课程介绍

* **为什么要做需求评审？**
* **需求评审时测试人员做什么？**
* **需求评审之实战演练**
* **需求评审之隐性需求**

1. 为什么要做需求评审

需求评审会听上去挺高大上，但对许多产品来说简直就是噩梦.

**为什么要做需求评审？**

**一、让团队中所有人知道我们这个需求的背景和目的**

无论什么职位都需要一个归属感，而不是一个接收任务和完成任务的工具，让团队中每个人都了解一件事情的来龙去脉更有利于这件事上的落实和达成共识。

**二、评估设计、开发、测试的周期，便于做决策**

做需求评审会时产品要带上需求文档、业务流程图、页面流程图、交互原型图，和团队的每个负责成员评估周期，如果评估结果时间超出预算时间，例如我们要配合运营在520当天上线一个活动，但以目前的人力无法按时上线，那么通过评估我们做出决策在当前情况下要和领导申请更多的资源。

**三、让参与者明确工作内容和交付时间**

这个就是在几次需求评审修订后最终定方案，团队中的每个人员都清楚接下来每个人的工作内容和需要阶段性的产出交付物。

1. 需求评审时测试人员做什么？

需求评审---测试人员参与项目的第一步。但是不少测试同学反馈，自己很少参与需求评审，需求会议也很少喊测试人员参与。

1. 流程上各角色配合的问题；

------主动找产品沟通，表达希望参与需求评审的意愿，要求他们在需求评审时叫上测试。

1. 测试在评审过程中没有体现出参与的价值；

------这就需要测试人员从自身上做改进了（测试就只是在那听，不能产生价值）。

作为测试，参与需求评审时我们可以贡献什么价值？

* 1. 需求评审的作用

回答上面的问题前，我们先看看需求评审到底是干嘛的？需求评审就两个作用：

* 同步产品对于需求的详细设计
* 收集大家对于需求的各种反馈

对于需求设计，肯定是产品发起并负责的了，那么作为测试人员参与需求评审，着重点就在于第二点，关于需求的反馈上面了。

* 1. 需求评审的形式
* 大家一起找个会议室，产品讲需求，开发和测试怼产品；
* 产品把需求往公司群里一扔，大家开始七嘴八舌的讨论；
* 产品直接跑过来，在开发和测试的座位上当面沟通一下。

其实这些都算需求评审，形式不重要，重要的是做这事的目的和效果。

* 1. 测试是否需要参与需求评审

仅仅从同步需求设计的角度看，当面的同步一下需求，肯定比文字上的传达效果要好的多了；

而最重要的其实还是测试在需求评审中提出的反馈，才是最宝贵的。

所以下面主要说一说测试对于需求反馈的价值主要都体现在哪些方面。

* 1. 需求评审之需求合理性

需求合理性，这是开发和测试怂产品最多的地方之一。

* 弹这么大个框，太打搅用户了吧？我建议缩小二分之一。
* 卸载个软件，还要确认这么多次，用户该烦了吧？我建议点击卸载按钮就完事。
* 首页内容已经很多了，再加一个会有效果么？是不是再精简点内容比较好？我建议一屏不超过 5 条内容。
* 这操作流程有点反人类呀，交互咋设计的呀？我建议主要操作一步即达，次要的三步以内完成。
* 根据用户手机壳的颜色改变App主题的颜色太鸡肋了吧？建议放弃。

虽然最后拍板可能还是产品是说了算，但是该提意见还是要提的，特别是有些地方其实产品也没有定论，这时候的意见非常有可能会被采纳，如果建议被采纳的次数多了，自己的建议就会更受大家重视，那么话语权也就会相应的有提升了。

然而，很多人其实是不会反驳需求合理性的，大不了就内心里吐吐槽「这么脑残的设计，亏你能想出来」，也许这是和公司环境有关系，但是如果自己真的有什么好的建议，还是建议找机会提出来，毕竟咱们是测试，用户体验的质量也是质量范畴内的事。

* 1. 需求评审之需求全面性

前面说的需求合理性，需要我们站在用户的角度去考虑问题，不是所有人都能做到，这也情有可原，但是需求全面性这个确实是需求评审中必须要考虑的问题，这个不仅仅针对产品设计，也包括开发实现逻辑。

* 如果用户登录超时了，产品怎么展现？
* 如果用户输入了非规定范围内的数据，逻辑上是否做了异常处理，怎么告知用户？
* 如果用户长时间不关机，逻辑上是否有问题，如何处理？
* 如果多用户同时登录，会出现啥问题？
* 如果系统休眠后恢复，产品如何处理？

针对这部分内容，大多是对于使用场景的覆盖，很多产品考虑需求时，只覆盖了常规用户的主要操作分支，而异常情况考虑的比较少，对于测试来说，异常场景的考虑正是我们的长处，所以在需求评审阶段尽可能多的和产品确认各种异常场景的处理，可以极大的避免在测试过程中出现问题后被返工的情况。

1. 需求评审之实战演练
   1. 需求描述

如果产品给出的需求描述是这样的：

现在有一个 PC 客户端的命令行工具，这个工具可以接收三个命令行参数，其中，前两个是数字，最后一个是运算符，运算符只支持加减乘除四种，工具的功能就是把前两个数字使用运算符做下运算，然后输出运算结果。

* 1. 模拟需求评审

下面是模拟针对这个需求的需求评审。

**先是需求合理性的讨论：**

测试：「命令行的计算器，干嘛用的，为啥不用系统自带的计算器？」

产品：「恩，目前是演示环节，先不用考虑使用者，请忽略这个问题。」

测试：「为啥是命令行工具？命令行的可控性太差，建议改成 GUI 实现。」

产品：「本次针对的是特定的 Geek 群体，习惯于命令行操作，而且市面上已经有很多 GUI 的实现工具了。」

测试：「前面两个是数字，最后是运算符，不太符合操作习惯，建议把运算符放中间。」

产品：「恩，这个我们回去考虑下。」

测试：「确定只需要支持加减乘除么？是不是功能太弱了？」

产品：「这是第一版迭代，后面会根据用户需求再酌情扩展，所以这地方开发记得别写死了。」

只是做了下简单的需求合理性讨论，就变更了一次需求---参数位置的问题，同时让开发在功能实现时提前考虑了可扩展性，这些问题如果是在测试阶段提出来，大部分的可能是先不动了，不然又得改代码，如果真的改，开发和测试的工作量都会相应增加，如果不改就会增加下次迭代时候的工作量，总之，早提出需求合理性讨论，有百利而无一害。

**接着是需求全面性的讨论：**

测试：「最大支持的运算数是多少？」

产品：「浮点型的最大值就行。」（懂技术的产品都是好产品。）

测试：「工具是每次运行后只做一次运算，还是一次运算结束可以继续接收新的参数输入？」

产品：「第一版不做太复杂，每次都需要重新执行，只接收直接执行时候的参数传入。」

测试：「三个参数之间用什么分隔？」

产品：「空格或逗号，两个都支持。」

测试：「这个得有个说明吧，不然用户会傻傻分不清。」

产品：「对，如果参数格式错误输出一个使用说明的提示。」

测试：「如果缺少参数提示什么错误信息呢？」

产品：「提示说，你输入的参数个数不正确，请按照 [运算数 运算符 运算数] 的格式输入。」

测试：「如果参数类型错误提示什么错误信息呢？」

产品：「提示说，你输入的参数类型不支持，请重新输入。」

测试：「这个提示不明确吧？参数类型不支持，那具体支持哪些类型呢？用户还是会懵逼呀。」

产品：「那改一下，你输入的参数类型不正确，运算数只支持浮点型，运算符中只支持+-\*/，分隔符支持空格和的逗号。」

测试：「如果除数为零，提示什么错误信息呢？」

产品：「提示说，你输入的除数为零，请重新输入。」

除了一个主分支的问题，其他的都属于旁支，旁支是对主分支的补充和完善，也是大家最容易忽视的地方，也是用户环境最容易出现问题的地方。

这么简单一个 if 语句就可以搞定的需求，竟然可以提出 12 个有效问题，如果这些是在测试过程中提出，考虑下每个问题从提出到产品确认，然后开发修复，然后测试验证，这过程的损耗有多大，而如果是在需求评审阶段提出的话，开发就可以完全按照既定的需求，提前考虑各种场景的处理，极大的减少了需求变化造成的沟通和返工成本。

以上是从一个简单的需求着手，说说关于需求评审的两个关注点，可以想象一下，如果是比较大的需求，测试要提出的问题会很多，那么就需要考虑一些策略的问题了，比如分批次进行评审，每一次评审确定下合理的颗粒度，方便大家聚焦，但是不管怎么说，测试参与需求评审的作用是很大的。

1. 需求评审之隐性需求

前面主要是对需求评审阶段要做的事情做了大概的说明，接下来，主要说一说需求评审过程中对隐形需求挖掘的重要性。

* 1. 关于原始需求的例子

我们先来看一个例子：

女儿：「爸爸，我想吃面条。」

父亲：「现在太晚了，饭店已经关门了，明天我带你去吃好不好？」

女儿：「不好不好，我就要吃面条。」

父亲：「你这孩子，这都 11 点了，哪还有面条？快给我睡觉去。」

女儿：「哇……」

脑补下宝宝大哭的画面。

来，我们现在换一个爸爸来对话：

女儿：「爸爸，我想吃面条。」

父亲：「今天太晚了，饭店已经关门了，明天我带你去吃好不好？」

女儿：「不好不好，我就要吃面条。」

父亲：「闺女是不是肚子饿了？我给你拿你最爱吃的面包好不好？」

女儿：「好呀好呀。」

虽然只是一句话的差别，但是得到的结果却是天壤之别。

上面例子中这句话如果按我们测试的行话说，就是发现了用户的隐性真实需求。

女儿是想吃面条，但是因为饿才想吃，而不是单纯的想吃，那么只要解决饿的问题就好了，她爱吃面条，同样也爱吃面包，所以就可以通过面包来解决她饿的问题了。

道理看起来很简单，但如果想不到这个点，就解决不了问题，还有可能造成负面影响。

这里想说的是，隐性需求，就是真实的原始需求。

* 1. 关于习惯性思维

我们继续拿之前那个简易计算器的需求作为第二个例子，重新描述下需求：

现在有一个 PC 客户端的命令行工具，这个工具可以接收三个命令行参数，其中，前两个是数字，最后一个是运算符，运算符只支持加减乘除四种，工具的功能就是把前两个数字使用运算符做下运算，然后输出运算结果。

针对上述需求的功能测试点：

* 三个参数的正常和异常情况（类型）
* 参数个数的正常和异常情况（数量）
* 数字参数的最大值情况
* 参数分隔符的正常和异常情况

参数个数和数字最大值，可以算到边界值分析法里面，如果不知道边界值分析，可能不会考虑到参数个数所有异常的覆盖情况，如果不懂编程，可能问不出来数字使用什么类型这样的问题，当然也就不知道所谓的最大值要怎么构造了，所以这个也可以算到隐性需求的范畴。

最后一个很少人考虑到的参数分隔符，肯定是我们要说的隐性需求了，这种没有明确说明的地方，有时候开发会按照自己自以为的方式给实现了，比如默认空格分割，但是测试后期发现很多人也会用逗号去分割，修改的话会造成新的修改成本，其实这么简单的地方，需求评审的时候提一下，就可以把需求明确了，难的是谁能想的到。

这里想说的是，隐性需求，就是把习惯性思维明确化。

第一个关于原始需求的例子，如果拿我们实际项目的情况来对应的话，可以找出来很多，比如某用户嫌某个软件的功能太多，其实他可能只是说他常用的功能入口太深而已，比如某用户嫌软件广告太多，其实可能只是推送的这些广告都不是他的关注点而已。

第二个关于习惯性思维的例子，实际项目中也经常发生，而且特别坑人，有个专有名词叫「经验主义」特别适合这个地方。

小明的数学还不错，最喜欢上的就是数学课，对各种数学题比较感兴趣。

有一次课间休息，小明不知道怎么就坐到一位成绩不太好的同学的座位上了，然后看到他桌子上放着还没做完的数学作业，一时兴起，就顺手给做完了，本来想着这同学回来该感谢他的，可是等到的却是责怪，责怪小明不该把他作业给做了，小明一时无语了。。。

这件事纯粹就是小明自己的一厢情愿，以为不爱学习的同学都不爱写作业，以为帮别人写了作业别人会高兴，所有这些都是「我以为」，或者说是小明的经验主义在作祟，让小明没能从实际情况去做出正确的判断。

这种事在需求评审过程中很常见，有一些貌似「不言自明」的逻辑，每个人都以为自己明白，别人也肯定能明白，并且和自己理解的一致，其实每个人可能理解的都不一样。

其实需求评审就是一个明确显性需求、挖掘隐性需求，然后相互确认理解一致的过程。

这里要说的就是，隐性需求，就是避免经验主义。

几个例子无非都想说明的是，隐性需求很重要，有时候，正确挖掘过的隐性需求会直接推翻现有的需求方案。

1. 课程总结
   1. 重点
   2. 难点
2. 课后练习
3. 扩展知识或课外阅读推荐（可选）
   1. 扩展知识
   2. 课外阅读