测试基础分享-用例

张敬峰
2014-8-27

- 一个好case具有的特征：
- 一：清晰简洁的格式。
- 二：合乎逻辑的流程。
- 二：完美的覆盖度。
- 三：良好的可执行性。
- 四：良好的可维护性。

- 用例设计的五个步骤：
- 一：格式
- 二：分析需求文档
- 三：模块划分
- 四：编写用例
- 五：整理与维护

第一部分：格式

- 一个清晰统一的格式为何重要?
- 1，让case的脉络清晰明了。
- 2，方便整个团队的认知与理解。
- 3，方便需求变化后的更新维护。
- 4，方便执行人员快速入手。

1.1首页

- 首页应该包含哪些内容?
- 1，case 名称
- 2，case 对应的游戏版本
- 3，编写人
- 4，编写日期
- 5，编写备注
- 6，修改人
- 7，修改日期
- 8，修改备注
- 9，spec的链接（地址或jira号，方便其他测试人员查看需求）

例子：

Case Name	Create version	Creater	Update data	Comment	Spec
city&field view combine	4.1.0	Zhang Jingfeng	8/18/2014		https://godzilla.
translation of keys	4.1.0	Zhang Jingfeng	8/19/2014		https://godzilla.

Case Name	Update Version	Update Person	Update data	Comment	Spec
city&field view combine	4.1.1	Zhang Jingfeng	8/18/2014	需求变更，增加新建筑	
	4.1.2	Zhang Jingfeng	8/19/2014	需求变更，增加新建筑	
	4.1.3	Zhang Jingfeng	8/20/2014	需求变更，增加新建筑	

Case Name	Update Version	Update Person	Update data	Comment	Spec
translation of keys	4.1.1	Zhang Jingfeng	8/18/2014	需求变更，增加新字符串	
	4.1.2	Zhang Jingfeng	8/19/2014	需求变更，增加新字符串	
	4.1.3	Zhang Jingfeng	8/20/2014	需求变更，增加新字符串	

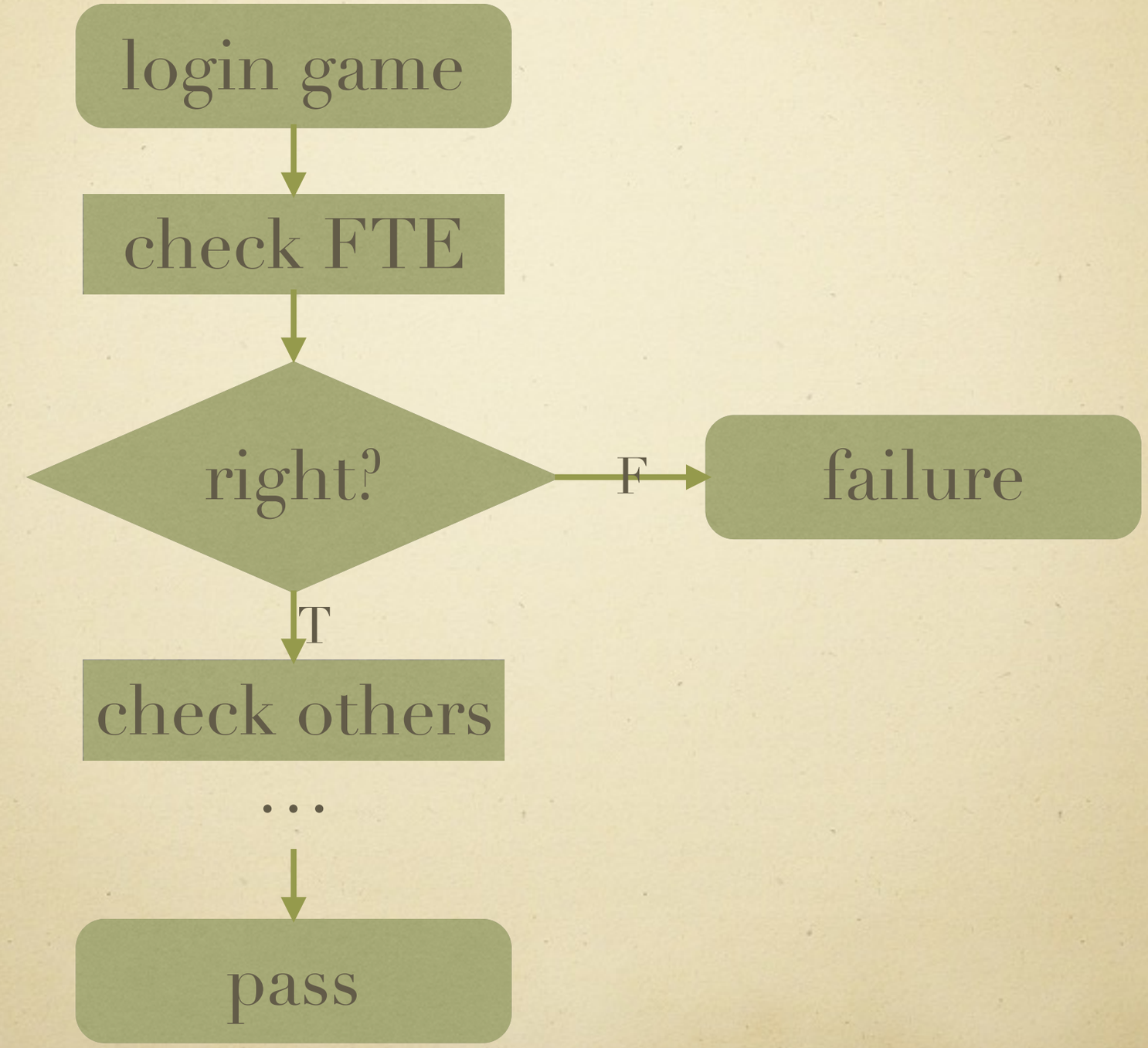
1.2 正文页

- 正文页应该包含的内容:
- 1，功能流程图（如果有）
- 2，写case时的思考（如果有）
- 3，模块id
- 4，模块名称
- 5，测试先决条件（或先决数据）
- 6，输入
- 7，输出
- 8，备注栏

例子：

ID	Case Name	Data	Steps	Expected	Comment	Result
功能主要针对去掉资源视图带来的影响：FTE, building&building list, city view, button, translation, compatibility						
VC-01	change of FTE	login game with a new account	1.check the FTE of change view steps	will delete the step of change view between city and field		
			2.check the FTE of field view steps	will delete the step of field view step		
			3.kill the game when the FTE not finish, login game again ,check the change view step and field steps	the change view steps and field steps will be deleted		
			4, complete the FTE step by step	will complete success		
VC-02	button in left lower	check the change view button	1.check the change view button in left lower	will change to 2 status from 3, delete the field status		
			2.click the button	will change view between city and map		
			3.check the icon of the button	will change correctly(just have 2 icons:map and city)		
			4.check the mini FTE of field view	will not appear		
			5.click the button in each dragon island view	will change to city view		
			6.click the button in each dragon keep view	will change to city view		

一个简易的流程图：



1.3关于格式的一些思考

- 1，必须保证逻辑清晰
- 3，尽量保证一个输入对应一个输出
- 4，尽量保证较高的可维护性
- 5，一个case内尽量保持格式一致

第二部分：需求文档分析

- 怎样开始分析文档?
- 1，详细阅读至少3遍。
- 2，逻辑梳理并勾勒出功能的大概流程图。
- 3，与产品探讨表述不清的地方。
- 4，细化流程图。
- 5，考虑正常流程可能存在的设计缺陷。
- 6，考虑正常流程中的测试难点。
- 7，考虑与其他功能的关联。
- 8，考虑非正常流程（特殊情况）的影响。
- 9，考虑版本数据兼容。

2.1 文档阅读

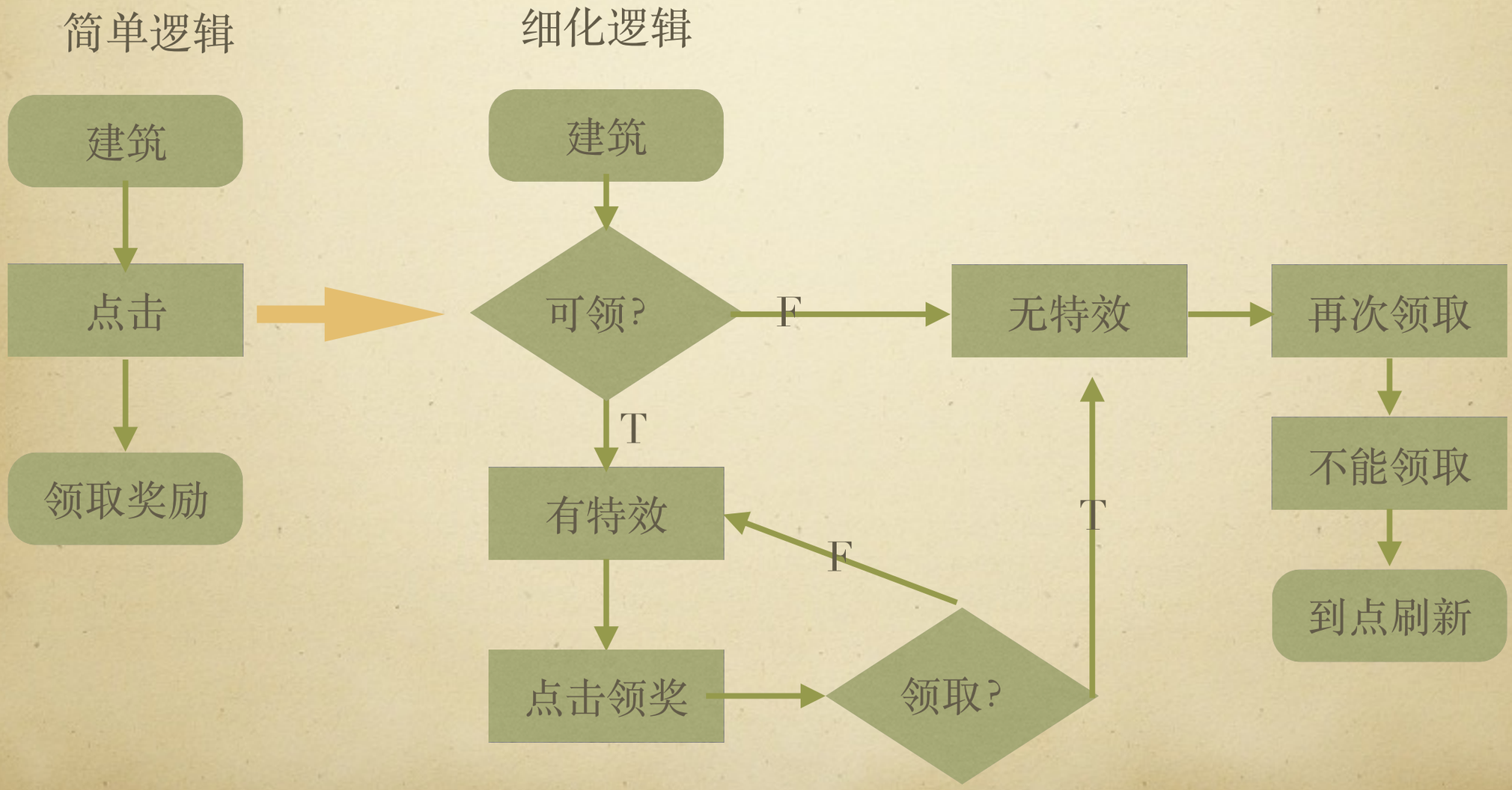
- 1，切忌不读文档，上来直接写。
- 2，我们需要理解产品的设计意图和设计思路。
- 3，避免粗略理解带来的测试用例遗漏问题。
- 4，一些关键设计可能隐藏在不起眼的语句中。
- 5，带着思考去阅读，如果让你设计这个功能，还有没有更好的实现方式？功能还有没有优化的空间？
- 6，加深对功能的理解，否则过段时间，我们自己都会忘记自己负责的功能细节。
- 7，只有理解充分，才能就功能问题与产品和开发探讨，才能给别的测试人员讲解这个功能。否则自己一知半解，跟别人探讨时会底气不足。

2.2 功能沟通探讨

- 1，不明白的地方需要要及时确认，切忌想当然。
- 2，功能确认最好要拉上相关的开发与产品一起。
- 3，尽量早确认，最好是case开始的时候就所有细节确认完毕。
- 4，关注需求变更，需求变更后一定也要与开发和产品确认。
- 5，关注时间，要能够根据功能的大小，复杂度，预估测试需要的时间。

2.3 逻辑梳理

- 文档的不一定是按照流程顺序书写的，而且经常存在功能交叉的情况。
- 简单例子：在城市里增加一个建筑，玩家每天可以领取免费的道具。可领取状态时建筑会有闪光特效



2.4 拓展思考

- 还以上面的例子来举例:
- 1，设计缺陷思考：既然是建筑，就会涉及到是否可以建造、升级、拆除等问题，上面的例子没有给出具体说明，需要让产品将细节写明。还有领取道具的种类、数量有没有限制等问题也需要设定明确。还有开关也需要考虑。
- 2，测试难点思考：如果每天刷新一次，那测试需要让程序给出调节刷新时间的方法，避免长时间等待。
- 3，关联度思考：需要考虑领取的道具在背包里的显示。
- 4，非正常流程的思考：网络，重启，背包上限，快速点击，切换服务器等。

2.5 兼容相关的思考

- 兼容基本包含以下几点：
- 1，数据兼容。主要涉及到数据变化时需要考虑，比如龙的数值修改对已有不同等级的龙的影响。
- 2，版本兼容。主要涉及新旧版本兼容，比如后端更新后，前端还没开启强升前的这段时间内同时存在新旧2个客户端版本的情况。
- 3，功能兼容。主要老功能添加新类型时需要考虑。比如新加一个tournament,新加一个兵种等。
- 4，操作系统版本兼容。主要考虑游戏能够在当前主流的几个安卓和ios版本上正常运行。
- 5，屏幕分辨率兼容。主要考虑游戏能够在主流机型的屏幕分辨率下正常运行。

第三部分：模块划分

- 划分模块时需要考虑的两个问题：
  - 1，划分模块需要遵循什么样的原则？
  - 2，用什么方法去划分？

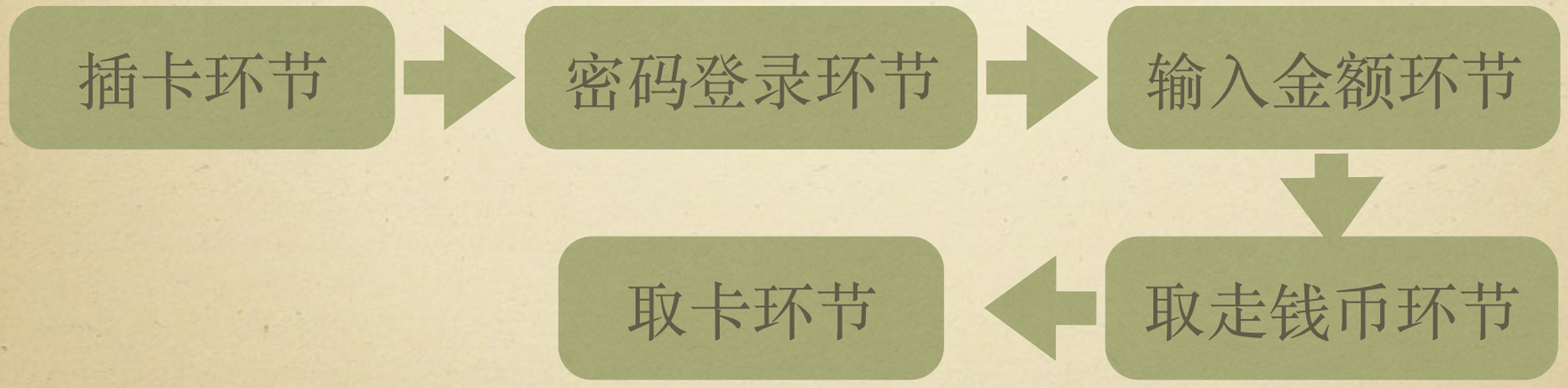
3.1 模块划分原则

- 1，高内聚，低耦合。高内聚，指的是单一模块划分出来后，模块内部的功能之间需要有紧密的关联度，假如进行分拆，分拆后很难作为单独的模块存在。低耦合，指的是模块与模块之间的关联度要低。举例：就ruby mine的续费功能进行子模块划分，240档和1600档应该划分成2个子模块，因为他们之间几乎没有关联度，符合低耦合的原则。240档内部没必要继续划分成ui和功能2个模块了，因为关联度太高，这符合高内聚的原则。

- 2，重整体，轻局部。重整体，指的是要有大局观，从功能整体的层面上去关注模块构成，去关注模块的流程，逻辑，覆盖。轻局部，指的是划分模块是不要纠结于具体的功能细节，从而导致模块划分不清晰，细节可以在模块划分完毕后的细化测试用例的步骤中进行具体分析。举例：ruby mine的模块划分，从整体上来看，应该划分为UI,购买与领取，续费，特殊情况4个大模块。然后再对每个大模块进行子模块的划分。此时无需关注于倒计时的显示位置等具体的细节问题。

3.2 模块划分方法（一）

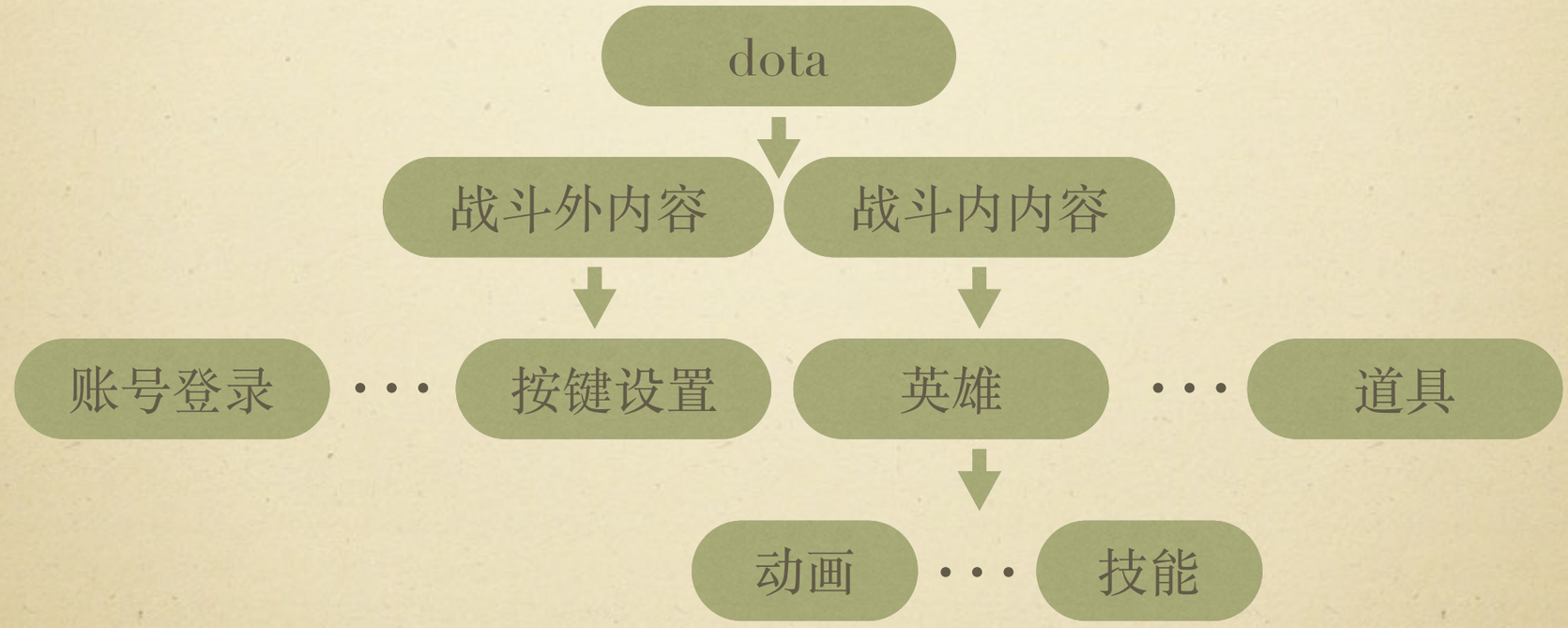
- **功能流程法**：将功能的基本流程画出来，根据流程的每个大的环节进行模块划分，然后再细化和查漏补缺。
- 举例：请就银行ATM的取款功能进行模块划分？
- 这种类型的问题就非常适合用功能流程法进行划分，我们来看下取款流程：



- 基本上我们就可以根据上面流程组成的几个部分划分成几个模块。

3.3 模块划分方法（二）

- **层次划分法**：按照逻辑层次逐层细化出模块的过程，比较适用于UI划分，大的系统模块划分等。
- 举例：请就dota这款游戏进行模块划分。



- 从上面的例子我们可以看出从大到小的层次关系，逐层划分的好处是逻辑非常清晰，不容易遗漏测试点。

3.4 模块划分方法（三）

- 类型划分法：按照功能包含内容的不同种类进行划分。
- 举例：兵种测试，龙测试等。兵种测试：可以划分为：可训练兵种 + 不可训练兵种。龙测试可以划分为1，2，3，4，5龙。
- 类型划分法比较适用于一个功能种类相对独立，种类之间关联度较低的情况。

3.4 模块划分的注意点

- 1，不同的方法适应不同的情况，要具体问题具体分析。
- 2，有时候一个功能要结合不同的方法进行划分。
- 3，也许还有很多别的方法，划分时可随意选用，划分方法不重要，划分原则很重要。
- 4，划分完毕后要根据需求文档重新梳理下，确保模块清晰，覆盖完整。

第四部分：用例编写方法

- 常用的几种测试用例编写方法：
- 1，边界值
- 2，等价类
- 3，判定表（参考：http://blog.csdn.net/xc5683/article/details/8125950）
- 4，其它（参考：http://blog.csdn.net/xc5683/article/details/8129955）
- 判定表和其它的方法用的不多，也很难在一页ppt内讲清楚，给大家发了网上讲的不错的例子，有兴趣可自己看看。

4.1 边界值

- 对输入或输出的边界值进行分析的一种方法。
- 首先需要确定边界值：一般选取正好等于，刚刚小于和刚刚大于3种情况作为测试数据。
- 通常适用的范畴：数值测试，字符串测试，数据类型测试等
- 举例：ELO值的测试。ELO值设计范围是2000-4000，那么我们需要对边界值测试，测试取值：4001，4000，3999，2001，2000，1999

4.2 等价类

- 等价类：指的是一个输入集合内，任何输入数据对于输出的验证来讲都是等效的，此时我们就可以选取少量代表性的测试数据来代表整体数据。

- 等价类分为2种：一种是有效等价类，指的是对输出来讲有意义的输入集合，可以验证程序的正常功能和流程。另一种是无效等价类，指的是对输出无意义的输入组合，用于验证非正常流程输入对输出的影响。

- 通常适用范围：大部分功能

- 举例：比如在地格上造建筑，我们一般不会把所有地格都建满，建造几个都正常我们就认为这一类是正确的。另一个例子是去攻打野外地块，我们也不会把所有野外地格都攻打一遍，大概挑几个不同的等级攻打一下没问题就认为这一类的功能没问题。

4.3 用例书写的注意点

- 1，输入条件要单一明确，尽量不要出现容易引起误解的词汇（如，可能，大概，也许等）。

- 2，输出（预期结果）要可判断且明确，比如“输出正确”，“显示正确”这种其实是不太好的，最好说清楚具体显示或输出的内容。复杂且混乱的输出会让这条case无法执行。

- 3，测试步骤要可执行。

- 4，能抽象的尽量抽象，避免无意义的冗余。

- 5，覆盖度一定要尽量高。

第五部分：整理与维护

- 1，需求变化后需要即使更新老的测试用例，并写清修改情况的备注（修改内容，产品和开发负责人）

- 2，测试用例应该尽量避免冗余，如果遇到重复的用例，需要根据实际情况进行修改。

- 3，注意测试用例的备份，写完后最好自己本地也备份一份，避免线上被人误删除。