**代码评审标准**

**一，计算机运行效率的维度：**

目的：运行效率高

手段：优化执行逻辑、减少执行时间、减少重复计算、对象池（空间换时间）......

**二，程序员方便理解的维度：**

目的：可读性

手段：命名方法、函数设计、格式、注释......

**三，程序员易于修改的维度：**

目的：可维护、可复用、可拓展、灵活性好

手段：设计模式六大原则......

**四，可读性量化标准**

在很多跟代码质量有关的书里都强调了一个观点：程序首先是给人看的，其次才是能被机器执行。

在评价可读性这个维度的好坏的时候，可采用一种方法，就是将代码翻译成中文读给其他程序员听。不可避免的会有需要联系上下文才能理解的句子，但这就是相对可以量化的地方，如果需要联系的上下文越多，意味着代码的可读性越差。

**五，代码的重构**

重构的定义：重构就是在不改变软件系统外部行为的前提下，改善它的内部结构。

项目的需求（即代码需要应对的情况）总是不断地迭代的，这是很正常的情况。一个好的程序架构，可以做到面对需求一定程度的变更（无论是增删还是改）都能轻松应对，好的程序架构对于需求应该是支持而不是限制。产品要做的就是将未来的各种需求，尽可能地考虑周全；而程序要做的就是要将未来的各种需求，都囊括到你现在写的程序架构可以尽可能轻松实现的范畴之中。

我们不能要求产品将未来的各种需求都细无巨细地考虑完全，所以一旦有超出目前程序架构能实现的需求出现，可能就面临着重构。当然这个时候，你也可以在老架构之上先将需求实现，这之后再来考虑重构的事，但这往往是代码变烂的开始，因为你已经欠下一笔账了。所以我认为重构代码应该是即时的（仅限于模块内的重构，项目重构涉及到项目周期等更多因素）。

重构再定义：保留对原有功能的支持，再增加对未来需求的适应性。

参考地址：https://blog.csdn.net/JohnBlu/article/details/83348413