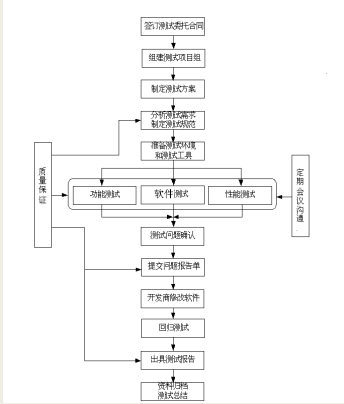
测试过程

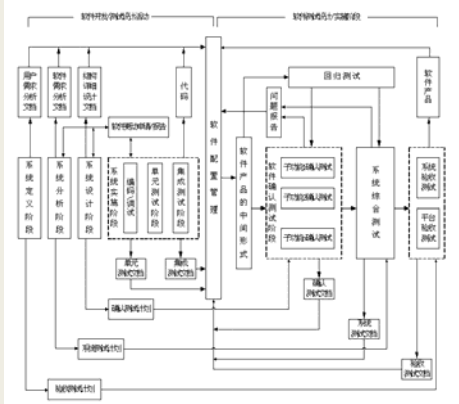
张程
Email：bootan@cqu.edu.cn
QQ:80463125

软件测试过程

测试工作流程



测试工作流程（续）



软件测试工作

软件测试过程

开始 → 测试计划 → 测试设计 → 测试执行 → 测试总结 → 结束

- 工作的输入是：软件测试任务书（或合同）和被测软件的需求规格说明。他们是开展软件测试计划的基础和依据
- 测试的计划与控制是整个测试过程中最重要的阶段，它为实现可管理且高质量的测试过程提供基础。这个阶段需要完成的工作内容是：拟定测试计划，论证那些在开发过程难于管理和控制的因素，明确软件产品的最重要部分
- 工作的输出：软件测试计划。
  - *软件测试任务书（或合同）和被测软件的需求规格说明，他们是开展软件测试计划的基础和依据。*

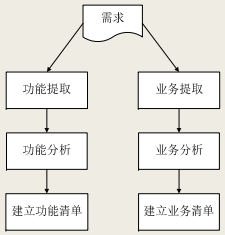
软件测试过程

开始 → 测试计划 → 测试设计 → 测试执行 → 测试总结 → 结束

- *需求分析*
  - *定义被测试对象和测试目标*
  - *确定测试阶段和测试周期的划分*
- *测试策略*
  - *测试方法的选择*
  - *测试工具的选择*
  - *测试用例设计方法的选择*
- *工作量估算*
  - *测试过程的任务定义*
  - *各任务的工作量估算进度安排*
- *进度安排*
  - *测试人员的时间任务安排*
  - *测试设备的配置*
  - *测试工具的配置*

测试需求分析

- 测试需求分析需要做两方面的事情，一是详细了解并深挖需求，二是进行测试范围分析，确定测试范围。
- 一般可依据软件产品需求说明书或产品原型确定功能测试范围。软件产品需求说明书清楚地描述了产品的功能特性。



测试范围分析

编码	需求名称	重要性	需求类型
AGJY-B	BS 业务建模	1：核心	1：业务
AGJY-B-T	目标	1：核心	1：业务
AGJY-B-T-01	系统登录管理业务	2：关键	1：业务
AGJY-B-T-02	身份核实业务	2：关键	1：业务
AGJY-B-T-03	信用卡交易管理业务	2：关键	1：业务
AGJY-B-T-0301	卡片处理业务	3：重要	1：业务
AGJY-B-T-0302	账户交易业务	3：重要	1：业务
AGJY-B-T-0302-01	卡片密码处理业务	3：重要	1：业务
AGJY-B-T-0302-02	还款业务	3：重要	1：业务
AGJY-B-T-0302-03	额度调整业务	3：重要	1：业务
AGJY-B-T-0302-04	积分业务	3：重要	1：业务

业务清单

软件测试过程

开始 → 测试计划 → 测试设计 → 测试执行 → 测试总结 → 结束

- 度量标准
  - *测试通过或失败的标准*
  - *测试挂起及恢复的标准*
  - *测试中需要进行度量的目标度量项*
- 风险评估
  - *定义项目中潜在的风险*
  - *制定相应的风险减缓措施和应急措施*
- 子计划制定
  - *度量分析计划*
  - *配置管理计划*
  - *质量保证计划*
  - *验证和确认计划*
  - *沟通计划*
- 计划评审

软件测试过程

开始 → 测试计划 → 测试设计 → 测试执行 → 测试总结 → 结束

- 软件测试计划的内容要素包括
  - *软件测试的范围*
  - *软件测试的策略*
  - *软件测试的需求*
  - *软件测试的资源要求*
  - *软件测试的人员要求*
  - *软件测试的进度*
  - *测试阶段停止测试的标准*
  - *测试用例设计的方法*
  - *测试中潜在的风险和问题区域*
  - *角色与职责*

测试计划文本

- 测试目标：包括总体测试目标以及各阶段的测试对象、目标及其限制。
- 测试需求和范围：确定哪些功能特性需要测试、哪些功能特性不需要测试，包括功能特性分解、具体测试任务的确定，如功能测试、用户界面测试、性能测试等。
- 测试风险：潜在的测试风险分析、识别，以及风险归避、监控和管理。
- 项目估算：根据历史数据，采用恰当的评估方法及时对工作量、测试周期以及所需资源做出合理的估算。
- 测试策略：根据测试需求和范围、测试风险、测试工作量和测试资源限制等确定测试策略，测试策略是测试计划的关键内容。
- 测试阶段划分：划分合理的测试阶段，并定义每个阶段的进入要求及完成的标准。
- 项目资源：各个测试阶段的资源分配，包括软、硬件资源分配和人力资源的组织和建设等，例如测试人员的角色、责任和测试任务等均属于人力资源管理的内容。
- 日程：确定各个测试阶段的结束日期以及最后测试报告的递交日期。
- 跟踪和控制机制：问题跟踪报告、变更控制、缺陷预防和质量管理等，如可能会导致测试计划变更的事件，包括测试工具的改进、测试环境的影响和新功能的变更等。

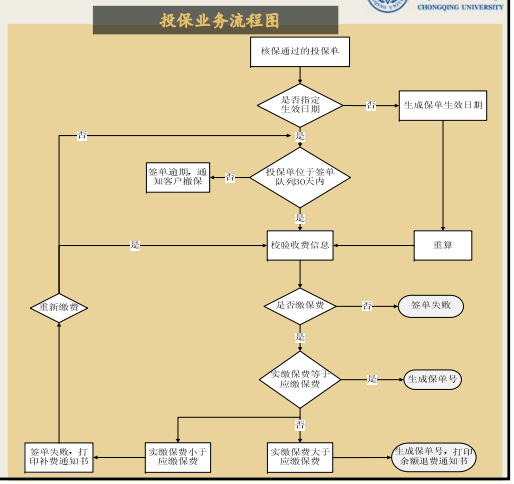
软件测试过程

开始 → 测试计划 → 测试设计 → 测试执行 → 测试总结 → 结束

- 工作的输入是：软件测试计划
- 软件测试设计阶段主要包含2个方面的工作：
  - *测试用例的设计*
  - *测试用例的开发和实现*
- 本阶段要完成的主要任务如下：
  - *在软件测试计划阶段中，通过测试需求分析得到细化后的每一个被测软件功能和特性，设计相应得软件测试用例。*
  - *针对每一个软件测试用例，确定其测试输入、测试步骤以及每一步骤的预期输出。*
  - *如果需要，开发和实现相应的测试输入。（自动化）*
  - *建立软件测试需求集和软件测试用例集之间的关联关系。（多对多）*
- 工作的输出是：测试用例和测试数据

测试建模

- 测试建模是将测试思路或测试内容形成条理清晰、系统全面的模型的过程。
- 一般地，可以针对系统中复杂的业务逻辑和功能进行建模，通过业务建模的方式来梳理这些复杂的业务逻辑和功能



用例设计

用例编号	业务场景	预置条件	测试步骤	预期结果
BX-F-0107-01	验证投保单缴费签单	星级业务员的人工核保通过的验缴保单	1. 检查保单的状态	1. 保单状态为有效
			2. 检查账户余额	2. 账户余额=已缴保费（9000）-应缴保费(8000)
			3. 检查保单号是否存在	3. 保单号生成
			4. 检查提示信息	4. 提示“签单成功”
			5. 进入综合查询，查看保单信息	5. 保单为有效保单，生成保单号，并打印退费通知书
BX-F-0107-02	实交保费等于应交保费	VIP客户的自动核保通过的投保单	1. 检查保单的状态	1. 保单状态为有效
			2. 检查账户余额	2. 已缴保费=应缴保费
			3. 检查保单号是否存在	3. 保单号生成
			4. 检查提示信息	4. 提示“签单成功”
			5. 进入综合查询，查看保单信息	5. 保单为有效保单
BX-F-0107-03	实交保费少于应收保费，不补交保费承保签单		1. 检查保单的状态	1保单状态为有效，投保单继续留在保单签发队列中
			2. 计算保单金额	2. 余额不足自动产生补费通知书
			3. 检查账户余额	3. 不补交，签单逾期，业务员通知客户撤保
			4. 检查保单	
BX-F-0107-04	实交保费少于应收保费，补交保费充足，承保签单		1. 检查保单的状态	1保单状态为有效，投保单继续留在保单签发队列中
			3. 检查保单号是否存在	2. 余额不足自动产生补费通知书
			4. 检查提示信息	3. 补交保费后，签单成功
			5. 进入综合查询，查看保单信息	

测试用例实例

软件测试过程

开始 → 测试计划 → 测试设计 → 测试执行 → 测试总结 → 结束

- 工作的输入是：测试用例和测试数据。
- 软件测试执行阶段，是在准备好的测试环境上依次执行各测试用例并详细记录每一步的测试结果。
- 本阶段主要完成的任务如下：
  - *获得被测程序*
  - *获得指定的测试资源*
  - *执行测试用例*
  - *记录测试过程和测试输出数据。*
- 工作的输出是：软件测试记录。

测试执行

- 代码提交测试之后，测试工程师就可以在测试环境中开始测试，这时往往先会执行冒烟测试。
- 冒烟测试：在一个编译版本发布后，先运行其最基本的功能，例如启动、登录、退出等。如果这些简单的功能运行都错误的话，测试人员没有必要进行下一步的深入测试，直接把编译版本退回给开发人员进行修改。
  - *冒烟测试通过后，测试人员就可以针对自己所负责的模块，根据测试用例进行详细测试。如果发现缺陷，则将缺陷提交至缺陷管理系统，当缺陷被开发人员修改后，测试人员再进行回归测试，以确认旧代码在修改后没有引入新的错误或导致其他代码产生错误。*
- 回归测试：测试执行过程中，回归测试往往要重复进行多次，在不断修复缺陷的过程中，测试工程师经常需要对主要流程及功能进行再次测试。为了提高测试人员的工作效率，可以将回归测试进行自动化处理，这也是自动化测试应用很重要的一个方面。

软件测试过程

开始 → 测试计划 → 测试设计 → 测试执行 → 测试总结 → 结束

- 工作的输入是：软件测试计划、测试用例、软件测试记录。
- 软件测试总结阶段的主要工作是根据软件测试的执行情况，作出两方面的评价：
  - *一是评价软件测试的效果;*
  - *二是评价被测试的软件*
- 本阶段要完成的主要任务如下：
  - *描述测试状态*
  - *描述软件状态*
  - *完成测试报告*
  - *保存测试文件*
- 工作的输出是：测试报告。

测试总结

- 缺陷分析
  - *1）缺陷密度：缺陷密度是指缺陷在软件规模（组件、模块等）上的分布，如每千行代码或每个功能点的缺陷数。一般来说，发现更多缺陷的模块，隐藏的缺陷也更多，在修正缺陷时也会引入较多的错误，结果产品的质量更差。所以说，缺陷密度越低意味着产品质量越高。*
    - 如果相对上一个版本，当前版本的缺陷密度没有明显变化或更低，就应该分析当前版本的测试效率是不是降低了？如果不是，意味着产品质量得到了改善；如果是，那么就需要额外的测试，还需要对开发和测试的过程进行改善。
    - 如果当前版本的缺陷密度比上一个版本高，那么就应该考虑在此之前是否为提交测试效率进行了有效的策划并在本次测试中得到实施。如果是，虽然需要开发人员更多的努力去修正缺陷，但质量还是得到更好的保证；如果没有，意味着质量恶化、质量很难得到保证。这时要保证质量，就必须延长开发周期或投入更多的资源。

测试总结

- 缺陷分析
  - *2）缺陷清除率：首先引入几个变量，F为描述软件规模用的功能点；D1为在软件开发过程中发现的所有缺陷数；D2为软件发布后发现的缺陷数；D为发现的总缺陷数。因此，D=D1+D2。*
    - 对于一个应用软件项目，则有如下计算方程式（从不同的角度估算软件的质量）：
      - *质量 = D2/F*
      - *缺陷注入率 = D/F*
      - *整体缺陷清除率 = D1/D*
    - 假如有100个功能点，及F=100, 而在开发过程中发现了20个缺陷，发布后又发现了3个缺陷，则D1=20, D2 =3, D=D1+D2=23
      - *质量（每个功能点的缺陷数）=D2/F=3/100=0.03（3%）*
      - *缺陷注入率=D/F=23/100=0.23（23%）*
      - *整体缺陷清除率=D1/D=20/23=0.869（86.96%）*
    - 整体缺陷清除率越高，软件产品的质量越高。缺陷清除率越低，质量也越低。

测试总结

- 缺陷分析
  - *2）缺陷清除率*
    - 整体缺陷清除率越高，软件产品的质量越高。缺陷清除率越低，质量也越低。

质量级别	潜在缺陷	清除效率(%)	被交付的缺陷
1	2000	85	300
2	1000	89	110
3	400	91	36
4	200	93	14
5	100	95	5

    - 阶段性缺陷清除率是测试缺陷密度度量的扩展，跟踪开发周期所有阶段中的缺陷，包括需求评审、设计评审、代码审查和测试等。因为大部分的编程缺陷是和设计问题有关的，进行正式评审或功能验证以增强前期过程的缺陷清除率，有助于减少缺陷的注入。

缺陷发现阶段	需求定义问题	设计问题	代码问题	阶段清除率
需求阶段—需求评审	10			需求阶段 10/32=31.25
设计阶段—设计评审	8	8		设计阶段 8/24=33.3%
编程阶段—代码评审	4	10	20	编程阶段 20/40 =50%
系统测试	8	5	15	
验收测试	2	1	5	
合计	32	24	40	

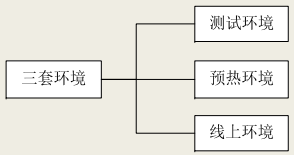
测试总结

- 编写测试报告
  - *测试报告基于测试中采集的数据以及对最终测试结果的分析，形成一份包含测试过程和测试结果的文档。测试报告是测试阶段最后的文档产出物，一份详细的测试报告要包含足够丰富的缺陷分析信息，还要包括对产品质量和测试过程的评价。*

1. 引言
   - 1.1. 项目背景
   - 1.2. 系统简介
     - 1.2.1 系统总体架构
     - 1.2.2 系统功能模型
   - 1.3. 引用文档
2. 测试概述
   - 2.1. 编写目的
   - 2.2. 测试组织
   - 2.3. 测试环境
   - 2.4. 测试范围
3. 测试过程
   - 3.1. 测试内容
   - 3.2. 测试时间
   - 3.3. 测试方法
4. 测试结果和缺陷分析
   - 4.1. 覆盖分析
     - 4.1.1 需求覆盖
     - 4.1.2 测试覆盖
   - 4.2. 缺陷的统计和分析
     - 4.2.1 缺陷统计
     - 4.2.2 缺陷分析
5. 测试总结与建议
   - 5.1 测试结论
   - 5.2 测试建议

系统上线与运维

- 测试工程师一般在测试环境下进行软件测试，测试通过后，往往会由运维人员将测试通过的代码版本部署至预热环境，预热环境测试通过后再部署到线上生产环境。



1. 测试环境
开发工程师开发完成后，会将代码提交至代码仓库，测试人员从代码仓库中拉取代码部署至测试环境，测试工程师在该环境下进行日常测试。
2. 预热环境
预热环境是测试环境到生产环境的过渡，预热环境往往会选取生产环境的某一个节点（也就是集群中的某一台机器）。代码部署至预热环境后，验证功能是否正常，如果在预热环境测试通过，那么再部署至生产环境。
3. 生产环境/线上环境
生产环境/线上环境是用户使用的环境，由特定人员来维护。生产环境一般会部署在多台机器组成的集群上，以防某台机器出现故障影响系统运行。在集群中，当某台机器出现故障时，其他机器可以继续运行，不会影响用户使用。集群环境也可以支持更多的用户访问系统。

测试用例

概述

- 测试用例：指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术和策略；内容包括测试目标、测试环境、输入数据、测试步骤、预期结果等，并形成文档
- 是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求
- 完整的测试用例包括：
  - *名称和标识*
  - *修改历史*
  - *测试用例分析*
  - *测试环境*
  - *每条测试用例的详细信息*
- 编写测试用例的依据
  - *单元测试用例编写依据：详细设计说明、软件需求规格说明书、软件测试计划；*
  - *集成测试用例编写依据：概要设计说明、软件需求规格说明书、软件测试计划；*
  - *功能测试用例编写依据：软件需求规格需求说明书、软件测试计划；*
  - *系统测试用例编写依据：用户需求（系统/子系统设计说明、软件开发计划等）、软件测试计划。*

测试用例的作用

- 实施测试指导的作用
- 指导测试数据规划的作用
- 指导脚本编写的作用
  - *软件测试行业也由原来的人工测试逐步向人工测试、自动化测试兼之并行的方向发展。而自动化测试的核心就是测试脚本。*
  - *自动化测试所使用的测试脚本编写的依据就是用测试用例来进行编写设计的*
- 作为评判基准的作用
  - *测试工作完成后需要评估并进行定论，判断是否合格，然后出具报告。*
  - *测试工作的评估审查以前是依据统计结果来判断的，但是这种方法相比依据测试用例来评审显的有些不够精细。所以现在的评判基准是以测试用例为依据的，测试结束后需要测试总结，总结中包括：*
    - 测试中检测到的Bug数目
    - 有效的Bug数目
    - 无效的Bug数目等等。
- 作为分析缺陷的基准的作用
  - *测试的目的就是为了发现缺陷（bug），测试结束后把得到的Bug进行复查，然后和测试用例进行对比看看这个Bug是因为没有检测到还是因为其他地方重复出现了这个Bug。如果是因为没有检测到，说明测试用例不够完善，应该及时补充相应的用例，如果是因为重复出现，则说明实施测试存在一些问题需要去处理。最终目的还是为了交付给用户一个高质量的软件产品*

测试用例设计原则

- 利用成熟的测试用例设计方法来指导设计
- 测试用例的正确性
- 测试用例的代表性
- 测试结果的可判定性
- 测试结果的可重现性
- 足够详细、准确和清晰的步骤
- 利用测试用例文档编写测试用例时必须符合内部的规范要求
- 测试用例设计时需注意：
  - *不能把测试用例设计等同于测试输入数据的设计*
  - *不能追求测试用例设计的一步到位*
  - *不能将多个测试用例混在一个用例中*
  - *不能由没有经验的人员设计测试用例*

测试用例评审

- 测试用例中用户需求和测试功能点是否与测试计划和测试方案对应。
- 测试用例标识是否按照测试方案的规则来编写。
- 测试环境描述是否清晰。
- 设计测试用例运用了三种或三种以上的设计方法。
- 是否每个测试用例的预置条件都被描述清楚?
- 每个测试用例的“输入”中是否列出了所有测试的输入数据?
- 步骤、输入和输出内容是否清晰。
- 测试用例的“预期结果”是否完整而且清晰?
- 是否明确说明了每个测试用例或测试用例集的重要级别?
- 是否明确说明了测试用例的执行顺序?
- 测试用例分析中测试深度是否描述了使用的测试技术和方法。

测试用例的基本要素

- 测试用例编号
  - *可参考：项目名称＋软件版本号＋测试阶段类型＋用例的优先级＋编号*
- 测试标题
  - *对测试用例的描述，测试用例标题应该清楚表达测试用例的用途。比如“测试用户登录时输入错误密码时，软件的响应情况”*
- 优先级别
  - *定义测试用例的优先级别，可以笼统的分为“高”和“低”两个级别。一般来说，如果软件需求的优先级为“高”，那么针对该需求的测试用例优先级也为“高”。至于优先级如何来确定，可以根据项目需求，或者用户的需求来确定，也可以根据实际经验对那些很容易产生缺陷的模块设置为高优先级。*
- 测试输入
- 操作步骤
  - *操作步骤要写测试执行过程的步骤。对于复杂的测试用例，测试用例的输入需要分为几个步骤完成？在操作步骤中详细列出：测试环境、用例编写人/日期、测试执行者/日期*
- 预期结果

测试用例的优点

- 在开始实施测试之前设计好测试用例，避免盲目测试并提高测试效率，减少测试的不完全性；
- 测试用例使软件测试的实施重点突出、目的明确；
- 根据测试用例的多少和执行难度，估算测试工作量，便于测试项目的时间和资源管理与跟踪；
- 减少回归测试的复杂程度；
- 在软件版本更新后只需修正少量的测试用例便可展开测试工作，降低工作强度、缩短项目周期；
- 功能模块的测试用例的通用化和复用化则会使软件测试易于开展；
- 根据测试用例的操作步骤和执行结果，可以方便地书写软件测试缺陷报告；
- 可以根据测试用例的执行等级，实施不同级别的测试；
- 为分析软件缺陷和程序模块质量提供依据；
- 可以最大程度地找出软件隐藏的缺陷；
- 测试用例内容清晰、分类组织

设计测试用例应注意的问题

- 不能把测试用例设计等同于测试输入数据的设计；
- 不能追求测试用例设计“一步到位”；
- 不能将多个测试用例混在一个用例中；
- 用例的设计人员最好是具有丰富的经验测试人员，没有测试经验的人员不能设计测试用例；
- 用例应该从系统的最高级别向最低级别逐一展开；
- 每个测试用例都应单独放在文档中；
- 系统中的所有功能都应该对应到用例中；
- 每个用例都应该依据需求进行设计。
  - *测试用例是多样的、复杂的而且也是简单的，设计的技术也不唯一，下面介绍测试用例设计的一些技术*

综合设计测试用例要点

- 白盒和黑盒测试用例的设计方法各有各的特点，但是每一个测试用例设计方法都给出了有用测试用例的一个特殊的集合，但没有一个可以贡献出完整的测试用例集合。在实际项目运作设计时常常共同使用各种测试用例设计方法进行用例的设计，以此来弥补它们各自的缺点
- 实际操作设计测试用例一般是先黑后白，即：先用黑盒技术设计一些用例，再用白盒技术做一些补充用例
  - *如果规格说明书中包含输入条件，用因果图法进行设计测试用例。*
  - *如果源码中遇到输入输出边界，用边界值分析法进行设计测试用例，这是输入输出边界的分析。边界值分析产生一组附加的测试条件，但是大多数或全部这些条件都可以组合到因果测试中。*
  - *为输入和输出识别有效和无效等价类。*
  - *使用错误推测方法来增加测试用例。*
  - *用逻辑覆盖方法来检查程序的逻辑，使用判定覆盖、条件覆盖、判定/条件覆盖和多条件覆盖准则（最完整），如果满足此方法不可能实现，那么设计足够的测试用例去让此方法被满足*

单元测试用例的设计要点

- 单元测试的进行是在一组单元模块设计完成以后就开始的测试。单元测试要以程序设计说明书为指导，测试模块范围内的主要控制路径，以揭露错误。重心点放在代码审查、测试用例、测试特性、用例描述、测试总结上
- 单元测试用例设计需要注意的有以下几点：
  - *被测单元模块声明初始状态时，即此模块单元测试的开始。*
  - *被测单元模块进行正面测试时常采用的技术有：*
    - 依据设计说明书设计用例；
    - 用等价类划分设计用例。
  - *被测单元模块进行负面测试时常采用的技术有：*
    - 错误猜测；
    - 边界值分析。
  - *被测单元模块需要其他特性测试时依据设计说明书设计测试用例*
  - *需要进行检测覆盖率测试时常采用的技术有：*
    - 分支覆盖；
    - 条件覆盖。

功能测试用例的设计要点

- 首先考虑等价划分类、边界值共用的方法设计用例，用错误估算法补充用例。
- 如果程序业务流程很清晰，应考虑主要采用场景法设计用例。
- 如果程序有详细的因果关系，应一开始就考虑用因果图法。
- 如果是文件配置类型的测试，应该考虑用功能图法

集成测试用例的设计要点

- 集成测试是按照详细设计说明书来设计的；
- 集成测试中的用例中的数据来自于UC；
- 集成测试中内部逻辑结构分析按单元测试来进行

性能测试用例的设计要点

- 性能测试用例设计通常不会一次设计到位，是一个不断迭代完善的过程，一个完整的性能测试通常包括：预期指标的性能测试、独立业务性能测试、组合业务性能测试、疲劳强度性能测试、大数据量性能测试、网络性能测试、服务器（操作系统，WEB服务器，数据库服务器）性能测试、一些特殊的测试等。
  - *预期性能指标测试用例依据需求和设计文档中明确的性能要求进行设计；*
  - *独立业务性能测试从单个模块功能要求性能要求出发进行设计用例；*
  - *组合业务性能测试从需求、设计文档、现场调查、系统采集数据进行用例设计；*
  - *疲劳强度性能测试编写测试用例时需要编写不同参数或者负载条件下的多个测试用例；*
  - *大数据量性能测试通过考虑数据处理能力用边界值分析法进行设计用例；*
  - *网络性能测试主要针对基于应用系统的测试设计时重点使用工具调整网络设置；*
  - *服务器测试一定要和前面的测试结合起来进行这类部分的测试用例一般不必单独编写*

系统测试用例的设计要点

- 系统测试主要是根据需求分析来检验软件是否满足功能、行为、性能和系统协调性等方面的要求。此部分测试是放在实际的系统环境中运行的。
  - *所使用的数据应具有代表性；*
  - *所使用的数据应和真实数据的大小和复杂性相当。*

验收测试用例的设计要点

- 验收测试用例应当在研发阶段测试用例的基础上重新组织和编写，而不能拿来直接使用；
- 验收测试用例应客户需求相对应，具有面向客户的特点；
- 设计过程中要把握客户的关注点并适当展示软件的独有特性；
- 在验收测试中发现软件的缺陷或与需求存在偏差的地方应与客户保持良好的沟通共同确定修复和改进计划

回归测试用例的设计要点

- 回归测试用例是软件系统修改后，在保证没有新的错误引入的前提下而进行的重新测试，它的测试用例不需要重复进行设计，只需要选择以前的测试用例。
- 选择测试用例时可以按照优先级别不同进行选择测试用例，如：测试用例库的用例是基于软件操作开发的，可以优先选择基于操作的测试用例，针对最重要或最频繁使用功能的测试用例

测试用例设计模板-举例

- 数据处理类测试用例

测试标题				用例的编号 ID	
测试技术		测试环境要求		特殊要求	
测试用例设计人员		测试人员		测试日期	
测试目的					
测试对象					
测试项	测试内容	测试方法与步骤	测试判断准则		测试结果
1					
2					
3					
⋮					
n					

测试用例设计模板-举例

- 输入数据/动作的编写测试用例
  - *测试中输入的数据/动作，一般就是具体执行的过程。*
  - *测试的输入一般有两种，一是输入数据；二是输入动作。要根据测试的不同选择输入数据或输入动作*

接口 A 的函数原型		
输入/动作	期望的输出/响应	实际情况
典型值		
边界值		
异常值		
接口 B 的函数原型		
输入/动作	期望的输出/响应	实际情况
典型值		
边界值		
异常值		
…		

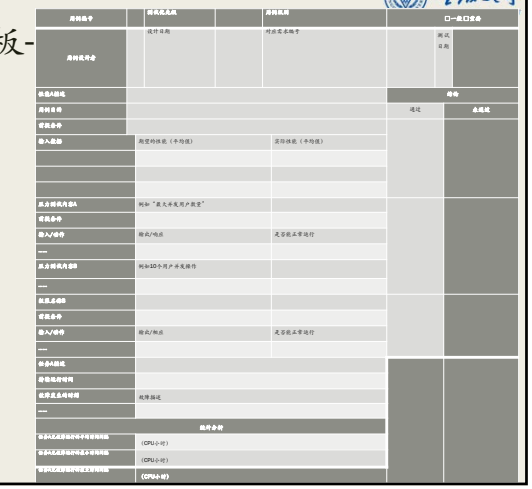
测试用例设计模板-举例

- 功能测试的测试用例

用例编号	测试点	测试摘要（目的）	操作步骤	预期结果	实测结果	测试结论	测试人员	测试时间	测试级别	备注
XXX-001	界面测试	检验用户界面是否友好	…	…	…	□ 通过 □ 未通过 □ 忽略	…	…	□一般 □重要	
XXX-002		检验界面的可用性	…	…	…	…	…	…	…	
XXX-003		检验界面的可靠性	…	…	…	…	…	…	…	
XXX-004		…	…	…	…	…	…	…	…	
XXX-005	安装测试	全部安装	…	…	…	…	…	…	…	

测试用例设计模板-

- 性能测试的测试用例



测试用例设计模板-举例

- 容错能力测试用例

用例编号		用例级别	□一般 □重要
容错性描述			
前提条件			
异常操作 A	预期（是否能够解决恢复）	实际结果	
异常操作 B	预期（是否能够解决恢复）	实际结果	

测试用例设计模板-举例

- 恢复能力测试用例

用例编号		测试优先级		用例级别	□一般 □重要		
用例设计者		设计日期		对应需求编号		测试日期	
异常输入/动作		容错能力/恢复能力		造成的危害、损失	结论	通过	未通过
错误的数据类型							
定义域外的值							
错误的操作顺序							
异常中断通信							
异常关闭某个功能							
负荷超出了极限							
……							

测试用例设计模板-举例

- 接口测试用例

用例编号		测试优先级		用例级别	□一般 □重要
用例设计者		设计日期		对应需求编号	
接口 A 的函数原型					
输入/动作	期望的输出/相应		实际情况		
典型值					
边界值					
异常值					
接口 B 的函数原型					
输入/动作	期望的输出/相应		实际情况		
典型值					
边界值					
异常值					
……					
测试日期					
结论	□ 通过		□ 未通过		

测试用例设计模板-举例

- 需求测试用例

用例编号		测试优先级		用例级别	□一般 □重要
用例设计者		设计日期		对应需求编号	
客户需求列表－需求说明书		开发人员－系统说明书－功能列表		测试人员－功能点测试列表	
1 注册功能		1 用户可以自动注册		（对比发现问题）	
2					
…… ……					
N					
测试日期					
结论	□ 通过		□ 未通过		

测试用例设计模板-举例

- 路径测试的测试用例

用例编号		测试优先级		用例级别	□一般 □重要	
用例设计者		设计日期		对应需求编号	测试日期	
检查项			结论			
			通过	未通过		
数据类型问题	存在不同数据类型的赋值吗					
	变量的数据类型有错误吗					
	存在不同数据类型的比较吗					
变量值问题	变量初始化或缺省值有问题吗					
	变量发生上溢或下溢吗					
	变量的精度不够准确吗					
逻辑判断问题	表达式中优先级有错误吗					
	由于精度原因导致比较无效吗					
	逻辑判断结果解释吗					

测试用例设计模板-举例

- 信息安全测试用例

用例编号		测试优先级		用例级别	□一般 □重要	
用例设计者		设计日期		对应需求编号	测试日期	
假想目标 A				评价		
				通过	未通过	
前提条件						
非法入侵手段	是否实现目标	代价－利益分析				
……						
……						
假想目标 B						
前提条件						
非法入侵手段	是否实现目标	代价－利益分析				
……						

测试用例设计模板-举例

- 界面测试用例

类别			特征
A类	单窗体	是 否	
B类	多窗体	是 否	
C类	资源管理器	是 否	

用例编号		测试优先级		用例级别	□一般 □重要		
用例设计者		设计日期		对应需求编号		测试日期	

检查项	测试人员 用户 相关人员	评价	通过	未通过
窗口切换、移动、改变大小时正常吗				
各种界面元素的文字正确吗				
各种界面元素的状态正确吗				
各种界面元素支持键盘操作吗				
各种界面元素支持鼠标操作吗				
对话框中的缺省焦点正确吗				
对于常用功能，用户能否不必阅读手册就能使用				
操作有风险的操作时，有“确认”、“取消”等提示吗				
操作顺序处理吗				
有联机帮助吗				
各种界面元素的布局合理、美观吗				
各种界面元素的颜色协调吗				
各种界面元素的形状美观吗				
字体美观吗				
图标直观吗				
……				

测试报告

概述

- 测试报告是把测试的过程和结果写成文档，并对发现的问题和缺陷进行分析，为纠正软件的存在的质量问题提供依据，同时为软件验收和交付打下基础。
- 测试报告是测试阶段最后的文档产出物，”优秀的测试人员”应该具备良好的文档编写能力，一份详细的测试报告包含足够的信息，包括产品质量和测试过程的评价，测试报告基于测试中的数据采集以及对最终的测试结果分析

测试报告-首页

- 报告名称（软件名称+版本号+XX测试报告）
- 报告委托方，报告责任方，报告日期等
- 版本变化历史
- 密级

测试报告-引言

- 编写目的：本测试报告的具体编写目的，指出预期的读者范围。
  - *实例：本测试报告为XXX项目的测试报告，目的在于总结测试阶段的测试以及分析测试结果，描述系统是否符合需求（或达到XXX功能目标）。预期参考人员包括用户、测试人员、、开发人员、项目管理者、其他质量管理人员和需要阅读本报告的高层经理。*
- 项目背景：对项目目标和目的进行简要说明。必要时包括简史，这部分基本不需要脑力劳动，直接从需求或者招标文件中拷贝即可。
- 系统简介：如果设计说明书有此部分，照抄。注意必要的框架图和网络拓扑图能吸引眼球。
- 术语和缩略语：列出设计本系统/项目的专用术语和缩写语约定。对于技术相关的名词和与多义词一定要注明清楚，以便阅读时不会产生歧义。
- 参考资料
  - *需求、设计、测试用例、手册以及其他项目文档都是范围内可参考的东东。*
  - *测试使用的国家标准、行业指标、公司规范和质量手册等等*

测试报告-测试概要

- 测试的概要介绍：包括测试的一些声明、测试范围、测试目的等等，主要是测试情况简介。
- 用例设计方法：简要介绍测试用例的设计方法。
  - *例如：等价类划分、边界值、因果图，以及用这类方法(3-4句)。*
    提示：如果能够具体对设计进行说明，在其他开发人员、测试经理阅读的时候就容易对你的用例设计有个整体的概念，顺便说一句，在这里写上一些非常规的设计方法也是有利的，至少在没有看到测试结论之前就可以了解到测试经理的设计技术，重点测试部分一定要保证有两种以上不同的用例设计方法。
- 测试环境与配置：简要介绍测试环境及其配置。
  - *提示：清单如下，如果系统/项目比较大，则用表格方式列出*
- 测试方法与工具：简要介绍测试中采用的方法和工具。
  - *提示：主要是黑盒测试，测试方法可以写上测试的重点和采用的测试模式，这样可以一目了然的知道是否遗漏了重要的测试点和关键块。工具为可选项，当使用到测试工具和相关工具时，要说明。注意要注明是自产还是厂商，版本号多少，在测试报告发布后要避免大多工具的版权问题。*

测试报告-测试结果与缺陷分析

- 主要汇总各种数据并进行度量，度量包括对测试过程的度量和能力评估、对软件产品的质量度量和产品评估。对于不需要过程度量或者相对较小的项目，例如用于验收时提交用户的测试报告、小型项目的测试报告，可省略过程方面的度量部分；而采用了CMM/ISO或者其他工程标准过程的，需要提供过程改进建议和参考的测试报告－主要用于公司内部测试改进和缺陷预防机制－则过程度量需要列出
- 测试执行情况与记录：描述测试资源消耗情况，记录实际数据。
- 测试组织
  - *可列出简单的测试组架构图，包括：*
    1. *测试组架构（如存在分组、用户参与等情况）*
    2. *测试经理（领导人员）*
    3. *主要测试人员*
    4. *参与测试人员*
- 测试时间：列出测试的跨度和工作量，最好区分测试文档和活动的时间。数据可供过程度量使用。
  - *例如XXX子系统/子功能：*
    实际开始时间－实际结束时间
    总工时/总工作日
    任务 开始时间 结束时间 总计

测试报告-测试结果与缺陷分析（续）

- 测试版本：给出测试的版本，如果是最终报告，可能要报告测试次数回归测试多少次。列出表格清单则便于知道那个子系统/子模块的测试频度，对于多次回归的子系统/子模块将引起开发者关注。
- 覆盖分析
  - *需求覆盖率是指经过测试的需求/功能和需求规格说明书中所有需求/功能的比值，通常情况下要达到100％的目标。*
  - *测试覆盖：需求/功能（或编号） 用例个数 执行总数 未执行 未/漏测分析和原因*
- 缺陷分析：本部分对上述缺陷和其他收集数据进行综合分析
  - *缺陷综合分析*
    - 缺陷发现效率 = 缺陷总数/执行测试用时
      用例质量 = 缺陷总数/测试用例总数 ×100％
    - 缺陷密度 = 缺陷总数/功能点总数
      缺陷密度可以得出系统各功能或各需求的缺陷分布情况，开发人员可以在此分析基础上得出那部分功能/需求缺陷最多，从而在今后开发注意避免并注意在实施时予与关注，测试经验表明，测试缺陷越多的部分，其隐藏的缺陷也越多。
    - 测试曲线图
      描绘被测系统每工作日/周缺陷数情况，得出缺陷走势和趋向

测试报告-测试结果与缺陷分析（续II）

- 残留缺陷和未解决的问题
  - *残留缺陷*
    - 编号：BUG号
    - 缺陷概要：该缺陷描述的事实
    - 原因分析：如何引起缺陷，缺陷的后果，描述造成软件局限性和其他限制性的原因
    - 预防和改进措施：弥补手段和长期策略
  - *未解决问题*
    - 功能/测试类型：
    - 测试结果：与预期结果的偏差
    - 缺陷：具体描述
    - 评价：对这些问题的看法，也就是这些问题如果发出去了会造成什么样的影响

测试报告-测试结论与建议

- 对上述过程、缺陷分析之后该下个结论，此部分为项目经理、部门经理以及高层经理关注，请清晰扼要的下定论。
- 测试结论
  - *测试执行是否充分（可以增加对安全性、可靠性、可维护性和功能性描述）*
  - *对测试风险的控制措施和成效*
  - *测试目标是否完成*
  - *测试是否通过*
  - *是否可以进入下一阶段项目目标*
- 建议
  - *对系统存在问题的说明，描述测试所揭露的软件缺陷和不足，以及可能给软件实施和运行带来的影响*
  - *可能存在的潜在缺陷和后续工作*
  - *对缺陷修改和产品设计的建议*
  - *对过程改进方面的建议*

测试报告-附录

- 缺陷列表
- 缺陷等级定义标准
- 测试通过标准