Don’t “imagine” request.

Keep it simple but dynamic, add complexity only when needed.

In an unknown environment, evolve fast > make things perfect.

If there's anything non-instinctive, prepare solid statement about why and give proof.

Don't limit to your scope. Some problems should be addressed by updating another service or package or sending request to other teams, feel free to do it.

Clearly differentiate complexity from problem itself and complexity from poor engineering.

大佬不会成心为难你，如果大佬的意见在你看来过分或者错得很厉害，往往是因为信息不对称，如概念的定义、项目的过程、某个服务的特性等。迅速找出这些不一致对推进讨论非常重要。当大佬提出方案和你的不同，这两个方案一定各有优势各有劣势。大佬指出你的方案的劣势有两种情况：1大佬是对的2大佬的判断缺少信息。后者通过找出并补全信息来解释，前者切勿硬反对。正确方法是列举和强调自己方案的优势，然后综合比较二者的优势和劣势并在此过程中支持自己。成功率取决于你对大佬的方案的迅速理解和分析并快速识别弱点（往往依赖经验），对自己方案优劣的清晰认知，是否认真考虑过替代方案。回忆一下怎么打辩论。

和客户保持联系。

每一次开会之前：这次会议的目标是什么。

每一个重大决定，和所有可能相关的大佬说，准备自己的方案并尝试convince所有人，不管成不成，最后一定要达成一致并所有人签字画押。就算有大佬中途加入问怎么回事，给他证据。

概念的一致性和明确性极端重要，即使细节也有巨大影响。实现的混乱和困难，理想选项不存在或不明确，经常源自概念问题。概念问题会随开发恶化，错误的实现会导致更多错误，影响开发难度乃至产品生命周期。

定义最容易混淆和不明确的地方之一就是‘空（null）’和‘缺（missing）’。