代码整洁之道

软件高质量代码体系最佳实践

一、为什么需要该课程

软件质量,不但依赖于架构,设计以及项目管理,而且与代码质量紧密相关.这一点,无论你使用什么开发技术,都不得不承认. 代码是程序员沟通最直接的手段，代码是技术交流的手段，代码是需求交流的途径。重视代码，回归本源，曾经我们远离代码，谈架构设计，谈 UML,谈开发流程。如今我们落地,找回软件的本源，彻彻底底看清代码、深入思考代码。那些一流的研发中心非常重视代码,Facebook 就有经典的 Code wins arguments（代码赢得争论）。在 Facebook 做 code review 时间大约占 50%，管理者对代码质量负有一定责任 。甚至代码质量高于一切：Facebook Code review 是重点 KPI 考核的对象，实行连坐制，如果因为代码质量问题，那么产生的 KPI 责任包括领导 30%、程序员 50%、审核人员 20%。

管理者最担心听到开发人员这样抱怨:“不能再增加功能了！我们得停下来重写代码。软件代码一团糟，就像纸糊的老虎，根本应付不了持续增加的用户需求。我们实在维护不下去了！最好推倒重写吧”

这一幕在很多公司上演过，现在依然在不断重演。一旦公司陷入这种困境，以前版本的开发者往往沦为替罪羊。新的开发者一般就会骂前人怎么写这么烂的代码。他们准备推倒重来，准备重写系统。在重写代码的过程中，用户无法看到产品的任何改进。你可能认为重写代码至多也就几个月，但是实际花费的时间无一例外要多得多。你只能坐在一旁，眼睁睁看着用户投奔竞争对手，而这个时候，竞争对手恰恰在不断地改进产品。

我们研发中心有一个理念”代码是债务而不是资产”。最开始，团队会编写代码，做出产品，并用它来赚钱，但是，之后团队应该尽可能地寻找减少代码的方法和使代码尽量整洁，从而降低成本。软件界有一个真理，你拥有的代码越多，维护代码所要付出的成本就越高。如果你的代码结构越好，你做了越多的单元测试，你的代码质量越好、越小、耦合越松，那么添加新代码所需要付出的成本就越少。因此大师 Craig Larman 说: “最好维护的代码就是没有代码，好的程序员的代码产量是负的，因为他通过减少代码来增加功能”。对比现实中，很多人以为，LOC(line of code)越多的 feature 越大，写 LOC 越多的程序员越牛。这其实是极其错误的观念.

因此我们必须有全面的管理制度让我们保持代码少而整洁。所以 Michael Feathers 认为"未来属于知道如何有策略地删除代码的公司”。持有代码的成本要比我们想象的大。意识到这一点的公司更具有竞争优势。

为了切实帮助软件企业降低企业项目开发成本，大面积提高软件工程师编程能力和代码质量管理能力，我们特别推出实战训练营. 分享多家大型研发中心代码管理经验给大家.

该课程适应于各个阶段的技术人员.初级工程师能够透过大师的眼睛来看待编程,了解编程的价值观和原则;具有丰富经验的设计师和架构师可以通过实现模式进行反思,探究成功实践背后的意义.把价值观,原则和开发实践结合;管理者通过学习业界著名研发中心的管理经验和失败的教训,来制定自己公司的代码管理策略.质量管理相关人员学习如何定制代码质量指标，通过哪些工具进行监控，怎样管理代码质量。

二、谁已经选择了我们的咨询和培训?

我们已经为几十家企业提供了多次培训和咨询服务，以下企业已经选择了我们的内训课程

- 互联网研发企业，比如百度研发中心 2 次 ，阿里巴巴 4 次，腾讯 6 次
- 电信研发企业，比如思科研发中心 5 次，阿尔卡特-朗讯研发中心 11 次，华为研发中心，摩托罗拉研发中心 1 次，大唐电信研发 1 次，广州从兴电子，亿阳通信 5 次， 爱立信研发中心，
- 广电行业：广州诚毅科技研发中心，
- 企业软件研发企业，比如 Adobe 中国研发中心，北京久其研发中心，博古中国研发中心，金蝶深圳研发中心，用友研发中心
- 嵌入式软件企业，比如阿尔卑斯中国研发中心，德国 M&M Software，西门子研发中心， Sony 研发中心，金立智能研究院，南车研发中心，德塞西威，霍尼韦尔研发中心
- 外包类企业，联盟计算机服务（天津）有限公司 ACS 3 次。
- 金融行业：恒生电子，华腾，中国人民银行研发中心，工商行研发中心，平安科技研发中心，建行研发中心，深圳登记结算研发中心,花旗银行中国研发中心

我们已经为几十期公开课，已经有 100 多家企业已经选择了我们的公开课程

腾讯（深圳）有限公司， EMC 中国研发中心，华为终端有限公司、 斯伦贝谢技术，通用电气医疗系统（中国）有限公司，华为技术有限公司，广州从兴电子开发有限公司、福建星网锐捷股份有限公司，广州菲特网络科技有限公司，盛立金融（杭州）软件公司 ，索尼中国研发中心， 爱德万，上海金慧软件有限公司，珠海世纪鼎利通信科技股份，兰吉尔仪表系统有限公司，珠海飞企软件有限公司，广东佳和通信技术有限公司，珠海一多监测科技有限公司，远光软件股份有限公司

三、你可以参加吗?

各类软件企业和研发中心的程序员、软件设计师、架构师, 项目经理,质量部门员工。

- 如果你不重视代码质量, 请不要参加. 本课程面向重视代码质量的管理者.
- 如果你不认为写好代码是一件重要,困难并且有趣的事情,请你不要参加. 本课程面向追求卓越的程序员,我们认为编程是一种态度.
- 如果你已经多年不写代码,最好不要参加,本课程面向一线还在编程的程序员/设计师/架构师

四、你的角色和收获

课程根据著名编程大师的理论：

编程是一种态度，编程是一种技艺，编程是一种习惯。

面向以下不同的人群，有不同收获。

角色	收获
技术负责人/技术总监	● 了解业内先进的代码审查的形式、技术、技巧和流程的成功经验，优化现有开发中心代码审核方法； ● 掌握业内成熟的自动化审核审查工具及方法，提升开发人员在代码结构分析、代码质量度量、代码覆盖率分析等方面的能力,并有效运用到项目研发工作中。

项目经理/项目管理人员/架构师/	● 学习其他研发机构的代码管理思想 ● 代码管理手段 ● 代码管理相关流程和相关工具 ● 代码监控
测试部门/质量管理部门	● 代码审查 ● 代码检查列表 ● 代码管理手段 ● 代码管理制度的建立
资深开发人员	● 掌握代码编码规范、代码评审要点等知识，引导开发人员养成正确的代码编写习惯； ● 编程技艺和相关编程实践 ● 重构手段
一般开发人员	● 编程技艺和相关编程实践 ● 重构手段 ● 代码坏味道

五、课程内容安排（该内容为 2 天版本）

第一篇： 编程是一种态度--------价值观		
主题	**培训内容**	**备注**
第 1 单元　代码就是债务(1~1.5 小时)	**内容一：代码是债务** 1. 代码的认识---代码就是债务 2. 代码是债务，越少越好 3. 你拥有的代码越多，添加新内容所要付出的成本就越高 4. 通过案例分析让代码库尽可能小的方法： **5. 通过国际研发中心电信计费系统演示代码是债务的思想，10 多年国外研发团队设计与研发第一版本，目前几百人在维护** **通过项目演示通过重构如何减少了一半的代码，维护的人员的减少** 项目的失败可能归咎于各种各样的原因。一些项目因糟糕的需求而失败，另一些则由于钱和时间超支了，还有少数单纯是因为糟糕的管理所致。如果我们探究其根本原因，是否会发现所有项目失败的罪魁祸首是糟糕的代码呢？ Bob 大叔坚信糟糕的代码所带来的成本之大足够让一个项目失败。 **内容二 软件界要以新视角看待代码** 1. 传统的软件工程对代码的错误认识 2. 代码的两面性，代码的静态结构和运行时行为	

	3. 客户和管理者往往仅仅关注代码的运行时的行为 4. 温伯格认为的主管必须关注代码 5. 软件设计与代码的关系—真正好的设计是在编码阶段一步一步而形成的,通过案例分析，设计如何根据代码进行演化 6. 编程真的是简单的劳动吗？ 7. **通过多家项目案例进行分析，传统思想对代码的种种误解，我们提出了从 3 种新的角度来观察代码，** a) **从管理者的角度，我们仅仅观察代码的运行时行为，导致代码的静态结构混乱的根源。这就是代码的冰山原理，大量垃圾代码隐藏在冰山之下。** b) **设计师的角度认为只要有好的设计，软件质量就可以保证。其实我们认为代码是真正唯一可以精确描述的设计文档。** c) **程序员的视角，编程真的很难，通过某一个项目案例分析，20 多人一周的工作量就为几行代码问题**	
第 2 单元编程价值观 **(1~1.5 小时)**	**内容一：编程价值观** 1. 编程的方法学 2. 什么是好的代码，我们却认为“Good code is not bad code！” 3. 编程价值观---沟通,简单,灵活 4. 价值观决定行为 5. 优秀代码的评价标准，什么是高质量编码？特征是什么？ 6. 软件代码的可读性 7. 代码的可扩展性 8. 糟糕代码的特征 9. 劣质代码的代价 10. 大师评价整洁代码的标准 **11. 通过大量项目案例分析，什么是好的代码，对好代码新的认识**	
第二篇： 编程是一种技艺-------实践篇		
第 3 单元 高质量函数	**内容一：高质量函数/过程 (2 小时)** 1. 为什么需要函数 2. 函数复杂度度量 3. 函数圈复杂度以及度量 4. 函数抽象层次-单一抽象层次原则 SLAP(Single Level of Abstrction Principle) 5. 函数实现模式之一组合函数(Composed Method) 6. 万恶之源—函数过长 7. 函数的单一职责 8. 函数第一原则:是要短小,函数第二原则:是还要短小,函数第三原则:是必须短小 9. 函数重构之道—抽取方法(Extract Method)和抽取对象函数 10. 函数命名—怎样取好的函数名 **11. 通过大量项目代码分析,函数的遇到的各种问题,如何编程高质量函数** **内容二：函数易理解与沟通 (1 小时)**	

	1. 函数主体流 2. 函数的异常处理 3. 函数主题流程简化方法 1-助手方法 4. 助手方法的应用场景 5. 助手方法的效果 6. 函数主题流程简化方法 2-函数对象（MethodObject） **7. 通过真实项目代码进行分析，如果提高代码的可读性** **内容三：函数灵活/易可扩展---函数接缝（1 小时）** 1. 历史遗留代码维护问题 2. 某电信研发中心的维护问题—开发维护的效率问题。 3. 增加一个功能特性的成本并不单单是为这些功能编码所花费时间的成本，还应该包括特性扩展的障碍成本。 4. 代码的可维护成本分析—通过大量案例分析 a) 确定需要修改哪些部分有多难 b) 必要的改动有多少 c) 实现改动对系统其他部分的影响有多大 5. 如何实现代码的易扩展—函数接缝 6. 接缝（seam），指程序中的一些特殊的点，在这些点上你无需做任何修改就可以达到改动程序行为的目的 **7. 通过案例分析，如何实现函数的灵活/易扩展。** **内容四：函数参数（0.5 小时）** 1. 函数参数过长 2. 最理想的参数数量是零,其次是一,再次是二,有足够的理由才能使用三个以上参数. 3. 函数参数重构之道-引入参数对象(introduce parameter object 4. 函数参数的顺序. 5. 不要把程序参数当做工作变量/临时变量 6. 函数参数模式-collecting parameter 7. 函数返回值 **8. 通过大量项目代码是函数参数问题** **9. 演示函参数的重构** **内容五：条件表达式 （1~1.5 小时）** 1. 复杂表达式的简化 2. IF/ELSE 语句应该如何编写 3. Switch/Case 语句应该如何编写 4. 复杂条件表示式的危害 5. 过分深层的缩进，或者“嵌套”，已经困扰了计算机界达 25 年之久，并且至今仍然是产生混乱代码的罪魁祸首之一 6. 复杂表达式重构之道—引入解释变量/分解条件/抽取方法计算条件 7. 表驱动法-多级嵌套 IF 语句的必然之道	

	8. 表驱动法使用总则 9. 某保险项目表驱动法应用案例分析 **10. 通过大量项目代码演示条件表达式编码问题** **11. 复杂表达式的注意事项,如何解决** **内容六：函数组织（0.5 小时）** 1. 尽管组织直线代码是一个相当简单的任务，代码结构上的不合理会对代码的质量，正确性，可读性和可维护性带来影响。 2. 把函数代码分成“段落” 3. 选择一个有意义的顺序，始终一致的使用它 4. 应该把代码组织得一次只做一件事情 5. 组织直线型代码。嵌套可以，但是不应该交叠 6. 系统应该由许多短小的函数而不是少量巨大的大函数组成！ 7. 系统应该由许多短小的类而不是少量巨大的大类组成！ 8. 把子程序提取另一个程序，不会降低整个程序的复杂度，只是把决策点转移到其他地方。 9. 但是可以降低你在同一时间必须关注的复杂度水平。重点是要降低你需要在头脑中同时考虑的项目的数量，所以一个给定子程序的复杂度是有价值的。 **10. 通过大量真实案例的代码进行分析函数的代码的组织技术** **内容七：函数的错误处理和异常管理(0.5 小时)** 1. 函数的错误处理 2. 使用异常而非返回码 3. 依赖磁铁（Dependeny magent） 4. 主体流-明确表达控制流的主体 5. 异常流-尽可能清晰地表达异常控制流，而不干扰对主体流的表达 6. 标准的异常处理 9 种方法 7. **通过大量真实案例的代码进行分析函数的错误处理和异常处理**	
第三篇： 编程是一种习惯-------编程习惯与管理篇		
第 4 单元 代码重构（1 小时）	**内容一：代码重构** 1. 重构必然性 2. 破窗效应与技术债务 3. 实际重构遇到的 4 大问题 4. 介绍常见的重构技术 5. 重构到模式的目录 **6. 通过多个案例分析，重构面临的问题和解决之道**	
第 5 单元 修改遗留系统代码(1.5~2 小时)	**内容一：修改遗留项目代码** 1. 必须修改遗留的代码起因 2. 遗留代码修改危险事项	

	3. 如何对依赖代码做测试 4. 依赖代码的感知与分离 5. 依赖代码修改的接缝技术 6. 修改依赖代码的工具 7. 降低风险的措施 8. 接依赖技术 **9. 通过多个大型项目案例分析，如何修改遗留代码，分析如何解耦** **内容二：拒绝退化-如何修改遗留系统，而不破坏现有系统结构** 1. 拒绝退化—“首先不要伤害” 2. Sprout Method 3. Sprout Class 4. Wrap Method 5. Wrap Method **6. 通过案例分析，如何修改遗留代码，而不破坏现有系统代码结构**	
第6单元 代码质量体系最佳实践(1 小时)	**内容一：代码质量管理 4 个现代化** 1. 代码管理的 4 个现代化 a) 质量量化（如何设置质量指标） b) 工具化（如何寻找合适的工具 c) 自动化（把流程自动化，忘记流程） d) 持续优化（反思与优化） 2. 多家电信研发中心，如何实现 4 个代码现代化 **内容二：代码静态分析工具** 1. 代码静态分析工具概述 2. 通过案例演示工具在项目之中的应用 **内容三：代码质量管理体系** 1. 结合国内多家研发中心的代码管理经验分享 2. 代码质量体系的建立	