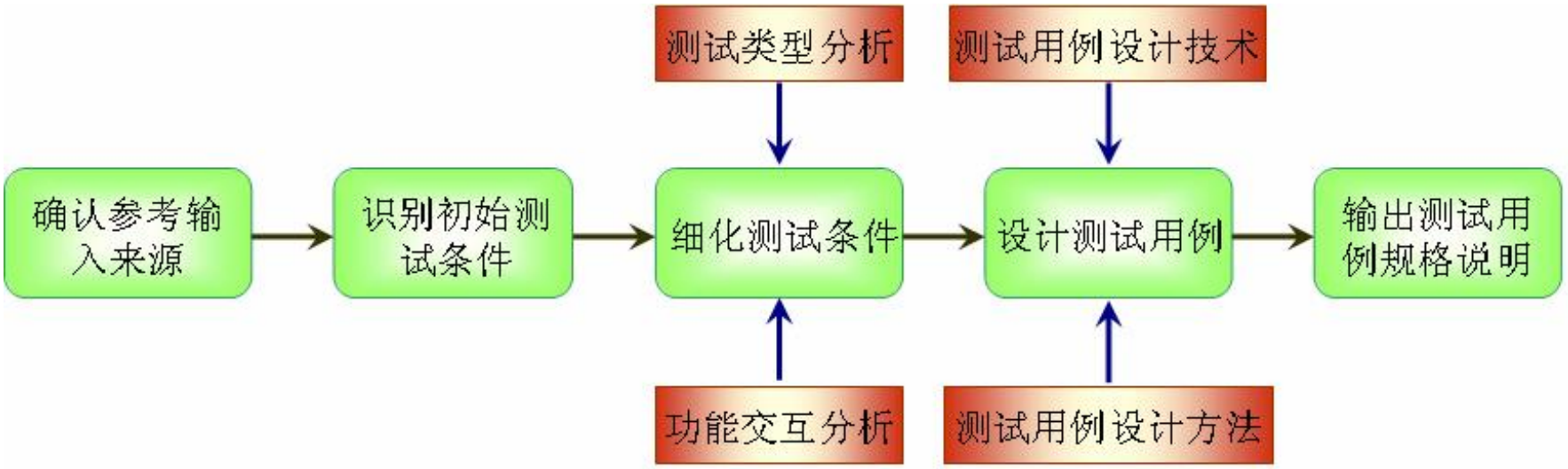
火龙果·整理
uml.org.cn

基于规格说明的测试设计

课程目录

1. 测试设计生命周期
2. 等价类划分与边界值分析
3. 状态转换测试
4. 决策表测试
5. 结对测试
6. 场景与用例测试
7. 需要评审吗

测试过程中的测试设计


测试类型分析
测试用例设计技术
确认参考输入来源
识别初始测试条件
细化测试条件
设计测试用例
输出测试用例规格说明
功能交互分析
测试用例设计方法

测试用例质量评估

课程目录

1. 测试设计生命周期
2. 等价类划分与边界值分析
3. 状态转换测试
4. 决策表测试
5. 结对测试
6. 场景与用例测试
7. 需要评审吗

等价类划分

等价类划分技术把所有可能的输入/输出数据，即软件或者系统的输入/输出域划分成若干部分，然后从每一部分中选取少数有代表性的数据做为测试用例的输入数据！

等价类划分技术可以应用在所有的测试级别中！

什么是等价类

等价类是指某个输入/输出域的子集合。在该子集合中，各个输入/输出数据对于揭露程序中的错误都是等效的。测试某等价类的代表值就等价于对这一类其它值的测试！

从测试人员的角度，对揭露软件或者系统中的缺陷来说，等价类中的每个元素是等效的！

等价类类型

有效等价类

是指对于软件或者系统的规格说明来说，是合理的，有意义的输入数据构成的集合！

无效等价类

是指对于软件或者系统的规格说明来说，是不合理的，无意义的输入数据构成的集合！

在设计测试用例时，要同时考虑有效等价类和无效等价类的设计！

等价类划分

参数x的取值范围是1000到20000；

有效等价类：1000 ≤ x ≤ 20000；

无效等价类：x < 1000或者x > 20000；

等价类划分的原则1

如果输入条件规定了连续的取值范围，则可以创建一个有效等价类和两个无效等价类；

等价类划分

注册的用户名长度应该是6至12位：

有效等价类： 6 ≤ 用户名长度≤ 12；

无效等价类：用户名长度大于12或者小于6；

等价类划分的原则2

假如描述的是允许数值的范围（离散的数字），则可以创建一个有效等价类（所有可能的有效值），和两个无效等价类（少于和多于有效的个数）；

等价类划分

根据学历来定岗位工资，学历可为:专科、本科、硕士、博士四种之一。

有效等价类：专科、本科、硕士、博士四个等价类；

无效等价类：其他的学历；

等价类划分的原则3

假如输入数据规定的是一组值（假定n个），并且软件对每一个输入值分别处理的情况下，可确立n个有效等价类和一个无效等价类；

等价类划分

预订飞机票的人可以是：小孩、青少年、成人、学生、残疾人或者退休人员，软件处理方式一样；

有效等价类：列表中的任何一个；

无效等价类：不包含在列表中的其他类型；

等价类划分的原则4

假如输入数据规定的是一组值（假定n个），并且软件对每个输入值都一样的处理，可确立一个有效等价类和一个无效等价类；

等价类划分

函数变量名必须以字母开头，并且长度为8；

有效等价类：字母开头，长度为8；

无效等价类：不是以字母开头，或长度不是8；

等价类划分的原则5

在规定了输入数据必须遵守的规则的情况下,可确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）；

等价类划分的测试用例

测试对象通常有多个输入参数，如何对这些参数等价类进行组合测试，来保证等价类的覆盖率，是测试用例设计首先需要考虑的问题！

等价类组合的原则

- 所有有效等价类的代表值都集成到测试用例中，即覆盖有效等价类的所有组合。任何一个组合都将设计成一个有效的测试用例，或者叫正面测试用例；

- 无效等价类的代表值只能和其他有效等价类的代表值（随意）进行组合。因此，每个无效等价类将产生一个额外的无效测试用例，或者叫负面测试用例；

火龙果·整理
uml.org.cn

等价类组合的数量

- 正面测试用例：即有效等价类的组合数量，是每个参数等价类数目的乘积；

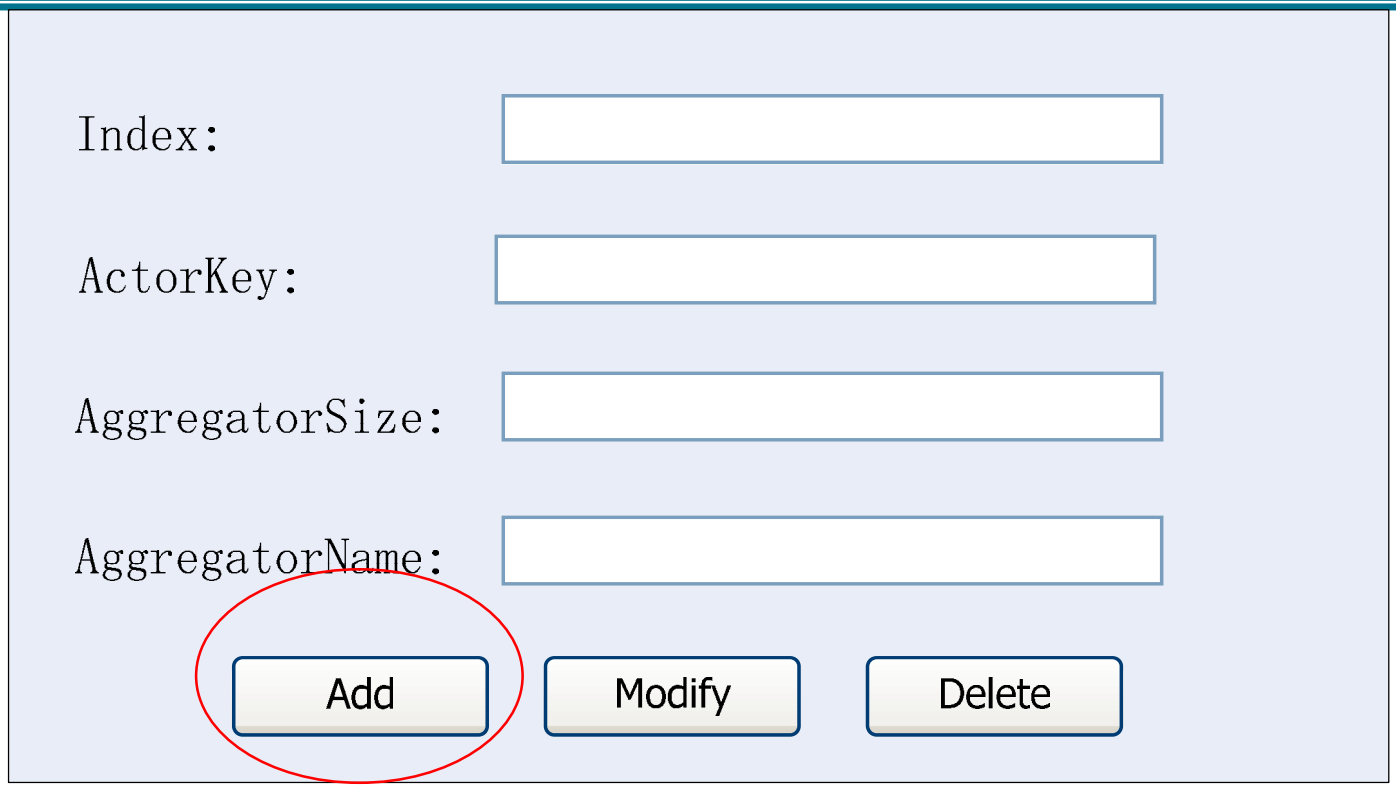
- 负面测试用例：即无效等价类的组合数量，是每个无效等价类数目之和；

等价类的组合，将产生数以百计的测试用例，如何有效的减少测试用例的数目？

选择等价类测试用例的原则

- 由所有代表值组合而成的测试用例按使用频率（典型的使用特征）进行排序，并按照这个序列设置优先级。这样就能仅对相关的测试用例（典型的组合）进行测试；
- 优先考虑包含边界值或者边界值组合的测试用例；
- 将一个等价类的每个代表值和其他等价类的每个代表值进行组合来设计测试用例（即双向组合代替完全组合）；
- 保证满足最小原则：一个等价类的每个代表值至少在一个测试用例中出现；
- 无效等价类的代表值不与其他无效等价类代表值进行组合；

等价类划分案例



- Index取值范围：10001 - 10006，整型；
- ActorKey取值范围：1 - 65535，整型；
- AggregatorSize取值范围：1 - 8，整型；
- AggregatorName：必须由字母组成，并且长度为1 - 8；

边界值分析

边界值分析就是对输入或输出的边界值进行测试的一种黑盒测试方法。通常边界值分析法是作为对等价类划分法的补充，这种情况下，其测试用例来自等价类的边界 ！

边界值分析方法需要测试边界值以及离边界值最近的上下两个值（等价类里面的值和等价类外面的值）。因此，通常情况下，对于边界值，一般可以得到3个测试用例！

为什么边界值分析

- 长期的测试工作经验表明，大量的错误是发生在输入或输出范围的边界值上，而不是发生在输入输出范围的内部。出现这种现象的原因有：
  - 需求文档或者规格说明中经常没有明确的定义边界值；
  - 或者编程人员对边界值产生误解；

常见的边界值类型

- 对16-bit 的整数而言 32767 和 -32768 是边界值；
- 屏幕上光标在最左上、最右下位置；
- 报表的第一行和最后一行；
- 数组元素的第一个和最后一个；

对于列表形式输入，一般情况下没有边界值。比如前面等价类划分中的“旅游者”的例子中，就不存在输入域的边界值，输入数据类型是不连续的，即6个元素组（儿童、青少年、成人学生、残疾人和退休人员）！

边界值分析

边界值的提示1

如果输入条件规定了值的范围，则应取刚达到这个范围的边界的值，以及刚刚超越这个范围边界的值作为测试输入数据；

边界值的例子

如果软件的规格说明中规定：“重量在10公斤至50公斤范围内的邮件，其邮费计算公式为……”。假如设计基于边界值的测试用例，应取10及50，还应取10.01，49.99，9.99及50.01等；

边界值分析

边界值的提示2

如果输入条件规定了值的个数,则用最大个数,最小个数,比最小个数少1，比最大个数多1的数作为测试数据；

边界值的例子

一个输入文件应包括1~255个记录，则测试用例可取1和255，还应取0、2及254、256等；

边界值分析

边界值的提示3

尽量选择非常庞大的数据结构、列表和表格等作为边界值分析的数据，比如：那些能使内存溢出、文件和数据存储到达边界的数据，来检查测试对象在这种极端情形下的行为；

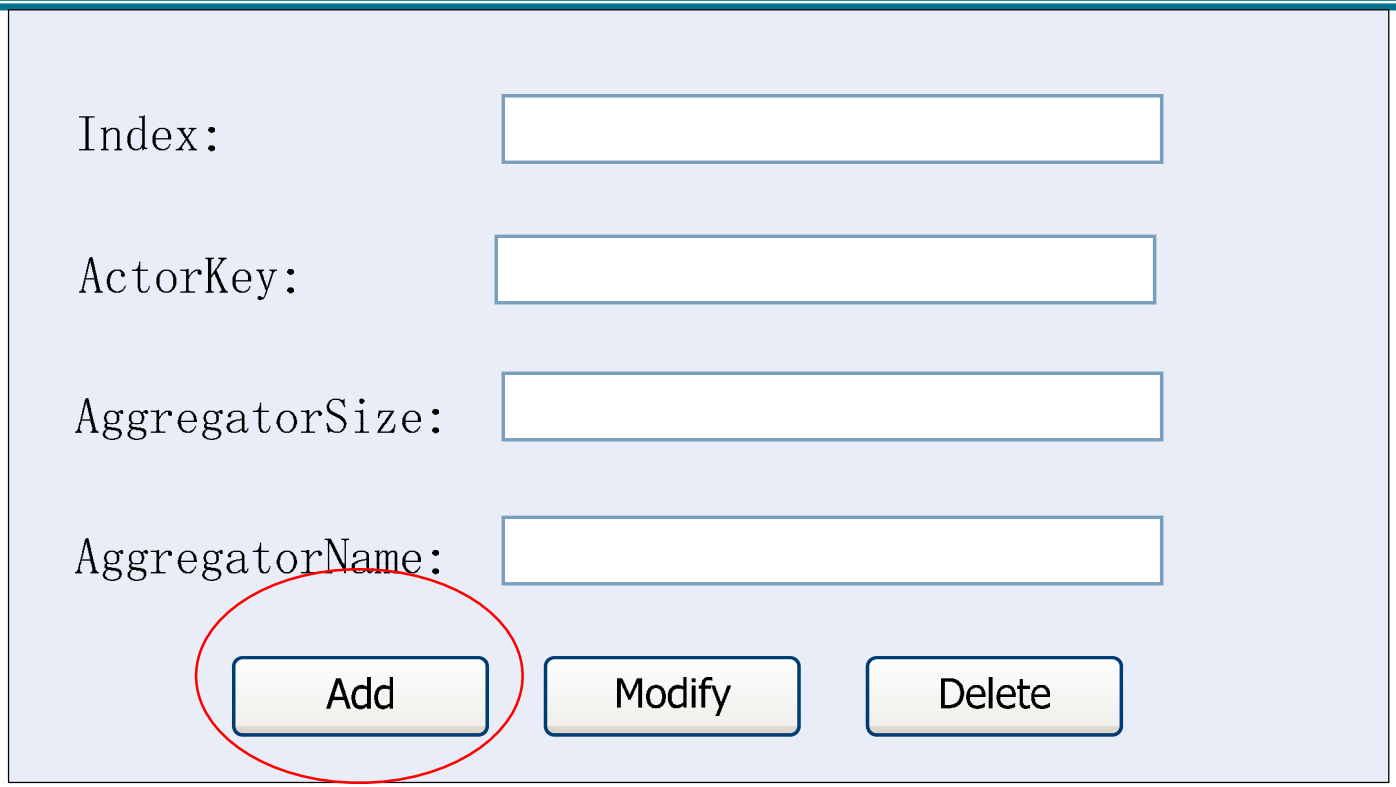
边界值的提示4

对于列表和表格，空列表和满列表以及列表的第一个元素和最后一个元素都是应该作为分析的对象，因为测试它们常常可以发现由于编程错误而导致的失效；

边界值分析步骤

- 边界值分析使用与等价类划分法相同的划分，只是边界值分析假定错误更多地存在于划分的边界上，因此在等价类的边界上以及两侧的情况设计测试用例；
- 将软件的输入或者输出参数进行等价类划分；
- 在等价类的基础之上进行边界值分析。一般情况下，假如边界值已经由等价类划分覆盖，则可以不予考虑；
- 将边界值进行组合，作为测试用例的输入数据；

边界值分析案例



- Index取值范围：10001 - 10006，整型；
- ActorKey取值范围：1 - 65535，整型；
- AggregatorSize取值范围：1 - 8，整型；
- AggregatorName：必须由字母组成，并且长度为1 - 8；

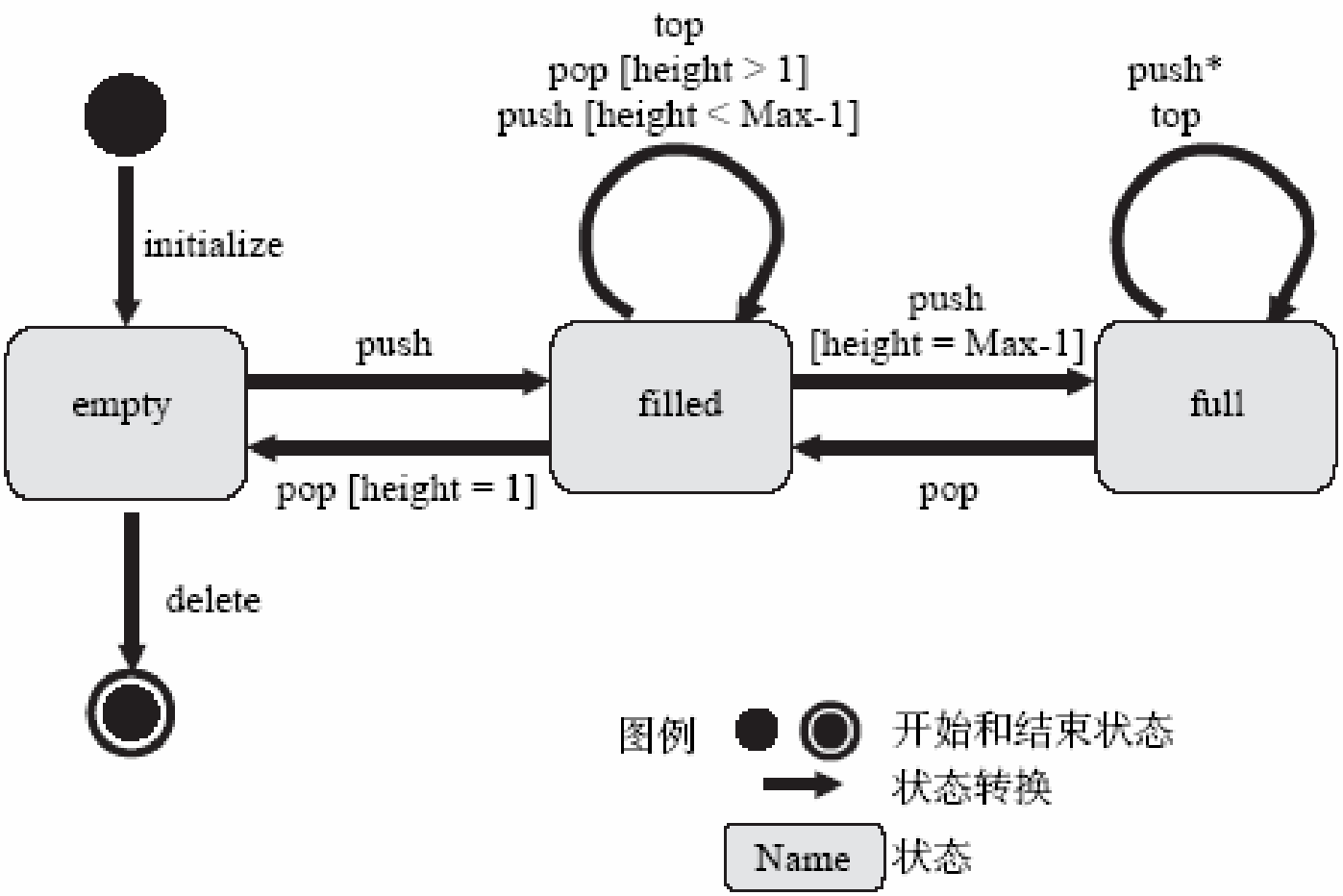
课程目录

1. 测试设计生命周期
2. 等价类划分与边界值分析
3. 状态转换测试
4. 决策表测试
5. 结对测试
6. 场景与用例测试
7. 需要评审吗

状态转换测试

- 测试对象的输出和行为方式不仅受当前输入数据的影响，同时还与测试对象之前的执行情况，或者之前的事件或以前的输入数据等有关；
- 通过引入状态图（state diagram）来描述测试对象和测试数据、对象状态之间的关系；
- 状态图中的各个状态是通过不同的事件驱动的，比如函数的调用；
- 基于状态图开展的测试称之为状态转换测试；

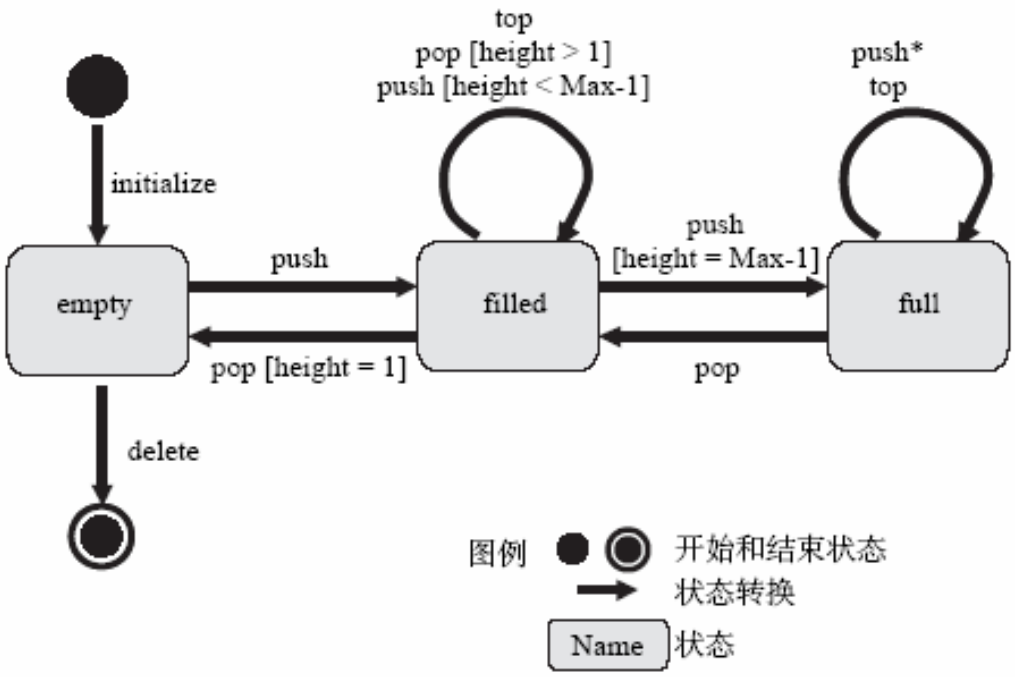
状态图


top
pop [height > 1]
push [height < Max-1]
push*
top
initialize
push
[height = Max-1]
empty
push
filled
full
pop [height = 1]
pop
delete
图例
开始和结束状态
状态转换
Name
状态

状态转换测试

状态转换测试之状态

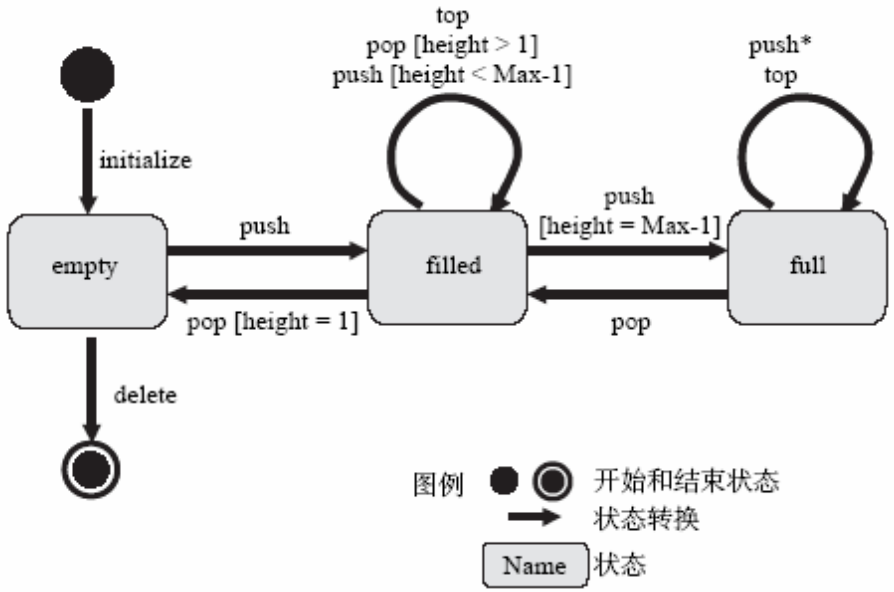
- start
- empty
- filled
- Full
- end



状态转换测试

状态转换测试之事件

- Initialize：初始化
- Push：加入一个元素
- Pop：移除一个元素
- Delete：删除堆栈

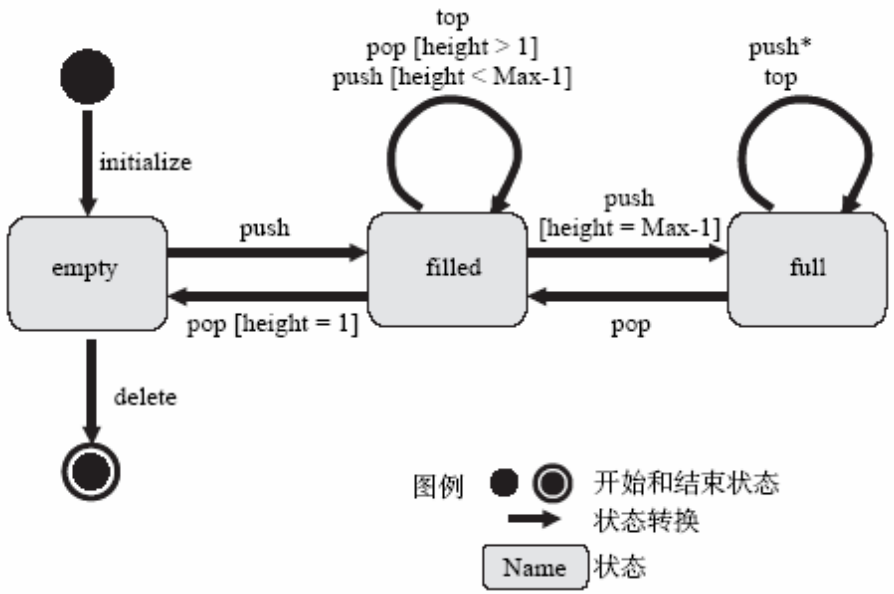


事件驱动着测试对象的状态发生变化，这里指的是调用Initialize、push、pop和delete函数！

状态转换测试

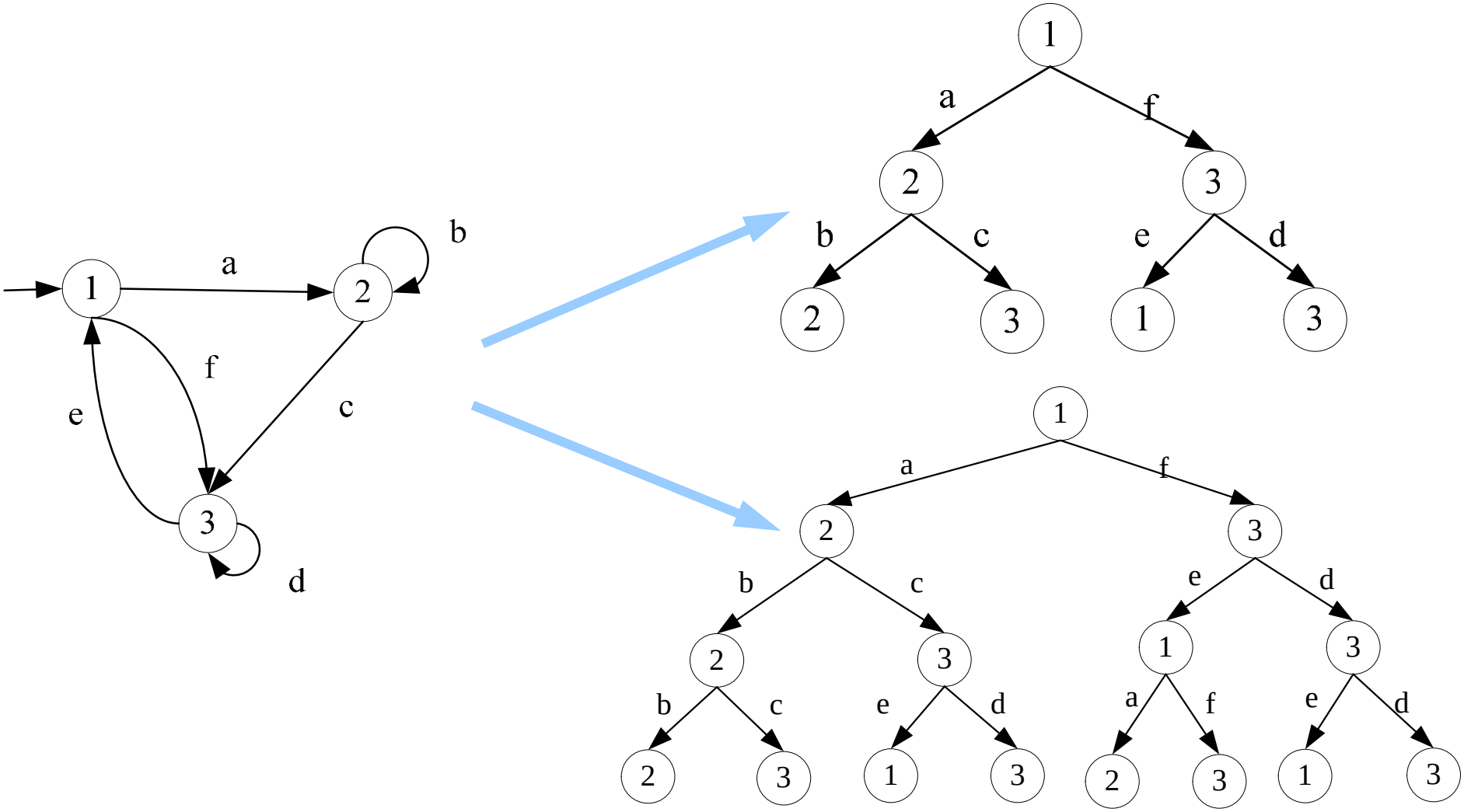
状态转换测试之信息

- Top：显示最上的元素
- Max：显示栈最大高度
- Show：显示栈的状态

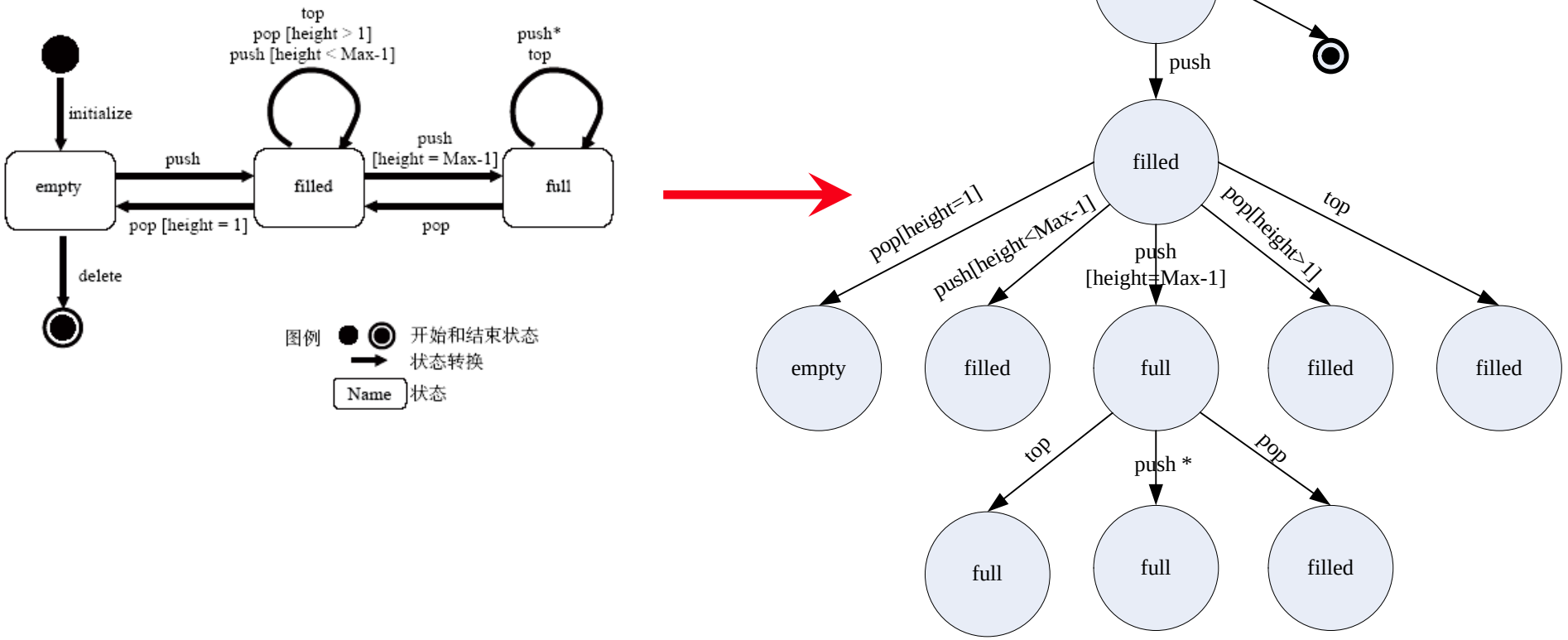


信息中的这些函数调用并不会改变测试对象的状态！

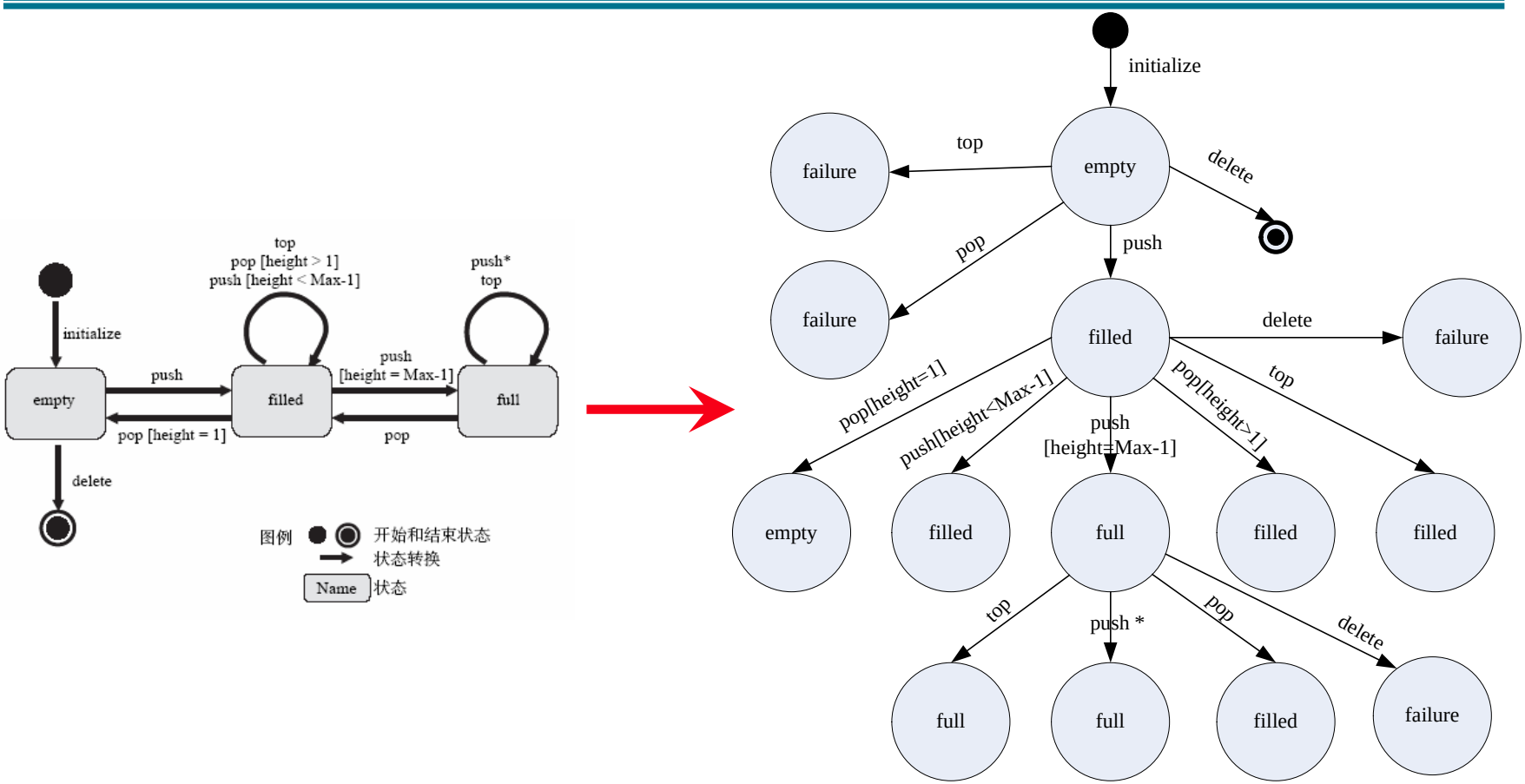
状态图到状态树



状态图到状态树


top
pop [height > 1]
push [height < Max-1]
push*
top
initialize
push
push
[height = Max-1]
empty
filled
full
pop [height = 1]
pop
delete
图例
开始和结束状态
状态转换
Name
状态
initialize
empty
delete
push
filled
pop[height=1]
push[height<Max-1]
push
[height=Max-1]
pop[height>1]
top
empty
filled
full
filled
filled
top
push *
pop
full
full
filled

状态树扩展



函数的错误调用：健壮性测试！

状态转换测试用例

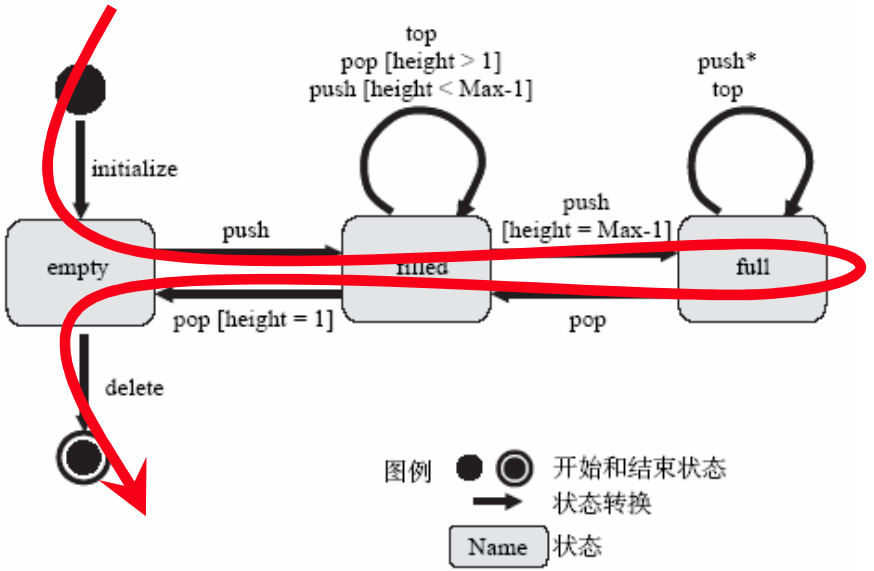
- 测试对象的初始状态（组件或系统）；
- 测试对象的输入；
- 期望输出或者期望行为；
- 期望的结束状态；

状态转换测试需要考虑什么

- 状态转换之前的状态；
- 触发状态转换的触发事件；
- 在状态转换时触发的期望反应；
- 接下来的期望状态；

强度定义1

覆盖测试对象的所有状态；

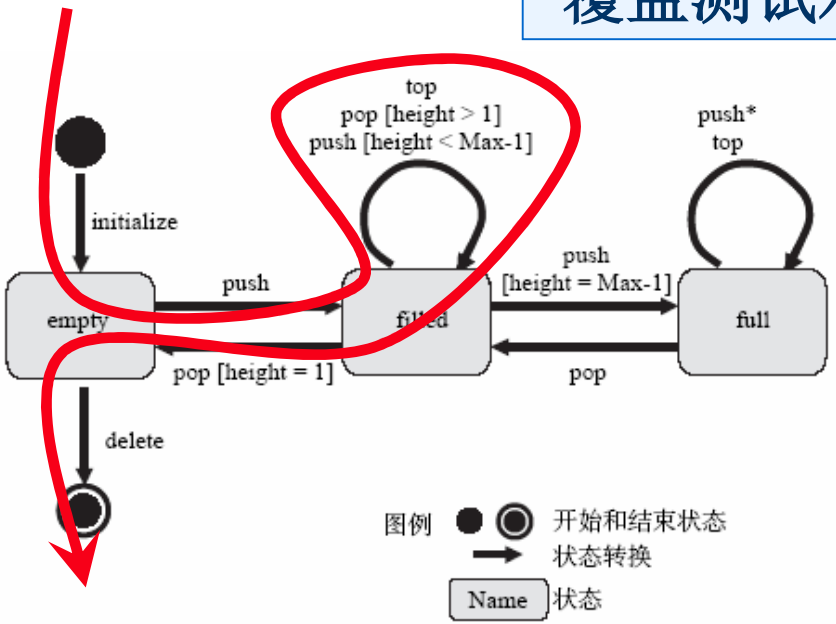


initialize[empty]，push[filled]…，push[full]，
pop[filled]…，pop[empty]，delete；

缺点：没有调用所有的函数；假设：Max ＝ 4

强度定义2

覆盖测试对象的所有函数；



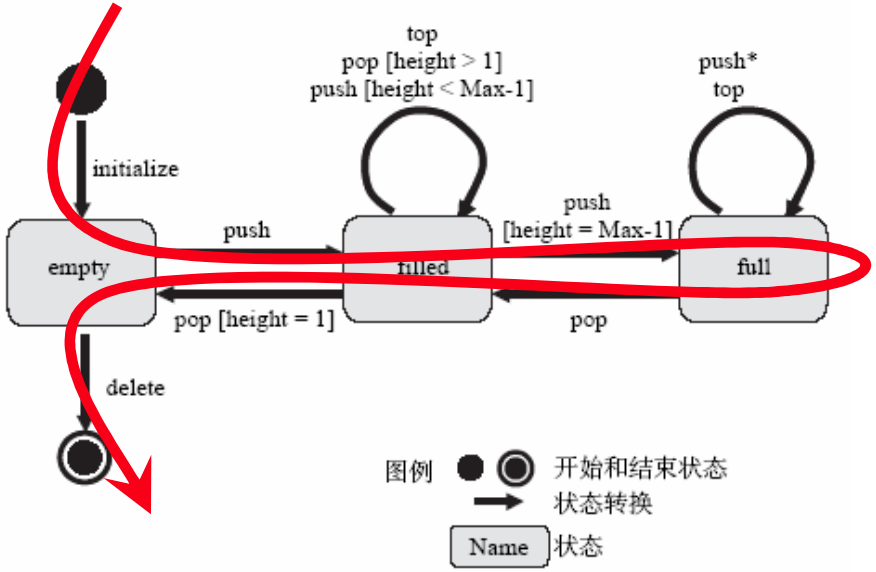
initialize[empty]，push[filled]，top，pop[empty]，delete

缺点：没有覆盖所有的状态；

假设：Max ＝ 4

强度定义3

每个状态至少转换一次；

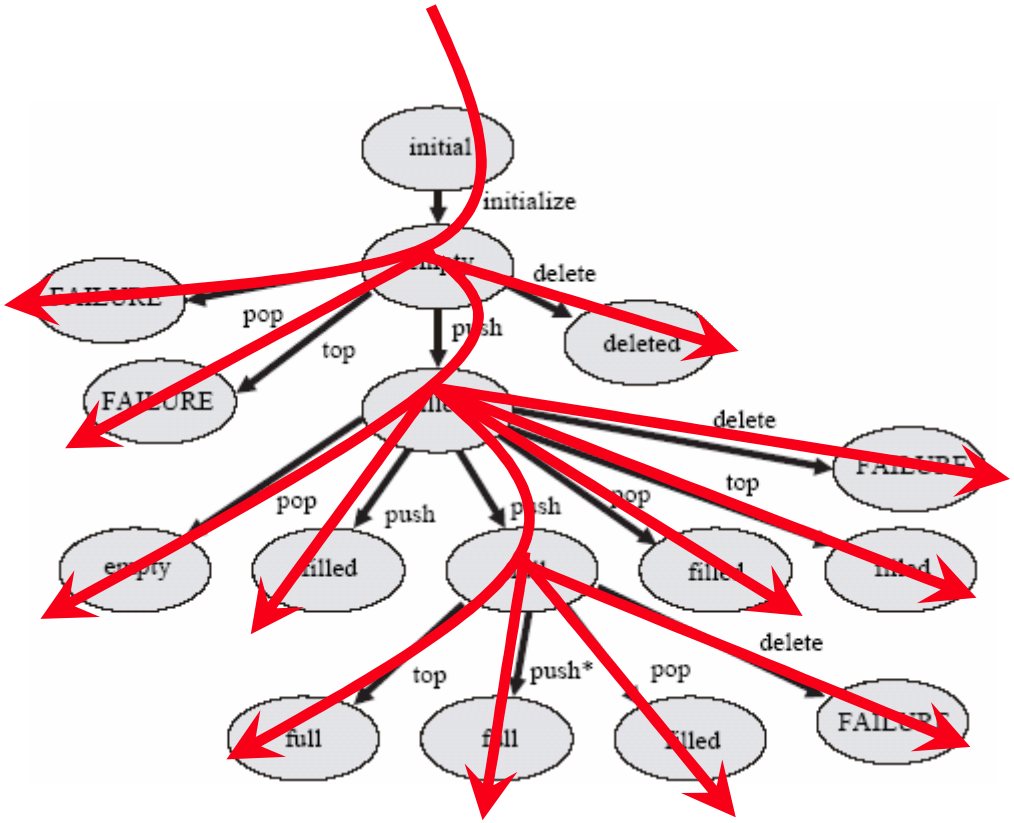


initialize[empty]，push[filled]…，push[full]，
pop[filled]…，pop[empty]，delete；

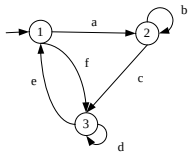
缺点：没有调用所有的函数；假设：Max ＝ 4

强度定义4

状态转换测试对于每个状态都应该将与此状态相关的函数至少执行一遍。通过扩展的状态树来实现；



状态转换树与状态转换表



？

状态转换树与状态转换表

课程目录

1. 测试设计生命周期
2. 等价类划分与边界值分析
3. 状态转换测试
4. 决策表测试
5. 结对测试
6. 场景与用例测试
7. 需要评审吗

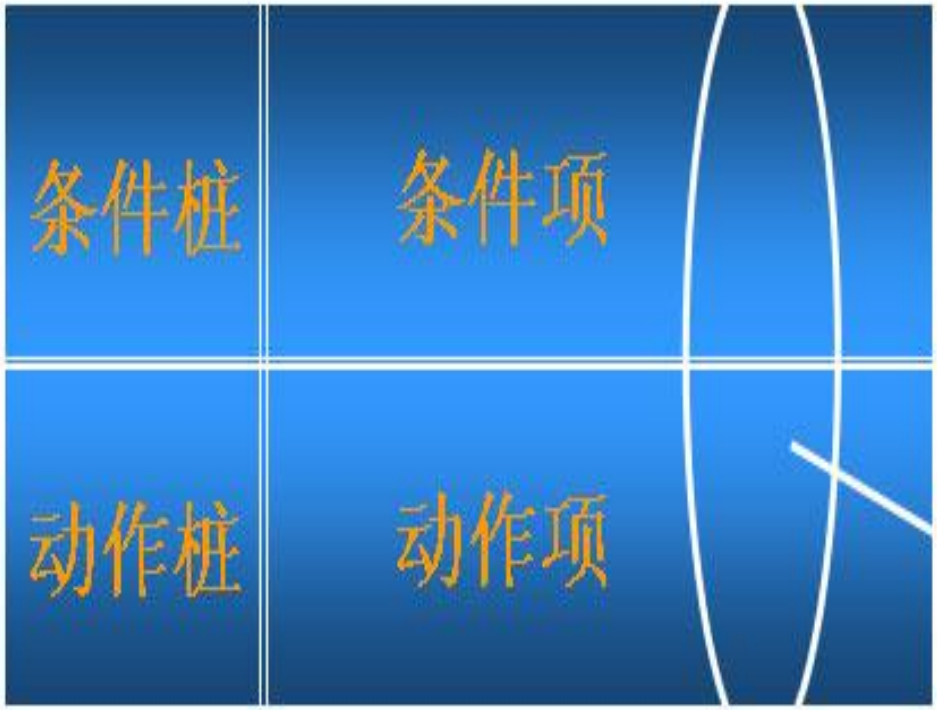
决策表测试

决策表测试是分析和表达多逻辑条件下执行不同操作的情况的工具 ！

决策表能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏。因此，利用决策表能够设计出完整的测试用例集合；

在一些数据处理问题当中，某些操作的实施依赖于多个逻辑条件的组合，即：针对不同逻辑条件的组合值，分别执行不同的操作。决策表很适合于处理这类问题；

决策表组成



决策表组成定义

- 条件桩（Condition Stub）：列出了问题的所有条件。通常认为列出的条件的次序无关紧要；
- 动作桩（Action Stub）：列出了问题规定可能采取的操作。这些操作的排列顺序没有约束；
- 条件项（Condition Entry）：列出针对它左列条件的取值。在所有可能情况下的真假值；
- 动作项（Action Entry）：列出在条件项的各种取值情况下应该采取的动作；

决策表测试案例

假如觉得疲倦，就休息一下；假如不觉得疲倦，并对当前阅读的章节感兴趣，那么继续本章的阅读；假如对本章内容不理解，那就重读本章。假如不感兴趣，那么跳到下个章节进行阅读；

		1	2	3	4	5	6	7	8
条件桩	觉得疲倦	Y	Y	Y	Y	N	N	N	N
	感兴趣	Y	Y	N	N	Y	Y	N	N
	糊涂	Y	N	Y	N	Y	N	Y	N
动作桩	重读					√			
	继续						√		
	跳下一章							√	√
	休息	√	√	√	√				

决策表测试规则

任何一个条件组合的特定取值及其相应要执行的动作就称为规则；

在决策表中贯穿条件项和动作项的一列就是一条规则；

决策表中列出多少组条件取值，也就有多少条规则，既条件项和动作项有多少列；

决策表测试步骤

- 确定规则的个数。假如有n个条件，每个条件有两个取值（0,1）,故有2^n种规则；
- 列出所有的条件桩和动作桩；
- 填入条件项；
- 填入动作项，得到初始的决策表；
- 简化合并相似的规则，比如相同动作，得到优化的决策表；
- 每列规则设计一个测试用例；

决策表规则合并

假如觉得疲倦，就休息一下；假如不觉得疲倦，并对当前阅读的章节感兴趣，那么继续本章的阅读；假如对本章内容不理解，那就重读本章。假如不感兴趣，那么跳到下个章节进行阅读；

有两条或多条规则具有相同的动作，并且其条件项之间存在着极为相似的关系；

		1	2	3	4
条件桩	觉得疲倦	Y	Y	Y	Y
	感兴趣	Y	Y	N	N
	糊涂	Y	N	Y	N
动作桩	重读				
	继续				
	跳下一章				
	休息	√	√	√	√

		1
条件桩	觉得疲倦	Y
	感兴趣	—
	糊涂	—
动作桩	重读	
	继续	
	跳下一章	
	休息	√

决策表规则合并

有两条或多条规则具有相同的动作，并且其条件项之间存在着极为相似的关系；

		7	8
条件桩	觉得疲倦	N	N
	感兴趣	N	N
	糊涂	Y	N
动作桩	重读		
	继续		
	跳下一章	√	√
	休息		

		7
条件桩	觉得疲倦	N
	感兴趣	N
	糊涂	—
动作桩	重读	
	继续	
	跳下一章	√
	休息	

决策表合并优化

		1	2	3	4	5	6	7	8
条件桩	觉得疲倦	Y	Y	Y	Y	N	N	N	N
	感兴趣	Y	Y	N	N	Y	Y	N	N
	糊涂	Y	N	Y	N	Y	N	Y	N
动作桩	重读					√			
	继续						√		
	跳下一章							√	√
	休息	√	√	√	√				

		1	2	3	4
条件桩	觉得疲倦	Y	N	N	N
	感兴趣	—	Y	Y	N
	糊涂	—	Y	N	—
动作桩	重读		√		
	继续			√	
	跳下一章				√
	休息	√			

决策表测试

决策表转化为测试用例

		1	2	3	4
条件桩	觉得疲倦	Y	N	N	N
	感兴趣	—	Y	Y	N
	糊涂	—	Y	N	—
动作桩	重读		√		
	继续			√	
	跳下一章				√
	休息	√			

编号	输入	期望结果
1	觉得疲倦	休息
2	不觉得疲倦、感兴趣，但糊涂	重读
3	不觉得疲倦、感兴趣且不糊涂	继续
4	不觉得疲倦，但不感兴趣	跳下一章

决策表测试应用

- 规格说明以决策表形式给出，或很容易转换成决策表；
- 条件的排列顺序不会影响执行哪些操作；
- 规则的排列顺序不会影响执行哪些操作；
- 每当某一规则的条件已经满足，并确定要执行的操作后，不必检验别的规则；
- 如果某一规则得到满足要执行多个操作，这些操作的执行顺序无关紧要；

决策表测试用例

从决策表的每一列可以清楚地看到条件及其输入的依赖关系，以及由这些输入组合得到的相应的输出动作和结果；

决策表定义了逻辑测试用例，为了执行这些测试用例，必须输入具体的数据值并且标识前置条件和后置条件；

决策表测试覆盖率

最基本的要求是至少用一个测试用例来执行决策表中的每一列，这样就验证了所有关心的输入条件组合和相应的输出结果；

决策表测试价值

根据可能的输入条件组合来定义决策表是一种系统化而且非常正式的方法，它可以覆盖一些在其他测试用例设计技术中没有包含的输入组合。但是，在优化决策表时可能会引入错误，比如忽略了需要考虑的输入和条件的组合。

正如前面提到，在条件的数量和依赖关系增加时，决策表的规模增加得非常快，从而失去可读性。如果没有工具的支持，使用决策表技术就不太容易；

火龙果·整理
uml.org.cn

课程目录

1. 测试设计生命周期
2. 等价类划分与边界值分析
3. 状态转换测试
4. 决策表测试
5. 结对测试
6. 场景与用例测试
7. 需要评审吗

测试某网站系统

某网站需要支持

- 不同的浏览器：IE5.0、IE5.5、IE6.0、Netscape6.0、Netscape6.1、Netscape7.0、Mozilla1.1 和 Opera7；
- 不同的插件：RealPlayer、MediaPlayer 或者没有任何插件 None；
- 不同的客户端操作系统：Windows95、Windows98、WindowsME、、WindowsNT、Windows2000 和 WindowsXP；
- 不同的 Web 服务器软件：IIS、Apache 和 WebLogic；
- 不同的服务器端操作系统：WindowsNT、Windows2000 和 Linux。

$$8 \times 3 \times 6 \times 3 \times 3 = 1296$$

采取什么测试策略？

8 X 3 X 6 X 3 X 3 = 1296

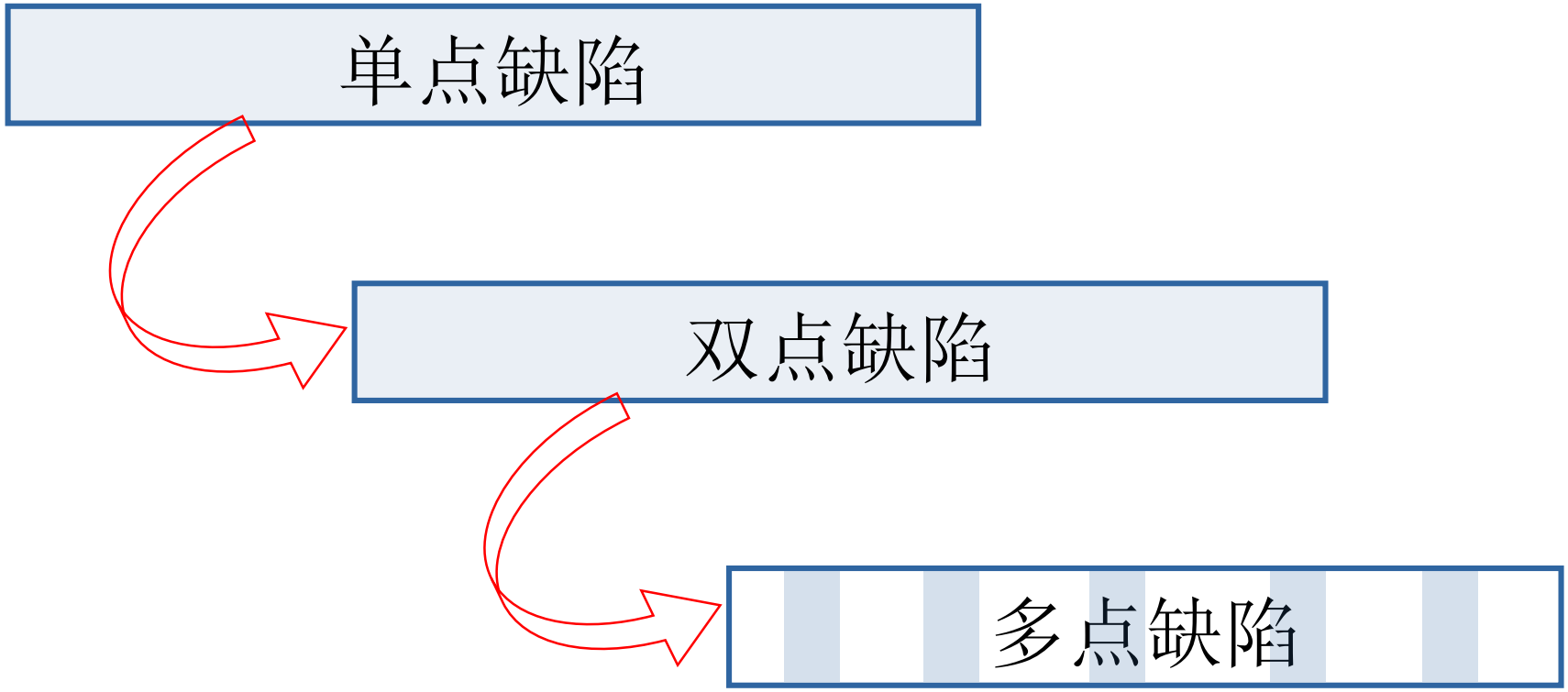
时间、成本、质量的平衡

结对测试

大幅减少测试用例

参数个数	输入值个数/参数	完全组合	结对测试
4	3	$3^4 = 81$	9
13	3	$3^{13} = 1594323$	15
20	10	10^{20}	180

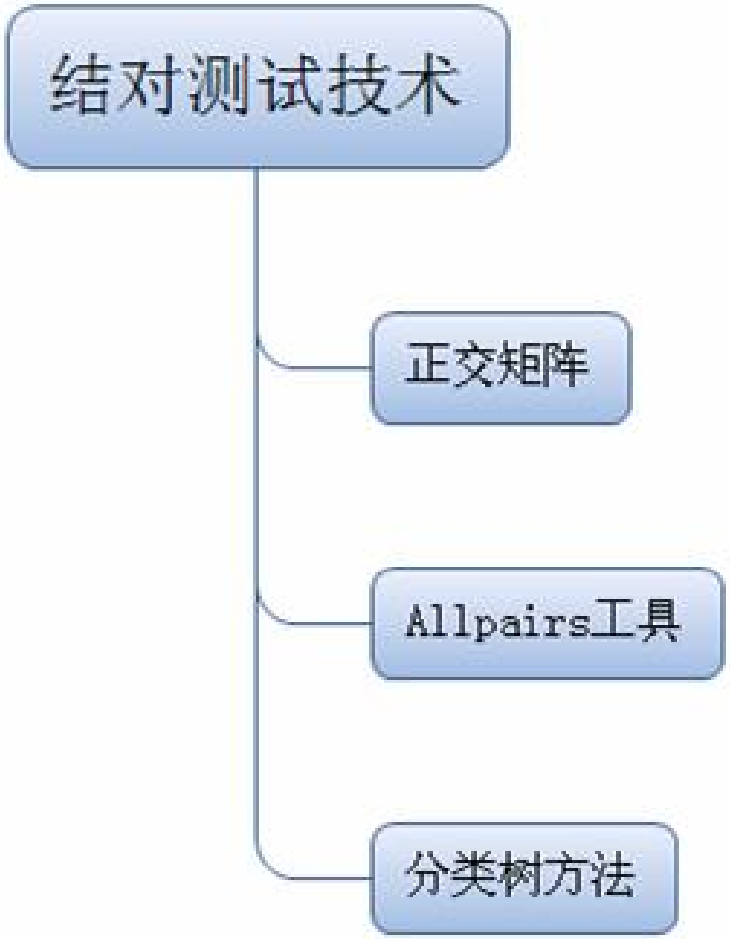
结对测试能发现什么？



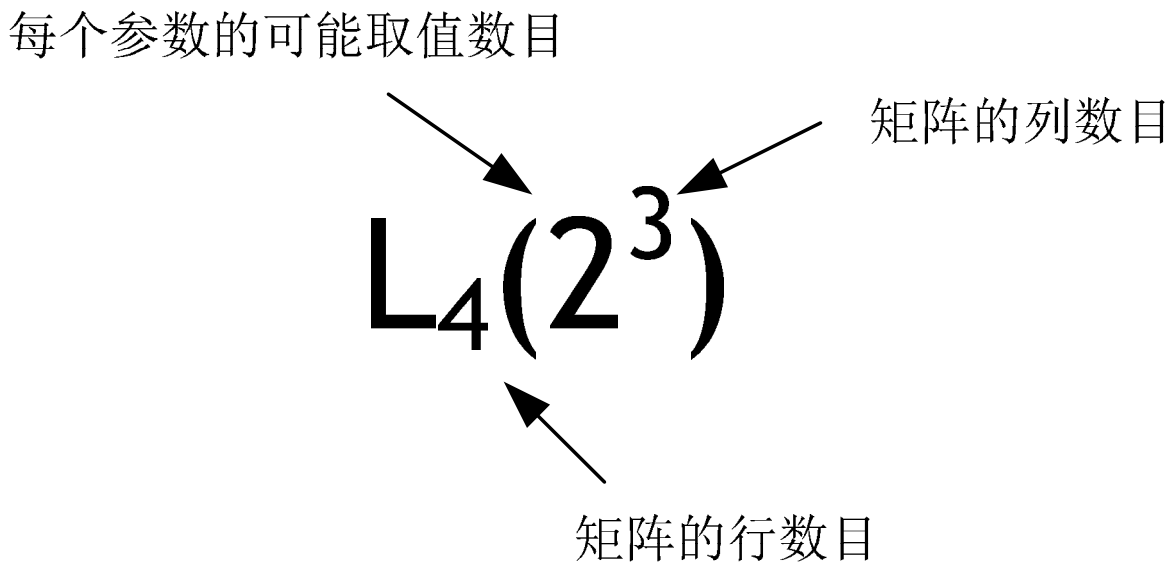
结对测试的有效性？

- 根据 AT&T 在对其基于局域网的邮件系统进行的测试中，应用结对测试技术得到的 1000 条测试用例比其原有的 1500 条测试用例多发现了 20%的缺陷，而测试工作量却减少了 50%。
- National Institute of Standards and Technology 在一项对医疗设备测试所进行的 15 年追踪中发现，有 98%的软件缺陷可以通过结对测试技术发现。
- 根据对 Mozilla 网页浏览器的缺陷分析显示，76%的缺陷可以通过结对测试技术发现。

如何开展结对测试？



正交矩阵



更多资料：http://www2.research.att.com/~njas/

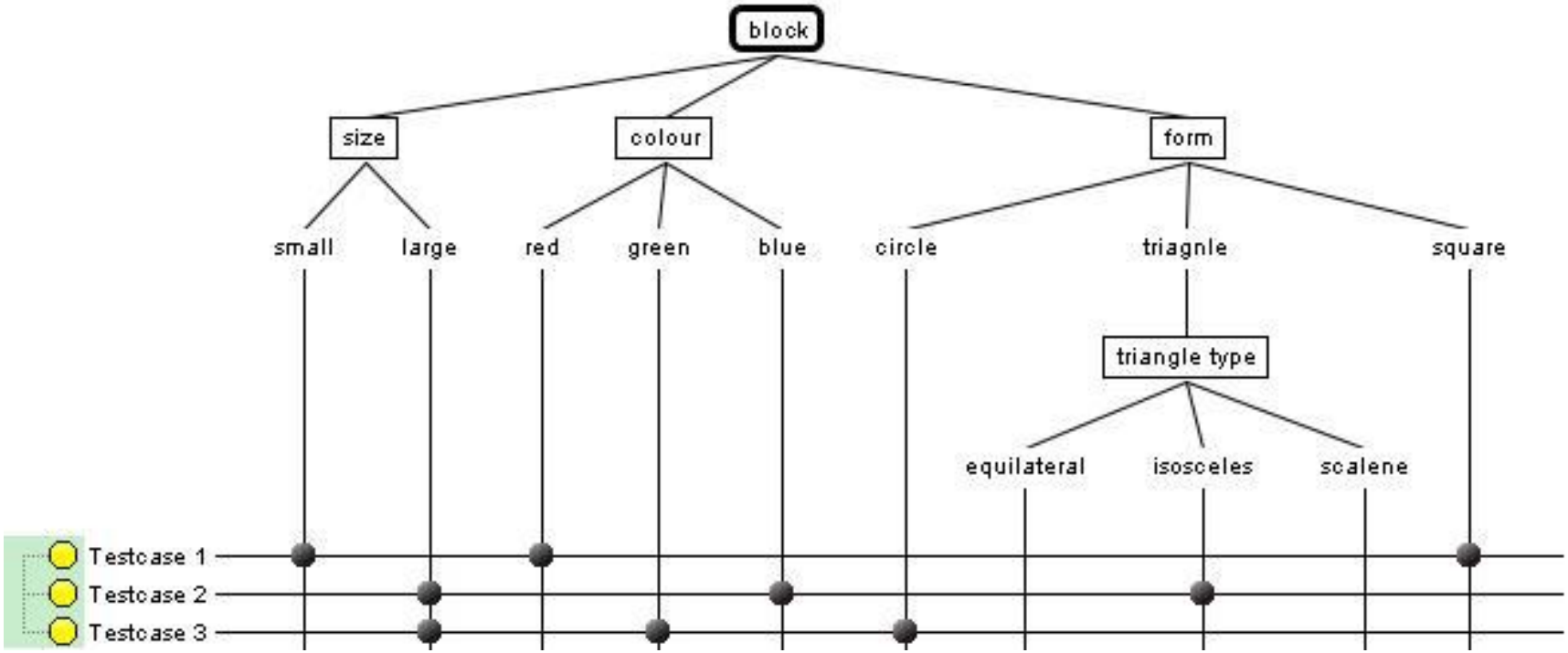
Allpairs工具

Allpairs工具下载：www.satisfice.com/tools.shtml

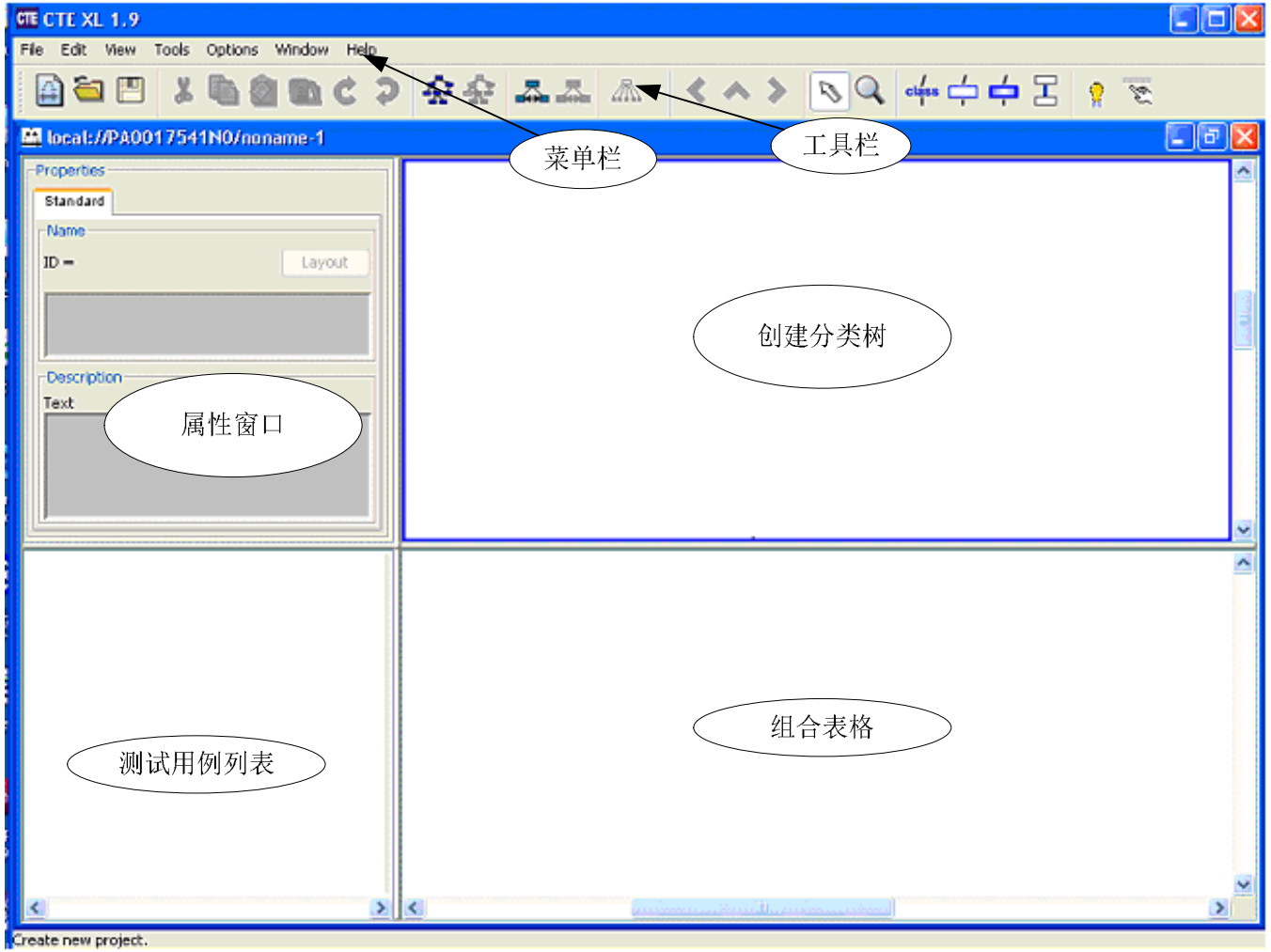
编号 ID	浏览器	插件	客户端操作系统	服务器	服务器端操作系统
1	IE 5.0	None	Windows 95	IIS	Windows NT
2	IE 5.5	RealPlayer	Windows 98	Apache	Windows 2000
3	IE 6.0	MediaPlayer	Windows ME	WebLogic	Linux
4	Netscape 6.0		Windows NT		
5	Netscape 6.1		Windows 2000		
6	Netscape 7.0		Windows XP		
7	Mozilla 1.1				
8	Opera 7				

分类树方法：CTE-XL工具

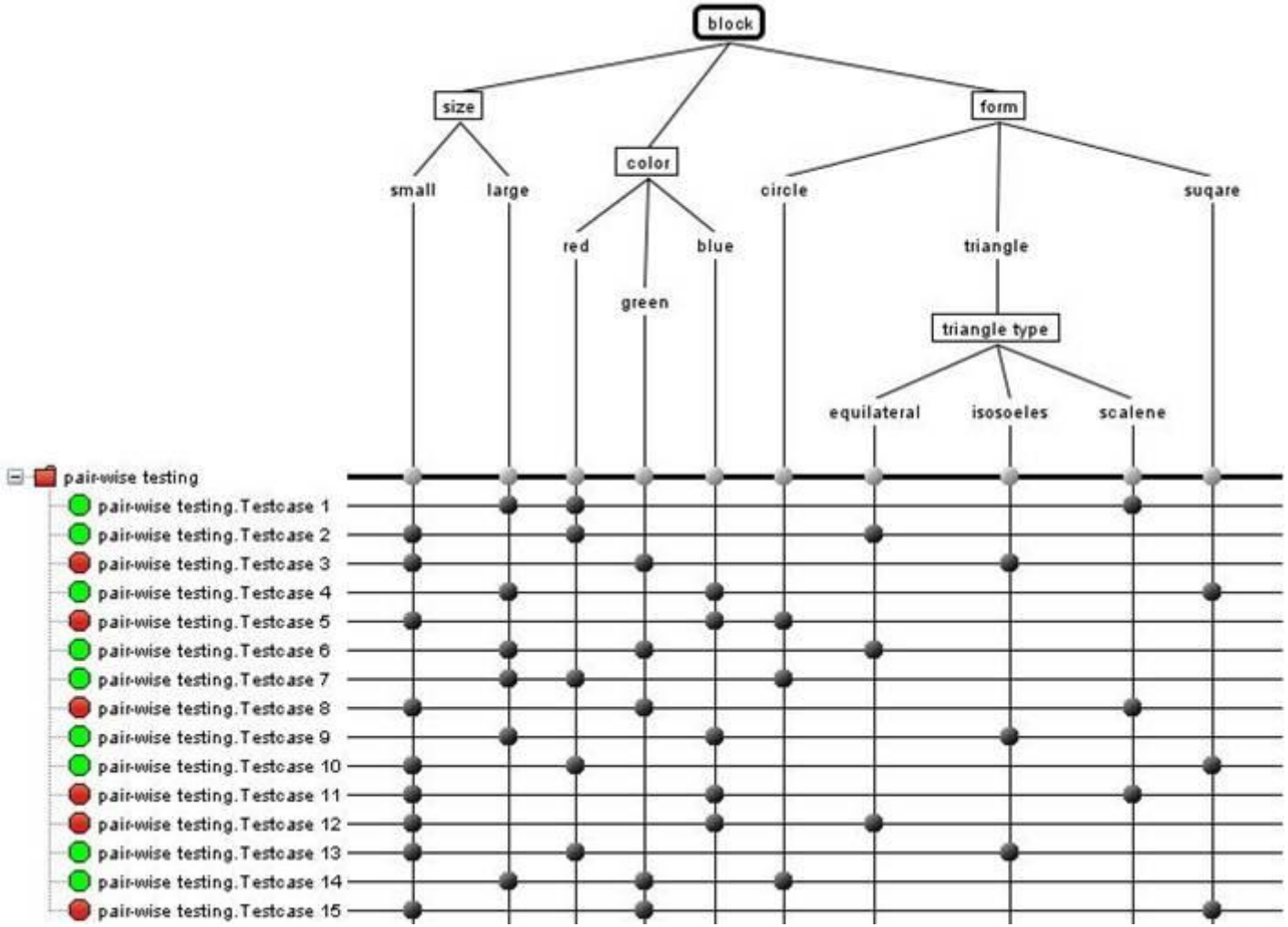
CTE-XL工具下载：http://www.systematic-testing.com/index.php



CTE-XL工具主窗口


CTE XL 1.9
File Edit View Tools Options Window Help
local://PA0017541N0/noname-1
Properties
Standard
Name
ID =
Layout
Description
Text
菜单栏
工具栏
创建分类树
属性窗口
测试用例列表
组合表格
Create new project.

CTE-XL工具（有依赖规则）



课程目录

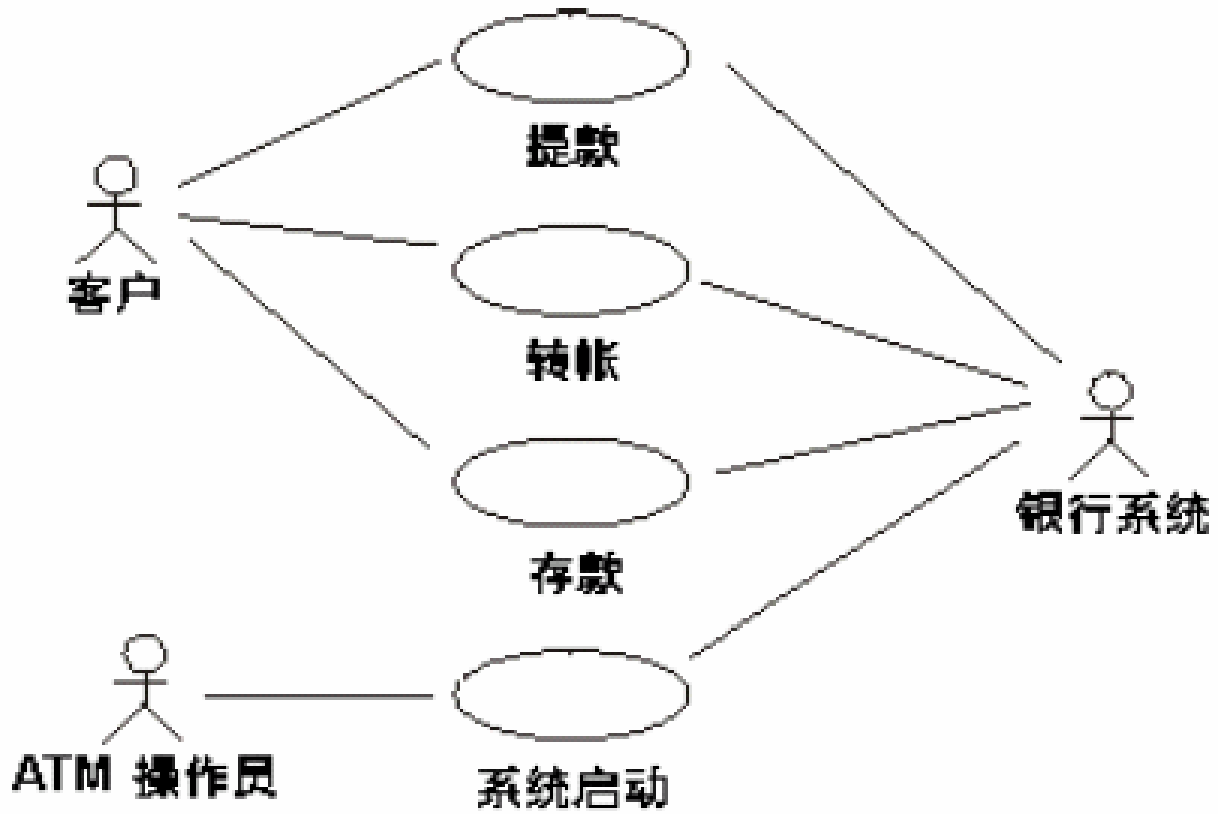
1. 测试设计生命周期
2. 等价类划分与边界值分析
3. 状态转换测试
4. 决策表测试
5. 结对测试
6. 场景与用例测试
7. 需要评审吗

用例测试

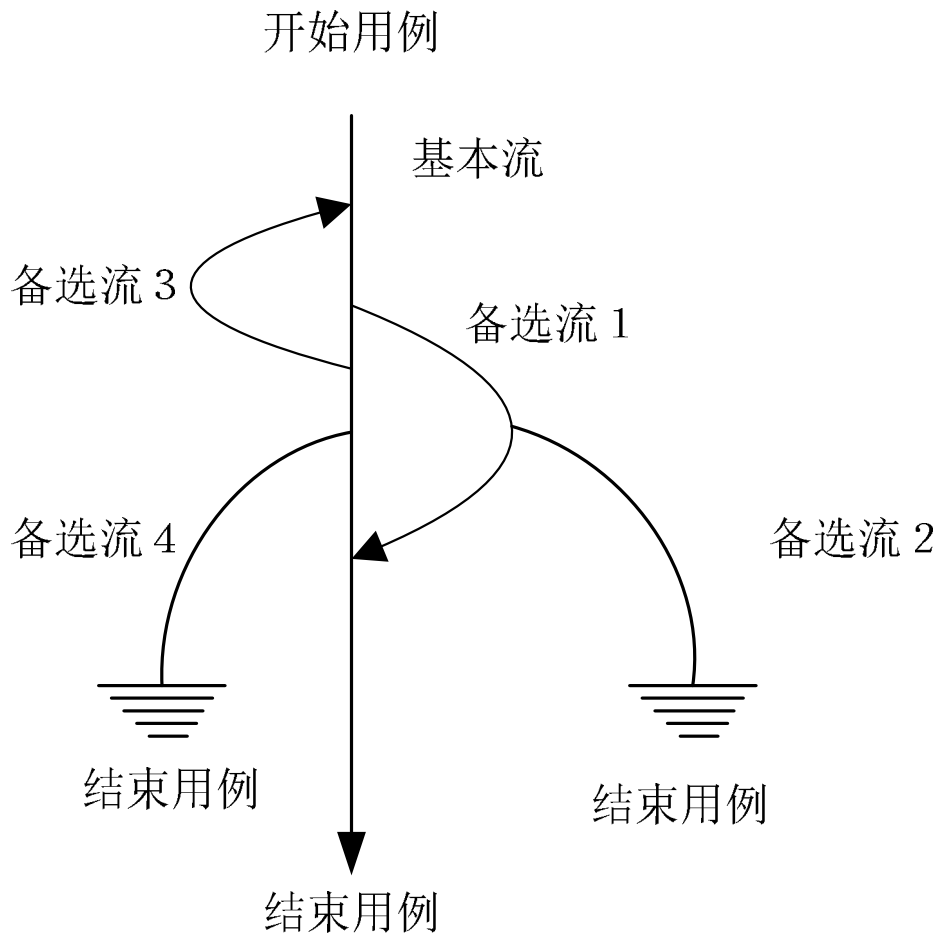
通过用例（Use Cases）或业务场景来设计测试，用例描述了参与者（包括用户与系统）之间的相互作用，并从这些交互产生一个从用户的角度所期望和能观察到的结果；

每个用例都有测试前置条件，这是用例成功执行的必要条件。每个用例结束后都存在后置条件，这是在用例执行完成后能观察到的结果和系统的结束状态；

用例测试案例



用例测试表现形式



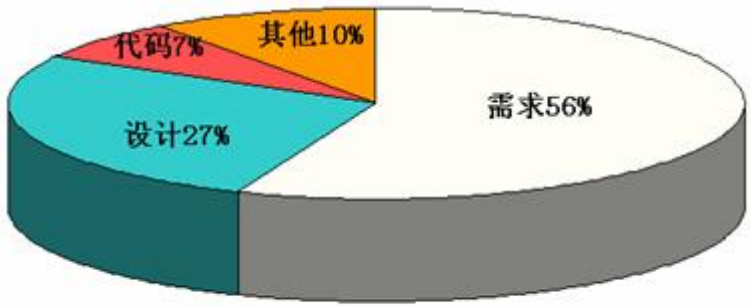
用例测试的价值

- 基于用例的测试技术适合于测试典型的用户系统交互。
- 其他测试技术（如边界值分析）可以在这方面做些辅助；

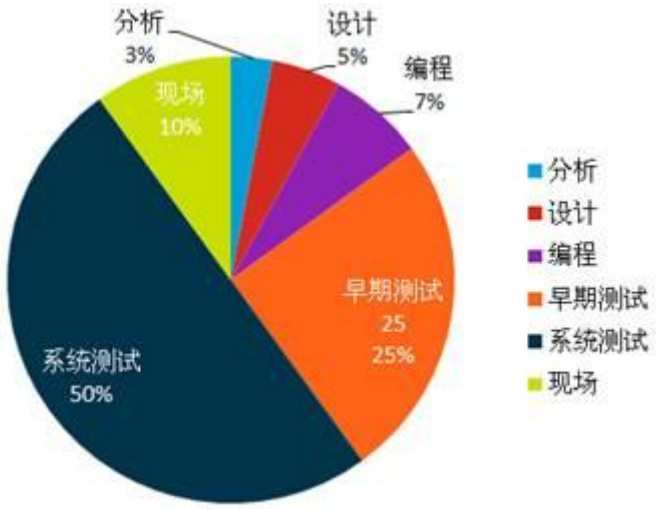
课程目录

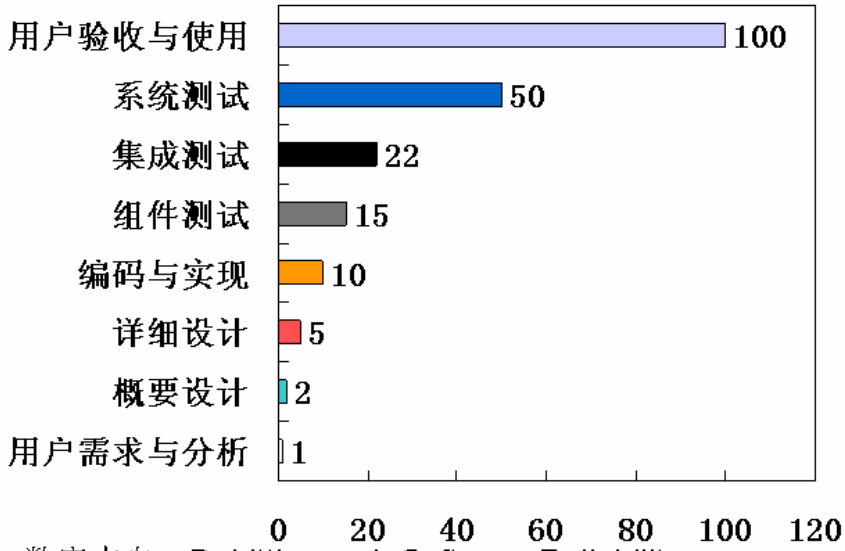
1. 测试设计生命周期
2. 等价类划分与边界值分析
3. 状态转换测试
4. 决策表测试
5. 结对测试
6. 场景与用例测试
7. 需要评审吗

为什么评审？



数据来自Dick Bender（1993）的"Writing testable requirement"，图中的"需求"包含了功能设计。

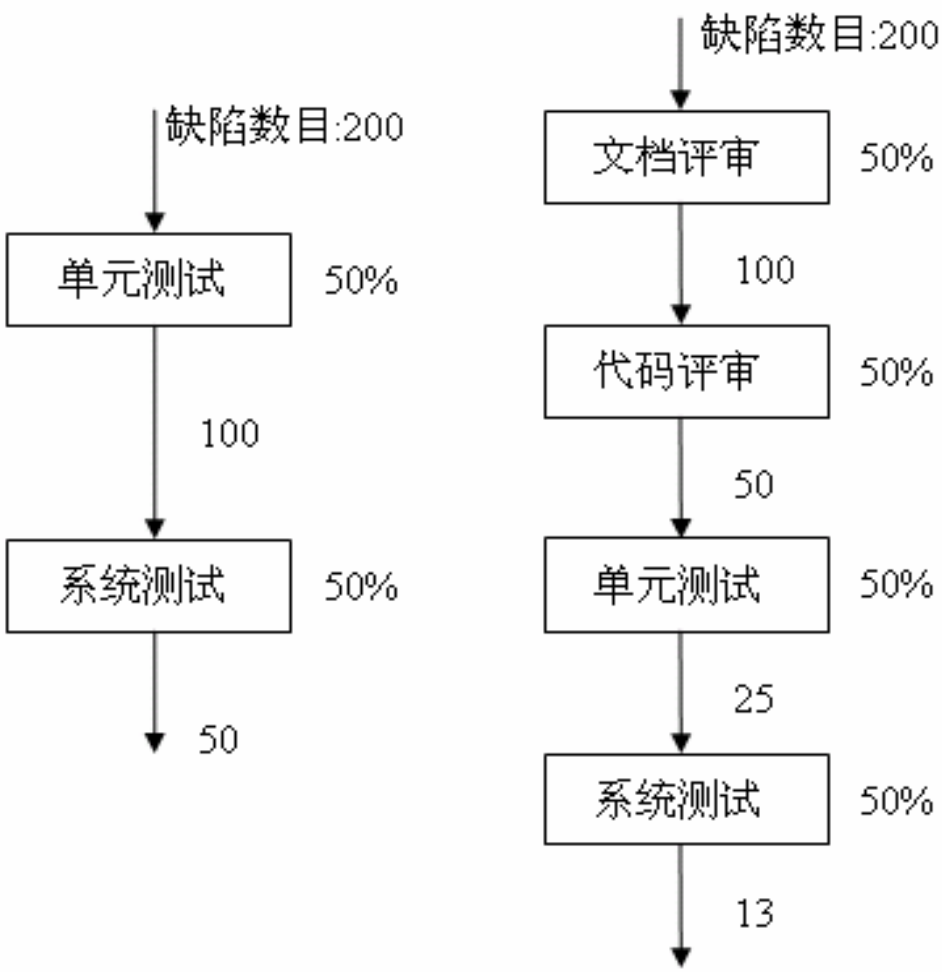




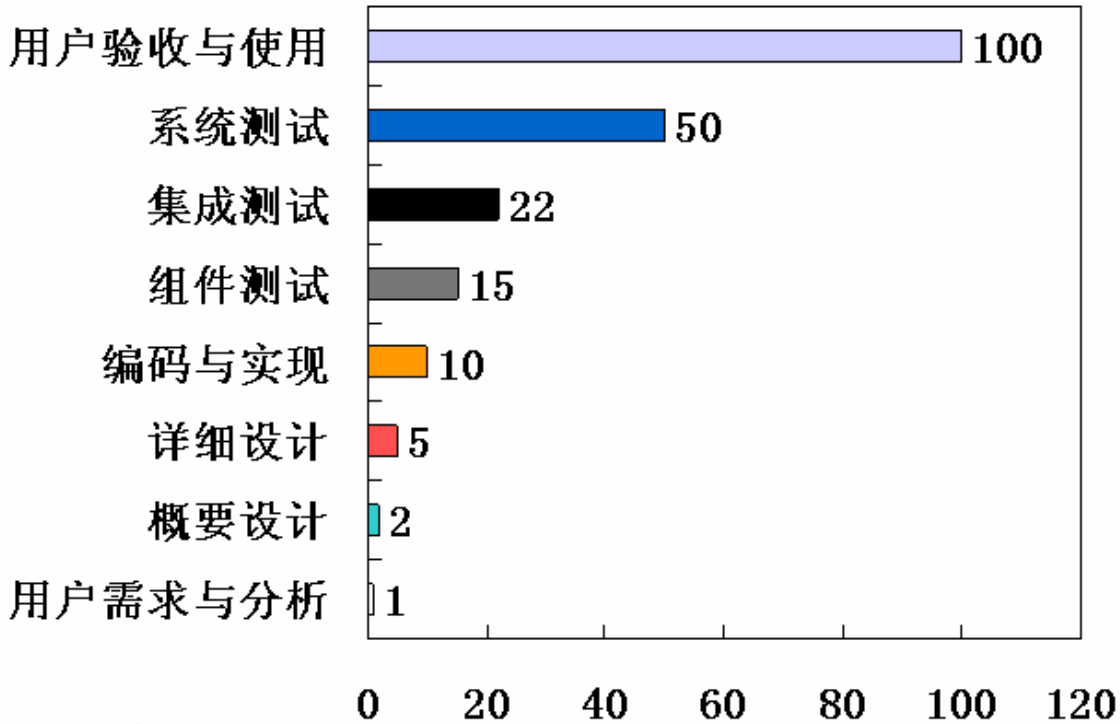
数字来自：B. Littlewood, Software Reliability: Achievement and Assessment，Blackwell Scientific Publications， November 1987

静态测试的优点

提高质量

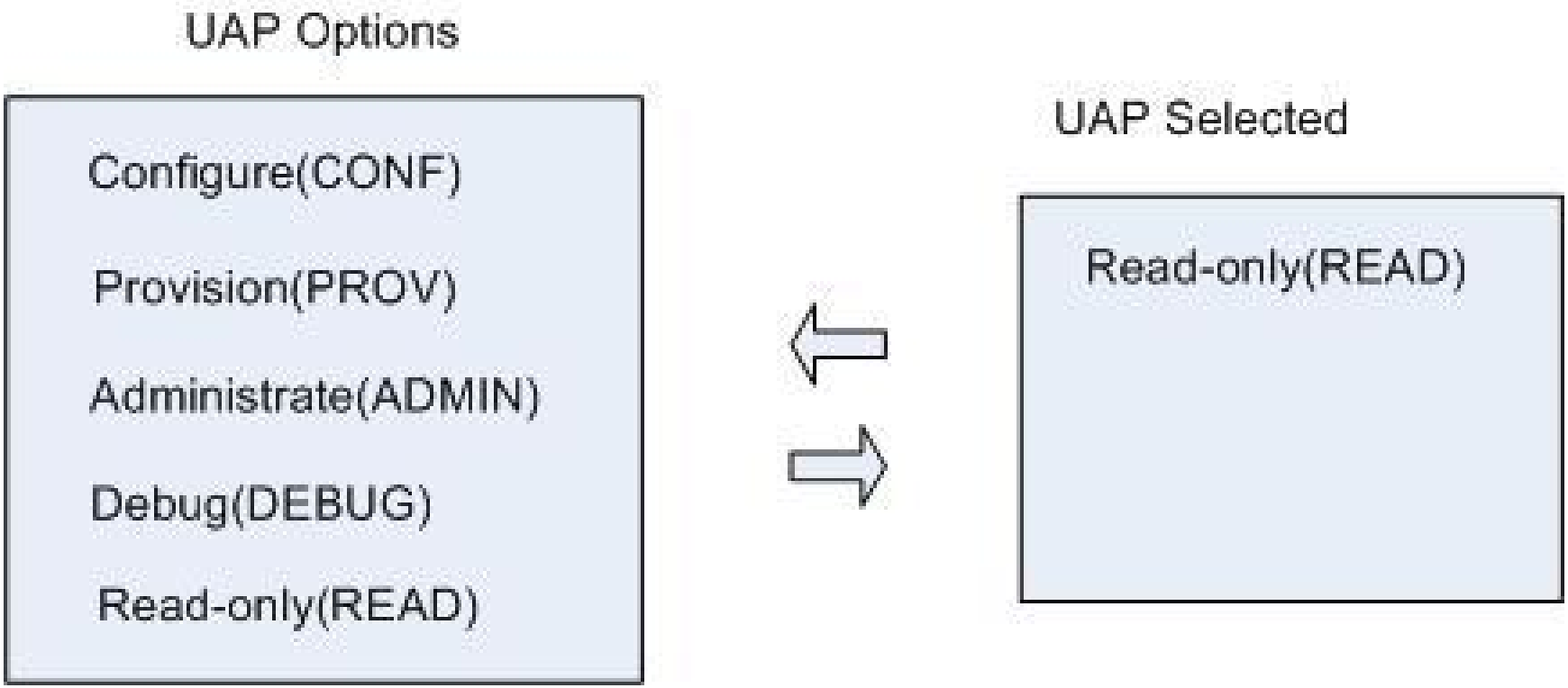

缺陷数目:200
单元测试
50%
100
系统测试
50%
50
缺陷数目:200
文档评审
50%
100
代码评审
50%
50
单元测试
50%
25
系统测试
50%
13

降低成本



数字来自：B. Littlewood, Software Reliability: Achievement and Assessment，Blackwell Scientific Publications， November 1987

加快进度

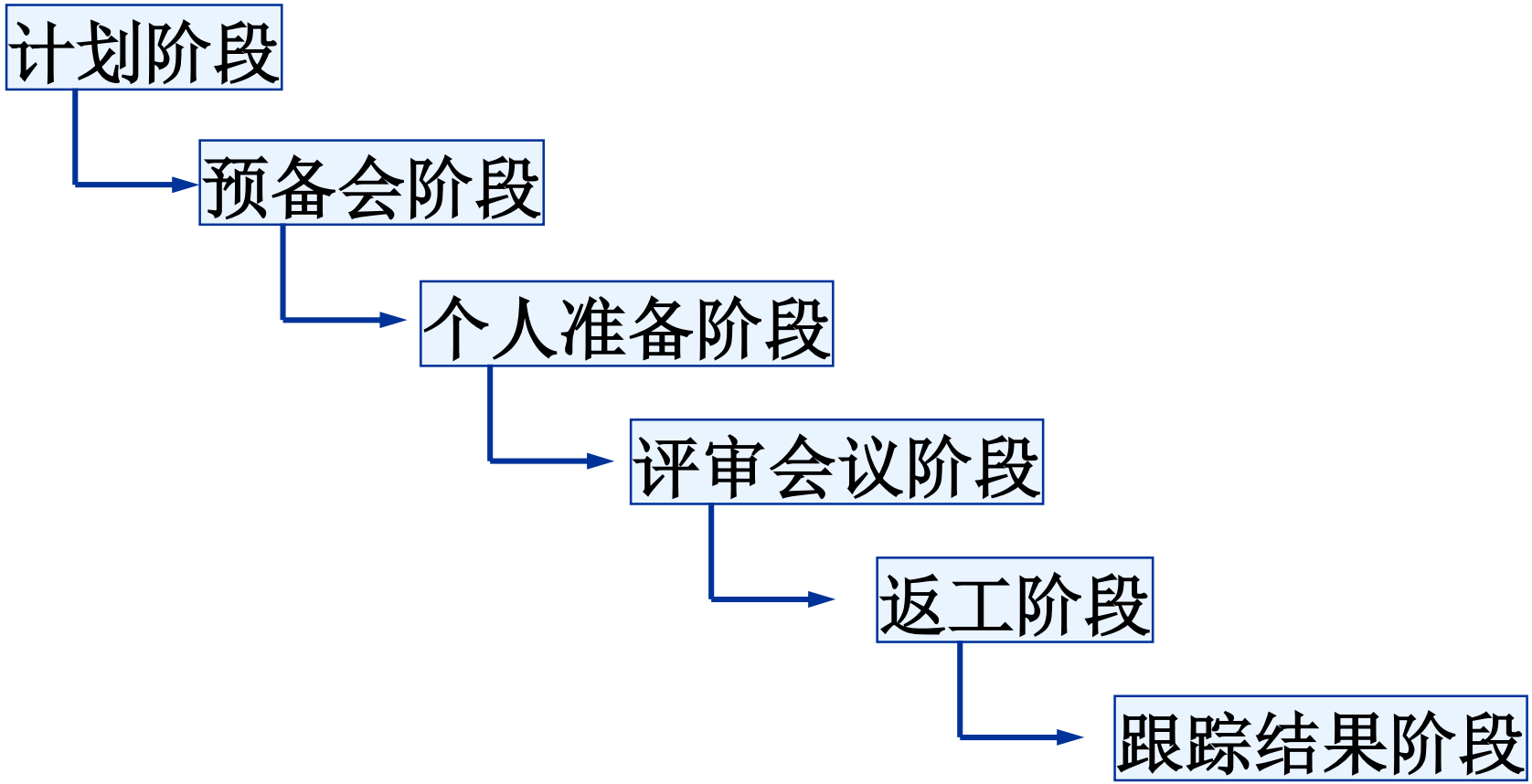

UAP Options
Configure(CONF)
Provision(PROV)
Administrate(ADMIN)
Debug(DEBUG)
Read-only(READ)
UAP Selected
Read-only(READ)

改进过程

？

如何改进过程

正式评审的过程



角色和职责

经理

决定是否进行评审，在项目计划中分派时间，判断是否达到评审目标

主持人

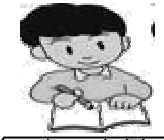
主持文档或文档集的评审活动，包括策划评审、召开会议和会议后的跟踪。假如需要，主持人可能还需要进行不同观点之间的协调。
主持人通常是评审成功与否的关键。

作者

待评审文档的作者或主要责任人。

评审员

具有专门技术或业务背景的人员，他们在必要的准备后，标识和描述被评审产品存在的问题。所选择的评审员应该在评审过程中代表不同的观点和角色，并且应该参与各种评审会议。



记录员

记录所有的事件、问题，以及在会议过程中识别的未解决的问题。

有效评审的实施

小结

1. 测试设计生命周期
2. 等价类划分与边界值分析
3. 状态转换测试
4. 决策表测试
5. 结对测试
6. 场景与用例测试
7. 需要评审吗

第2章：基于规格说明的测试设计