让客户做决定

“客户就是上帝”！当我们在业务功能上遇到一些困难需要作出抉择时，不要轻易的由项目成员作为决定，因为项目成员也许并没有搞清楚客户真正想要的功能。把抉择权留给客户，没人比客户更了解需求和业务。当然，在和客户进行交流时，我们应该理清脉络，以客户能理解的表述方式向他讲述目前项目中遇到的困境，然后等待客户作出抉择。当然，我们也需要提供我们所能提供的所有有关信息，以帮助客户作出最好的判断和决定。

如果客户最终的决定和我们预想的不一致，不要惊慌，不要始终认为是客户或是自己错了，此时，将客户的决定和我们预想的结果作比较，或是根据已有信息作出权衡，深入挖掘导致产生不同结果的本质原因。

让客户做决定，让客户感觉到是他们在驱动着项目的进行，同时，也避免了我们主观判断而导致的错误，从而避免了维护成本的增加。

让设计指导而不是操纵开发

软件开发过程中只有变化是永恒的，进行编码前不需要写出非常详细的设计文档。详细的设计文档往往是过时的设计。需求是不断变化的，我们不可能在编写需求文档的时候就预想到客户增加或将来一定会增加的所有需求。我们也不可能在设计文档中就定义好所有的细节，这些东西都是不稳定的，是易于变化的。

相反，我们只需要保证文档能给我们指引正确的方向，因此在编码前我们几乎不需要详细设计文档，只需要概要设计，告诉我们应该有哪些类，类的职责和类之间的协作关系就可以了。好的代码即是好的详细设计说明书。

好的设计应该是正确的，而不是精确的。也就是说，它描述的一切必须是正确的，不应该涉及不确定或者可能发生变化的细节。它是目标，而不是具体的处方。

合理的使用技术

软件开发行业最突出的特色就是新技术层出不穷，有时我们会被新技术的广告或是其他同行的观点所迷眼，而过于盲目的使用新技术去解决不存在或不适用的问题。它有可能会带来一些好处，但是却会有更深的隐患。

在使用一门新技术之前，需要进行详细的评估，比如要判断该技术是否能解决我们目前遇到的问题？在解决问题的同时，是否会带来额外的隐患？是否需要大量资金去购买这种技术的使用权或环境？是否需要较大的学习成本？是否和我们目前使用的技术相斥？。。。

我们必须认识到新技术是有风险的，在我们真正认识到新技术的方方面面之前，就不要轻易去使用它。

欲速则不达

有时我们在修改bug的时候，尤其是在维护并非自己写的大型系统的时候，为了快速的解决问题，在没有真正理解代码的情况下就做出不明智的修改。这带来的隐患可能比原先的bug还大。

在我们动手去修改一个bug之前，需要花时间去探索这段代码。当确信自己真正理解之后，再去做出修改。有时，这样的探索过程也许会花费很长的时间与精力，但是，我们必须抱着“让将来的维护人员更好过”的想法，如果那段代码实在太难以理解，结构太混乱了，我们就可以考虑重构它。

当然，对于大型项目，十万百万行的代码，我们无法做到精通全部细节，但是，我们至少要做到从整体到局部的理解代码。理解每段代码在整个系统中的层次位置和与其他模块的协作关系。

我们必须做到“防微杜渐”，不在坠入快速的简单修复之中，要投入精力和时间保持代码的清晰和整洁。

当然，有很多方式可以帮助我们理解代码，比如层次划分，编写单元测试，请教其他开发者等。同时，在自己编写模块的时候，也不要孤立编码，通过互相阅读彼此的代码，代码复审，单元测试，详细的注释，都可以提示代码的可读性，为以后的维护人员提供便捷。

永远不要让代码之中留有无人敢问津的黑暗角落。

打破沙锅问到底

软件开发人员的基本素养应该是有探索未知的精神，当面临一个问题的时候，除非给我们完全清晰的答案，否则我们应该不停的问“为什么?”。因为不清晰的地方可能会给我们带来灾难。

同时，问别人为什么，也是在迫使别人思考。注意，对别人的提问，不是简单的三个字，我们的问题要有助于别人告诉我们本质的答案。

记住，不能只满足于别人告诉你的表面现象，要不停的提问直到你明白问题的根源。