软件质量保证与测试

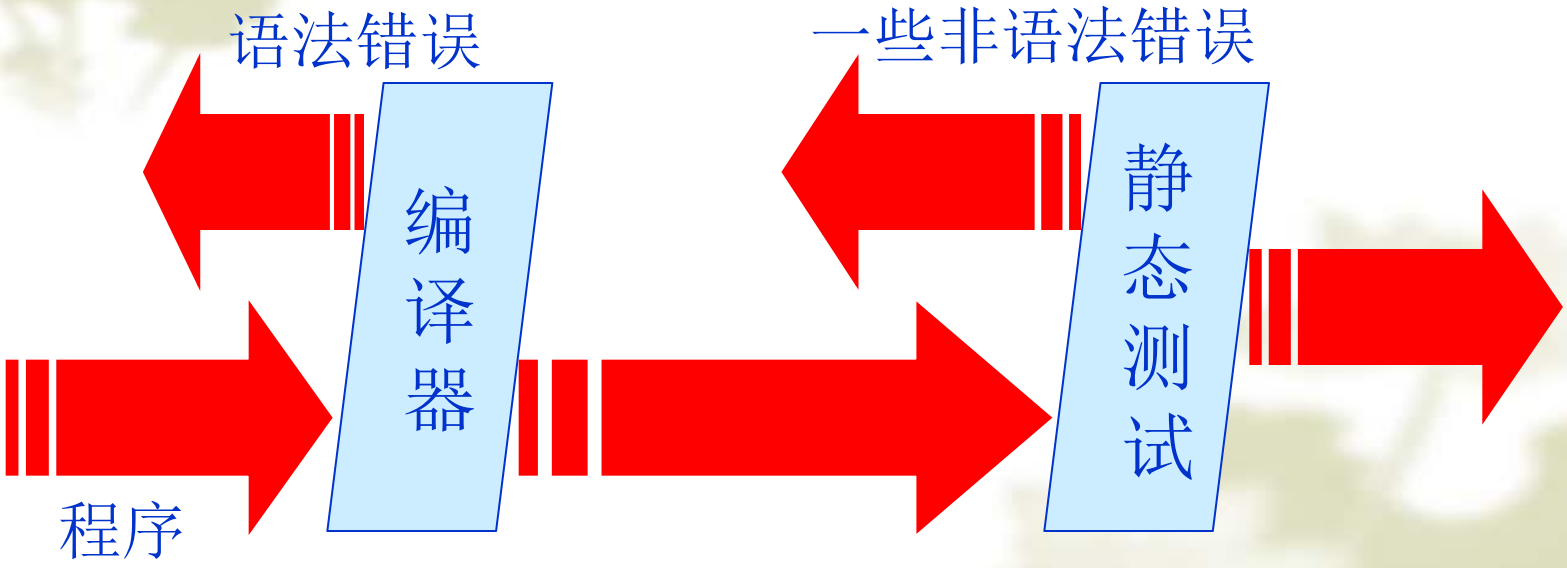
Software Quality Assurance and Testing

第 4 章 白盒测试

4.2 程序的静态测试

金陵科技學院

静态测试



静态测试

程序静态测试通过分析或检查源程序的结构、过程、接口等来检查程序的正确性，找出不妥和可疑之处，例如不匹配的参数、不适当的循环嵌套和分支嵌套、不允许的递归、未使用过的变量、空指针的引用和可疑的计算等。静态测试结果可用于进一步的查错，并为测试用例选取提供指导。

最常见的静态测试包括代码检查、静态结构分析、静态质量度量等。

静态测试

代码检查法主要是通过桌面检查，代码审查和走查方式，对以下内容进行检查：

1. 检查代码和设计的一致性；
2. 代码的可读性以及对软件设计标准的遵循情况；
3. 代码逻辑表达的正确性；
4. 代码结构的合理性；
5. 程序中不安全、不明确和模糊的部分；
6. 编程风格方面的问题等。

常见的代码检查项目

代码检查项目	检查结果
所有设计要求是否都实现？	
代码编制是否遵照编码规范？	
所有的代码是否风格保持一致？	
所有的注释是清楚和正确？	
所有代码异常处理是否都有注释？	
每一功能目的是否都有注释？	
是否按注释类型格式编写注释？	
代码注释量是否达到了规定值？	
所有变量的命名是否依照规则？	
循环嵌套是否优化到最少？	
……	

代码检查

代码检查一旦发现错误，通常能在代码中对其进行精确定位，这与动态测试只能发现错误的外部征兆不同，因而可以降低修正错误的成本。

另外，在代码检查过程中，有时可以发现成批的错误，典型的如分散在多处的同一类错误，而动态测试通常只能一个一个的测试和报错。

代码检查
- 桌面检查
- 代码码审
- 代码码走

桌面检查

桌面检查可以说是最早的一种代码检查方法，一般是程序员对自己的代码进行一次自我检查，阅读程序，对源程序代码进行分析，对照错误列表检查程序，对程序推演测试数据，检验程序中是否有错误等。

总体而言，桌面检查的效率是相当低的：

1、桌面检查随意性较大，除非有严格的管理和技术规范来约束，否则检查哪些内容，如何检查，检查到哪种程度，基本上取决于程序员个人。

2、自己一般不太容易发现自己程序中的问题。

桌面检查

在实践中，可以采用交叉桌面检查的方法，两个程序员可以相互交换各自的程序来做检查，而不是自己检查自己的程序。

但是即使这样，其效果仍然逊色于代码审查和走查。因为代码检查和代码走查以小组的形式进行，小组成员之间存在着互相监督和促进的效应。

代码审查

代码审查是由若干程序员和测试员组成一个审查小组，通过阅读、讨论、评价和审议，对程序进行静态分析的过程。

代码审查分两步。第一步，小组负责人提前把设计规格说明书、控制流程图、程序文本及有关要求、规范等分发给小组成员，作为审查的依据。小组成员在充分阅读这些材料后，进入审查的第二步，召开程序审查会。通过会议和集体讨论、评价和审议，以集体的智慧和不同的角度，找出程序中的问题，提出修改意见和建议。

代码审查

代码审查是软件开发中常用的手段，和其它测试手段相比，它更容易发现和架构以及时序相关等较难发现的问题，还可以帮助团队成员提高编程技能，统一编程风格等。

代码审查有相关辅助的辅助工具可以选择使用。

代码走查

走查是让人充当计算机，把数据代入程序，模拟代码的执行，看程序能不能正常执行下去，证件过程和状态是否正确，并最终得出符合预期的结果。

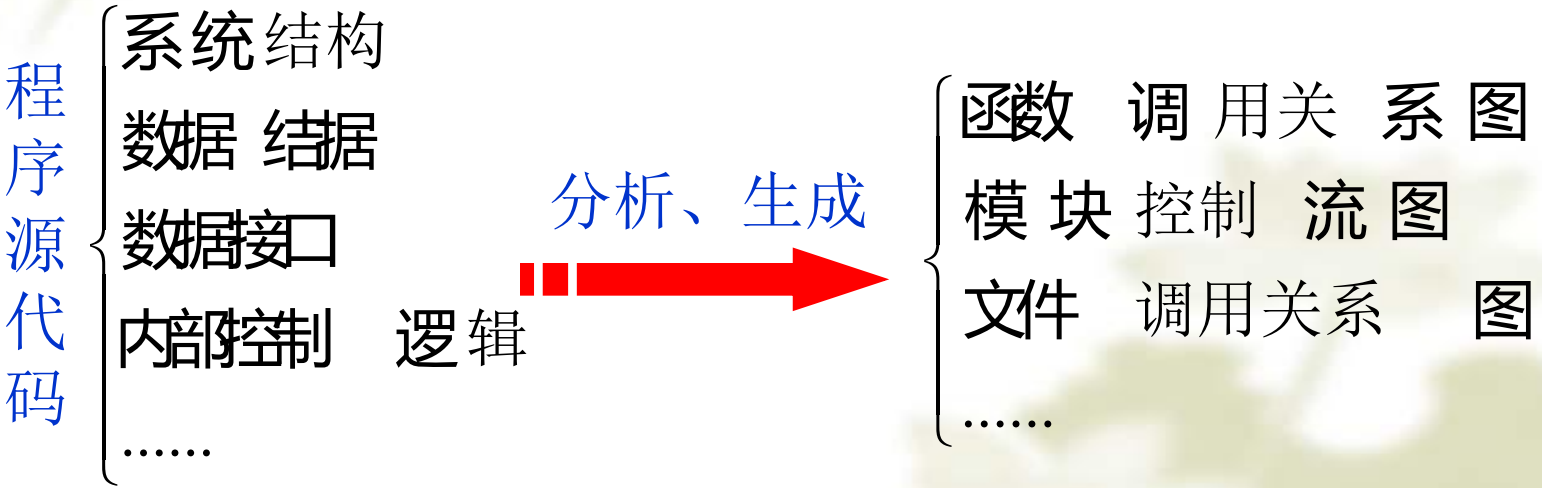
代码走查与代码审查有类似的地方，都是以小组为单位进行，但代码走查重在模拟程序的执行，它需要推演每个测试用例的执行过程和结果，把测试数据沿程序的运行逻辑走一遍，过程和中间状态记录在纸张或白板上以供监视检查。

代码走查

与代码审查基本相同，代码走查过程也分为两步。第一步把材料先发给走查小组每个成员，让他们认真研究程序，然后组织代码走查会议。但开会的过程与代码审查不同，不是简单地读程序和对照错误检查表进行检查，而是让与会者“充当计算机”，由测试人员为被测程序准备一批有代表性的测试用例，提交给走查小组，走查小组开会，一起把测试用例代入程序，模拟代码的执行，分析检查程序的执行过程和结果。

静态结构分析

一个软件通常由多个部分组成，总是存在着一定的组织结构。



清晰地呈现整个软件的组成结构，使程序便于被宏观把握，和微观分析。

静态结构分析

- 控制流分析
- 数据流分析
- 接口分析
- 表达式分析
- ……

发现程序当中的问题或者不合理的地方，然后通过进一步检查，就可以确认软件中是不是存在问题、缺陷或错误

静态结构分析

静态结构分析法通常采用以下一些方法：

1. 通过生成各种图表，来帮助对源程序的静态分析，常用的的各种引用表主要有：

- 标号交叉引用表
- 变量交叉引用表
- 子程序（宏、函数）引用表
- 等价表
- 常数表

静态结构分析

2. 通过分析各种关系图、控制流图来检查程序是否有问题。主要有：

- 函数调用关系图：列出所有函数，用连线表示调用关系，通过应用程序各函数之间的调用关系展示了系统的结构。
- 模块控制流图：由许多结点和连接结点的边组成的图形，其中每个结点代表一条或多条语句，边表示控制流向，可以直观地反映出一个函数的内部结构。

静态结构分析

3. 其它常见错误分析：分析程序中是否有某类问题、错误或“危险”结构。如：

- 类型和单位分析
- 引用分析
- 表达式分析
- 接口分析

程序流程分析

一个程序要能够正常执行，不留下问题隐患，在流程上会有一些基本要求。

可以分别从控制流和数据流的角度来对程序做流程上的分析。

控制流分析

从控制流的角度来说，程序不应存在以下问题：

1. 转向并不存在的标号

   如果这样的话，程序执行就会意外中止。

2. 有没有用的语句标号

   没有用的语句标号类似于有定义而未使用的变量，没有任何作用，却需要对其进行管理并占用资源。

3. 有从程序入口无法到达的语句

   从程序入口无法到达的语句，就意味着这些语句根本就不会被执行到，相应的功能也无法被调用。

4. 不能到达停机语句的语句

   不能到达停机语句的语句如死循环。

数据流分析

早期的数据流分析主要集中于变量定义/引用错误或异常，包括如下三方面：

1. 变量被定义，但是从来没有使用(引用)
2. 所使用的变量没有被定义
3. 变量在使用之前被定义多次

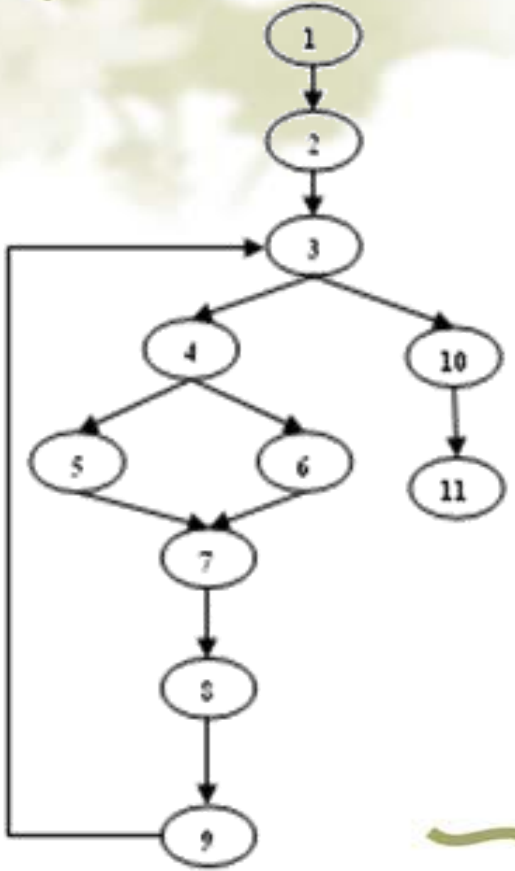
这些问题光靠简单的语法分析器或者是语义分析器是无法检测出来的。而软件测试工具检测到上述问题后，就可以给出一些警告信息，如“所定义的变量未被使用”等。

数据流分析

变量被语句定义是指：某一语句执行时能改变变量V的值，则称V是被该语句定义的。变量被语句引用是指：某一语句的执行应用了内存变量V的值，则称变量被语句引用。

语句：X:=Y+Z		定义了变量 X，引用了Y，Z
语句：if Y>Z	then …	引用了变量 Y 和 Z
语句： READ	X	引用了变量 X
WRITE	X	定义了变量 X

数据流分析



节点	被定义变量	被引用变量
1	X，Y，Z	
2	X	W，X
3		X，Y
4		Y，Z
5	Y	V，Y
6	Z	V，Z
7	V	X
8	W	Y
9	Z	V
10	Z	Z
11		Z

没有定义过

第一次执行循环时没有定义过

没有使用过

没有使用过

数据流分析

左图为一个程序的控制流图，右表为控制流图各节点的数据操作，分为变量定义和变量引用。通过数据流分析，我们可以发现一下一些错误或者异常：

1、2号节点引用了变量W，但在此之前W没有定义。
2、5号、6号节点引用了变量V，但第一次循环执行到5号、6号节点时，变量V尚未定义
3、8号节点定义了变量W，但之后程序没有引用变量W。
4、6号节点、9号节点都定义了变量Z，但两次定义之间并没有引用变量Z。

数据流分析

上页左图为一个程序的控制流图，右表为控制流图各节点的数据操作，分为变量定义和变量引用。通过数据流分析，我们可以发现一下一些错误或者异常：

1、2号节点引用了变量W，但在此之前W没有定义。
2、5号、6号节点引用了变量V，但第一次循环执行到5号、6号节点时，变量V尚未定义
3、8号节点定义了变量W，但之后程序没有引用变量W。
4、6号节点、9号节点都定义了变量Z，但两次定义之间并没有引用变量Z。

数据流分析

一般来说，情况1和2属于错误，引用未经定义的变量可能会让程序执行出错；情况3是一种疏漏，定义了变量但并没有引用，那么这种定义操作就没有实际意义；情况4是一种异常，这种异常有时可能是由于程序员疏漏所致，需要予以修正，但有时也可能是程序实际的需要。

分析工具

对程序的控制流分析和数据流分析，除了有的编译器就带有相关分析功能之外，很多都可以采用辅助工具来完成。

本节内容就讲到这里，谢谢，再见！

金陵科技學院