怎样写出清晰的需求文档



注：插图来自《启示录》

不得不佩服漫画的魅力，明明讽刺的就是我们这些产品经理，可是我看了还是差点笑喷，骂得真是太有道理了~

长期以来，大家吐槽的重点都只是放在产品经理永无止尽地改需求上面，而糟糕需求文档的危害却被严重低估了。如果把一个项目比作做一件衣服，需求文档写得不够清晰就会导致衣服到处破洞，每发现一个洞需要打一个补丁（改需求）。所以改需求只是病症，需求不够清晰才是病因。

那么，怎么才能写出清晰的需求文档呢？我总结了5点心得。

1.目的明确

比细节表述清楚更重要的是：让大家理解背景、目的，知道所有功能的来龙去脉。如果缺失了这些信息，大家看需求文档就和翻字典一样，索然无味，细节写得再精准大家也记不住。

所以我在口述需求或写需求文档时，一般喜欢先介绍下这个版本/功能的目的，介绍步骤可以分解为：

\* 这个功能是为了解决什么问题？（目的）

\* 为了解决这个问题，我给出的解决方案是什么？一般尽量带上数据或理论支撑。

\* 为了实现这个解决方案，我们需要A、B、C这3点功能。

讲完这些，再接着介绍功能细节，大家就很好理解了，心理上会忽然觉得需求文档薄了很多。

目的明确的一个意外收获是：即使功能细节描述出现歧义，但是当大家发现只有1种解读方式是和目的匹配的，就自然会选择那个正确的解读方式。

2.结构清晰

拆解完目标，我们会得到一个功能list。学会将功能list归纳很重要，一个将15个零散功能堆在一起的需求文档会让人感觉很杂乱，但如果根据逻辑关系整理成3大块功能、每大块里面5个小功能，就会清晰很多。归纳功能并按优先级排列，会让整个需求文档显得结构清晰，大家也更愿意读下去。

3.图文并茂

虽然我之前曾在博客中提到——用什么工具画原型图、以什么形式写需求文档不足以影响一个产品经理是优秀还是平庸，但是单纯从需求文档的表达效果而言，我认为还是存在优劣之分的。对于绝大部分涉及到界面的功能需求，图文并茂都是最好的表达形式，一图胜千言。想象下如果没有示意图，就会出现这样的奇葩表达方式“页面首屏1/3处第2个按钮，点击后跳转到xx页面”。如果有示意图，就可以相对随意地写“如图所示，页面中部有个xx按钮，点击后调到xx页面”，大家一看图就明白了。毕竟需求文档的目的不是秀文采或语言表达能力，而是精准地传达需求。

一般对于独立的功能，我常采取的形式是左图右文：左侧是示意图，在右侧写下关于这个页面的所有策略。如果需要表达几个功能步骤的跳转关系，就会先将几张步骤图横向排列，然后在每张图的下方附上对应的策略。比如下图这样：



使用的工具是Axure，每个原型图+对应的策略在一个画布里搞定，整体导出PDF就成了一份需求文档。

多说一句，由于word天然不适合施展图文并茂的方式，导致我每次看到还有产品经理在用word写需求文档，心里都会默默鄙视一番“我去，都什么年代了，怎么还能看到这种老古董！”

4.描述顺序合理

我描述需求的方式是从编程思想中偷学而来。编程里有一种比较成熟的模式叫做MVC，即将同一个功能的代码拆成3部分——model<数据层>、view<展现层>、controller<操作层>，开发每一部分时只用重点关心一件事情，比如开发数据层就关心数据结构怎么设计、开发展现层重点关心样式、开发操作层重点关注用户交互。这样做的好处是代码结构更加清晰，以及各层次耦合度低，调整时不需要牵一发动全身。

写需求也是一样，可以按照“有什么”->“长啥样”->“怎样操作”的顺序来描述。以常见的点赞功能为例，可以这么描述：

\* 页面底部添加一个点赞按钮，按钮上内容为一个红心icon+“赞”的文案。

\* 点攒前按钮和文案为灰色，赞过之后按钮和文案为红色。

\* 未赞同状态下：点击点赞按钮，弹出浮层（浮层文案为“点赞成功”），1秒后浮层消失，点赞按钮变为红色。

\* 已赞同状态下：点击点赞按钮，弹出浮层（浮层文案为“取消点赞”），1秒后浮层消失，点赞按钮恢复为灰色。

5.量体裁衣，保持合适的详细程度

一般需求中的功能，大致可以归为这几类：

\* 全新需求，需要描述得最为详细。一个新功能需求一般需要功能目的、功能入口、展现内容、样式、交互操作一应俱全。

\* 优化需求，稍微简略一些。由于功能已经存在于线上，所以大家已知的信息量多一些，一般不需要把现有功能的背景再啰嗦一遍。只需要讲明线上版本存在什么不足、希望怎么改、预期效果，然后重点介绍改动部分即可。

\* 修复bug，最简略。由于是bug，一般不涉及功能逻辑调整，所以比普通功能描述简略一些。但我们常犯的错误在于太简略，只说要改什么就一笔带过了。一个修复bug的需求哪怕浓缩成一句话，也应该包含这几点元素：功能/策略原本应该是什么样的、现在是什么样、这样会引发什么问题、希望怎么改（一般是希望改成原本应该的样子，所以这点一般可以略去）。

做好以上几点，虽谈不上让需求文档突然就文采飞扬，但至少可以循序渐进地引导研发愿意读并读懂需求，避免许多在开发过程中因需求理解不一致而走的弯路。我相信大部分产品经理和我一样，每次看到吐槽产品经理的段子，未尝不是笑中带泪，唯有自勉以赢得一丝丝尊严吧~