代码走查指引

目标

通过代码走查把控代码风格及设计要求。

经验表明，使用代码走查方法能够有效地发现30%到70%的深层次逻辑设计和编码错误，是一种无法替代的提高软件质量的研发活动。同时代码走查对提高程序员的技术水平也非常有帮助，是一种非常高效的、团队成员之间相互学习的方法，会在几方面上帮到开发团队：

- 一次代码审查可以将 增、删、改 等功能性改动清楚明了地传达给团队成员，以便其开展后续的工作
- 审查者可以学习到提交者所使用的某种技术或算法。更概括的说，代码审查有助于组织内部的质量提升
- 审查者可能掌握着能够改善或精简所提交代码的编程技术知识或代码库；举例来说，某人也许正好也在开发类似的特性或修复类似的问题
- 积极的交互和沟通会加强团队成员之间的社交连结
- 代码库中的一致性让代码易读易懂，有助于预防 bug，并能促进开发者之间的合作
- 代码片段的易读性对于将其亲手写出的作者来说是难以判断的，而对于没有完整上下文概念的审查者则容易的多。易读的代码更容易复用、bug 较少，也更不易过时
- 意外错误 (如错别字) 及结构错误 (像是无效代码、逻辑或算法错误、性能或架构上的关注点) 经常更容易被旁观者清的挑剔审查者找出来。有研究显示，即便是简短、非正式的代码审查也能显著影响代码质量和 bug 的出现的频次
- 合规合法的环境通常需要审查。代码审查是避免常见安全陷阱的很好途径

原则

- 通过MR来进行代码评审活动。
- 尽早开始代码评审，早期的评审可以看作是设计评审。
- 尽早给出评审意见，建议不超过3个工作日。
- 一个MR不要有多个任务，除非它们是紧密关联的
- 一次评审少于 200–400 行的代码。
- 被评审的代码必须编写了测试。
- 检查结果的内容可以加入到开发规范中，并考虑用工具来实现自动化检查。

规范

评审关注点

测试

- 代码是否可以测试？比如，不要添加太多的或是隐藏的依赖关系，不能够初始化对象，测试框架可以使用方法等。
- 是否存在测试，它们是否可以被理解？
- 是否包含了正确路径、异常情况的测试？还有没有其它场景需要纳入进来？
- 测试覆盖率是否没有下降？
- 单元测试是否真正的测试了代码是否可以完成预期的功能？
- 是否检查了数组的“越界“错误？

常规项

- 代码能够工作么？它有没有实现预期的功能，逻辑是否正确等。
- 代码是否尽可能的模块化了？
- 是否有可以被替换的全局变量？
- 是否有被注释掉的代码？
- 循环是否设置了长度和正确的终止条件？
- 是否有可以被库函数替代的代码？
- 是否有可以删除的日志或调试代码？

安全

- 所有的数据输入是否都进行了检查（检测正确的类型，长度，格式和范围）并且进行了编码？
- 在哪里使用了第三方工具，返回的错误是否被捕获？
- 输出的值是否进行了检查并且编码？
- 无效的参数值是否能够处理？

文档

- 所有的函数都有注释吗？

- 对非常规行为和边界情况处理是否有描述?
- 第三方库的使用和函数是否有文档?
- 是否有未完成的代码？如果是的话，是不是应该移除，或者用合适的标记进行标记比如‘TODO’？

可读性与可维护性

- 命名是否足够清晰的描述了变量、方法、类的含义与用途
- 是否有注释，并且描述了代码的意图或业务规则，而非if... then...的简单描述?
- 能够很快看明白方法的目的么?
- 异常信息能够看明白吗?
- 所有的代码是否简单易懂?
- 是否存在多余的或是重复的代码?

推荐实践

- 通过创建MR来开是一个评审过程。
- 尽早创建MR，如果设计方面需要多方意见，甚至可以创建完分支即开启MR。
- 未完工的MR加上 WIP: 前缀，表明 work in progress。
- 如果CI结果失败，不用浪费时间做评审。
- 每一个代码工程，根据工程特点，维护一个OWNER列表，用于MR创建时，自动分配reviewer。
- 被分配到MR时，reviewer会收到自动发出的通知，reviewer尽早开展评审活动，建议不晚于3个工作日给出意见与清单。
- 问题清单应当创建到具体的代码文件或者代码行，便于交流。
- 针对评审人提出的清单，被评人在解决完后，需要回复，以通知评审人哪些问题得到了修正。
- 清单的状态由评审人改变。
- 全部解决后，评审人Approve这个MR，代码合并至Master分支。

指标

- MR包含的代码变更行数
- MR合并平均等待天数
- 每天合并MR的数量
- Review问题数目 / MR包含的文件数