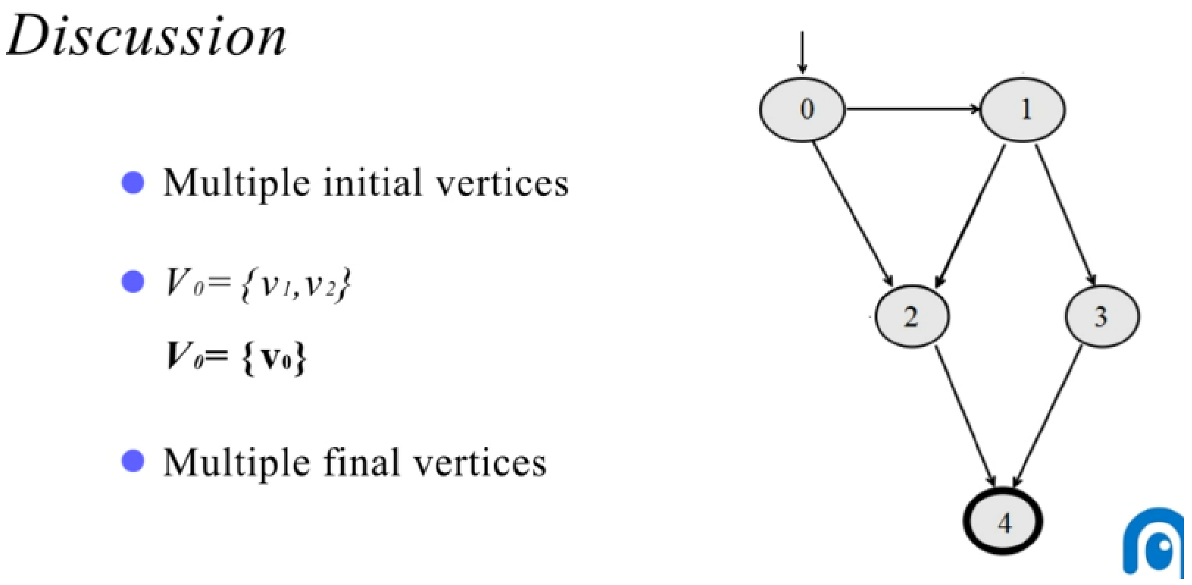
P1 测试中的图/路径

A graph G is defined as follows: G=(V, E)

- V: a **finite**, **nonempty** set of vertices V={v_1, v_2, v_3, v_4}
- E: a set of edges (pairs of vertices) E={(v_1,v_2), (v_3, v_4)}
- V_0初始节点（集合）、V_f终结节点（集合）都是V的子集
- 

Discussion

- Multiple initial vertices
- $V_0 = \{v_1, v_2\}$

  $V_0 = \{\mathbf{v_0}\}$
- Multiple final vertices

- 可以通过添加**哑节点**的方式，将多个初始（终结）节点变成一个（0是12的哑节点）----这门课约定：只有一个初始（终结）节点。

Path：A sequence of vertices ----- [v_1, v_2.... v_n]

- Each pair of vertices is an edge.
- Length: the number of edges, a single vertex is a path of length 0
- Subpath: a subsequence of vertices in p（它前提是一个路径，不能断开）

Test Path: a path that **starts at an initial vertex and ends at a final vertex**.

- Test paths represent execution of test cases.
    - Some test paths can be executed by many tests.
    - Some test paths cannot be executed by any tests.（不可判定问题）
- path(t): the test path executed by test t.
- path(T): the set of test paths executed by the set of tests T.

P2 测试中的图

P3 可达/覆盖/测试准则

Reach可达

- Syntactic reach语法可达：A path exists in the graph
- Semantic reach语义可达：A test exists that can execute that path

Cover覆盖

- A test path p covers a vertex v if v is in p
- A test path p covers an edge e if e is in p

Structural Coverage结构覆盖

- Defined on a graph just in terms of vertices and edges

Test Criteria测试准则

- Test Requirements(TR): Describe properties of test paths
- Test Criterion: Rules that define test requirements
- 给定一个测试准则C，对应派生的一个测试需求集TR，我们称之为一个测试用例集T满足这个准则，是指测试需求集TR里面的每一个测试需求tr都可以被测试用例集T的某一测试用例t所满足。

P4 顶点覆盖/边覆盖/边对覆盖

Vertex Coverage顶点覆盖(VC)

- Test set T satisfies vertex coverage on graph G if and only if for every syntactically reachable vertex v in V, there is a path p in path(T) such that p covers v.
- TR contains each reachable vertex in G.

Edge Coverage边覆盖(EC)

- Test set T satisfies edge coverage on graph G if and only if for every syntacically reachable edge e in E, there is a path p in path(T) such that p covers e.
- TR contains each reachable edge in G.

满足EC即满足VC，但满足VC不一定满足EC

Edge-Pair Coverage(EPC)

- TR contains each reachable path of length to up 2, inclusive, in G.

Complete Path Coverage(CPC): TR contains all paths in G.

n-Path Coverage(nPC): TR contains each reachable path of length up to n, inclusive, in G.

- VC(n=0), EC(n=1), EPC(n=2), CPC(n=∞)

Subsume蕴含

- C1 subsumes C2, denoted by C1≥C2:
  - For any T, if T satisfied C1 implies T satisfies C2.
- C1≥C2 does not imply that T_1 satisfying C1 can detect any fault detected by T_2 which satisfies C2.

P5 控制流图

Control Flow Graph: a **control flow graph(CFG)** is a representation, using graph notation, of all paths that might be traversed through a program during its execution.

- Vertex:
  - Statement
  - Block
  - Function
  - Module
- Edge:
  - Flow
  - Jump
  - Call

Soot自动为Java程序产生控制流图。

P6 数据流机覆盖准则

Data Flow

- Beyond structure
- Goal: try to ensure that values are computed and used correctly
- 数据流中对变量的操作
  - Definition(def): a location where a value for a variable is stored into memory
    - def(n) or def(e): the set of variables that are defined by node n or edge e.
  - Use: a location where a variable's value is accessed
    - use(n) of use(e): the set of variables that are used by node n or edge e.
  - The values given in defs should reach at least one, some, or all possible uses.

    - DU pair:
        - a pair of locations (l_i, l_j) such that a variable v is defined at l_i and used at l_j.
    - Def-clear:
        - a path from l_i is def-clear with respect to variable v if v is not given another value on any of the nodes or edges in the path.
        - if there is a def-clear path from l_i to l_j with respect to v, the def of v at l_i reaches the use at l_j.
- DU Path
    - du-path: a simple subpath that is def-clear with respect to v **from a def of v to a use of v**.
    - du(n_i, n_j, v): the set of du-paths from n_i to n_j
    - du(n_i, v): the set of du-paths that start at n_i

覆盖准则

- All-defs coverages(ADC): for each set of du-paths S = du(n, v), TR contains **at least one** path d in S.
- All-uses coverage(AUC): for each set of du-paths to uses S = du(n_i, n_j, v), TR contains **at least one** path d in S.
- All-du-paths coverage(ADUPC): for each set S = du(n_i, n_j, v), TR contains **every** path d in S.

P7 JUNIT使用示例

JUnit是一个强大的java的单元测试框架。

P8 随机测试

Test cases are generated purely at random:

- input domain must be known,
- pick random points within input domain,
- automation(自动化).

Problems in Random Test:

- define input domain
- generate random sequence(在计算机中很难得到真正的随机数)
    - psseudo random sequence algorithm
    - seed of pseudo random sequence algorithm
    - randomness and integrity service(random.org)

Random Test in security testing

- Fuzz testing(Fuzzing)
    - a special from of random testing, aims to breaking the software
    - providing invalid, unexpected, or random data to the inputs of a program

能够导致程序出错的测试用例一般归结于：矩形状分布；条带状分布；散点状分布。（前两种分布可以利用聚集特性来提高测试效率）

FSCS-ART algorithm自适应随机测试

- randomly generate an input t, run t, add t to T,
- while(stop criteria not reached)跳出条件可能是找到错误，或测试资源耗尽
  - randomly generate k candidate input c_1,...c_2
  - for each candidate c_i
    - compute min distance d_i to T // 通过程序来自行定义distance如何计算
  - end for
  - select one candidate t with **max** distance
  - run t, add t to T
- end while

Anti-Random Testing（对于输入域不连续的情况）

- Process
  - select one test case randomly
  - select a test case with the **maximum Sum** of Hamming Distance to existed test cases
  - Repeat...

P9 等价类划分

Equivalence Partitioning

- partitioning the **input domain** into a collection of subsets(or equivalent classes)
- partition criterion
  - different computational results
  - relation based on control flow or data flow
  - distinction between valid and invalid inputs
- valid class and invalid class
  - valid class: test the function of program
  - invalid class: test the exception handle of program
- 完备性原则和无冗余原则
  - complete: equivalent classes must cover the whole input domain(both valid and invalid)
  - no redundant: intersection of different equivalent classes should be empty
- method
  - partition according to the range: for a given range of input variable, there are one valid class and two invalid classes.
  - partition according to processing methods: program processes input with n different methods for n sub-set of input values. there are n valid class and one invalid class.

  - partition according to the single condition: value of input variable comes from a set or satisfies a single condition. there are one valid class and one invalid class.
  - partition according to multiple conditions: value of input variable satisfies multiple conditions(totally n conditions). there are one valid class and n invalid classes.

P10 边界值分析

P11 组合测试

All Combination: test all possible combinations of parametric values.

Pair-wise testing: 任意两个之间的取值组合都出现

T-wise/T-way combinatorial testing: ...

Variable strength combinatorial testing: ...（可变粒度）

P16 PIE模型

Fault, Error & Failure:

- software fault: a **static** defect in the software(在coding的时候犯错)
- software error: an incorrect **internal** state that is the manifestation of some faults 运行过程中触发了错误的中间状态
- software failure: **external**, incorrect behavior with respect to the requirements or other description of the expected behavior.测试人员或用户可以观察到这个出错行为

PIE Model

- Execution/Reachability: the location or locations in the program that contain the fault must be reached.
- Infection: the state of the program must be incorrect.
- Propagation: the infected state must propagate to cause some output of the program to be incorrect.

P17 测试术语

Test Case:

- Test Input: test data
- Test Oracle: expected output
- others: environment

Testing vs. Debugging:

- Testing is to reveal a bug: by executing test and observing failure.
- Debugging is to fix a bug: by locating, understanding and correcting fault.

Verification vs. Validation确认

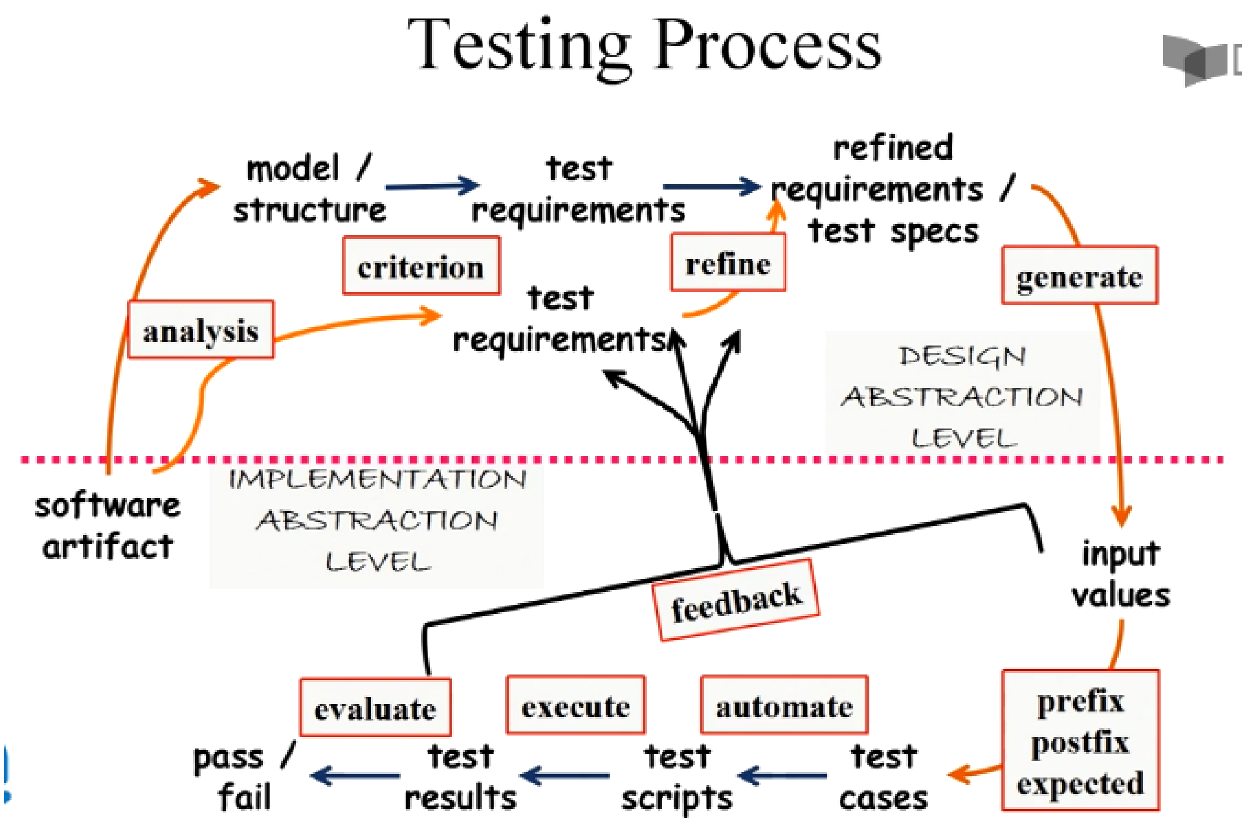
- Verification: the evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process.
- Validation: the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers.

Static Testing vs. Dynamic Testing: 是否需要运行程序

Black-box Testing vs. White-box Testing: 是否有源代码(灰盒是指通过其他代码或反编译手段等获得了部分源代码信息)

Testing Level:

- System testing（build后）
- Integration testing(模块集的组合)
- Module testing
- Unit testing



P18 Fault反思

P19 功能测试简介

功能测试：根据产品特性和设计需求，验证一个产品的特性和行为是否满足设计需求

功能测试常用步骤：

- 根据需求来细分功能点

- 根据功能点派生测试需求
- 根据测试需求设计功能测试用例
- 逐项执行功能测试用例验证产品

相关的测试类型

- 正确性：产品功能是否与需求和设计文档一致
- 可靠性：用户交互是否引发软件崩溃和其它异常
- 易用性：软件产品完成特定任务的难易程度

P20 探索式测试

探索式测试：

- 一种软件测试风格
- 强调独立测试人员的个人自由和职责
- 将测试学习、设计、执行、结果分析作为相互支持的活动
- 在整个项目过程中并行地执行
- 为了持续优化其工作的价值，将测试学习、测试设计、测试执行、测试结果分析在一个迭代循环中快速地迭代，通过测试进行优化

产品元素：需要测试的对象

- 结构：产品的物理元素，如代码、硬件、配置文件、数据文件等
- 功能：产品的功能
- 数据：产品所操作的数据
- 接口：产品所使用的或暴露出的接口
- 平台：产品所以来的外部元素
- 操作：产品被使用的方式
- 时间：影响产品的时间因素

测程（session）：用来组织软件测试

- 任务：通过漫游测试，建立Notepad的初始功能列表（Function List）
  - 功能列表的特征：
    - 建立了被测对象的整体模型
    - 提供了可扩展的测试设计框架
    - 提供了测试覆盖的目标
    - 用简洁的形式提供丰富的信息
- 潜在目标：学习软件功能，建立测试模型

P26 性能测试简介

性能测试度量方法

- 不同的关注对象采用不同的性能的度量方法
- 服务端性能采用CPU、内存等使用率来度量
- 客户端性能通常根据系统处理特定用户请求的响应时间来度量

响应时间

- 响应时间是指系统对请求作出响应所需要的时间
- 响应时间划分为：
    - 服务端响应时间是指从请求发出开始到客户端接受到最后一个字节数据所消耗的时间。
    - 客户端响应时间是指客户端收到响应数据后呈现/响应用户所消耗的时间。

并发用户数：其取决于测试对象的目标业务场景。需要先确定业务场景，然后基于场景采用某些相应方法计算并发用户数。

吞吐量：指单位时间内处理的用户请求数量，如访问人数/天、页面数/秒、请求数/秒等

负载测试：用于验证应用系统在正常负载条件下的行为，性能行为通过一些性能指标体现

- 两种方式：要么直接到达负载数，要么逐步增加负载数

压力测试：

- 评估应用系统处于或超过预期负载时的行为。
- 压力测试关注的行为不一定是性能行为，可能是某种bug，比如同步问题，内存泄漏等。
- 在压力级别逐渐增加时，系统性能应该按照预期缓慢下降，但是不应该崩溃。
- 压力测试还可以发现系统崩溃的临界点，从而发现系统中的薄弱环境。

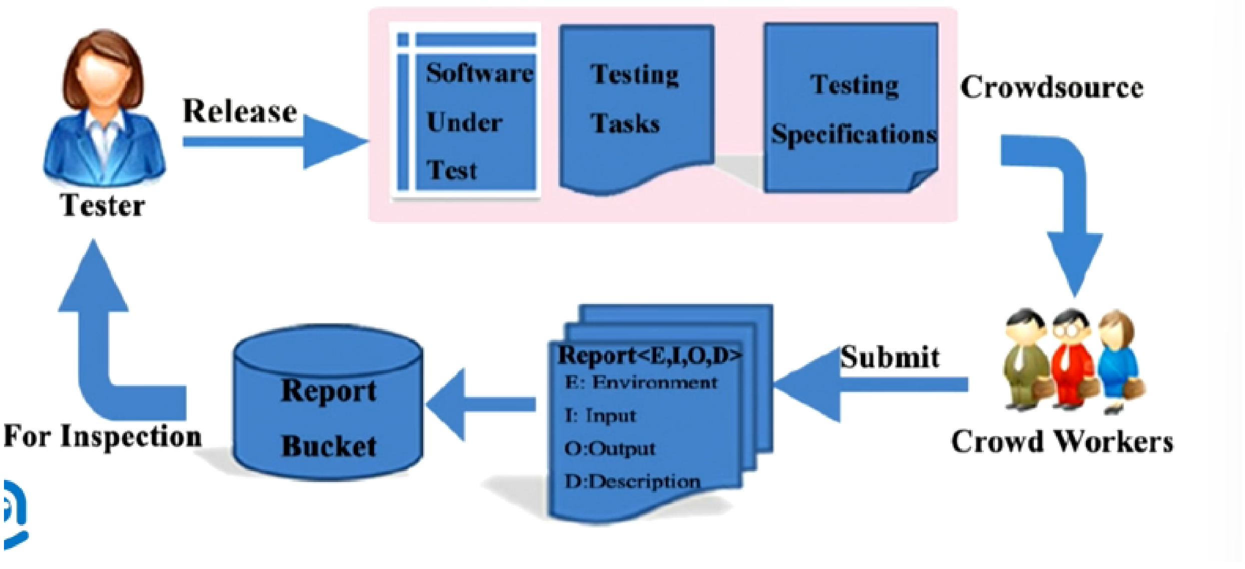
P33 JMeter使用演示1

P34 JMeter使用演示2

JMeter中集合点的使用。

P35 移动应用测试简介

众包测试


Tester
Release
Software Under Test
Testing Tasks
Testing Specifications
Crowdsource
Crowd Workers
Submit
Report<E,I,O,D>
E: Environment
I: Input
O:Output
D:Description
Report Bucket
For Inspection