CS10102302
软件系统测试

叶晨 教授级高工

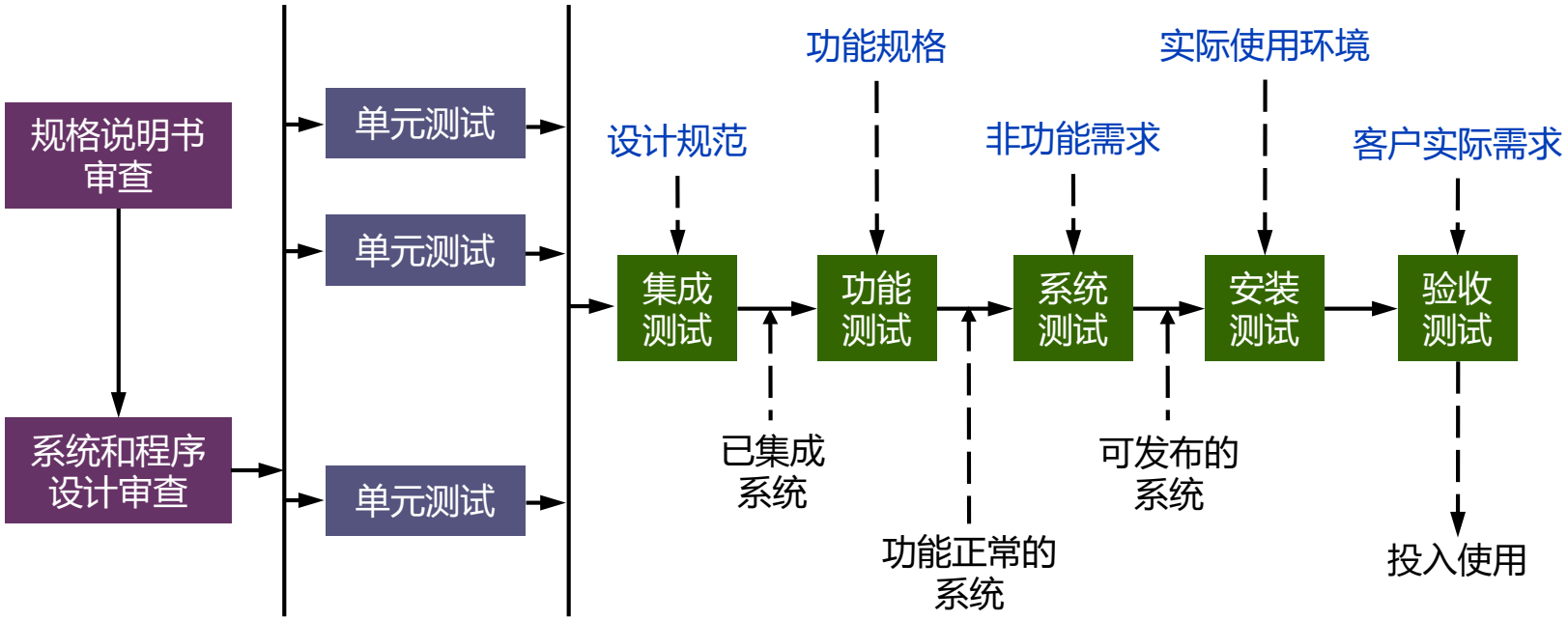
计算机科学与技术系 电子与信息工程学院

同济大学

软件系统测试

1. 集成测试
2. 功能测试
3. 性能测试
4. 软件缺陷管理

回顾：软件测试阶段


规格说明书
审查
系统和程序
设计审查
单元测试
单元测试
单元测试
设计规范
功能规格
非功能需求
实际使用环境
客户实际需求
集成
测试
功能
测试
系统
测试
安装
测试
验收
测试
已集成
系统
功能正常的
系统
可发布的
系统
投入使用

软件测试的目的是

- ○ 找出软件的所有错误
- ○ 评价软件的质量
- ○ 证明软件是正确的
- ○ 发现软件的错误

关于软件测试基本概念描述错误的是（）。

- ○ 软件测试方法分为静态分析法和动态测试法
- ○ 白盒法是一种静态分析方法，可尽可能早的发现缺陷，主要用于模块测试
- ○ 软件测试的基本步骤的顺序依次为：单元测试、集成测试、系统测试、验收测试
- ○ 软件测试通常很难用“穷举法”进行测试

下列说法错误的是（）。

- ○ 等价类划分法属于白盒测试法
- ○ 条件覆盖属于白盒测试法
- ○ 边界值分析法属于黑盒测试法
- ○ 分支覆盖属于白盒测试法

以下哪种测试方法不属于白盒测试技术

- ○ 循环覆盖测试
- ○ 边界值分析测试
- ○ 逻辑覆盖测试
- ○ 基本路径测试

使用白盒测试方法时,确定测试数据应根据( )和指定的覆盖标准

- ○ 该软件的编辑人员
- ○ 程序的内部逻辑
- ○ 程序的功能
- ○ 程序的复杂程度

Apifox

常用解决方案

API 设计者

Swagger

API 文档设计

Postman

API 开发调试

后端开发

Mock.js

API 数据Mock

前端开发

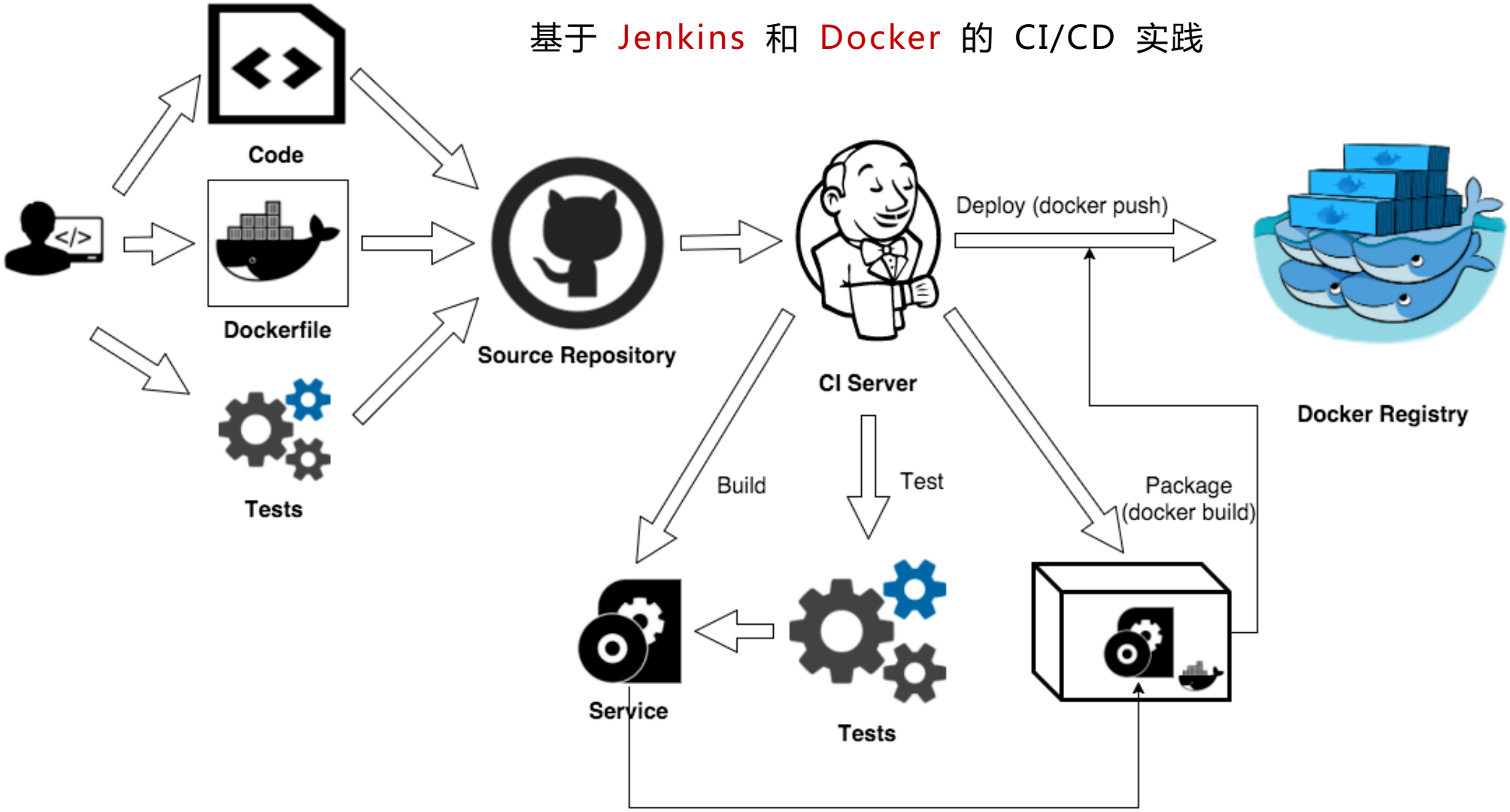
JMeter

API 自动化测试

API 压力测试

测试人员

基于 Jenkins 和 Docker 的 CI/CD 实践



Reference

- GitLab的CI/CD配置流程
  - ✓ https://juejin.cn/post/7309158055018905626

- Gitlab配置Docker自动部署
  - ✓ https://blog.csdn.net/qq_43692950/article/details/134217402

- Gitlab配置Docker自动部署
  - ✓ https://www.bilibili.com/video/BV1BV411G7Q5/?share_source=copy_web&vd_source=0564a7a68c171ecceef424b200481fe8

教学提纲

1 集成测试

- 集成测试概念
- 基于功能分解的集成
- 持续集成
- 集成测试原则

集成测试

集成测试（Integration Testing）是在单元测试的基础上，将所有软件单元按照总体设计的要求组装成为子系统或系统进行的测试，其目的是发现各单元接口之间存在的问题。

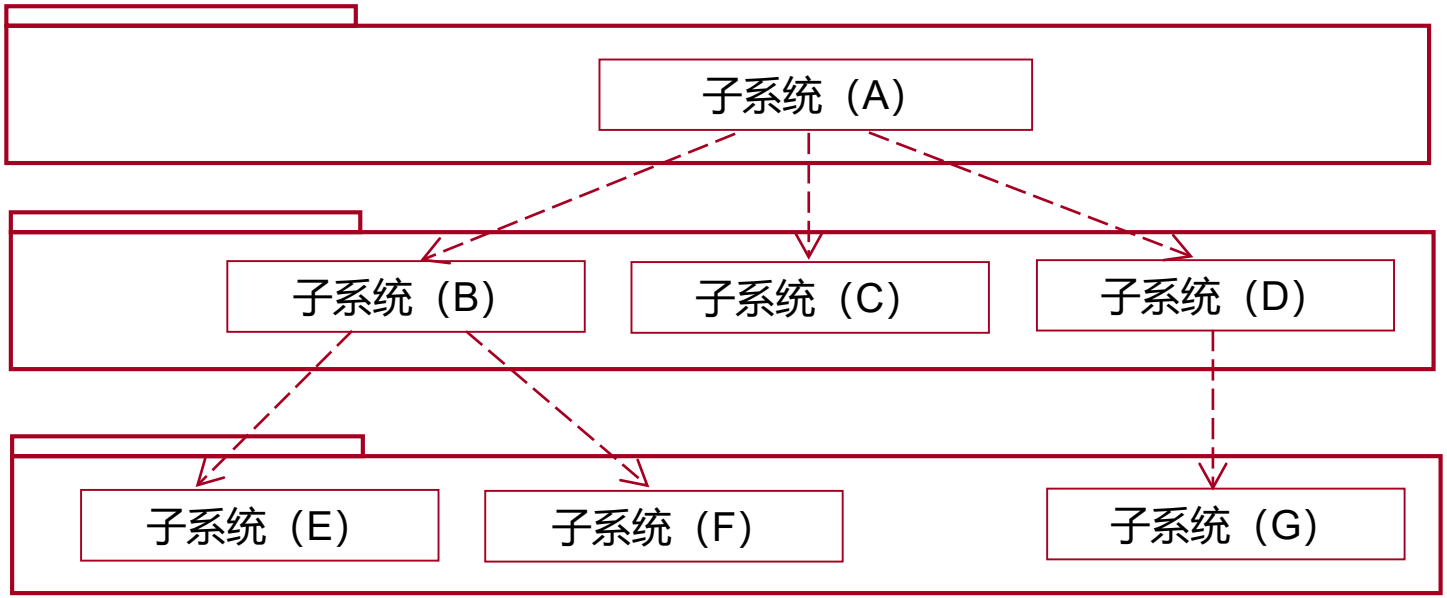
- 穿越模块接口的数据是否会丢失
- 一个模块的功能是否会对另一个模块的功能产生不利的影响
- 各个子功能组合起来能否达到预期的总体功能
- 全局数据结构是否有问题
- 共享资源访问是否有问题
- 单个模块的误差经过集成的累加效应

单元测试、集成测试与系统测试

测试类型	对象	目的	测试依据	测试方法
单元测试	模块内部的程序	消除局部模块逻辑和功能上的错误或缺陷	模块逻辑设计 模块外部说明	白盒测试
集成测试	模块间的集成和调用关系	找出和软件设计相关的程序结构、模块调用关系以及模块间接口方面问题	程序结构	结合使用白盒测试和黑盒测试，较多采用黑盒测试
系统测试	整个系统包括硬件、支持软件、人员等	对整个系统进行一系列整体的、有效性测试	系统结构设计目标说明书 需求说明书	黑盒测试

集成测试策略

问题：下面的三层结构系统如何进行集成测试？



集成测试策略

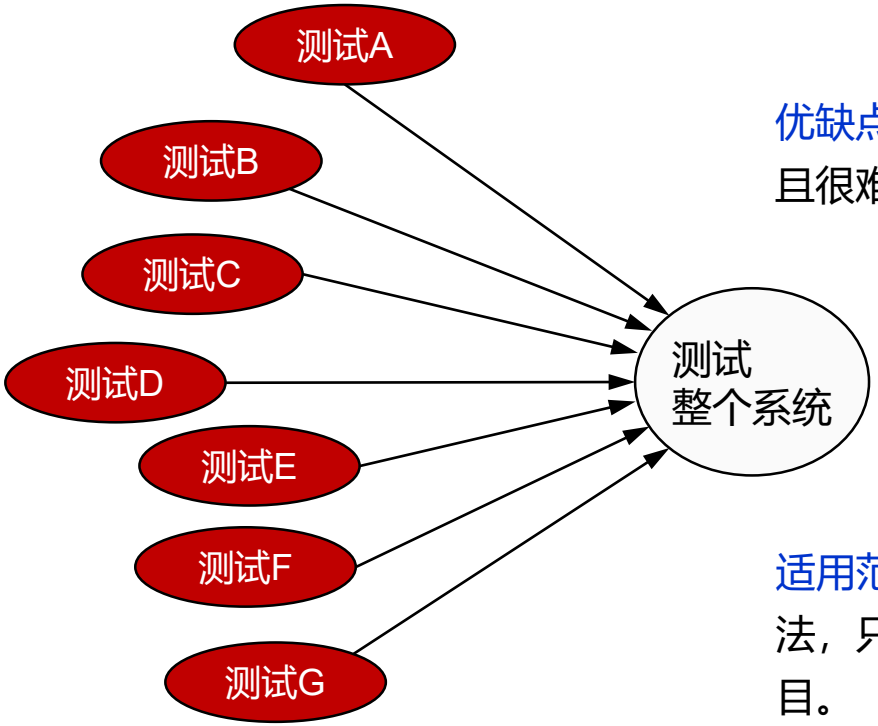
爆炸式集成

•将所有通过单元测试的模块一次性地集成到一起进行测试，也称为一次性组装或整体拼装。

递增式集成

•将要测试的模块逐渐集成到已经测试好的模块里面，边集成边测试，测完一部分再连接一部分。

爆炸式集成



优缺点：操作简单，但发现错误比较晚，且很难确定出错的真正位置和错误原因。

适用范围：一般情况下不建议使用该方法，只适合功能简单、规模小的小型项目。

基于功能分解的集成

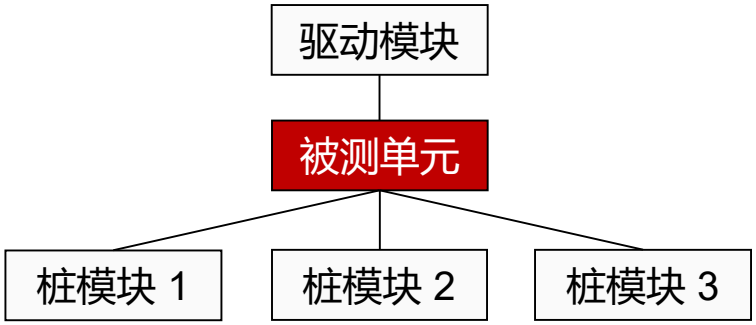
- 自顶向下集成：从顶层开始采用同设计顺序一样的思路对被测系统进行测试，一般集中于顶层的组件，然后逐步测试处于底层的组件，被上层单元调用的下层单元以桩出现。

- 自底向上集成：从最底层组件开始，按照分解树的结构，逐层向上集成，调用下层单元的上层单元以驱动出现。

- 三明治式集成：这是一种混合增殖式测试策略，它将自顶向下和自底向上的集成方法有机地结合起来，从顶层和底层向中间集成，可以减少桩模块和驱动模块的开发。

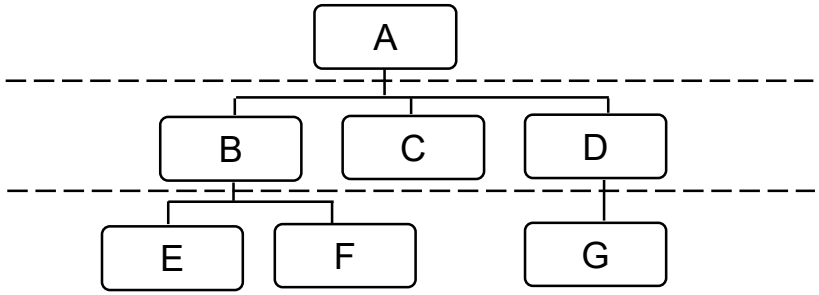
驱动模块与桩模块

驱动模块：用以模拟被测模块的上级模块，在集成测试中接受测试数据，把相关的数据传送给被测模块，启动被测模块并打印出相应的结果。



桩模块：用以模拟被测模块工作过程中所调用的模块，它由被测模块调用，以便于检验被测模块与其下级模块的接口。

自顶向下集成

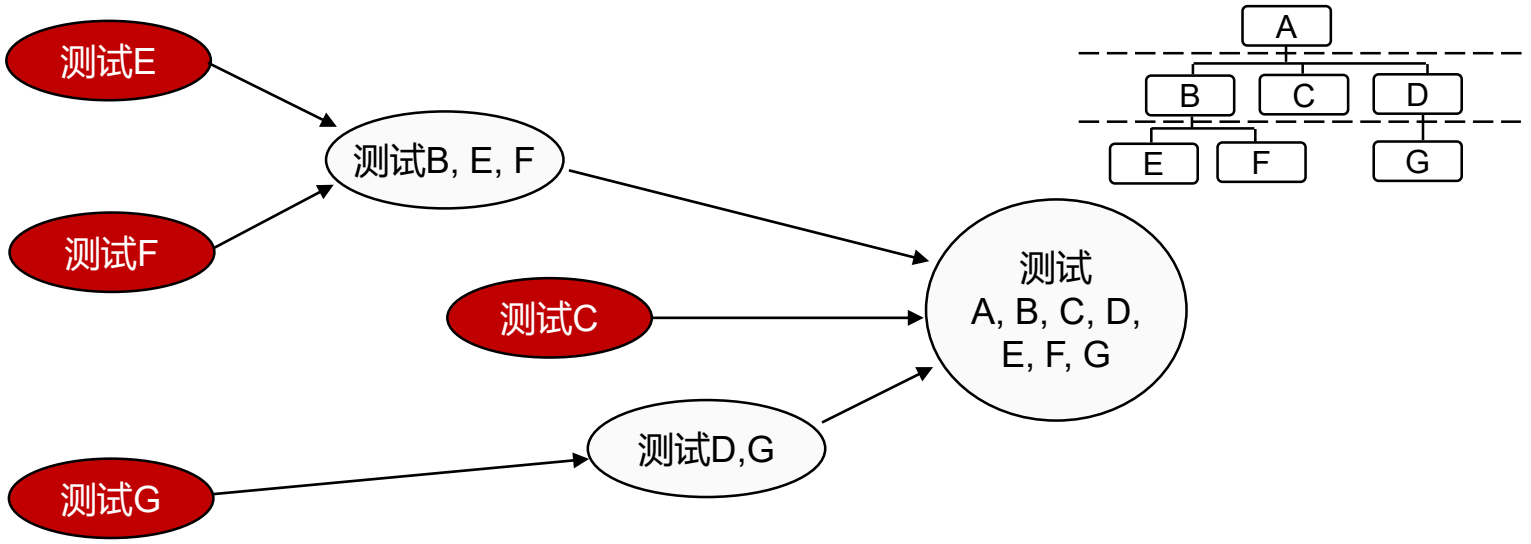


广度优先策略：A — B — C — D — E — F — G

深度优先策略：A — B — E — F — C — D — G

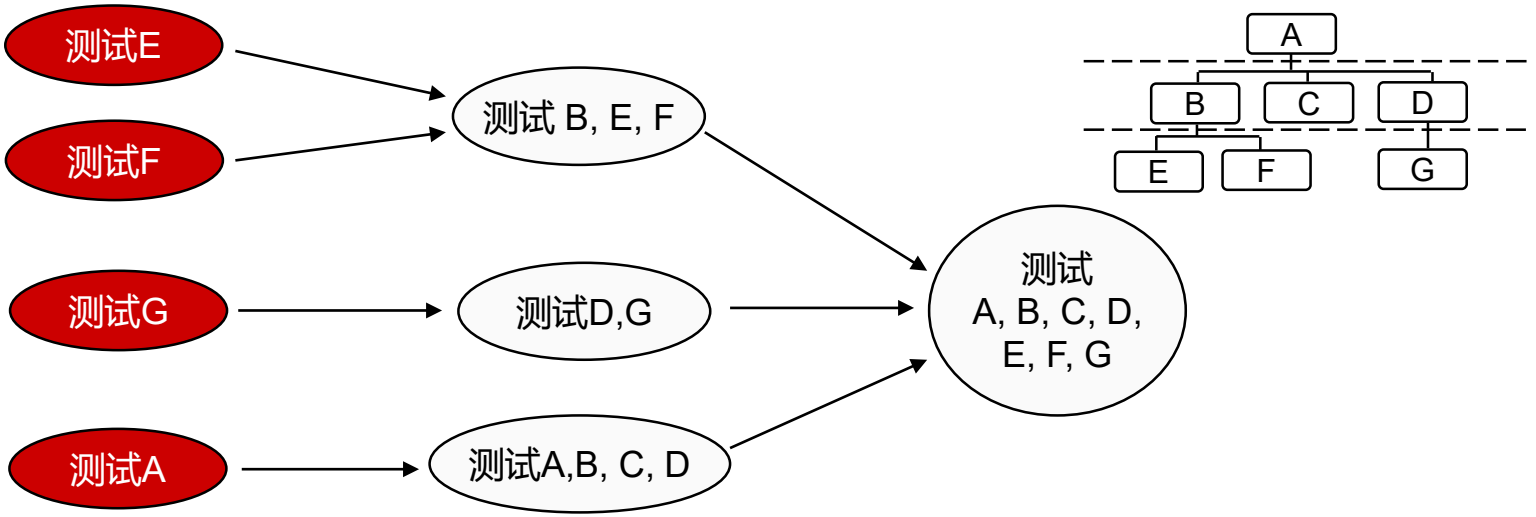
总结：该方法需要桩模块，可以尽早地验证系统的主要功能，较早地发现上层模块的接口错误，但不能及时发现底层关键模块的错误；适合控制结构比较清晰和稳定，底层模块接口未定义或更改频繁的系统。

自底向上集成



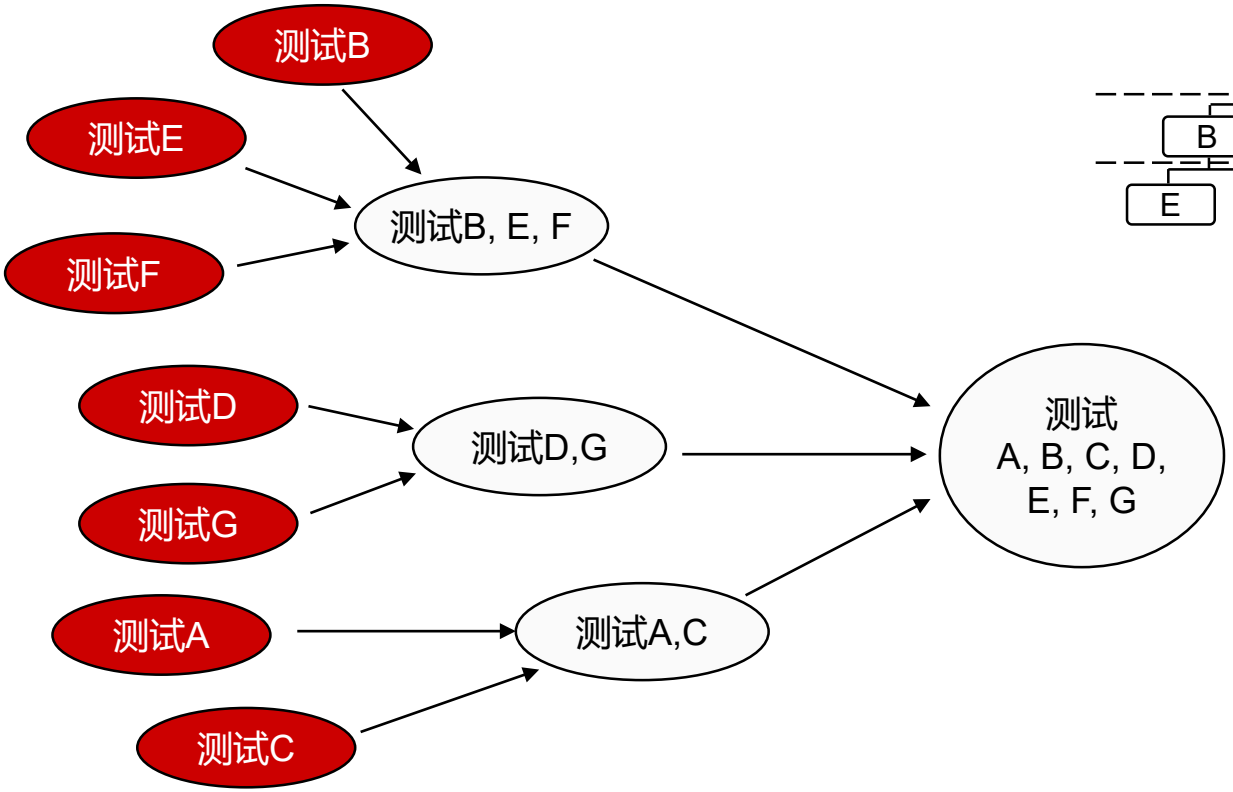
总结：该方法需要驱动模块，可以尽早地验证底层模块的行为，但不能及时发现高层模块设计上的错误；适合底层模块接口比较稳定、高层模块接口变更比较频繁的系统。

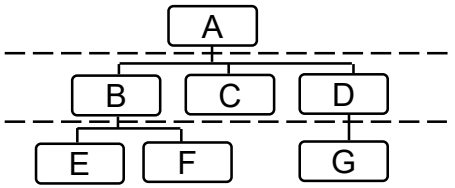
三明治式集成



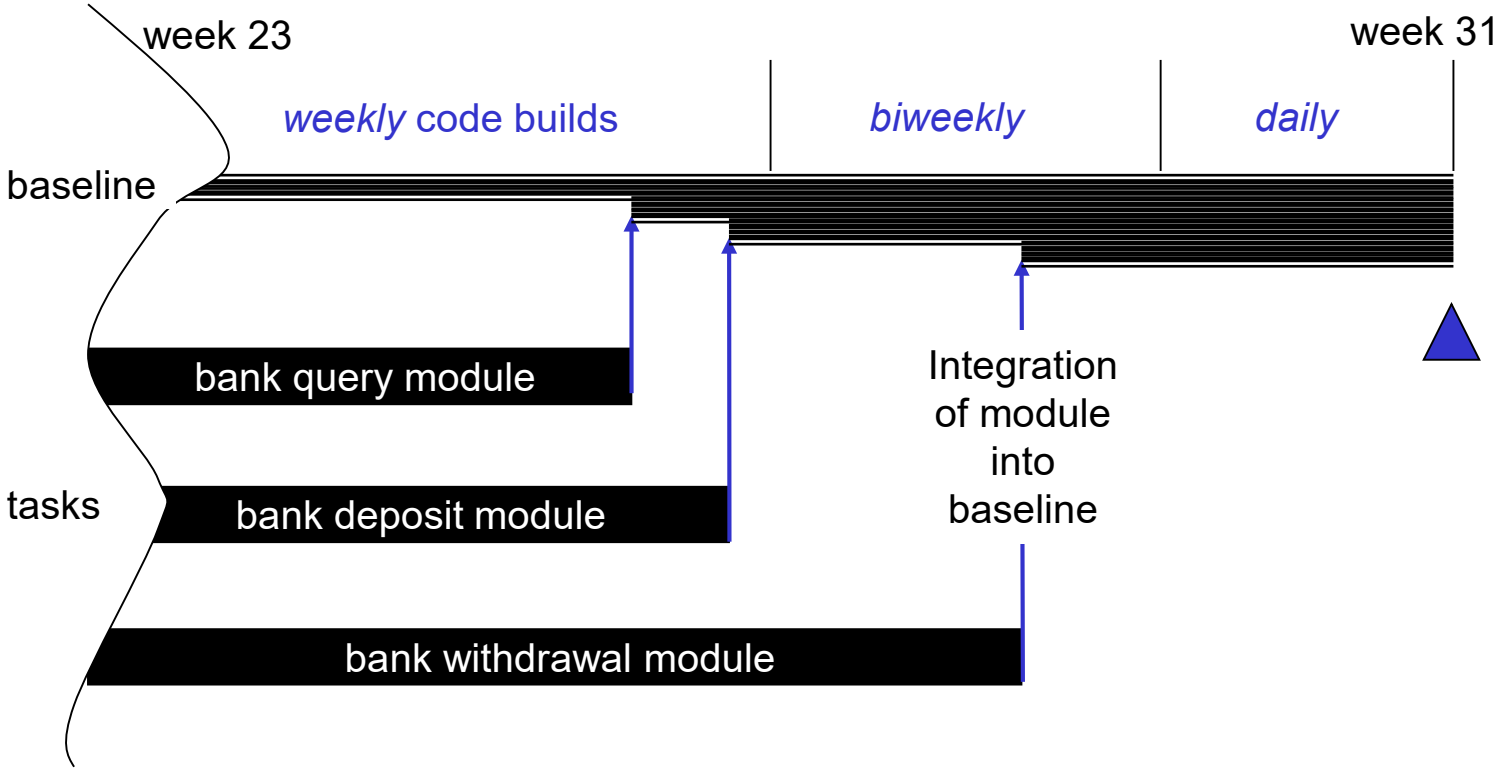
总结：该方法具有自顶向下和自底向上两种集成策略的优点，运用一定技巧可以减少桩模块和驱动模块的开发，在被集成之前中间层不能尽早得到充分的测试；适用于大多数系统。

改进的三明治集成


测试B
测试E
测试F
测试B, E, F
测试D
测试G
测试D,G
测试A
测试C
测试A,C
测试
A, B, C, D,
E, F, G

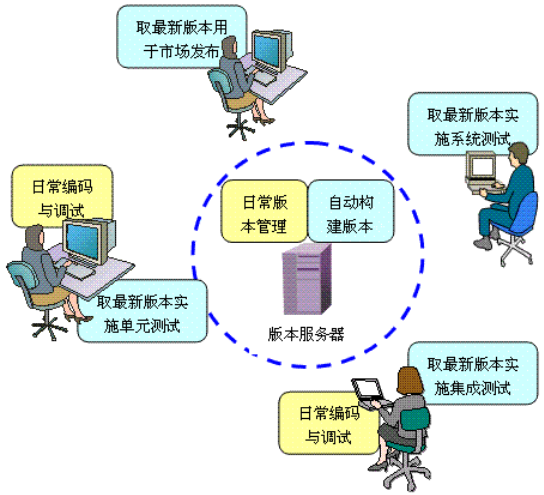

A
B
C
D
E
F
G

持续集成


week 23
week 31
weekly code builds
biweekly
daily
baseline
bank query module
tasks
bank deposit module
Integration of module into baseline
bank withdrawal module

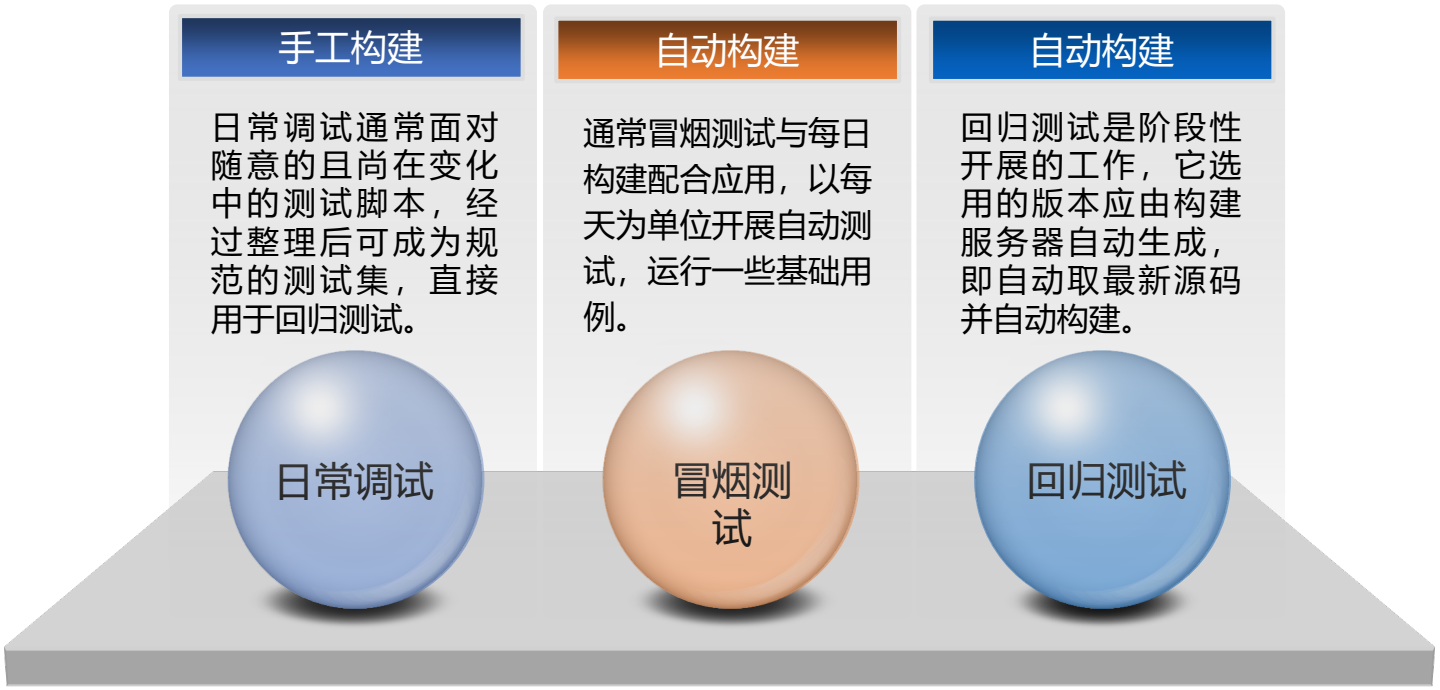
持续集成

持续集成是一种软件开发实践，团队开发成员经常集成自己的工作，通常每个成员每天至少集成一次，即每天可能会发生多次集成。



- 所有开发人员需要在本地机器上进行本地构建，然后再提交到版本控制库中，以免影响持续集成
- 开发人员每天至少向版本控制库中提交一次代码，至少从版本控制库中更新一次代码到本地机器
- 需要有专门的集成服务器来执行集成构建，并通过自动化的构建（包括编译、发布、自动化测试）来验证，从而尽快地发现集成错误

持续集成


手工构建
日常调试通常面对随意的且尚在变化中的测试脚本，经过整理后可成为规范的测试集，直接用于回归测试。
日常调试
自动构建
通常冒烟测试与每日构建配合应用，以每天为单位开展自动测试，运行一些基础用例。
冒烟测试
自动构建
回归测试是阶段性开展的工作，它选用的版本应由构建服务器自动生成，即自动取最新源码并自动构建。
回归测试

关于集成测试说法不正确的是（）。

- ○ 自顶而下增量集成的优点是能够尽早发现系统主控方面的问题
- ○ 集成测试在单元测试完成以后进行
- ○ 集成测试以黑盒法为主
- ○ 自底而上增量集成的优点是能够尽早发现系统主控方面的问题

关于回归测试说法不正确的是（）。

- ○ 在修正发现的软件缺陷后，要认真设计修改，不需要对变化的部分再进行测试
- ○ 回归测试是指有选择地重新测试系统或其组件，以验证对软件的修改没有导致不希望出现的影响
- ○ 回归测试需要测试所有新集成的程序
- ○ 回归测试需要测试所有修改或者修正过的程序

集成测试原则

尽早测试关键模块：关键模块实现了系统的主要功能，尽早测试有利于提高系统的测试效率。

关键模块的特征：

- 满足某些软件需求
- 在程序的模块结构中位于较高的层次（高层控制模块）
- 较复杂、较易发生错误
- 有明确定义的性能要求

尽早测试包含I/O的模块：为以后的测试提供方便。

集成测试原则

为了做好集成测试，需要遵循以下原则：

- 所有公共接口都要被测试到
- 关键模块必须进行充分的测试
- 集成测试应当按一定的层次进行
- 集成测试的策略选择应当综合考虑质量、成本和进度之间的关系
- 集成测试应当尽早开始，并以总体设计为基础
- 在模块与接口的划分上，测试人员应当和开发人员进行充分的沟通
- 当接口发生修改时，涉及的相关接口必须进行再测试
- 测试执行结果应当如实记录

教学提纲

2 功能测试

- 功能测试概念
- 功能测试类型
- Web应用功能测试
- 功能测试工具

功能测试

功能测试（Functional Testing）是在已知产品所应具有的功能基础上，从用户角度来进行功能验证，以确认每个功能是否都能正常使用。

功能测试

界面 | 数据 | 操作 | 逻辑 | 接口

- 程序安装、启动正常，有相应的提示框、错误提示等
- 每项功能符合实际要求
- 系统的界面清晰、美观
- 菜单、按钮操作正常、灵活，能处理一些异常操作
- 能接受正确的数据输入，对异常数据的输入有提示、容错处理等
- 数据的输出结果准确，格式清晰，可以保存和读取
- 功能逻辑清楚，符合使用者习惯
- 系统的各种状态按照业务流程而变化，并保持稳定
- 支持各种应用的环境
- 配合多种硬件周边设备
- 软件升级后，能继续支持旧版本的数据
- 与外部应用系统的接口有效

朋友圈点赞功能测试

1.点赞后是否显示结果；
2.点赞后是否可以取消；
3.点赞取消后是否可以重复点赞；
4.共同好友点赞后，是否有消息提醒；
5.非共同好友点赞后，是否有消息提醒；
6.点击点赞人昵称，是否可以跳转到他/她的主页；
7.自己能否给自己点赞；
8.屏蔽了该用户，共同好友点赞是否提示；
9.点赞人有备注时，是否展示备注昵称；
10.点赞后删除好友，是否继续展示其点赞；

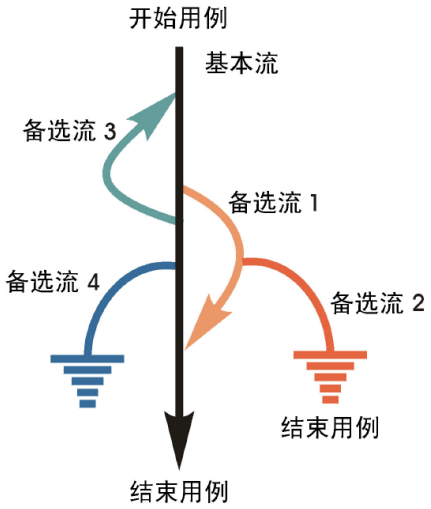
适合性测试

适合性测试是评估和确认软件产品是否能够帮助用户执行期望的任务或者实现了用户需要的功能。

适合性测试应该从客户或用户的角度评判

适合性测试

场景法（也称用例测试方法）是正确性测试经常采用的方法，因为用例清楚地描述了用户任务和软件产品实现的功能。



场景1	基本流
场景2	基本流、备选流1
场景3	基本流、备选流1、备选流2
场景4	基本流、备选流3
场景5	基本流、备选流3、备选流1
场景6	基本流、备选流3、备选流1、备选流2
场景7	基本流、备选流4
场景8	基本流、备选流3、备选流4

准确性测试

准确性测试是验证软件产品提供具有所需精度的正确或者相符的结果或者效果的能力。

- 预期的准确性：在运行期间执行任务的实际结果与预期结果的差别。

- 计算的准确性：由于计算本身的错误和数据精度方面的问题导致的错误计算结果。

准确性测试

举例：网上购物系统的“购物结算”

商品名	单件积分	市场单价	网站单价	优惠	数量	删除
WENGER双肩电脑包	0	￥620.00	￥290.10	减￥91.10	1	删除
空间大师DPC003金属四层车	0	￥256.00	￥100.00	减￥15.00	1	删除
欧姆龙电子血压计HEM-8102	269	￥438.00	￥269.00	无	1	删除
爱国者 aigo（4GB）U盘 8380	0	￥99.00	￥69.00	减￥4.00	2	删除
好孩子木餐椅MY303A-E531绿色	339	￥369.00	￥339.00	无	1	删除
讲故事学折纸（全3册）	399	￥65.40	￥39.90	无	2	删除
	您共节省：￥910.00 可获得商品积分：1406			商品金额总计：￥1101.80		结算->

互操作性测试

互操作性测试是验证软件产品在多种指定的目标环境下（如硬件、软件、中间件、操作系统、浏览器等）是否可以正常工作。

- 输入互操作性：软件产品可以正常使用与之集成的其他软件产品或系统的输出（数据或协议格式等）。
- 输出互操作性：软件产品必须输出标准或协议规定的格式。
- 软件产品是否具有自动检测与该软件交互的系统的通信请求并作出相应的处理。

安全性测试

安全性测试是查找软件自身程序设计中存在的安全隐患，并检查软件产品对非法侵入的防范能力。

- 对应用程序或数据进行未授权的复制
- 未授权的访问控制
- 缓存区溢出
- 服务拒绝
- 在网络上窃听数据传输来获得敏感信息
- 破解保护敏感信息的加密代码
- 逻辑炸弹

安装测试

安装测试（Installing Testing）是确保软件系统在不同的目标环境中都能够进行安装，并在安装后可立即正常运行。

安装测试应该包括哪些方面？

安装测试

软件安装测试

- 安装过程中对于缺省目录及任意指定目录，是否都能正确安装；
- 若是选择安装，查看能否实现其相应的功能；
- 在所有能中途退出的位置退出安装程序后，验证程序并未安装成功；
- 软件安装后，对其它已经安装的软件是否有影响；
- 裸机安装后，各功能点是否可用；
- 在安装前，安装程序是否判断可用磁盘空间大小，如果不满足安装空间要求，安装程序能否继续；
- 安装时查看版权声明、版本信息、公司名称、LOGO等是否符合标准；
- 安装过程中界面显示与提示语言是否准确、友好；
- 重复安装时系统是否有提示、是否可以覆盖安装、是否可以升级安装、是否允许多版本共存；
- 是否有注册码或硬件加密狗，在没有它们（或错误）存在的情况下能否顺利安装。

安装测试

软件卸载测试

- 卸载后注册表中的注册信息及相关的程序安装目录是否能完全删除掉；
- 卸载过程中完全删除共享文件后，看其它程序能否正常运行；
- 卸载后，是否对其它已经安装的软件有影响；
- 系统卸载后用户建立文档是否保留；
- 软件卸载画面上的软件名称及版本信息是否正确；
- 在所有能中途退出卸载的位置是否能正确退出；
- 卸载过程中界面显示与提示语言是否准确、友好；
- 卸载后安装此系统能否打开原来保存的文件，并一切运行正常；
- 卸载程序如果要求重新启动机器，在重启之前是否提示用户保存现有的已运行程序的资料；
- 是否可以选择组件进行卸载；
- 卸载过程中，对意外情况的处理（掉电等）。
- 在卸载过程中，是否有终止或者结束按钮。

安装测试

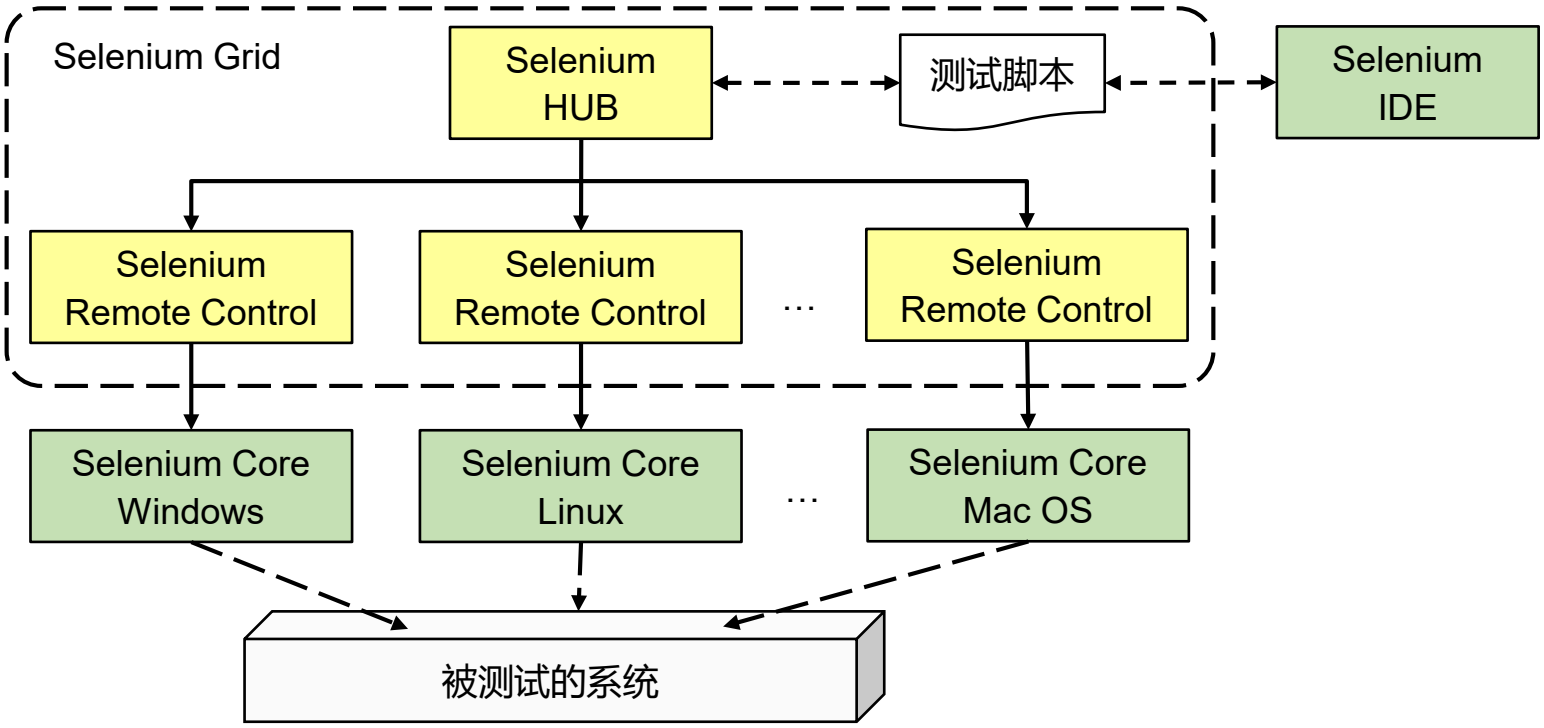
运行与关闭测试

- 运行时是否与其它应用程序有冲突（内存冲突）；
- 是否可以同时运行多个程序；
- 任务栏有无程序运行提示；
- 若有未保存的数据，关闭系统时是否有提示；
- 后台服务程序在点击关闭按钮时是否有确认提示；
- 运行时是否过份占用系统资源、退出时能否完成释放占用的系统资源。

功能测试工具

- HP QuickTest Professional（简称QTP）提供符合所有主要应用软件环境的功能测试和回归测试的自动化。它针对的是GUI应用程序，包括Windows应用程序以及Web应用。
- IBM Rational Functional Tester（简称RFT）是一个面向对象的自动化测试工具，可以支持各种应用程序的测试，如Web、.Net、Java、Siebel、SAP、终端仿真、PowerBuilder、Ajax、Adobe Flex、Dojo Toolkit、GEF、Adobe PDF 文档、zSeries、iSeries 和 pSeries 等。
- Selenium是ThoughtWorks公司开发的一套基于WEB应用的开源测试工具，可以直接运行在浏览器中，模拟用户的操作。它可以用于单元测试、回归测试、冒烟测试、集成测试和验收测试，并且可以运行在各种浏览器和操作系统上。

Selenium组成


Selenium Grid
Selenium HUB
测试脚本
Selenium IDE
Selenium Remote Control
Selenium Remote Control
…
Selenium Remote Control
Selenium Core Windows
Selenium Core Linux
…
Selenium Core Mac OS
被测试的系统

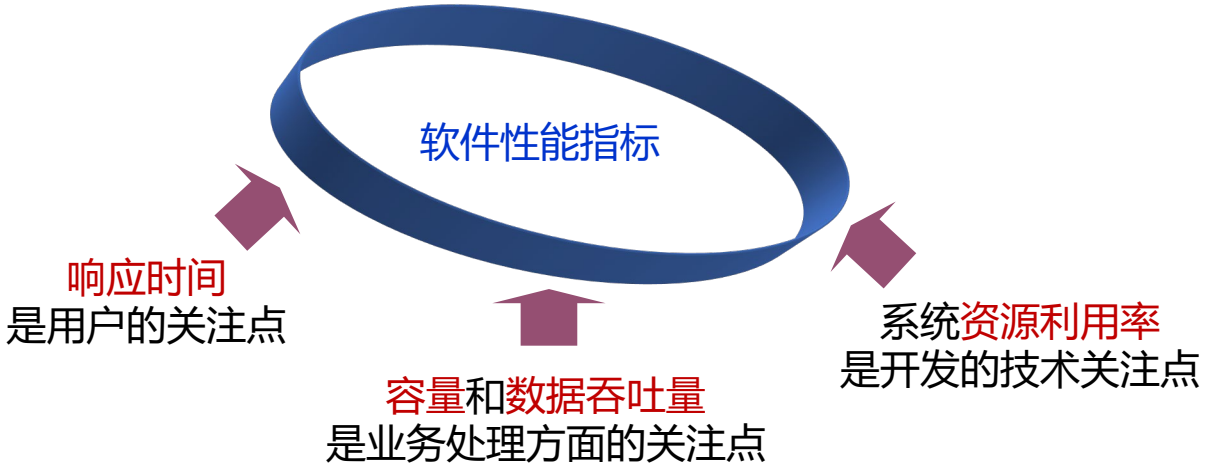
教学提纲

3	性能测试

- 软件性能指标
- 性能测试类型
- 性能测试策略
- 性能测试工具

软件性能

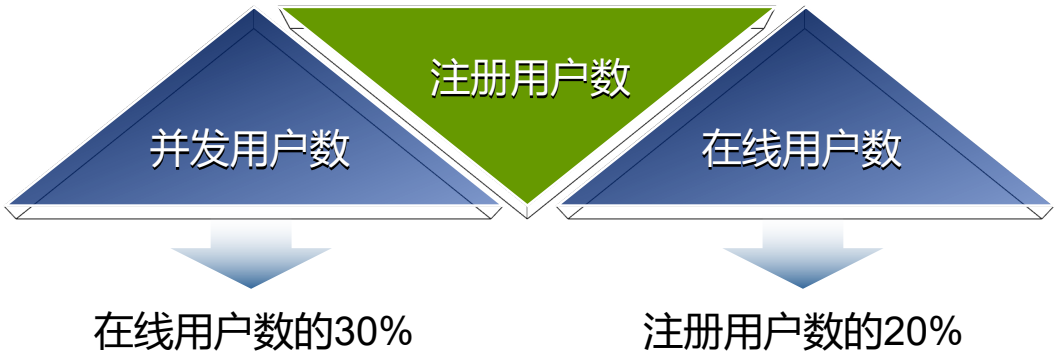
软件性能是软件的一种非功能特性，它关注的不是软件是否能够完成特定功能，而是在完成该功能时展示出来的及时性、资源占用、稳定性、安全性、兼容性、可扩展性、可靠性等。



性能指标

用户数

- 注册用户数：所有在系统注册的用户数目
- 在线用户数：所有正在访问系统的用户（不一定做操作）数目
- 并发用户数：在某一给定时间内某特定时刻进行会话操作的用户数



性能指标

举例：新浪微博系统

•假设用户只有注册后才能使用，但注册用户并不是每时每刻都在使用该网站，因此具体一个时刻只有部分注册用户同时在线。

•在线用户在浏览网站时会花很多时间阅读网站上的信息，因而具体一个时刻只有部分在线用户同时向系统发出请求。

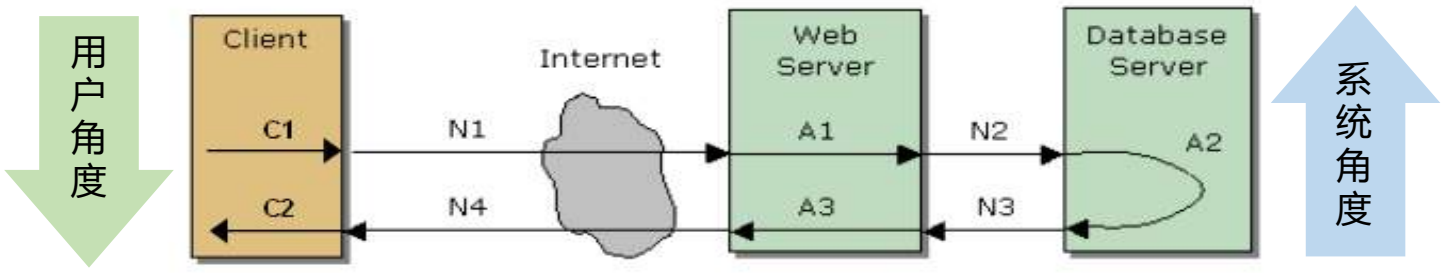
•由于注册用户可能长时间不登陆网站，使用注册用户数作为性能指标会造成很大的误差。在线用户数和并发用户数都可作为性能指标，但性能测试更多关心并发用户数。

性能指标

响应时间

- 从客户端发出请求到获得响应的整个过程所经历的时间

响应时间 = (A1+A2+A3)+(N1+N2+N3+N4)



响应时间 = (C1+C2)+(A1+A2+A3)+(N1+N2+N3+N4)

性能指标

C1：用户请求发出前在客户端需要完成的预处理所需的时间

C2：客户端收到服务器返回的响应后，对数据进行处理并呈现所需的时间

A1：Web Server 对请求进行处理所需的时间

A2：DB Server 对请求进行处理所需的时间

A3：Web Server 对DB Server 返回的结果进行处理所需的时间

N1：请求由客户端发出并达到Web Server 所需的时间

N2：如果需要进行数据库相关操作，由Web Server将请求发送至DB Server所需时间

N3：DB Server 完成处理并将结果返回Web Server 所需的时间

N4：Web Server 完成处理并将结果返回给客户端所需的时间

性能指标

吞吐量

- 吞吐量（Throughout）是指单位时间内系统处理的客户请求的数量，直接体现软件系统的性能承载能力。
- 从业务的角度，吞吐量可以用访问人数 / 天或处理的业务数 / 小时等单位来衡量；从网络的角度，可以用字节数 / 天等单位来考察网络流量。

资源利用率

- 资源利用率是指系统资源的使用程度，例如服务器CPU利用率、内存利用率、磁盘利用率、网络带宽利用率等。
- 一般由“资源实际使用 / 总的资源可用量”表示，如CPU利用率68%、内存利用率55%等。

性能测试

性能测试是通过自动化测试工具或手段模拟多种正常、峰值以及异常负载条件对系统的各项性能指标进行的一种测试。

通过逐渐增加系统负载，测试系统性能的变化，最终确定在满足性能指标的情况下，系统能承受的最大负载量。

目标：在特定的运行条件下验证系统的能力状况。

通过逐步增加系统负载，测试系统性能的变化，最终确定在什么负载条件下系统性能处于失效状态，从而获得系统能够提供的最大服务级别的测试。

目标：发现在什么条件下应用程序的性能会变得不可接受。

性能测试

压力测试类型

- 稳定性压力测试：高负载下持续运行24小时以上。
- 破坏性压力测试：通过不断加载的手段，快速造成系统的崩溃，让问题尽快地暴露出来。
- 渗入测试：通过长时间运行，使问题逐渐渗透出来，从而发现内存泄漏、垃圾收集或系统的其他问题，以检验系统的健壮性。
- 峰谷测试：采用高低突变加载方式进行，先加载到高水平的负载，然后急剧降低负载，稍微平息一段时间，再加载到高水平的负载，重复这样过程，容易发现问题的蛛丝马迹，最终找到问题的根源。

性能测试

大数据量测试

- 独立数据量测试是针对某些系统存储、传输、统计、查询等业务进行的大数据量测试。
- 综合数据量测试是系统在具备一定数据量时，在负载压力测试下考察业务是否能够正常进行的测试。
- 目标：测试数据量较大时系统的性能状况。

疲劳强度测试

- 采用系统稳定运行情况下，长时间运行系统的测试。
- 目标：通过综合分析交易执行指标和资源监控指标来测试系统长时间无故障稳定运行的能力。

性能测试

失效恢复测试

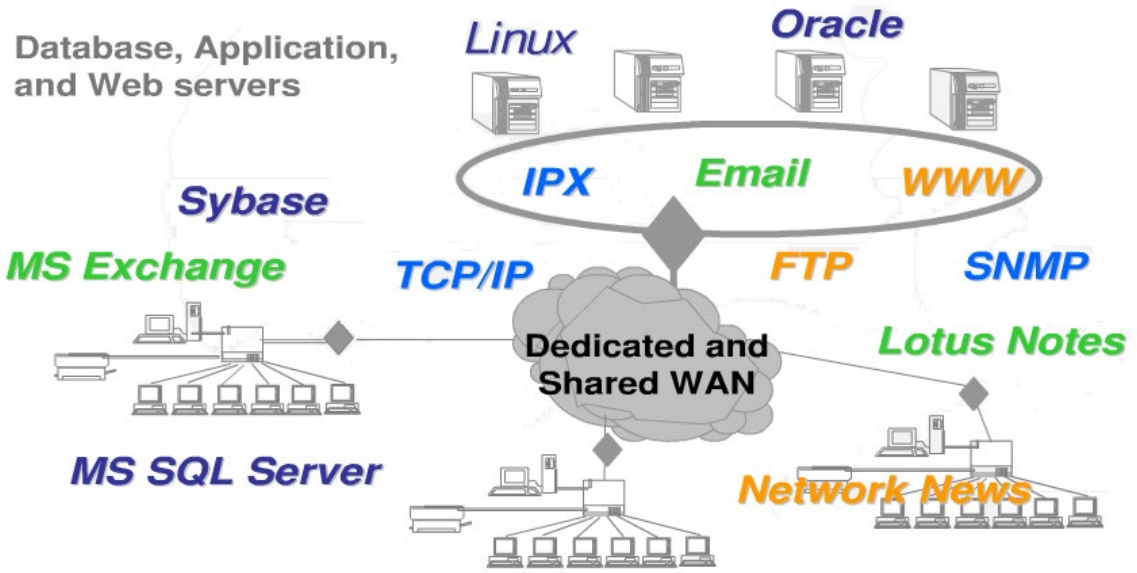
- 对于有冗余备份或负载均衡的系统来说，检验如果系统局部发生故障，系统灾备措施是否可以正常启动，用户是否可以继续使用。
- 目标：通过实施失效恢复测试，评估系统的健壮性和可恢复性。

基准测试

- 通过设计科学的测试方法、测试工具和测试系统，实现对一类测试对象的某项性能指标进行定量的和可对比的测试。
- 例如对数据库管理系统的ACID、查询时间和联机事务处理能力等方面的性能指标进行基准测试。

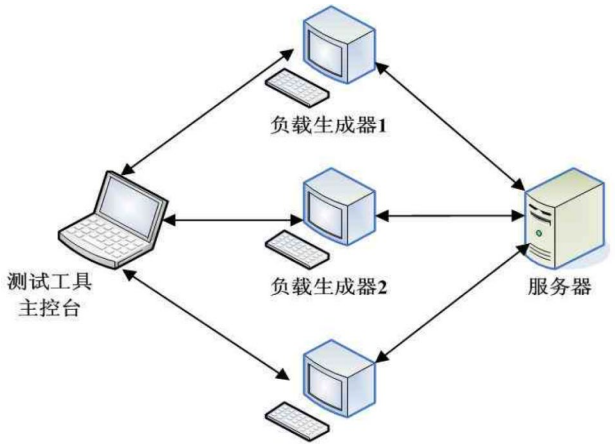
性能测试策略

问题：如何找出问题的根源？



性能测试策略

应用在客户端性能的测试：逐渐增加并发虚拟用户数负载，直到系统的瓶颈或者不能接收的性能点，通过综合分析交易执行指标、资源监控指标等来确定系统的并发性能。



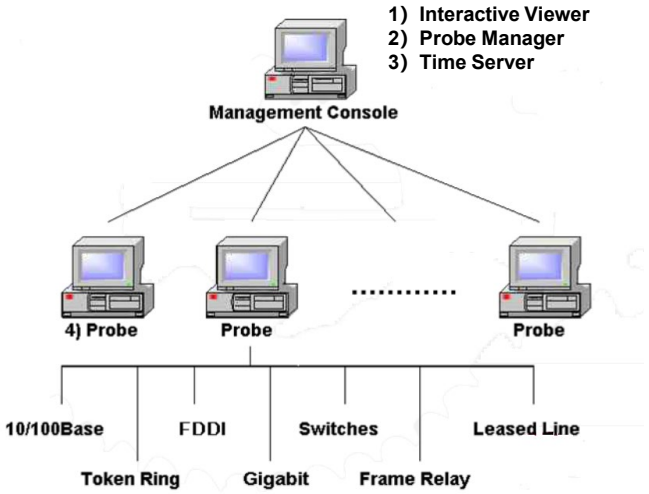
自动化性能测试工具

实现机制：通过在一台或几台PC机上模拟成百上千的虚拟用户同时执行业务的情景，对应用程序进行测试，从而度量系统性能并确定问题所在。

步骤：录制—改进—执行

性能测试策略

应用在网络性能的测试：测试网络带宽、延时、负载和TCP端口的变化如何影响用户的响应时间。



网络测试监控工具

多个捕捉点，一个分析

捕捉点：Agent 被动监听数据包来实现实时数据采集。

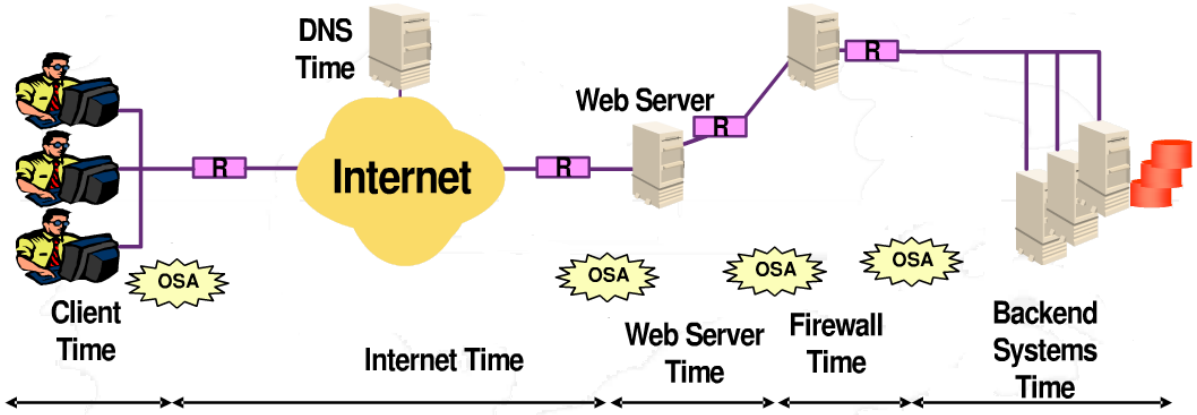
分析：Agent Manager完成对所跟踪到的数据的分析。

性能测试策略

典型的故障诊断策略：设置多点代理服务

•在应用逻辑路径上多点采集

•在任何两个节点之间进行数据整合，测量分段响应时间，分析应用故障



性能测试策略

应用在服务器性能的测试：测试关键点是资源占用情况、数据库性能和故障报警。对于应用在服务器上性能的测试，可以采用工具监控，也可以使用系统本身的监控命令。

监控服务器
操作系统

- CPU
- 内存 & SWAP
- 磁盘管理
- 网络
- 文件系统
- 活动的进程

监控中间件

监控数据库

- 数据库中的关键资源
- 读写页面的使用情况
- 超出共享内存缓冲区的操作数
- 上一轮询期间作业等待缓冲区的时间
- 共享内存中物理日志和逻辑日志缓冲区的使用率
- 磁盘的数据块使用情况以及被频繁读写的热点区域
- 用户事务或者表空间监控事务日志
- 数据库锁资源
- 关键业务的数据表的表空间增长
- SQL执行情况

性能测试工具

- HP LoadRunner是一种预测系统行为和性能的负载测试工具，它通过以模拟上千万用户实施并发负载及实时性能监测的方式来确认和查找问题。LoadRunner能够对整个企业的架构进行测试，通过使用该工具，企业能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。

- JMeter是Apache组织开发的基于Java的压力测试工具，不仅用于Web服务器的性能测试，也涵盖数据库、FTP、LDAP服务器等各种性能测试，可与Junit、Ant等工具集成。它可以针对服务器、网络或其他被测试对象等大量并发负载进行强度测试，分析在不同压力负载下系统的整体性能。

教学提纲

4 软件缺陷管理

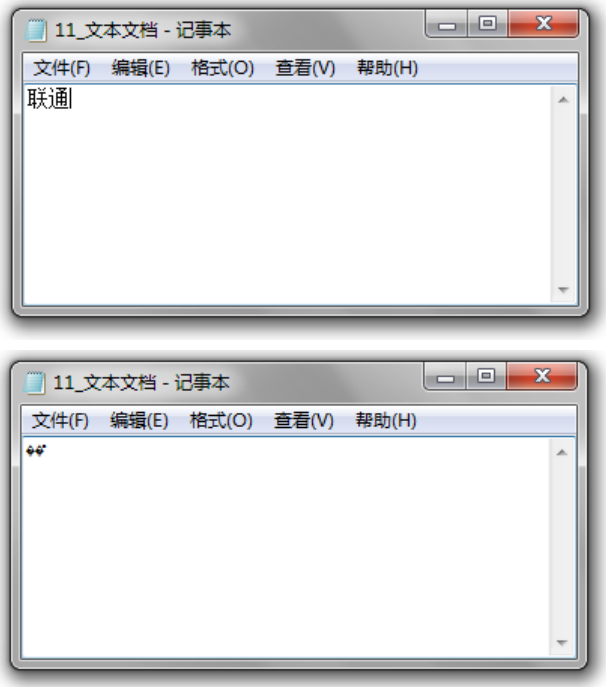
- 缺陷严重性与优先级
- 软件缺陷生命周期
- 软件缺陷属性
- 缺陷管理系统

软件缺陷举例

举例：记事本应用程序

- 在Win7系统中使用记事本创建一个新文件，输入“联通”两个字，之后保存并退出。

- 退出后，再重新打开这个文本文件，正确的结果应该是文本内容为“联通”，但实际上刚才输入的内容变成乱码。



严重性与优先级

缺陷的严重性是指缺陷对软件产品使用的影响程度，缺陷的优先级是指缺陷应该被修复的紧急程度。

- 一般情况：缺陷越严重，越要优先得到修正，缺陷严重等级和缺陷优先级相关性很强。
- 特殊情况：如有些缺陷比较严重，但由于技术的限制或第三方产品的限制，暂时没法修正，其优先级就会低。

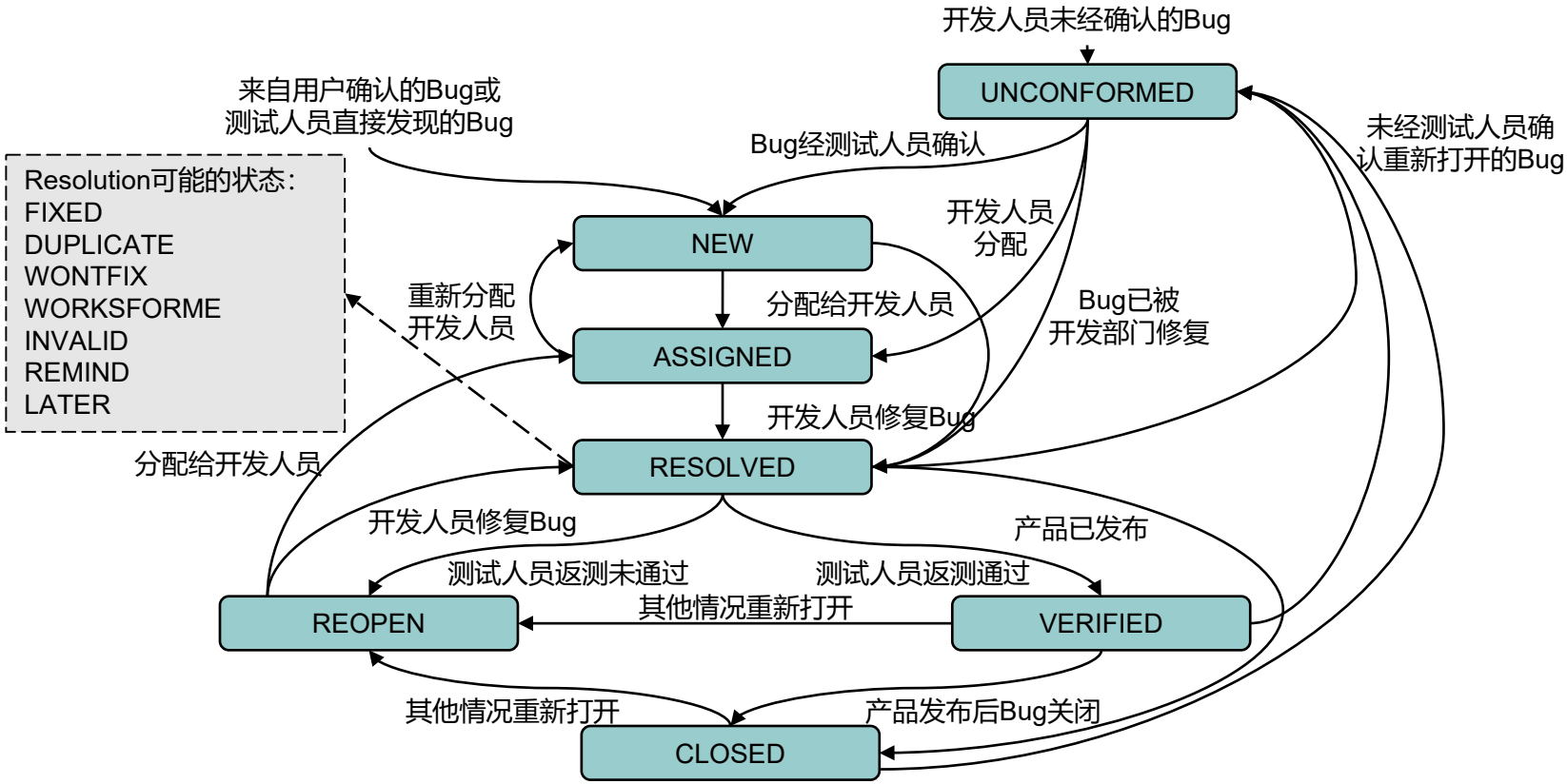
严重性与优先级

缺陷严重性	描述
致命的（1级）	造成系统或应用程序崩溃、死机、挂起，或造成数据丢失、主要功能完全丧失。
严重的（2级）	系统功能或特性没有实现、主要功能部分丧失、次要功能完全丧失或者致命的错误声明。
一般的（3级）	这类缺陷虽不影响系统的基本使用，但没有很好地实现功能，没有达到预期效果，如次要功能丧失、提示信息不太明确、用户界面差、操作时间长等。
微小的（4级）	对功能几乎没有影响，产品及其属性仍可使用，如存在个别错别字、文字排列不整齐等。

严重性与优先级

缺陷优先级	描述
立即解决（P1）	缺陷导致系统几乎不能使用或测试不能继续，需要立即修复
高优先级（P2）	缺陷严重，影响测试，需要优先考虑
正常排队（P3）	缺陷需要正常排队等待修复
低优先级（P4）	缺陷可以在开发人员有时间的时候被纠正

软件缺陷生命周期

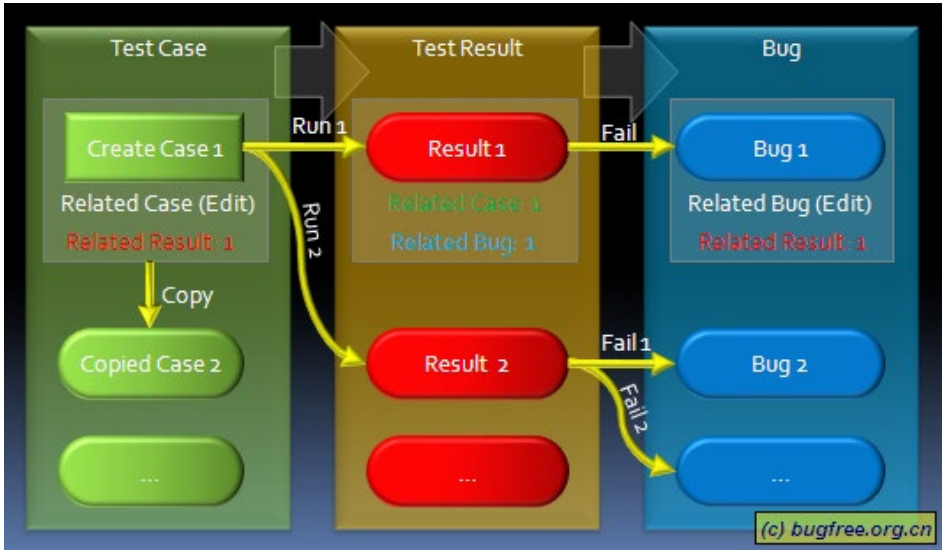

开发人员未经确认的Bug
UNCONFORMED
来自用户确认的Bug或
测试人员直接发现的Bug
Bug经测试人员确认
未经测试人员确
认重新打开的Bug
Resolution可能的状态:
FIXED
DUPLICATE
WONTFIX
WORKSFORME
INVALID
REMIND
LATER
NEW
开发人员
分配
重新分配
开发人员
分配给开发人员
Bug已被
开发部门修复
ASSIGNED
开发人员修复Bug
分配给开发人员
RESOLVED
开发人员修复Bug
产品已发布
测试人员返测未通过
测试人员返测通过
其他情况重新打开
REOPEN
VERIFIED
其他情况重新打开
产品发布后Bug关闭
CLOSED

软件缺陷属性

- 缺陷编号：编号是唯一的，可由缺陷跟踪系统自动生成。
- 缺陷标题：简要说明缺陷的类型和内容。
- 缺陷所属：指明缺陷所在模块或所属文档的名称。
- 缺陷严重程度：缺陷对软件产品使用的影响程度。
- 缺陷优先级：缺陷应该被修复的紧急程度。
- 缺陷处理角色：提交人、解决人、确认人
- 缺陷处理状态：
  ① 确认结果：确认软件缺陷的修正工作是否有效
  ② 解决结果：预计缺陷修改后能达到的结果
  ③ 处理结果：缺陷最后的实际处理结果
- 时间信息：提交日期、预计解决日期、确认日期、最终处理日期

缺陷管理系统

BugFree是基于PHP和MySQL开发的免费开源的缺陷管理系统，其服务器端在Linux和Windows平台上都可以运行，客户端无需安装任何软件，通过浏览器可以自由使用。



Bug状态

Bug的初始状态

任何新建的Bug状态都是Active，可以通过编辑修改Bug的内容，并指派给合适的人员解决。

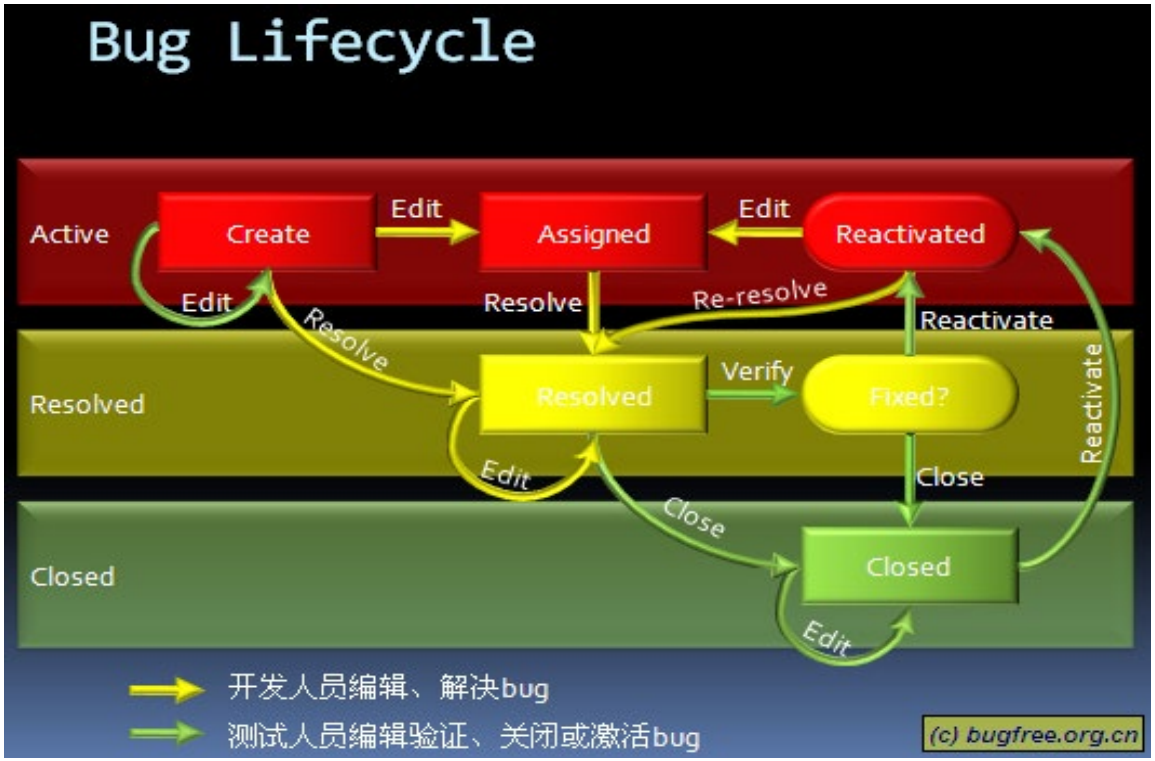
解决Bug之后的状态

①By Design　设计如此
②Duplicate 已发现
③External　外部因素
④Fixed　已解决
⑤Not Repro　无法重现
⑥Postponed　推迟解决
⑦Won't Fix 不值得解决

Bug的结束状态

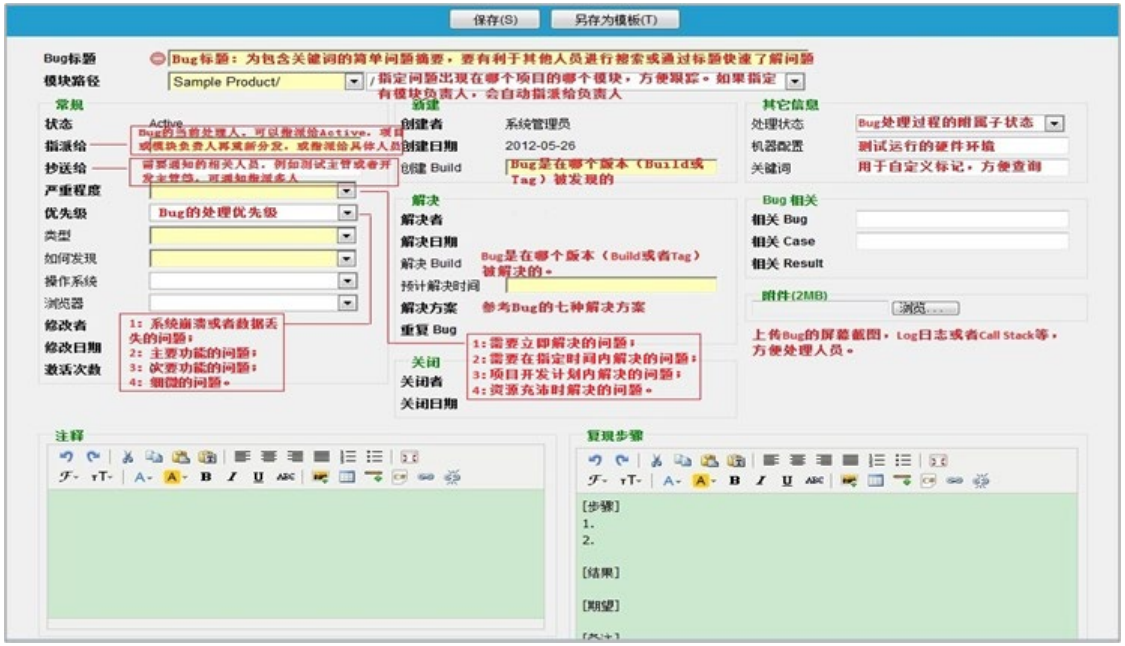
测试人员和项目经理检查标记“已解决”的Bug是否解决，并决定关闭该Bug或是再激活它。

Bug处理流程


Bug Lifecycle
Active
Create
Edit
Assigned
Edit
Reactivated
Edit
Resolve
Re-resolve
Reactivate
Resolved
Resolve
Resolved
Verify
Fixed?
Reactivate
Edit
Close
Closed
Close
Closed
Edit
开发人员编辑、解决bug
测试人员编辑验证、关闭或激活bug
(c) bugfree.org.cn

创建Bug

创建方式：可以通过运行用例来创建，也可点击首页【新建Bug】直接创建。



关闭Bug

开发人员提交了“解决”信息后，测试人员对处于“解决”状态的Bug进行验证。如问题没有得到解决或在修改过程中引入了新问题，该Bug会被再度激活（Active）；如经验证确实已解决，则将该Bug关闭（Closed）。

Reference

- 清华大学国家级精品课程 《软件工程》主讲人 刘强 副教授 刘璘 副教授
  - https://www.icourses.cn/sCourse/course_3016.html
  - https://www.xuetangx.com/course/THU08091000367/5883555?channel=learn_title

谢谢大家！

THANKS