CI--> rewiewer-->TestDrive

代码持续集成，审核及测试方式

IBM代码质量要素

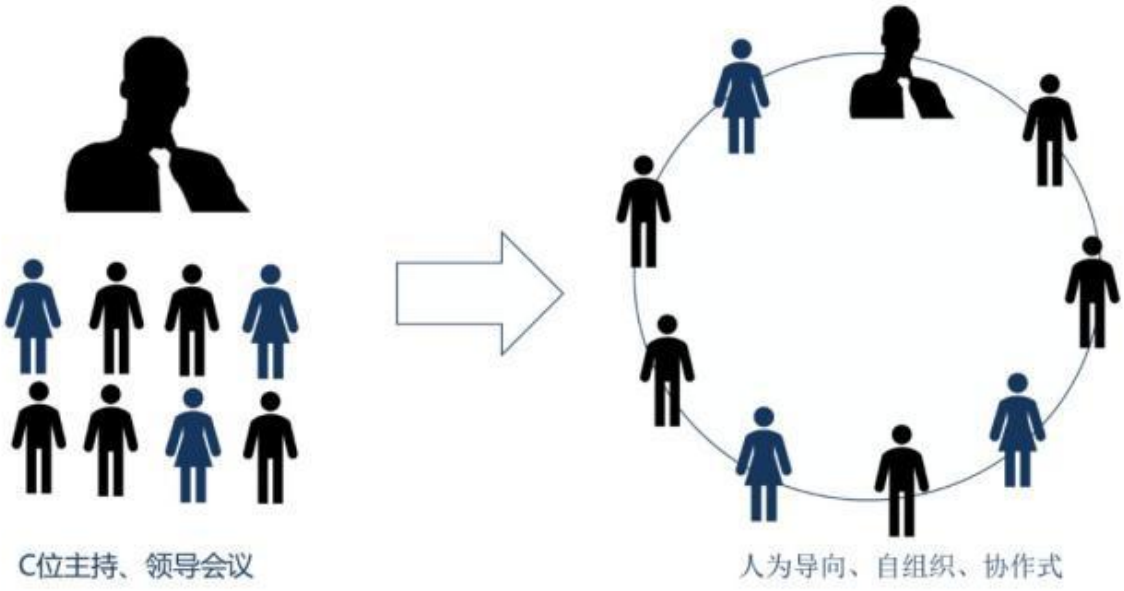
1. 意识培养 + 成文的规范保证

新程序员入职的培训，主要包含以下内容：

- 代码整洁之道
- 编程最佳实践
- 团队自己成文的编码规范（类似开发者手册,例如google C ），涵盖命名、圈复杂度、单元测试和覆盖率等指标性要求
- 严格的高效率代码注释（IBM 的doxygen注释语法）

2. 强大的 CI 守护系统

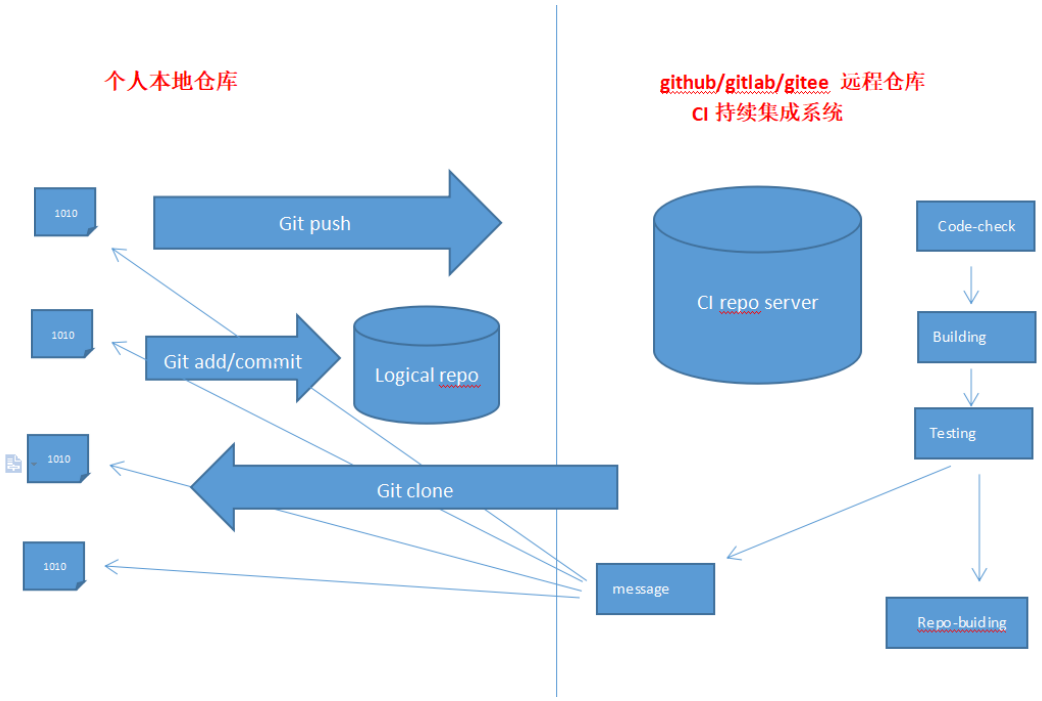
CI（Continuous integration，持续集成）在企业级开发中扮演着非常重要的角色。所谓 CI，简单的理解就是：我们每次提交代码时，都会做各种各样的检查和测试以保证质量（CI 防护网），通过之后才有可能将代码集成到代码库的主干上去。由于这个过程很频繁，所以叫做持续集成。现在还发展出了 CD（持续部署）。



实际工作流程如下：

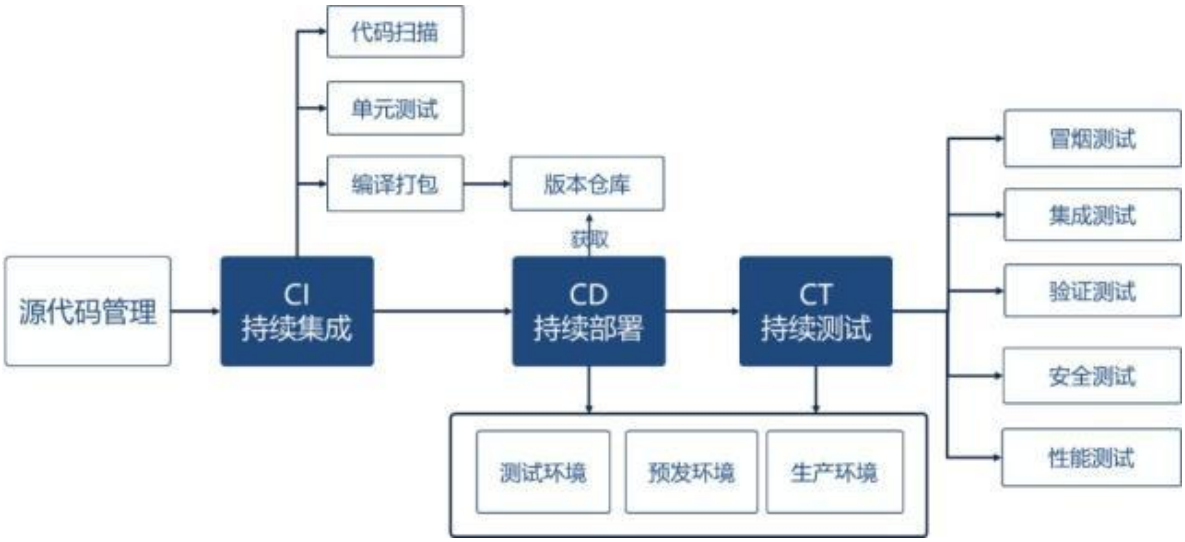
1. 自查语法，并在本地执行各种检查。包括但不限于圈复杂度、PCLint、安全分析、单元测试、代码覆盖率等等。
2. 本地检查通过后，再推代码入库。此时会触发 CI 防护。CI 自动合并代码，将各个库的代码放到同一目录下（或者通过软连接来实现），然后编译。编译之后做静态检查。如果不能通过，则返回至提交者。

分层多维度CI：每次提交都会进行 CI 防护（Verify CI），每天凌晨执行 Daily CI。Daily CI 比 Verify CI 更加全面，Verify CI 通常是做增量检查，Daily CI 则是全量检查。



3. 严格的 code review

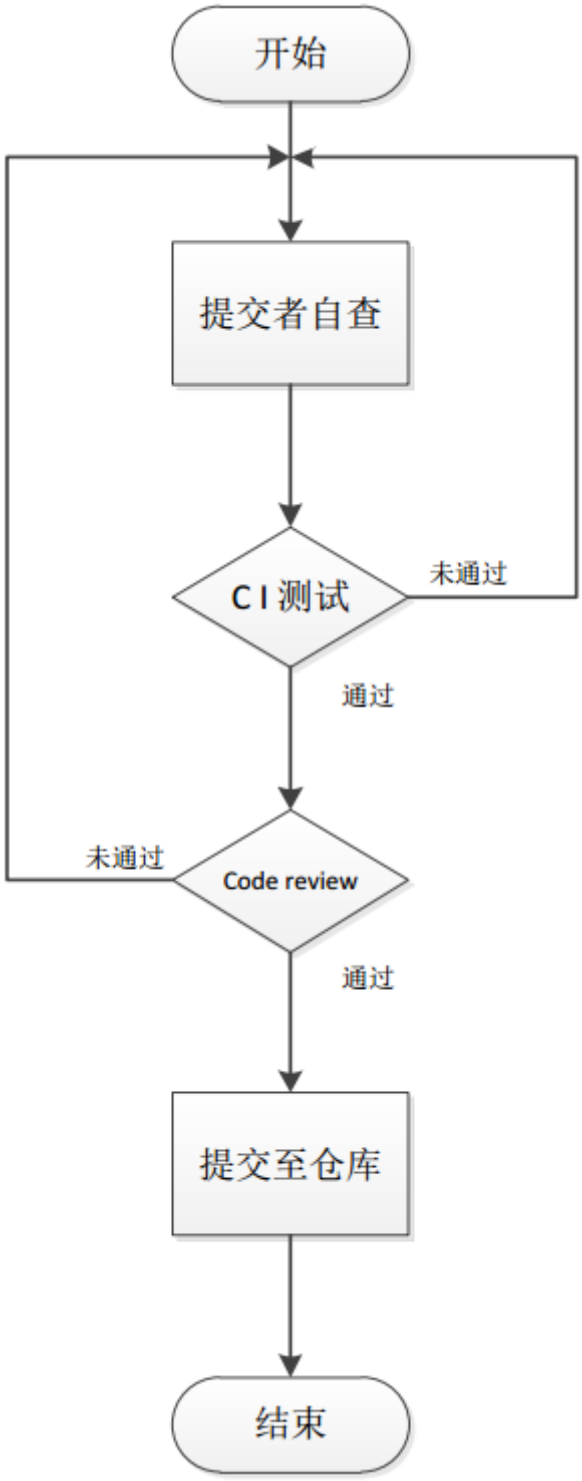
接上述流程，如果 CI 检查通过，流程走到代码审查（CI中已经安装了自动代码审核工具）。人工代码审查通常由 team leader 或骨干成员执行，如通过审查，则提交至主干，如未通过，则返回提交者继续修改。对所有的issue及PR认真复核及审查，各个代码分支合并前的参数测试，输出值等的注释及说明



4. 强大的度量系统

每人每月的代码量，譬如增删多少行，净增多少行，都有详细报表。CI 通过率也会有详细统计。

二、整体流程图



三：代码单元化测试，永远保持小而专的代码及测试

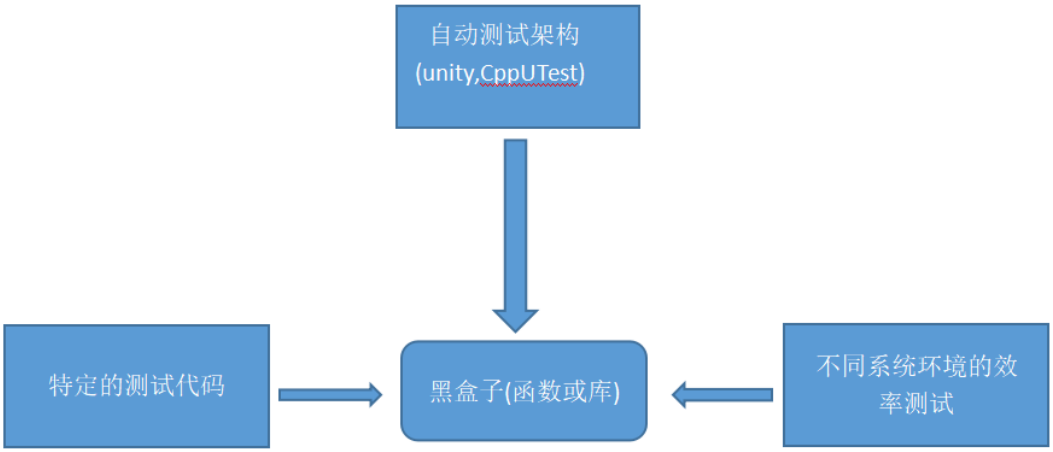
打包建黑匣子

功能单元通过以上流程之后就可以打包生成静态或动态的代码库（黑匣子），严格的代码注释在形成黑盒子之后体现尤其重要，因为测试人员及库使用者没有详细的入口参数及输出说明的情况是很难有效使用打包的库文件或函数

引用IBM的程序员入职的第一条格言：

如果一个程序在仅仅看完它的注释之后，你还不能了解它的作用，那么把这个程序扔了吧！

打包成黑匣子的代码，就要收到测试工具及测试员严格的各种严刑拷打；



Gerard Meszaros所著的在"xUnit Testing Patterns"的书中，四个阶段的测试模式践行以上流程

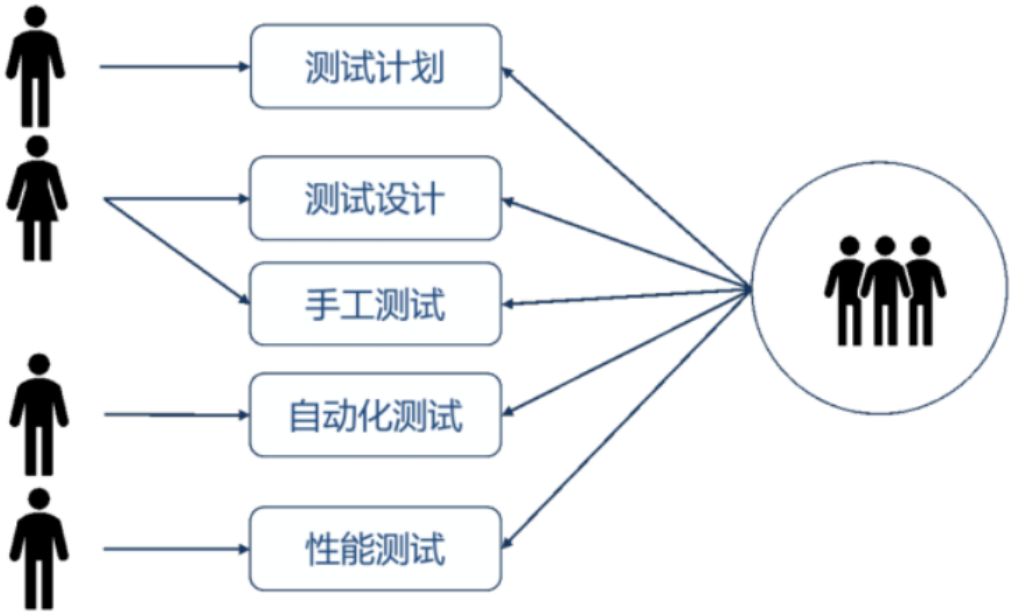
1）建立：建立测试的前置条件

2）运行：对被测试系统进行操作

3）验证：检查预期输出

4）破坏（分解）：用常规，非常规的越界，超内敛拆解库还原到初始代码

四 :方法与手段



1）编写测试列表（列出说有测试的内容及流程）

2） 编写测试代码 （类似FPGA的功能测试）

3）编译环境的兼容性（在不同的开发环境中测试黑匣子的兼容性）

4）注入依赖参数，测试输出结果

5）黑匣子的重构性能测试（对黑匣子的引用并测试引用后功能正确性）

6）编写测试结果文档，把库放进公司私有代码库，纳入公司机密档案申请流程