VcTester 插装原理与各种覆盖率配置

2007-3-28

VcTester 与常见 C 语言覆盖测试工具一样，提供多种覆盖率统计，已涵盖语句覆盖、分支覆盖、分支条件这 3 种最常见的覆盖统计标准。本文讲解 VcTester 的插装实现原理、描述该工具的覆盖率特点，以及如何定制代码覆盖统计标准，包括：LICC、LDCC、调用覆盖、分支覆盖、条件覆盖、分支条件覆盖。

一、VcTester 插装实现原理

VcTester 是基于函数调用进行覆盖统计的，比如下面被测语句：

```
int i = printf(”I am here”);
void *ptr = &printf;
```

源码经过插装改造，变为：

```
int i = (z000002(),printf)(”I am here”);
void *ptr = &(z000002(),printf);
```

VcTester 借助 C 语言的逗号表达式实现插装，跟据 ANSI C 规范，逗号表达式的结果值是最后一个表达式的计算值，比如“expr1,exprt2, expr3”，这 3 个表达式依次运算，最后返回值是 expr3 的计算值。所以，不难看出，运行上述插装代码等效于插装前代码，只有一点差别，每次 printf 函数调用前插装打点函数 z000002 都被调用一次。

借助 C 语言的宏替换功能可以让插装过程变得很简单，比如：

```
#define printf (z000002(),printf)

int i = printf(”I am here”);
void *ptr = &printf;
```

同理，VcTester 将 if、while、for 等控制语句也按如下方式改造：

```
#define if(expr)   if( (expr)? if_1():if_0() )
```

改造后，插装进去的打点函数（如 if_1、if_0）在相应条件成立时就被执行。如此，分支覆盖统计就支持了，对于条件覆盖统计，VcTester 改造“&&”与“||”连接的条件子句，比如下面代码：

```
if ( i > 0 && i < 100 ) printf(”OK”);
```

改造成：

```
#define COND(expr)  ((expr)?cond_1():cond_0())

if ( COND(i > 0) && COND(i < 100) ) printf(”OK”);
```

二、VcTester 覆盖率统计特点

上面我们讲解了 VcTester 的覆盖插装是基于函数调用实现的，该工具对外提供 2 种规范的覆盖率统计形式：LICC 与 LDCC，这两种覆盖率都是基于函数相对调用实现统计的，由 LICC 与 LDCC 扩展开来，可实现调用覆盖、分支覆盖、条件覆盖、分支条件覆盖。

下面我们先看一下 LICC 与 LDCC 如何定义的。前者是位置无关调用覆盖率（Location-independent call coverage，LICC），后者是位置相关调用覆盖率（Location-dependent call coverage，LDCC）。定义如下：

LICC = (已覆盖的不重复的函数调用个数 / 全部不重复的函数调用个数) * 100%
LDCC = (已覆盖的函数调用个数 / 全部函数调用个数) * 100%

比如某函数中调用了 3 个子函数，其中第 1 个子函数调用在函数定义的两个地方出现，其余 2 个子函数都只在一处调用（即，只使用了一次）。如果这个 3 个子函数都被调用过，而且第 1 个子函数只一个位置调用了，另一个位置尚未覆盖到。这时，我们计算 LICC 是“3/3 = 100%”，而 LDCC 是“3/4 = 75%”。

LICC 指标主要用于粗测，确保某模块具备一定的初始稳定度，适合于与其它模块集成，LICC 指标尝试说明新定义的函数是否跑到过，而不关心每个地方调用它都跑到。LDCC 则在正式测试中使用，是比较完整的评估方式。

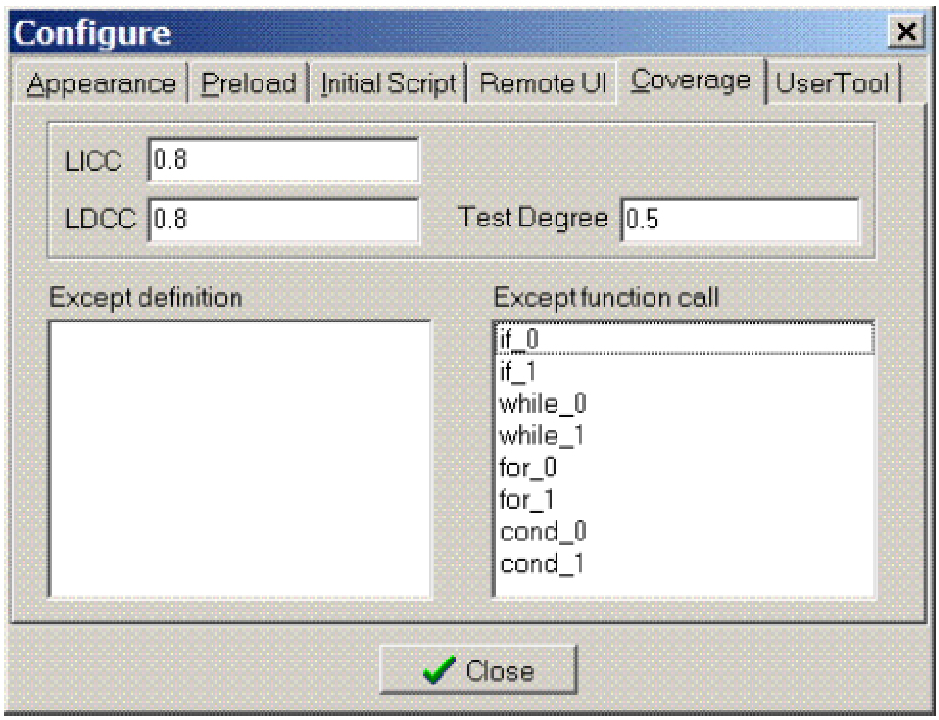
如何从 LICC、LDCC 扩展出调用覆盖、分支覆盖、条件覆盖与分支条件覆盖？VcTester 是通过两种配置实现这一点的，一是插装配置，二是覆盖率统计配置。插装配置是决定是否对源码进行插装，如果插装，还进一步配置是否对 if 语句、while 语句、for 语句，以及条件子句进行插装，这 4 项分别对应 vtPreprocess.exe 预处理程序的 4 个插装选项：“/if”、“/while”、“/for”、“/cond”。

覆盖率统计配置包括“例外被测函数列表”与“例外函数调用列表”，前者决定特定被测函数不纳入测试统计，比如某些特定的异常处理函数，因为实现逻辑简单，但模拟出异常实现覆盖测试的代价很高，就可以把这些异常处理函数定义到“例外被测函数列表”。从 LICC、LDCC 扩展出各种覆盖率主要通过配置“例外函数调用列表”来实现，比如，我们想获得纯正的函数调用覆盖率，即：去掉对 if、while、for 的分支统计，以及不对条件子句进行插装，如此获得的 LDCC 覆盖统计就是纯正的调用覆盖率。值得一提的事，函数调用覆盖的作用近似于语句覆盖，两者主要不同是：赋值语句未纳入调用覆盖，因为赋值语句总是可靠的，它有无跑到对结果统计影响甚小。

不难看出，VcTester 的覆盖率统计比较灵活、适应性强，尤其针对同一产品的不同模块，因代码复杂度不同需按不同覆盖标准，或者异常处理的风格不同需定制覆盖统计方式，或在不同阶段（如原型阶段、联调阶段、测试阶段）分别按不同质量标准进行控制。实际操作中，覆盖统计能定制，可保证持续集成得以顺利实施，如果缺少这种定制能力，产品开发全过程就只能维持一种质量标准，测试情况随代码特性波动而波动，最终使持续集成难以平稳的向前推进。

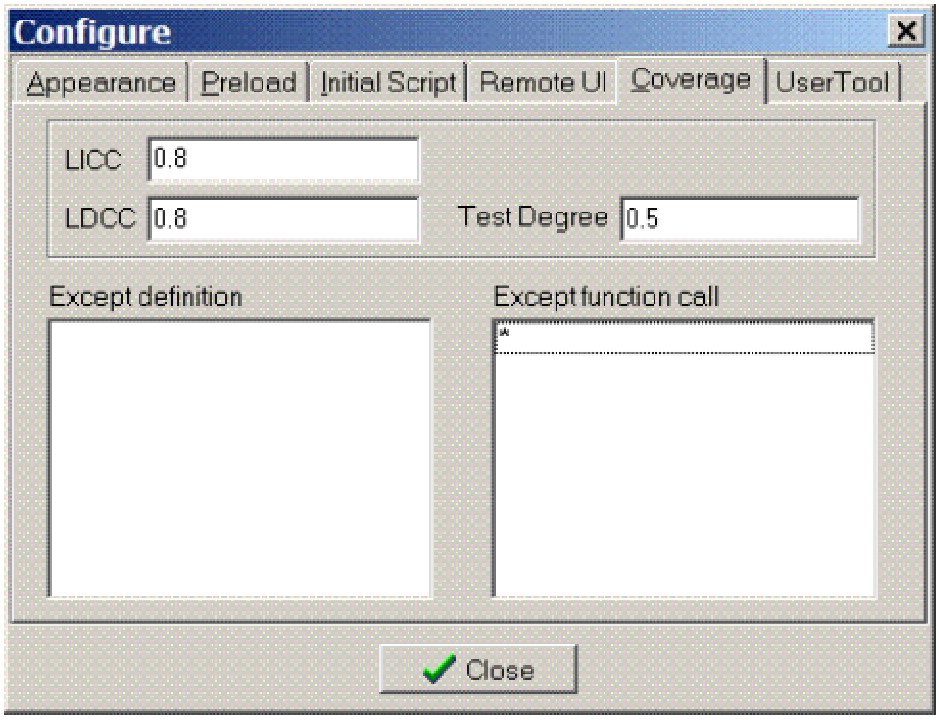
三、定制覆盖率统计

a) 调用覆盖（Call Coverage），将 4 个插装选项都置上（/if、/while、/for、/cond），然后配置如下图配置例外函数调用列表：



当然，如果插装配置去掉“/cond”，上图例外调用列表可以将 cond_0 与 cond_1 去掉，结果是一样的，都是纯正的调用覆盖。

b) 分支覆盖（Decision Coverage，也称判定覆盖），设置/if、/while、/for 插装选项，然后把“*”添加为例外函数，如下图：



“*”表示系统内部函数之外的所有函数，VcTester 缺省将 if_0、if_1、while_0、while_1、for_0、for_1、cond_0、cond_1 这几个函数看当系统内部函数。

c) 条件覆盖（Condiction Coverage），设置/cond 插装选项，然后把“*”添加为例外函数即可。

d) 分支条件组合覆盖（Condition/Decision Coverage，或称判定条件组合覆盖），设置/if、/while、/for、/cond 插装选项，然后把“*”添加为例外函数。

四、VcTester 覆盖率现实应用

由于调用覆盖、分支覆盖、条件覆盖、分支条件覆盖都从 LICC 与 LDD 定制而来，现实应用中，我们通常只使用 LICC 与 LDCC 两种覆盖标准，通常将/if、/while、/for、/cond 四个插装选项都置上，然后定义数个“例外被测函数”与“例外函数调用”，再根据不同代码的质量要求，给定 LICC 与 LDCC 测试通过的阈值就可以了。

只有特定情况下，比如 QA 为了给不同测试工具的实施效果做横向比较，可以配置特定的覆盖统计形式，得到标准的调用覆盖、分支覆盖、条件覆盖等覆盖率指标。

参考文献：

1. ezTester，《VcSmith 使用手册》
2. ezTester，《VcTester 使用手册》