1. 没有永不过时的解决方案

这件事和软件危机的其中一个表现类似，不可能有万金油一样的普适性方案。尤其是 CS 领域，几乎是日新月异。硬件、软件、技术、框架、语言等都在随时更新，解决方案也必须随之更新，甚至于达到瓶颈之后要进行重构重新设计，或者再换一个解决方案。就如我们学校的 OOAD 课程，每年都根据这一年来 Java 领域的技术和框架的发展，对前一年的代码进行部分更新和重构，并根据当前流行技术提出新的需求。

2. 对最终用户而言，界面就是系统

软件的使用者大多是非专业人士，甚至是从没接触过相关领域的人，他们几乎不懂得软件背后的结构和逻辑，这就像我们平时挑选商品一样，首先吸引我们的大多数情况下都是商品的外观，其次才是之后的质量等。如果一个软件因为界面布局不合理或者不合用户胃口，那么会带给用户糟糕的用户体验。因此，我们在软件开发中，既要注意后端的性能，也要注意前端的设计。
一个我个人感觉比较合适的例子是 Windows 系统和 Linux 系统。从本质上看，二者都是操作系统，或者说只是一个用来统一管理硬件的系统，本质上区别不大。但 Linux 更多的情况下是靠 Terminal 终端的命令行运行，甚至简单的编辑文本也需要 vim 编辑器，不懂指令的人几乎无法正常使用 linux 系统。而 Windows 系统的图形化界面几乎是用户友好型的，即使是没有电脑基础的人也可以轻松上手

3. 不要在一棵树上吊死

软件开发过程中不要“不撞南墙不回头”，如果只执着于一个方案，在行不通或者效果不好时不去采取新方法，会把问题拖的更大，耗费更多的人力物力时间成本，从而让软件开发进度停滞甚至倒退。所以从一开始就要先设计出多种解决方案以及应急策略，并清楚每种方案的优缺点，以便在之后的开发过程中方便更换。