目录

[概要 2](#_Toc1772)

[测试风险 2](#_Toc2991)

[1.风险识别 2](#_Toc28528)

[2.风险评估 2](#_Toc17665)

[3.风险缓解 2](#_Toc31998)

[4.风险监控 2](#_Toc14693)

[测试策略 3](#_Toc27592)

[敏捷会议评审 5](#_Toc8138)

[版本控制 5](#_Toc23673)

[测试进度跟踪 5](#_Toc11706)

[测试用例 6](#_Toc12217)

[接口测试方法 6](#_Toc25957)

[1. 功能测试 6](#_Toc31383)

[2. 边界测试 6](#_Toc7373)

[3. 异常测试 6](#_Toc17269)

[4. 性能测试 6](#_Toc20371)

[5. 安全测试 6](#_Toc29484)

[测试人员KPI 7](#_Toc6134)

[1.工作内容和质量 7](#_Toc1157)

[1.需求熟悉程度 7](#_Toc20739)

[2.测试用例覆盖度 7](#_Toc3127)

[3.测试用例完成质量 7](#_Toc13062)

[4.有效BUG率 7](#_Toc17924)

[5.BUG描述质量 7](#_Toc1210)

[6.测试报告质量 7](#_Toc30647)

[7.按时完成测试工作 7](#_Toc3792)

[8.项目进度更新，项目BUG跟踪 7](#_Toc16651)

[2.工作效率 7](#_Toc11449)

[1.测试用例产出率 7](#_Toc26560)

[2.BUG发现率 7](#_Toc27880)

[3.测试执行率 7](#_Toc26988)

[4.测试过程中问题沟通效率 7](#_Toc986)

[3.素质能力 7](#_Toc10445)

[1.工作态度 7](#_Toc12458)

[2.沟通能力 7](#_Toc31395)

[3.培训指导能力 7](#_Toc26364)

[4.加分项 7](#_Toc13912)

[1.学习研究新技能研究 7](#_Toc29054)

[2.需求阶段发现巨大需求漏洞 7](#_Toc20745)

[3.其他好的建议，并能推动实施 7](#_Toc6495)

[5.减分项 7](#_Toc32168)

[1.测试项目有要点BUG的遗漏 7](#_Toc603)

[2.个人操作出错引起严重问题 7](#_Toc25419)

[性能测试 7](#_Toc10321)

[测试报告 8](#_Toc17131)

# 概要

测试管理由两部分组成：计划 和 执行。

计划包含风险的识别，测试策略的输出，测试资源，人员的分配。

执行包含测试继续监控，问题管理，测试的反馈和评估

# 测试风险

对测试工程中常见的风险做一个大致罗列，如：

技能、培训和人员不足；

项目团队成员出现的个人问题，人员流失；

组织内部协作不调，缺乏有效的沟通渠道；

测试时间受到压缩；

交付的代码质量低下，加重测试以及返工任务量；

产品的缺失配置管理，变更流程等控制手段；

团队的合作效率问题；

风险管理一般通过下面四个阶段来完成：

1.风险识别

2.风险评估

3.风险缓解

4.风险监控

在进行风险管理的过程中，要把握好四个原则：

可避免的风险，采取好的过程管理和流程控制来应对；

不可避免的风险，采取降低和转移的措施；

做好风险管理计划；

做好应急方案；

对于目前可以想到的风险有以下类型：

对于测试过程风险以及解决方案：

需求的计划外变更-做好变更控制和配置管理，为可能的变更预留时间和人员的调整空间

测试用例执行率不足-日常跟踪所有工作过程，及时发现阻碍测试执行的因素并协调解决

测试分析产生偏差-更完善的测试分析流程，对于经验不足的人员安排指导，用评审的手段予以检查

测试用例设计不足-更完善的测试分析流程，充足的人员技能培训和指导，用例评审的把关

测试与生产环境差异-尽量缩小测试环境与生产环境的差异，比如使用更大的数据量，更强的客户响应以确定用户生产环境的特性

偶现类问题-倡议充分的问题记录和分析流程

代码质量过低-更好的单元测试实行，倡议和建立规范的提测控制流程

回归测试覆盖率不足-适合的回归测试策略，自动化的回归测试覆盖

对于人员风险以及解决方案：

人员流失-积极响应人员诉求，创造更积极的工作流程及环境，做好人员技能储备

人员不可用状态（休假等）-建立良好的文档归档流程，建立完善的工作后备机制（不可让某项工作只有某一个人能完成）

新人工作准备-建立良好的人员培训机制，建立帮助新人融入的导师机制

# 测试策略

个人建议 是图文化的测试策略，相对于冗长的文字测试策略，图文化的更加便捷。

图示化就是用图来描述测试策略的内容。我们考虑用图来表示测试策略的各个关键组成部分，并且绘在一页纸上。

当然，一页纸的测试策略只是将关键信息以图示化的方式呈现出来，并不是整个测试策略的全部，在一页纸的背后是团队的充分沟通和对策略各个方面达成的一致认识，是需要团队一起来做很多工作的。这种高度简化的呈现形式，是为了给团队更多的讨论空间，一页纸也更易于修改，从而更能适应变化，真正满足需求。

建议的测试策略图包含下列信息：

指导性原则：团队为质量负责

测什么：可能包括功能、性能和安全等

如何测：涵盖测试流程、测试类型、测试方法等。

谁去测：人员的分配，根据测试人员的需求熟悉程度，业务技能分配测试任务

多少时间测：版本的预期时间和模块的规划测试时间

在哪里测：测试环境，预发布环境，正式环境

测试类型：

功能

接口

性能

安全

兼容

界面

金字塔的含义是测试比例的多少，体现为底层单元测试较多，越往上层测试比例越少，呈现为金字塔结构。

越往底层的测试越接近代码，编写成本更低、执行速度更快、定位问题也更准确，但是离业务较远，不能很好的体现业务价值；越往上层的测试越接近业务，更能反应业务价值，但有着不够稳定、执行速度慢、实现成本较高的不足。因此，需要权衡利弊，根据项目具体情况，真实的目标来确定每层测试的比例。

对于稳定的产品型项目，个人推崇是版本迭代之间使用接口自动化测试脚本，性交比极高，还可以将脚本与Jenkins结合使用，构建版本之后立即执行，版本质量就可以有个大概的预估。

新需求和过往功能需要手工测试和冒烟测试。

一页纸搞定的测试策略，优势非常明显，比传统策略文档更加简洁、清晰，关键信息一目了然。

图示化的测试策略是高度精简的，具有更大的讨论和发挥空间，在防止僵化、保持演进方面的优势明显。

# 敏捷会议评审

每个测试环节中的迭代都少不了评审会议，这保证了测试输出的内容方向是否正确，比如：测试策略，测试用例，测试报告，都需要项目干系人参与会议评审。

# 版本控制

1. 如果缺乏版本控制，测试的质量和进度都是难以保证
2. 使用代码管理工具比如git去管理代码版本
3. 使用项目管理工具管理测试版本
4. 不允许随意融入新需求，新需求应该放入需求池中，在下个版本加入，或者启动版本变更响应机制，项目干系人签字确认，延长版本周期等。

# 测试进度跟踪

使用相关的管理软件，可以将需求，需求分解出的工作内容，bug，测试用例，相关联，从而得出测试进度，相关模块的bug率，相关测试人员的测试覆盖率，执行效率，项目进度等等许多KPI。

# 测试用例

快速掌握用例其实没有什么窍门，只有多问，多看，多想，多写，多评审。

测试用例设计主要是采用等价类、边界值、场景设计法、判定表、因果图、正交法、错误猜测法。

用例需要会议评审。

# 接口测试方法

1. 功能测试
   1. 正常场景
   2. 错误场景
2. 边界测试
   1. 所有必填参数
   2. 组合可选参数
   3. 参数的个数，顺序，类型
   4. 参数的大小，范围
   5. 参数长短
   6. 参数特殊字符
3. 异常测试
   1. 幂等
   2. 并发
   3. 事务
   4. 分布式
   5. 环节异常
   6. 大数据量
4. 性能测试
   1. 响应时间
   2. 吞吐量
   3. 并发量
   4. 服务器使用资源
5. 安全测试
   1. SQL注入
   2. 是否加密

# 测试人员KPI

1.工作内容和质量

1.需求熟悉程度

2.测试用例覆盖度

3.测试用例完成质量

4.有效BUG率

5.BUG描述质量

6.测试报告质量

7.按时完成测试工作

8.项目进度更新，项目BUG跟踪

2.工作效率

1.测试用例产出率

2.BUG发现率

3.测试执行率

4.测试过程中问题沟通效率

3.素质能力

1.工作态度

2.沟通能力

3.培训指导能力

4.加分项

1.学习研究新技能研究

2.需求阶段发现巨大需求漏洞

3.其他好的建议，并能推动实施

5.减分项

1.测试项目有要点BUG的遗漏

2.个人操作出错引起严重问题

# 性能测试

# 测试报告

测试结论（测试是否通过/是否满足发布要求/是否能够发布）

罗列发现的主要问题（或者说该版本存在的主要风险）

测试版本（客户端，服务器）

测试内容（测试范围）

测试用例执行情况

发现的严重缺陷有哪些

风险（出现的风险，是否解决，可能出现的风险预警，解决方案）

总结（问题，建议）

报告需要会议评审。