**整体测试策略指引**

|  |  |  |  |
| --- | --- | --- | --- |
| **文件类别** | ○手册 ○流程 ●指引/标准/规范/ ○其他 | | |
| **发布版本号** | V1.0 | **发布日期** |  |
| **密 级** | ○公开 ●秘密 ○机密 ○绝密 | | |

明源软件股份有限公司

修订记录

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 修订日期 | 修订说明 | 版本号 | 修订人 | 审核人 | 审核日期 |
| 1 | 2019-12-15 | 创建 | V1.0 |  |  |  |
|  | 2020-03-07 | 修订 |  |  |  |  |
|  |  |  |  |  |  |  |

# 目的

“测试策略”的目的简单来讲就是6个字，指引测试：“测什么”和“怎么测”。

# 适用范围

本指引适用于公司大型专项测试策略的制定。

哪些需求需要制定测试策略？以什么形式制定？由谁来制定？可以按以下几个维度分：

* 整体工作量>=20天，需要制定测试策略
* 如果需求任务涉及多套子系统，或者任务是超过上百人日的专项，可采用整体策略+迭代批次子策略的模式。
* 如果本次需求任务是多人协作型，应由PM指定测试Leader制定测试主策略，交由测试中台评审，如果是大型的复杂项目，还需要架构师、设计、专业线一起评审。
* 测试人员制定子策略交由测试Leader评审。子策略在整体策略上进行拆分和细化。

注：整体策略需按模板制定，子策略可以根据复杂程度由测试LD确认按模板制定或并入测试用例中

* 如果本次需求是单人操作型则由测试人员自己制定【注：如是ODC独立测试，需提交测试策略给对应的ABU测试评审，如果是ABU独立测试需提交测试中台评审】
* 如果需求任务是单系统单人的可只制定整体策略，可以根据任务复杂度单独制定策略，或将测试策略并入测试用例中【注：将测试策略并入测试用例的在命名时注明是测试策略+测试用例】
* 整体工作量<20天可不制定测试策略。特殊类型除外，如升级类、主动治理【性能优化类、安全漏洞修复类】

如需制定形式及人员安排同>=20天

# 测试范围

测试策略编写思路如下：

* 在开始制定测试策略前，首先要明确项目的目标及项目整体规划，项目目标包括项目交付时间目标和质量目标。
* 制定目标后要明确范围，重点是识别需求类别，再针对每类需求分析测试范围和不需要测试的范围、测试的重难点重难点及测试的优先级，范围包含功能和非功能，非功能包含性能、安全性、兼容性等，通过显性需求挖掘出背后的隐性需求，即需求文档中没有明确提出的。
* 依据分析结果，明确测试类型，并选择合适的测试方法、测试环境、工具。

整体流程及切入点如下图：



## 明确目标

### 时间目标

* 明确整体交付节点：由项目组与一线沟通确认后形成(如果是专项拆分为多个批次的，需跟一线明确每个迭代批次的交付节点及交付范围)
* 明确每种类型测试的关键节点及里程碑节点，

### 质量目标

* 功能部分可以从以下几个维度来考虑：

a、开发内部bug率<=1

开发内部bug率=（1级bug数\*1.2+2级bug数\*1+3级bug数\*0.8+4级bug数\*0.5）/开发计划工作量

b、测试内部bug率>=1.6

测试内部bug率=（1级bug数\*1.2+2级bug数\*1+3级bug数\*0.8+4级bug数

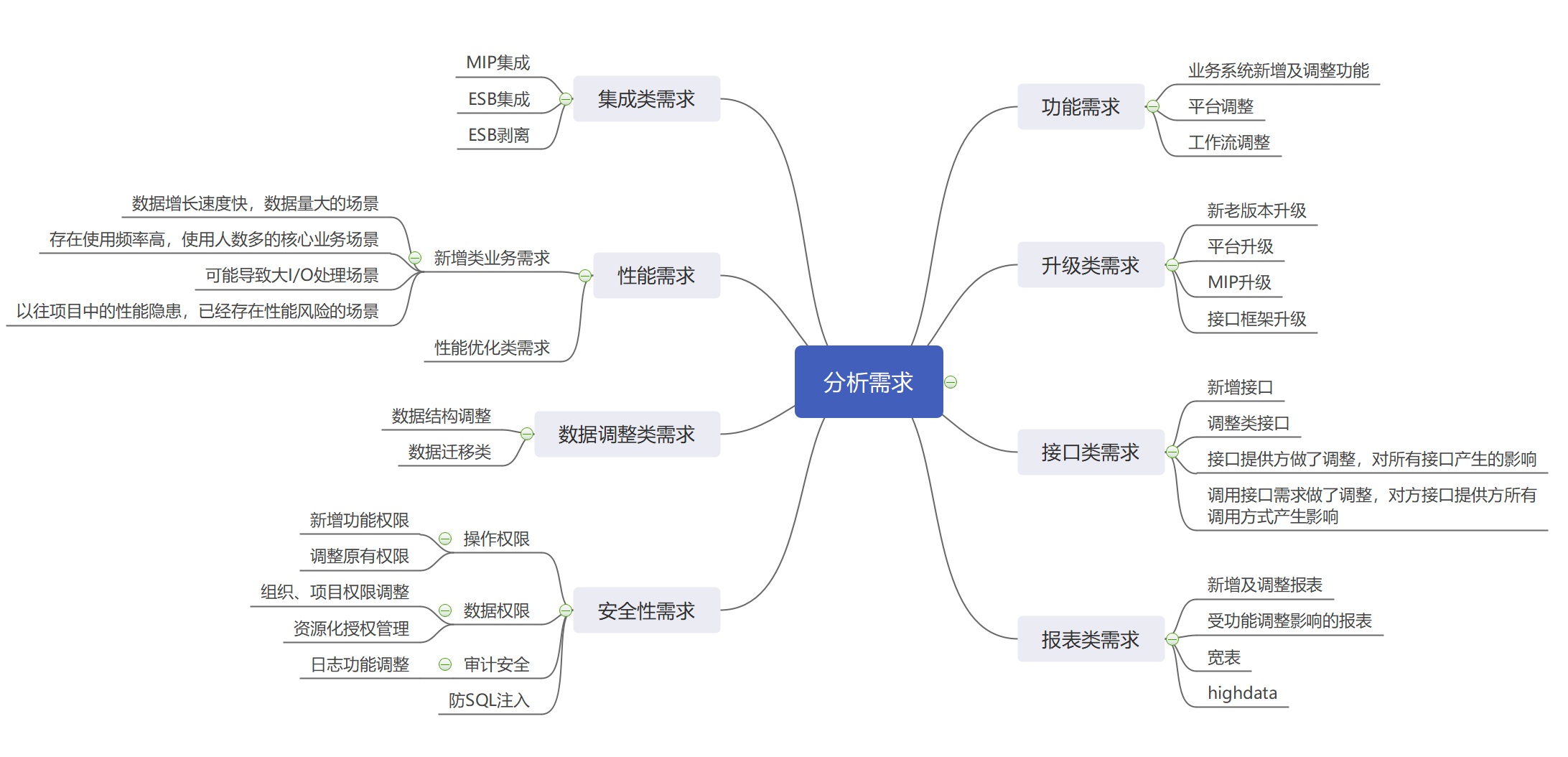
c、外部bug率<=5%

外部bug率=外部bug/交付工作量

* 非功能部分如接口功能、业务流程正常，客户实际数据量及并发量下，二开模块性能正常
* 数据迁移后数据准确，能正常用迁移数据进行功能操作

## 分析需求

测试需求前先识别整体需求中包含的类别，详细分析维度如下：



## 制定策略

依据业务需求和功能需求，根据各测试类型的分析维度，分析包含的测试范围，并对影响范围进行分析，最终得到版本所包含的完整测试范围详细清单。

### 3.3.1功能测试

#### 测试目标

按照需求迭代批次，输出测试用例（场景、功能、数据），保证所有测试用例通过评审。基于通过评审的测试用例全部执行通过，缺陷全部关闭（如有暂不修复的缺陷，需明确修复计划，保证在上线前全部关闭）

#### 测试方法

手工测试与WEBUI自动化测试

#### 测试范围

##### 功能测试：

1. 功能测试的范围：

* 新增和调整功能首先需要考虑。
* 理清新增调整功能影响范围，将新增功能或调整功能引发的其它系统或功能的所有调整点列出
* 需求中覆盖了很多场景，区分正常场景、异常场景，模拟错误、无效的流程
* 涉及复杂算法的，拎出需求中的复杂算法，通过前置数据分析分析功能的入口、出口，得出影响范围，测试用例中需包含复杂算法的测试用例及数据用例，
* 数据测试范围（历史数据，数据库调整）
* 功能并发控制分析

1. 功能测试中的重难点识别，可以按照以下几个维度来确立：

* 被测功能在系统中的地位
  + - * 客户最关注的功能应作为测试的重点，对非核心功能但是会核心功能产生影响的模块，也应确立为测试的重点
* 客户能够容忍哪些错误的存在
  + - * 针对客户不能接受的错误，可以确立为测试的重点
* 被测功能的使用频率
  + - * 使用频率高的发生错误的可能性更大，核心功能中使用频率较高的模块可以确立为测试的重点
* 被测功能的提出人、使用人
  + - * 针对高层提出的或者高层使用的模块，可以确立为测试的重点
* 错误产生的影响
  + - * 不同模块出现错误产生的影响不同，针对某些模块出现问题会导致其他模块正常运行的作为测试的重点，另外涉及金额及资金流失的也可确立为测试的重点

##### 数据库和完整性测试

* SQL语句执行正确：

更新SQL，执行成功，没有黄色警告及报错现象

已经更新的脚本需检查脚本是否可重复执行

* 数据表结构分析

主健及索引可以简单检查下

* 作业

如果有作业，需要提供作业的详细操作安装步骤

* 历史数据

客户存在历史数据，首先要考虑历史数据的处理

##### UI测试

* 新增，调整功能UI检查
* 参照设计文档说明书中的UI描述进行测试【如有高保真，则需内部明确UI测试是参照高保真，还是参照设计文档的UI截图，参照高保真图片进行测试时，要同步检查设计文档上UI截图是否与高保真一致，当发现不一致时，发送正式邮件让一线、设计修改文档上的UI截图）】
* 参照通用测试用例中标准控件测试标准进行测试

### 3.3.2非功能性

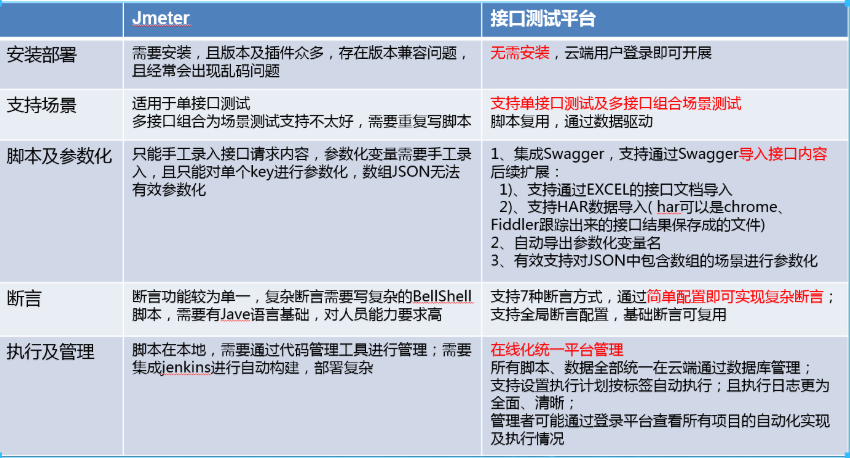
##### 3.3.2.1接口测试

##### 3.3.2.1.1测试目标

保证外部系统与系统之间数据传递的通畅正确性。

##### 3.3.3.1.2测试方法及工具

* 接口测试工具



* 接口测试的方法

1. 手工测试

1、有界面接口测试可以通过功能按钮触发，直接查看日志中的发送报文和返回报文

2、无界面接口测试（如定时调度任务，无手工调用按钮）可以通过工具调用接口来测试，如postman：

检查接口返回的数据是否与预期结果一致，测试接口是否联通（检查“状态码”、“消息体”中的内容）

根据接口文档测试接口的传入参数的合法性（验证字段长度，属性，是否必填，取值是否正确，关联业务场景等）

根据接口文档测试接口的传出参数的合法性（验证字段长度，属性，是否必须，逻辑，异常处理等）

检查接口的容错性，假如传递数据的类型错误时是否可以处理

 常见的http状态码：

 200   2开头的都表示这个请求发送成功，最常见的就是200，就代表这个请求是ok的，服务器也返回了

 300   3开头的代表重定向，最常见的是302，把这个请求重定向到别的地方了

 400   400代表客户端发送的请求有语法错误，401代表访问的页面没有授权，403表示没有权限访问这个页面，404代表没有这个页面

 500   5开头的代表服务器有异常，500代表服务器内部异常，504代表服务器端超时，没返回结果

1. 自动化测试
2. 准备环境（若环境为外部客户环境，则要求可以外网访问）

程序包：用最新的程序包搭建独立的自动化分支环境【环境必需等同于最新的集成环境】

数据库：可以用产品标准数据库，与标准脚本匹配的版本，通过跟项目数据库对比结构差异性来同步数据库，然后部署数据库环境

是否涉及外部接口，是否需要配置Mock环境

配置Mock的调用路径地址

Mock的正确返回参数模板格式和失败返回模板格式

1. 编写用例【可以与功能用例结合】
2. 编写脚本：

* 内部接口：公司系统内部接口互相调用，使用云测平台编写接口脚本；
* 外部接口：

1. 公司—>第三方接口：使用云测平台编写接口脚本：校验传出值
2. 第三方—>公司接口：使用云测平台编写接口脚本：通过Mock方式，校验传入值【当第三方未通的时候，可以采用MOCK方式模拟第三方返回结果】
3. 脚本维护

#### 性能测试

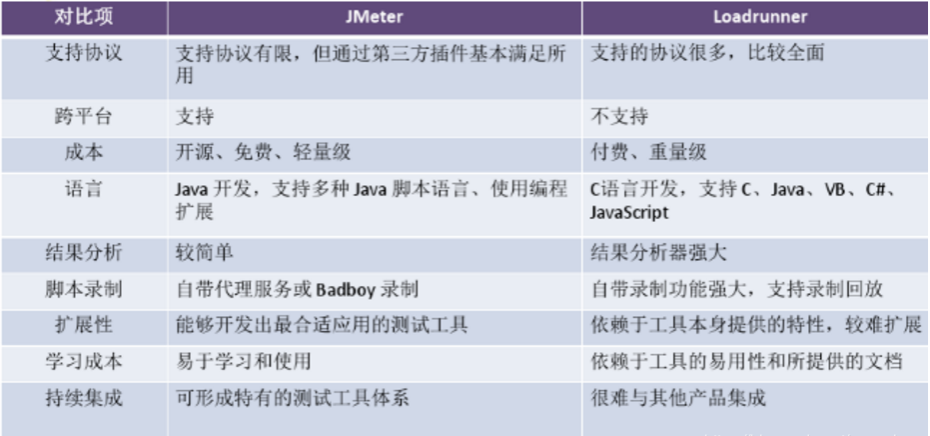
##### 测试目标

保证功能正常后，单用户大数据测试、单场景/组合场景并发条件下，性能满足通过要求；

##### 测试方法

* 性能测试工具选择

Loadrunner或Jmeter，两者区别如下：



* 性能测试的步骤

1. 性能测试前准备工作，在进行测试之前先要通过指标值分析出性能测试的范围

类别一：明源系统内部升级，可以通过以下两种方式来分析提取调用频率、高峰应用时间段、最大并发数

* 获取典型客户数据库，通过系统日志分析，哪些是每日高频、特定时间高频、中频、较少使用。以此安排优先级
* 获得该客户的业务规模，根据客户总的在线人数，定位最高峰每小时访问次数较高的模块

如：有使用过销售系统，可查看近三年的销售数据，得出业务规模

未使用过销售系统，由产品经理咨询一线确认该客户的业务规模

类别二：从非明源产品升级到明源系统时,通过访问客户的老系统，查询数据库确认性能测试范围，主要从以下三个维度分析来提取数据

* 高层操作的模块
* 高频操作的核心模块（调用频率、高峰应用时间段、最大并发数）
* 存在性能隐患的部分清单列表

1. 环境准备

性能测试环境要求，需保持客户正式环境相同的配置（如果客户无法提供生产一致的环境，一方面可以考虑在客户测试环境压测，另一方面实在满足不了，结合实际情况现在低配置上压压测识别初步初步性能问题，但需要得到客户同意），专用，不安装非需要的服务，这样能保证性能测试的准确性。具体的环境配置需根据客户当前配置进行明确。

|  |  |  |  |
| --- | --- | --- | --- |
| **设备** | **数量** | **硬件配置** | **软件配置** |
| DB  服务器 |  | CPU ：  内存：  硬盘（本地）：  硬盘（存储阵列）：  网卡： | 操作系统：  数据库： |
| WEB服务器 |  | CPU ：  内存：  硬盘（本地）：  硬盘（存储阵列）：  网卡： | 操作系统：  IIS: IIS 7.0 |
| 压力  测试PC机 |  | 2 CPU：  内存：  网卡： | 操作系统： |

1. 数据准备

* 从生产库导数据
* 存储过程生产数据，也可以写脚本---建议登录用户数据使用存储过程生产
* loadrunner或jmeter生产---过程业务数据建议使用工具生产，数据比较真实
* 针对数据量，通过跟客户沟通测算未来几年的数据增长量，分三批压测，一级为当前生产数据量，二级翻一倍，三级翻两倍。【具体数据量跟客户明确】

1. 性能测试场景分析
2. 单用户场景测试

* 分页列表的查询
* 非分页列表的查询与操作
* 主从列表的同时保存
* 导入、导出
* 文档上传/下载
* 批量操作

1. 单交易并发场景测试步骤

* 第一步，锁定高访问繁率模块，通过系统日志分析，定位排名前十的模块，最高峰每小时访问次数高于20次的模块
* 第二步，基于业务分析访问模块的业务场景及操作
* 第三步，分析各类场景的高峰期业务数据量
* 第四步：依据各类业务高峰期每分钟的业务量得出该场景的并发用户数
* 第五步：依据被分析数据库的规模推算出3000亿规模时的并发用户数

1. 组合交易场景测试步骤

* 第一步：锁定最高访问量的时间(TOP20)
* 第二步：分析锁定时间的特征
* 第三步：取每类特征的两个时间点，分析该时间段各个模块的访问情况
* 第四步：分析不同特征时间段的模块访问情况的差异性，确认组合模型的种类及各类模型的访问模块和访问比例
* 第五步：依据各模块的单交易并发场景，确认组合场景中各个模块的业务场景、各场景并发用户数、业务数据量
* 性能测试报告分析

#### 平台升级类测试

参考文档《集成测试任务指引和模板》

#### 安全测试

##### 测试目标

* 保证安全测试用例全部执行通过；

##### 测试方法

* 手工测试。

##### 覆盖策略

* 公司最近扫雷行动，主要防止木马侵入，导致客户资料泄露；前期需要与PM沟通本次专项是否考虑的扫雷行动。
* 如果是新客户，可以建议客户先打下安全补丁
* 如果是老客户，与PM沟通是否需要进行扫雷行动，扫雷行动放在哪个阶段。
* 软件狗及版本测试
* 如果是二次开发增加个性化模块，不需要进行软件狗测试
* 如果客户想要新的产品，需要进行软件狗测试，如电子开盘属于新产品，客户需要新增这个模块，必须先写狗
* 权限点测试
* 权限点测试一般放在功能测试中
* 权限点组合测试
* 团队权限和用户权限测试：如销售系统要着重测试团队权限和用户权限过滤。
* 密码策略测试
* 如果需求有涉及到密码策略方面，需要进行密码策略测试
* http改https
* 可以分别用http还有https登录试试。如果用https可以正常登录，地址栏显示一把锁头，那么这个网站是有部署SSL的。
* 如果http和https都能够正常登录，进一步说明该网站没有设置强制https登录，或者说没有设置http链接自动跳转https链接.
* 相反如果用http登录，结果跳转到https页面，说明网站部署了SSL，而且设置了http自动跳转https
* 也可以通过SSL体检网站，输入域名进行SSL体检。这样可以直观看出网站有没有部署SSL

#### 兼容测试

##### 测试目标

保证在不同浏览器不同版本下，WEB端访问正常，无兼容性问题；保证在不同类型手机，及不同操作系统的不同版本下，移动端访问正常，无兼容性问题。

##### 测试方法

工具与方法：手工测试

##### 覆盖策略：

* 主要包括产品之间的兼容（如ERP和EKP）、ERP各版本之间的兼容以及我们的产品和第三方系统之间的兼容（如：和其他公司财务软件、办公平台之间的兼容）。
* 目前ERP和EKP系统是能实现兼容，但ERP是不支持各版本之间的兼容的。未来的规划是，ERP各子系统各版本之间实现可拔插的集成兼容，如：ERP2.5.3的成本系统可和ERP254的全面预算系统集成在一起使用。
* 针对不同浏览器，不同系统版本进行测试

#### 数据调整测试

##### 测试目标

保证迁移数据展示正常、历史数据处理正常、数据流转正常、数据结果正确

##### 测试方法

使用手工测试活使用导数工具(待专业线规划)

##### 测试要点

* 数据量一致性测试

   要做到新老系统无缝切换，就必须保证数据的正确性和一致性，首要条件就是迁移的数据量是保持一致的，否则无法进行其他测试。

方法：①、可以通过文本统计工具或者数据库连接工具将迁移前的数据库表名、字段、数量等进行统计，然后将迁移后的新数据库表名、字段、数量等进行统计，然后进行比较。

     ②、通过MD5生成工具，对新老数据文本进行MD5值比对测试，如果一致则表示数据量一致，如果不一致，则表明迁移后的数据有部分存在问题。

* 数据库表结构变化测试

   这种测试分2种情况：一种是新老数据库表结构完全不存在关系，新数据库表的字段都是给定的默认值；还有一种是新数据库表字段是由原有系统数据库表字段转换而来。

方法：针对第一种情况，因为新增的字段都是给定的默认值，所以只需要根据开发提供的填写规则，检查该字段的所有值是否满足填写规则。

     针对第二种情况，可以通过编写自动化测试脚本或者人工抽样或者切片方式进行，具体的抽样选择根据数据量等具体情况选择合适的比例即可。

* 业务逻辑测试

完成上面的数据迁移测试后，需要在新的系统中进行回归测试，以确保迁移过来的数据是100%可用的。

回归测试前需要和业务、开发确认哪些业务模块涉及了哪些表，然后根据具体情况，尽可能的提高测试用例覆盖率，做一次全系统的功能回归测试（可以考虑自动化测试来替代一部分手工测试）。

##### 测试注意点

* 是否存在跨公司、跨项目的迁移；
* 要注意null值的处理（系统间的容错处理不同）；
* 提前识别是否有拼接字段；
* 转换规则是否全面：新老系统之间所有字段名字的匹配、字段内容的匹配转换（比如老系统存储是空，到新系统规则应该是有默认值或null，避免新系统报错；再比如金额长度，如小数点的精确度，2位和4位，避免高位转低位，导致数据丢失）

# 安排计划

**测试工序及各工序策略安排**

**安排计划**

定4 **5W1H原则**

* 测试工序依据5W1H原则制定：即为什么制定该措施（WHY）? 达到什么目标（What）?在何处执行（Where）？由谁负责完成（Who）？什么时间完成（when）？如何完成（How）
* 测试工序依据测试过程指引分为四个阶段：可接受性测试，功能测试，系统测试，回归测试。根据需求的难易度以及测试过程中测试质量和测试进度的把控，可以根据实际情况增加一轮测试来保障质量。
* 需求中如涉及压力测试、自动化化测试多人执行，可按照流程图中的测试介入点和事项。
* 自动化测试、及压测测出的bug，需先在功能分支上修复验证通过之后再同步到自动化及压测的分支上验证。
* 针对重难点功能在计划安排上尽量避免新人测试或对核心业务不熟悉的人员，可针对bug集中的模块进行多轮回归或探索性测试。
* 对于有迭代批次的测试任务，或多分支并行的测试任务，前期可以分开独立进行测试，但需在测试计划中预留集成测试的时间，可以在每次分支集成后排布半天时间每个测试人员进行核心场景功能的验证，避免集成时代码合并导致功能被覆盖。

### 4.1资源分配

测试资源分配可以从掌握的工具、技能、认证情况、项目经验、外部bug漏测数五大维度分析筛选：



### 4.2测试计划

#### 可接受性测试

* 如果有多个测试人员，安排多个测试人员进入可接受性测试
* 可接受性测试范围：圈定可接受性测试范围

如果是场景为主的需求，可接受性测试范围可圈定为主场景

如果是功能为主的需求，可接受性测试范围可圈定为功能模块不报错，功能操作可正常操作

如果是数据为主的需求，可接受性测试范围可圈定为主要数据取数正确（脚本不报错和数据总条数正确）

* 可接受性测试打回：依据可接受性测试打回标准
* 可接受性测试时间制定

#### 功能测试

* 功能测试优先级

确定功能测试的优先级，有些测试范围需要有前期进入，可以控制后期质量问题。根据发包优先级规划、流程上下游来确定，规划需要优先发包则先测，属于上游的功能先测（否则下游先测的话做前置数据受阻）；如：某模块有性能问题，且性能问题影响到整个架构，需要重新调整方案，如果后期才发现这个BUG,测试进度，周期都会受到严重的影响。需要将此内容放在前期测试，部分功能的性能测试可以并到功能测试中执行，如批量操作的性能，场景级或系统级性能建议单立子项做

* 功能测试重难点

复杂的场景，算法，取数要着重测试，可以依据对应的测试用例来验证

* 功能测试时间制定

需要制定功能测试时间周期，功能测试包括功能验证以及BUG验证。时间周期确定后可以通过关键节点来把控测试的进度

* 多个测试人员互相协作，模块的分配

多个测试人员协作测试，需要将每个测试人员细化到对应需求点，分配时要注意同一种类型的分配到一个测试人员身上，最好不要交叉导致重复测试。如果业务有关联，需要在集成测试阶段测试每个测试人员关联的业务，

* 功能测试范围

根据测试需求分析，分析出哪些范围需要在功能测试中进行

#### 系统测试

* 系统测试时间制定

确定明确的系统测试进入时间点，出来时间点，通过关键节点来把控测试进度

* 系统测试策略

依据需求业务，如果需求较复杂，可增加系统测试用例，这个阶段可根据系统测试且例进行测试；如果需求较简单，依据测试用例和功能规格设计说明书进行测试，并进行发散测试

* 系统测试策略

根据测试需求分析，分析出哪些范围需要在系统测试中进行

#### 回归测试

* 回归测试时间制定
* 回归测试范围

回归测试需要测试哪些内容

### 4.3风险预估

1. 识别风险 ：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **风险类别** | **子类别** | **风险项** |
| 1 | 需求风险 | 需求定义类 | 1. 需求范围定义不够明确，在进一步详细定义的情况下可能会超出范围 2. 在制定需求的过程中，客户参与不足，导致后期需求不稳，甚至交付时不满足需求 3. 需求定义模糊，难以理解工作困难 |
| 需求变更 | 1. 需求已经成为项目基准，但仍然会继续变化 2. 在已经确定的需求基础上增加额外的需求 3. 需求的变更未进行规范管理 |
| 2 | 进度风险 | 工作量估算 | 1、需求描述模糊部分，工作量较难估算，耗时较长使项目进度推延  2、估算资源、现有资源和产品资源不完全一致  3、估算结果不符合实际情况  4、对需求描述的领域不熟悉，导致估算的工作量不准确  5、未采用标准的估算方法，使估算的工作量不准确  6、错误的估计了辅助性工具对项目的助力，导致估算工作量不准确 |
| 计划编制类 | 1、项目预算可以被维持的时间不确定  2、预算得不到实际的保证  3、软件总体规模(代码行数、功能点、与前一产品规模的百分比)比预计的规模大  4、项目目标提前完成，但是没有相应的调整产品范围或可用资源，使资源可能会出现空置状态  5、计划仓促计划和任务定义不够充分，可能带来进度方面的问题  6、任务之间的关联关系未整理，不够明确  7、计划不够完整，未覆盖全部过程 |
| 项目进度类 | 1、项目进度由于某些原因不能够被维持，导致项目不能按时交付  2、项目进度由于某些原因不能继续进行，需要暂停或放弃进行  3、项目进度改变，要求提前完成任务等。 |
| 3 | 质量风险 | 产品质量类 | 1. 开发过程中，基础版本资源升级，需要进行功能移植，从而需要额外的工作量 2. 质量较差，缺陷较多，可能会出现返工的情况 3. 产品的性能是否可以满足需求和符合使用目的不确定 4. 开发了额外的功能，可能影响到项目进度 |
| 4 | 技术风险 | 规格设计类 | 1. 设计的质量较差，缺乏实现的可行性，不能很好的满足需求，需要重复设计，可能会影响到产品质量和项目进度 2. 设计的实现方法不符合实际情况，或者不能使用现有的代码和库来实现 3. 设计出现遗漏，需要重新设计补全 |
| 功能开发类 | 1. 开发环境未准备完成，例如：设施未到位、开发工具未准备完成、开发工具可用性较差 2. 基础版本资源的质量较差，直接影响到产品质量 3. 开发开始或中途使用新的开发工具，需要对新工具进行培训学习等 4. 需求和设计同开发人员之间的信息传递不够清晰明确，使开发的功能不满足需求 |
| 产品测试类 | 1. 测试准备不足，测试不能达到质量要求，例如：未编写测试大纲等 2. 测试时间不足，测试匆忙，不够充分，可能出现漏测的情况 |
| 5 | 管理风险 | 组织协作类 | 1. 实施的任务提交延迟，计划时间延长 2. 管理层的审查决策工作进度缓慢，计划时间延长 3. 预算削减，打乱项目计划 |
| 人员管理类 | 1. 组内成员流失，甚至导致项目进度停滞 2. 组内成员是新员工，工作经验不足，对工作环境不够了解，需要培训或磨合，对其工作量的评估较难把握 3. 组内个别成员能力欠佳，其工作成果需要投入较多工作量进行验证 4. 人力资源紧缺，临时任务较多，资源不足 5. 项目团队对业务了解程度不足 6. 团队成员间关系不佳，导致沟通不畅，接口出现问题 |
| 6 | 外部风险 | 客户 | 1、客户不满意交付的产品，要求重新设计和开发  2、客户的意见未被采纳，造成交付时不满足客户需求  3、客户对需求的提出、规划、规格决策的时间过长，导致计划延迟  4、产品交付后，客户回复时间过长 |

1. 分析风险产生的影响

|  |  |  |
| --- | --- | --- |
| **序号** | **级别** | **级别定义** |
| 1 | 1级 | 影响范围较小，不会影响项目关键节点的完成时间或者项目中开发bug加权数和开发工作量相等 |
| 2 | 2级 | 对项目影响较大，会影响项目关键节点的完成时间或者项目中开发bug加权数是开发工作量的1.2倍，但是可通过调整局部计划或动用储备资源，保证项目进度和质量不受影响 |
| 3 | 3级 | 严重影响项目且对项目关键节点有不可控的影响或者项目中开发bug加权数是开发工作量的1.5倍，严重滞后进度的，或项目不能使用 |

1. 根据风险项定位风险产生的原因
2. 制定风险应对措施，如：
   * 为规避开发质量风险，可以和开发约定，提供自测报告，必要时提供用例设计指导或者直接提供冒烟用例给开发

* 为规避变更频繁带来质量的风险，可以与PM沟通，进行需求控制，或者裁剪部分低优先级需求。
* 为规避测试质量产生的风险，可以申请测试中台或者资深测试参与评审测试用例；对于多人的工作安排，多进行互相审查，如：交叉回归
* 为规避流程风险，可以制定标准文档和流程规范，建立约束机

1. 风险解除后可关闭风险。

# 计划跟进及质量管控

## 计划跟进

1. 测试LD和开发LD每天跟进开发的计划，倒逼上游按规定提测时间达成，审视开发进度是否对测试有影响，如有风险，及时向PM、QA报风险；
2. 测试人员每日在日报中主动上报自己的进度、风险；测试LD每天审视测试进度是否按时达成，及时暴露进度风险
3. 设置“里程碑节点”

每个项目从开始要编写开发和测试计划，相应的在测试计划中也会对应每个里程碑，可以在项目进度的25%、50%、75%、100%的时候设置高等级风险卡点，进行阶段性审查

## 质量跟进

### 5.2.1 Bug分析：

1. bug趋势图：根据专项每日新增bug数的趋势图，只有bug呈收敛趋势时，需求包的质量才是可发包的状态，如果bug趋势为上涨趋势，证明最后收尾时的质量并不可靠（修复bug产生更多bug）。

建议措施：再进行一轮回归测试/bug分析，以免漏测。

1. bug等级占比，当1级bug的占比大于20%时，则开发质量差。

建议措施：需要推动开发深入分析，然后推动开发自测、暴露风险给PM，让PM协调高级开发人员协助解决

1. 开发bug解决情况：反馈专项中每位开发的bug解决情况，与前一日对比，当日解决bug数，以及剩余待解决bug数。

建议措施：如果有开发人员解决bug时间长，bug数量多现象，暴露风险给PM，让PM协调高级开发人员协助解决

1. 以上所有通过bug分析出来的现象，给出的建议措施，需要第二天跟PM确定实际解决措施和责任人，并在第二天的测试日报中反馈解决情况和进度

### 5.2.2问题跟进：

1. 测试过程中遇到的所有妨碍测试工作的问题以及疑问，都可以在日报中反馈，并同时寻求解决，在第二天编写日报中，回溯解决情况，以免遗忘
2. 测试建议/风险：若在测试过程中发现任何资源，环境，技术，工具等风险，及时暴露出来，并沉淀在日报中，第二天日报时可复盘前一日的风险是否有关闭

5.2.3测试报告

编写人员：专项测试leader 编写时间：测试介入后每日 收件人：专项相关干系人（PM、设计、开发、测试） 抄送人：部门经理、QA、一线

5.2.4测试总结

1. 本期遗留问题：专项中因为各种原因和客户沟通一致，可以延期处理的bug，方便追溯以及作为项目复盘的数据。
2. 测试总结：
3. 1级bug数，2级bug数，3级bug数，4级bug数

目的：体现本专项的整体开发质量，通过1级bug数和4级bug数，可分析开发的能力以及态度问题，供后续项目复盘做数据支撑。

1. bug分布模块

目的：根据导出的bug，对bug进行模块划分

建议措施：一线根据bug分布模块，制定验收测试策略，针对bug多的模块，可重点验收。

1. 冒烟bug数，冒烟不通过次数

目的：体现开发自测质量

建议措施：自测质量不过关需要给出详细自测用例以及让开发leader/设计验收自测结果

1. 开发每日bug量：内部bug/开发工作量（大于2，则标红）

目的：体现开发质量

建议措施：若开发质量差，则可以推进开发主控针对质量差的开发进行bug分析和复盘，推动团队整体质量的改善

1. 设计bug数

目的：体现设计质量

建议措施：若设计质量差，则可以推进设计主控针对质量差的开发进行bug分析和复盘

1. 验收（一线联测验收）bug数，验收bug漏测根因，改进措施

目的：体现测试质量

建议措施：若验收bug数多，需要深入分析，分析根因以及给出可落地的解决措施。