授客细说场景测试用例设计与实践

by:授客 QQ：1033553122

测试是一种思想，短视者把工具当目的，远视者把工具当手段.....

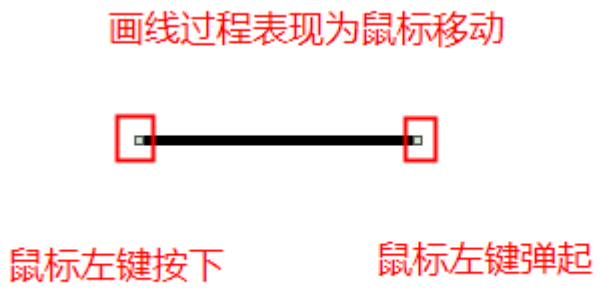
软件设计

1）单个用户操作 -> 触发单个事件 -> 事件处理

2）按顺序执行多个用户操作 -> 按顺序触发多个事件，形成事件流

注：通常事件是操作触发的，和操作往往是一一对应的关系，所以，这里为了便于理解，暂且把“操作”名称，称为事件名。

举例：在 windows 画图榜中画线为例



这里简单说，触发了三个事件：鼠标左键按下，鼠标左键弹起，鼠标移动。

事件处理：

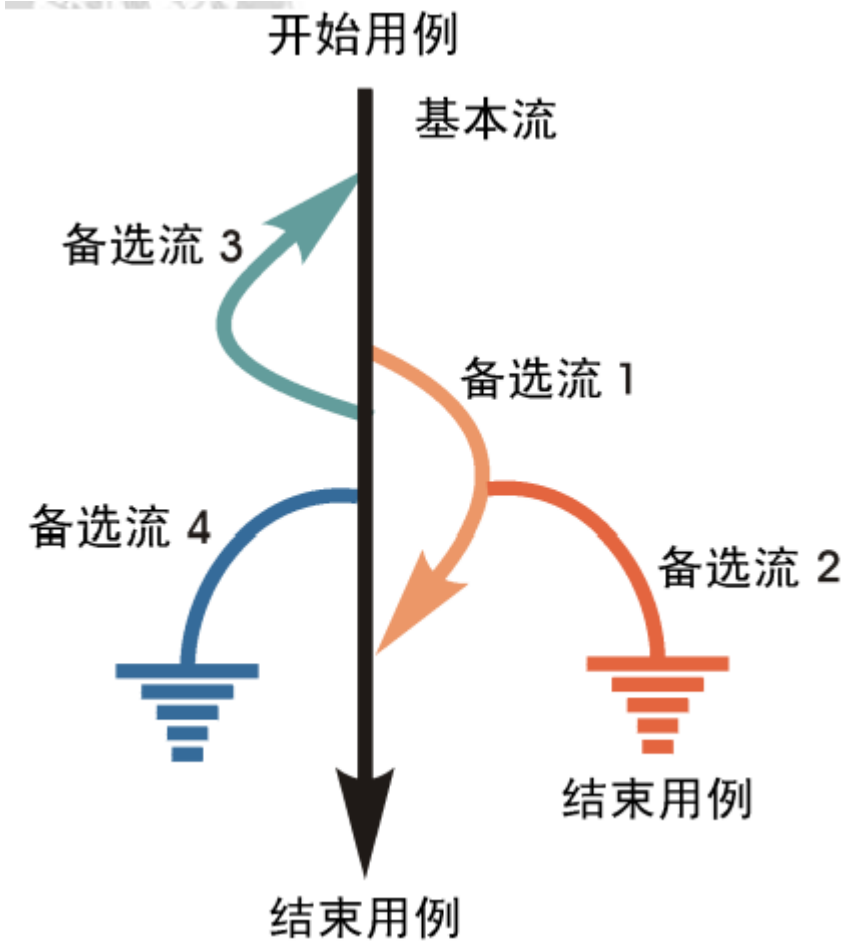
鼠标左键按下时，用两个不同名称的变量保存鼠标的点击点，作为直线的起点和终点；

鼠标移动时，不断用最新的鼠标点代替线条中线条终点，并擦除之前画的线条；

鼠标左键弹起时，保存最后一个点作为直线终点，并画线。

当然，我们可以稍微宏光的把多个“较小”的用户操作整合为一个“较大”的用户操作，比如上述的三个操作（按下鼠标左键，移动鼠标，松开鼠标左键），可以整合为一个操作--“画线”。

借鉴软件设计的思想，引进“按场景设计用例”的思想



基本流用黑色表示，是经过用例的最简单的路径。
备注：个人理解，这个称为“最主要”的路径会比较合适，具体理由见下文说明

备选流用不同的彩色表示，一个备选流可能从基本流开始，在某个特定条件下执行，然后重新加入基本流中（如备选流 1 和 3）；也可能起源于另一个备选流（如备选流 2），或者终止用例而不再重新加入到某个流（如备选流 2 和 4）

什么叫场景

通俗的将，场景为用户活动和活动环境的结合。
其中，用户活动通常是由一系列操作组成，活动环境则通常是操作时的软、硬件环境。

-> 按场景来设计用例，其实就是设计不同系列的操作，按顺序去触发每个系列的操作，查看其结果是否和预期保持一直。

问题来了，那么多用户操作，每个系列的操作要怎么安排？？

设计思想：

产品是给用户使用的 -> 用户是怎么使用产品的？ -> 操作产品：
1）如果顺利完成操作，产品功能好

2）如果不能完成操作，产品功能差

-> 测试人员要模拟用户操作 -> 用户怎么操作的？：
1）用户会按模拟的那样，操作产品(不管是有意还是无意)，测试投入有价值
2）用户永远不会那么操作，测试投入约等于无价值。

-> 按优先级模拟操作：优先模拟用户最有可能的系列操作，即用户场景，然后模拟次可能的场景操作。

注意：不管是不是按场景设计用例，这也是作为用例优先级安排的一条最最基本的原则。

看完了似乎还是没解决怎么安排的问题，对吧
烦先看文章“细说软件产品和业务 & 业务过程(流程) & 业务逻辑”

看完了文章，可以容易得出
1）业务逻辑之业务过程是用户最有可能执行的场景操作 -- 应设计为 基本流
2）业务逻辑之业务规则是用户次有可能执行的场景操作 -- 应设计为 被选流

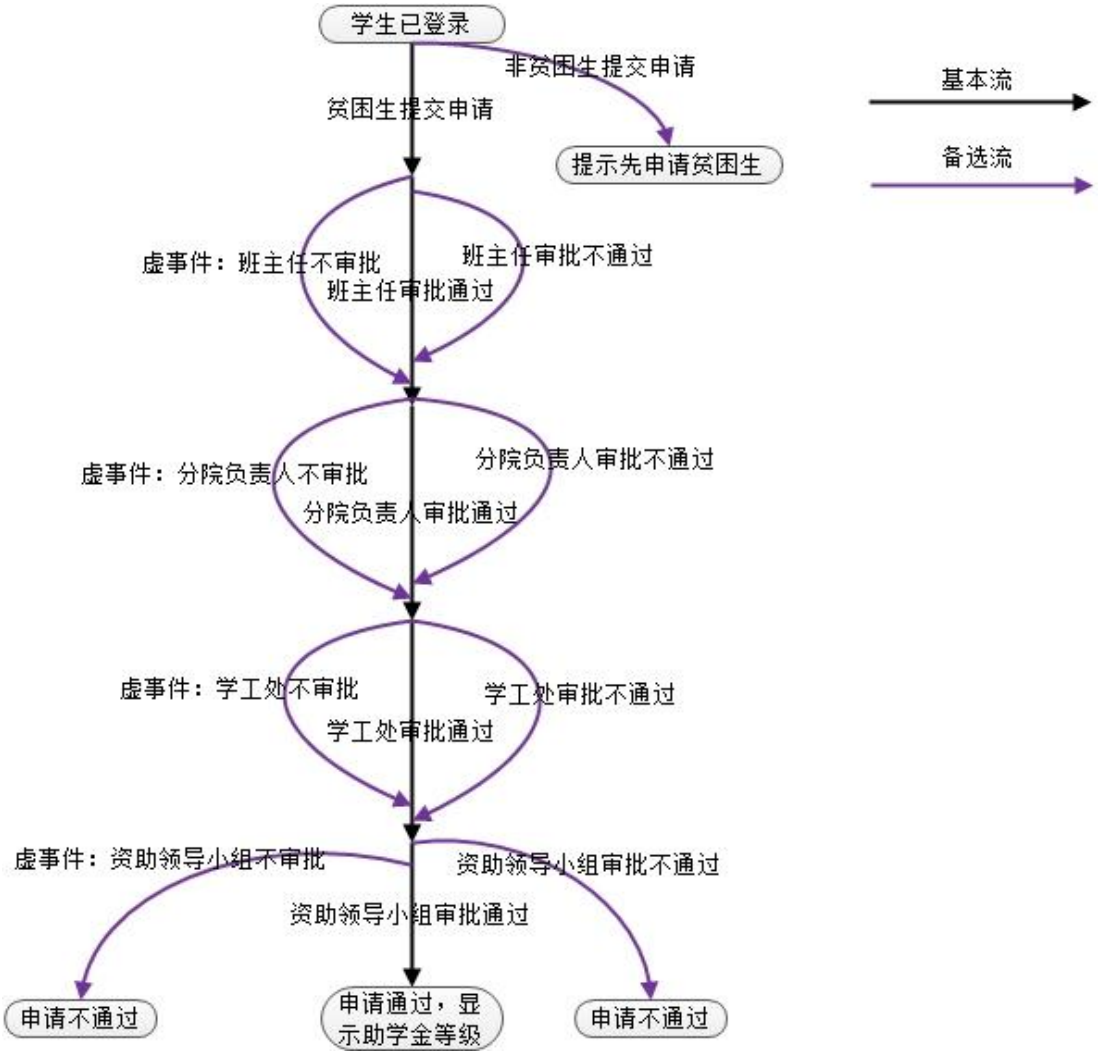
注：以上这种对应关系仅是大致思想，基本是那样，并不绝对。

设计实践
1.绘制事件流景
2.描述事件流
3.用例设计

例子：以学校学生申请助学金为例子
业务过程：
学生申请助学金 -> 班主任审批 -> 分院负责人审批 -> 学工处审批 ->资助领导小组审批

附加说明：
审批时可选择助学金等级：1 等，2 等，3 等
1.班主任仅可见其管理班级的学生提交的申请表
2.分院负责人仅可见其管理院系的学生提交的申请表
3.学工处和资领小组审批可见所有的申请表
4.职位较低的审批人拒绝或不审批，不影响较高职位的人对申请进行审批，如果他有权限的话

绘制事件流图



特别说明：

1.如图，为了画图和事件流描述方便、易于理解，我们可以增加“虚事件”--不需要实际操作去触发的事件，之所以说是虚事件，因为没有用户、系统提供实际操作，就不会产生事件。

2.如图，为了便于理解，通常把“事件流”拆分成一个一个事件(过程中，某个过程节点上的主选事件和备选事件，分别用不通颜色代替)，也就是说上面每根带箭头的线条，宏光上仅代表一个事件，所谓的事件流是由这些事件按一定顺序触发后才形成的。

描述事件流

推荐书写格式

场景名称(描述这一整个事件流为了完成什么事情？目的)

事件 1

事件 2

...

事件 N

基本流

1	**贫困生申请助学金通过**
1.01	贫困生提交申请
1.02	班主任审批通过
1.03	分院负责人审批通过
1.04	学工处审批通过
1.05	资助领导小组审批通

备选流

1	**非贫困生申请助学金**
1.01	非贫困生提交助学金申请
2	**贫困生申请助学金，班主任审批不通过**
2.01	贫困生提交申请
2.02	班主任审批不通过
2.03	虚事件：分院负责人不审批
2.04	虚事件：分院负责人不审批
2.05	虚事件：资助领导小组不审批
3	**贫困生申请助学金，分院负责人审批不通过**
3.01	贫困生提交助学金申请
3.02	班主任审批通过
3.03	分院负责人审批不通过
3.04	虚事件：分院负责人不审批
3.05	虚事件：资助领导小组不审批
4	**贫困生申请助学金，学工处审批不通过**
4.01	贫困生提交助学金申请
4.02	班主任审批通过
4.03	分院负责人审批通过
4.04	学工处审批通过
4.05	虚事件：资助领导小组不审批
5	**贫困生申请助学金，资助领导小组审批不通过**
5.01	贫困生提交助学金申请
5.02	班主任审批通过
5.03	分院负责人审批通过
5.04	学工处审批通过
5.05	资助领导小组审批不通过
6	**贫困生申请助学金，班主任审批不通过，学工处审批通过，资助领导小组审批不通过**
6.01	贫困生提交助学金申请
6.02	班主任审批通过
6.03	虚事件：分院负责人不审批
6.04	学工处审批通过
6.05	资助领导小组审批不通过
7	**贫困生申请助学金，班主任审批不通过，学工处审批通过，资助领导小组审批通过**
7.01	贫困生提交助学金申请
7.02	班主任审批通过
7.03	虚事件：分院负责人不审批
7.04	学工处审批通过
7.05	资助领导小组审批通过
8	**贫困生申请助学金，班主任审批不通过，学工处不审批，资助领导小组审批通过**

8.01 贫困生提交助学金申请
8.02 班主任审批通过
8.03 虚事件：分院负责人不审批
8.04 虚事件：学工处不审批
8.05 资助领导小组审批通过

9 **贫困生申请助学金，班主任审批不通过，学工处不审批，资助领导小组审批不通过**
8.01 贫困生提交助学金申请
8.02 班主任审批通过
8.03 虚事件：分院负责人不审批
8.04 虚事件：学工处不审批
8.05 资助领导小组审批不通过

10 **贫困生申请助学金，学院负责人审批不通过，学工处审批通过，资助领导小组审批不通过**
10.01 贫困生提交助学金申请
10.02 班主任审批通过
10.03 分院负责人不通过
10.04 学工处审批通过
10.05 资助领导小组审批不通过

11 **贫困生申请助学金，学院负责人审批不通过，学工处审批通过，资助领导小组审批通过**
11.01 贫困生提交助学金申请
11.02 班主任审批通过
11.03 分院负责人不通过
11.04 学工处审批通过
11.05 资助领导小组审批通过

12 **贫困生申请助学金，学院负责人审批不通过，学工处不审批，资助领导小组审批通过**
12.01 贫困生提交助学金申请
12.02 班主任审批通过
12.03 分院负责人不通过
12.04 虚事件：学工处不审批
12.05 资助领导小组审批通过

13 **贫困生申请助学金，学院负责人审批不通过，学工处不审批，资助领导小组审批不通过**
13.01 贫困生提交助学金申请
13.02 班主任审批通过
13.03 分院负责人不通过
13.04 虚事件：学工处不审批
13.05 资助领导小组审批不通过

用例设计

通常情况下，可以把每个场景当作一条用例。这里需要注意的是，这里的事件流侧重事件触发逻辑顺序，设计用例时，还要注意测试数据(按我的观点，测试逻辑和测试数据一般是要分开的)。

根据上述例子中的附加说明，每条用例可能有多条测试数据。因为审批过程中是可修改助学金等级的，这个很重要，所以要测试不同等级的审批结果。

适用范围：

通常，按场景设计用例，比较适合流程性比较强的测试，比如业务测试。

当然，这种思想，也可以应用用在局部功能的测试上，具体参见文章“测试用例设计实践总结”描述中，其核心思想和这个场景测试是差不多的。