于存放被测的.c文件以及.h文件，如src目录。
- 工程目录：用于存放RTRT的工程以及临时编译文件，如test目录。
- 脚本目录：用于存放增强的测试脚本，如scripts目录。
- 报告目录：用于存放生成的测试报告，如reports目录。

目的：方便测试环境的维护和共享。

安装配置VC和RTRT

- 安装VC
- 安装RTRT

说明：最好先安装VC，然后再安装RTRT，因为涉及到一些环境变量的设置，后安装VC将需要自己手动设置这些环境变量，如后安装则不需要手动设置。

创建工程并配置

说明：为了保证整个测试工程拷贝到其它机器能继续使用，建议工程配置中所有与目录或者文件路径相关的项都填成相对路径。

专题提纲

- 单元测试介绍
- RTRT工具介绍
- 用RTRT搭建单元测试环境的基本过程
- 简单的演示一个单元测试例子

简单的演示一个单元测试例子

- 被测代码介绍
- 实际演示测试环境的搭建
- 单元测试演示（所有测试用例测试通过）
- 根据代码覆盖率增加测试用例
- 单元测试演示（添加的测试用例测试失败）
- 调试定位问题
- 单元测试演示（添加的测试用例测试成功）

被测代码介绍

对函数code_int(int x, char* buffer)进行单元测试

- 功能：对整数x进行倒序编码，如输入x=2，buffer=“”，输出buffer=“I12”；如输入x=34，buffer=“”，输出buffer=“I243”；如输入x=56，buffer=“I243”，输出buffer=“I243I265”。
- 数据流：先把指针移动到输入的buffer末尾，写入字符I，然后留出一个字符来记录x的长度，再倒序写入x，最后将长度填入。