[**做开发十年，我总结出了这些开发经验**](https://www.cnblogs.com/qcloud1001/p/6606060.html)

康亮，腾讯高级工程师。历经网易在线游戏事业部、百度客户端部门、腾讯研究院、腾讯MIG。横跨多个平台10年开发，目前负责腾讯翻译君app。

在一线做了十年的开发，经历了网易、百度、腾讯研究院、MIG等几个地方，陆续做过3D游戏、2D页游、浏览器、移动端翻译app等。

积累了一些感悟。必然有依然幼稚的地方，就当抛砖引玉，聊为笑谈。

**一、对于团队而言，流程太重要了**

行军打仗，你需要一个向导；如果没有向导，你需要一个地图；如果没有地图，至少要学习李广，找一匹识途的老马；如果你连老马也没有，那最好可以三个臭皮匠好好讨论，力图胜过一个诸葛亮；如果三个臭皮匠连好好讨论也做不到，那就是典型的乌合之众了，最好写代码前，点上三炷香，斟上一杯浊酒，先拜拜菩萨，再拜拜谷歌。

我个人属于性格温和的（程序员大多性格不错），但确实见过少数强势的人，说很多强势的话。在技术上一言而决，一听到任何反对就上升到私人恩怨。这样的风格，到底是刚愎自用，还是胸有成竹，就需要仔细判断了。

为什么说流程重要呢？实际上，如果团队上有孙悟空存在，去西天取经，大概也不需要什么流程，只要方向就可以了。 但作为普通的战士，应该先虑败。找人算命时，应该先听听不好的地方，好的地方就不用听了，总归是好的，不好的地方一定要听，这样才能规避。

这就是我的态度：先悲观一点，划清底线，考虑在这个底线上你该怎么做？

这是我做开发的一个习惯，但这个习惯肯定不适用于买房。

怎么划清底线呢？就是假想团队中没有孙悟空了，光靠你唐玄奘、猪八戒和沙和尚，应该怎么去取经。

这个月走什么地方，遇到山怎么走，遇到河怎么过，遇到路上有妖怪劫道，谁去抵挡。遇到路上有少女要搭救，怎么办？这就是流程，是原则。

我经历过一个流程很混乱的阶段。都是很多年前的事情了，可以拿出来说说，不涉及单个人。

2011年在百度浏览器团队时遇到几件让人影响深刻的事情。 有一次开会，产品拿出Google某个产品的DEMO，里面有一段很酷炫3D 效果，要求开发加上，只给2天时间，大家目瞪口呆。后续的开发为了赶节奏，导致非常多的bug，又为了修改bug，leader将所有的bug按照人员平均分配，导致不同模块间的同学相互修改。。。。。实在难以想象。好比让做花卷的厨子，去修改西湖醋鱼的味道。

最初的现象是：bug下降的慢，延伸bug反而增加，每个人都累的半死，代码风格极其杂乱，为了赶工导致的临时方案层出不穷；

到了中期：人员离职越来也多，代码难以维护，新加的需求与之前的临时方案冲突。

到了后期：想做一些修复，想调整架构，又要保证正常运行，其难度好比在一架飞行的飞机上拆换零件。

然后我也急忙离职了。。。。实在看不到成功的可能性。

后来到了腾讯的团队，感觉流程就规范多了。需求和bug有Tapd跟踪，产品发布按照节奏，需求提出前会和开发反复讨论可行性，有专门的质量跟踪，有专门的用户反馈，每天知道要做什么，也知道明天要做什么。有产品需求，也有开发需求！这个非常重要。很多团队，都是只有产品需求，开发好像牛一样，耕完地就不管了？

流程其实没那么复杂，就是各司其责+节奏。我们都是“哆瑞咪发梭拉西多”中的一员，各自有各自的责任，然后组合在一起，按照一个节奏跑起来。把该做的事情与该跑的节奏定好。

**二、不要炫技，老老实实写代码**

网上有一个段子，说有人要用JS实现一个简单的功能，然后朋友给他推荐了几十个库。

真的有必要吗？具体情况具体分析。

居家过日子，你只需要一套普通的工具就可以了；如果你是修车的，你需要一套修车的工具；如果你是光头强，你需要一台伐木机。 吃饭用筷子，用刀叉，都可以，但不要用杀猪刀，不要用丈八长矛！，当然也不能用牙签。

用什么工具，用什么库，问问过来人，多在KM上搜索一下。举个例子：android上加密，用[SQLChpher](http://link.zhihu.com/?target=https%3A//www.qcloud.com/document/product/238/7518%3FfromSource%3Dgwzcw.57237.57237.57237)就可以了，微信也在用，你当然可以学习；数据库ORM思想，用KM上推荐的GreenDAO就可以了；PC上3D引擎，用OGRE就可以了；小型游戏DEMO，用Irrlicht足够；写WebGL，用ThreeJS足够。

首先想想：一些大库hold的住吗，后续发展如何？这些库对安装包的体积影响有多大？有没有调研过同样的产品在用什么？

想清楚了再决定用什么，最好是跟随成功项目的脚步。

**三、架构上实用+适用**

很喜欢曾国藩的一句话：结硬寨、打呆仗。

一字长蛇阵、八门金锁阵，哪个好？iOS都是单个进程，微信Android版本3.5以前是单进程，3.5以后有独立的网络进程； PC浏览器的进程架构更加复杂，UI进程、内核进程、Render进程，而且还有根据页面多少的进程调节模型。

这些设计都很好，各有各的道理，都适用于当前的产品。所以我的观点是：首先分析当前产品的规模、性质，然后再设计架构。

在当前阶段达到：开发效率+架构的平衡；并向后展望3个月，或者半年左右，看看架构能不能适应。

我做腾讯翻译君时，曾反复犹豫要不要模仿微信加入独立的网络进程。后来逆向了有排在第一二位的竞品，最终采用了现在的主功能单进程模型。

产品规模、人员规模、功能阶段，具体问题具体分析。

**四、既要有攻城之力，也要有熬战之气——BUG**

产品开发完成后，必然有bug。其实开发人员在工作过程中，是有一定的直觉或者心理预判的，即：某个功能模块的质量如何。 这里面的质量包括：可维护性、扩展性、算法\渲染效率，还有就是bug与崩溃率。

功能开发完成后，就要开始守城了。

bug，一部分产生是由于架构带来的，例如比较复杂的架构，会导致复杂的实现细节；

但还有很大部分bug，其实是基于如下三个原因产生的：

1 . 对于某个api的不了解，或者对于某个平台，或者SDK版本的不了解。 举例而言：andrid里面非主线程，是不能直接处理UI相关的事情的；JAVA的内存释放也不是绝对的，相互指向是无法释放的；函数个数是有DEX问题制约的---------------------这些bug的产生，也是开发人员摸索学习的过程，经历过一次就不会再犯了。这是学习广度与熟练度的问题；

2 . 还有一些bug，是由于粗心大意导致的。例如空指针的问题，野指针的问题。在C的开发中，野指针的问题，GDI句柄的释放问题，这些都是严谨的代码需要避免的； 而又一些工具，或者方法是可以规避这些问题的，例如android中的利用@Nullable和@NonNull加强空指针检测等方法；

3 . 还有一些bug，是由于“使用情况各异导致的”。例如：偶现在某个模块crash。这里的本质还是因为逻辑的异常边界没有处理好。例如android上的OOM问题，还有PC上UI焦点导致的对象释放问题。这些异常情况，一部分靠测试发现，一部分靠用户反馈，还有一部分就靠自己的异常处理。例如Android中的try catch机制，其实就是遇到异常了，你能纠正错误的机会。

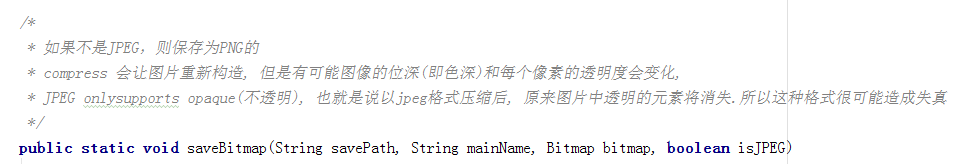
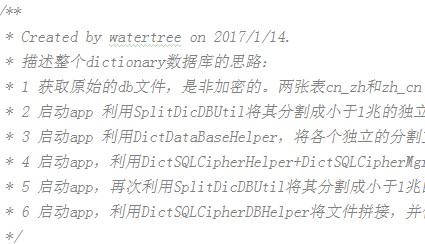
**五、自审**

每过一段时间，都要站在高空俯视自己，问问：到底是在承担过去，还是在改变未来。

如果之前程序代码质量不好，后面修改问题的时间就会比较多。到了开发的中期，得多问问自己，你在不停的改正以前的错误，还是在做新的东西。 如果修改错误的时间多一点，那就要注意自己的代码质量了！

**六、注释**

我很喜欢写注释。有大牛说：代码就是最好的注释。 可惜我还没有达到那个程度。所以，我会把注释写的非常清楚。其一：为了自己以后维护的方便； 其二：为了其他人接手的方便。



这是我在翻译君项目中写注释的方式。1：对于很复杂的逻辑，务必用12345的顺序依次写清楚；2 ：对于函数中的某个参数，需要解释为什么要设置这个参数，尤其是公用工具类里面的函数---说清楚参数的背景含义，可以让其他调用者理解的更加清晰。

我一般不用英文写。虽然这样看起来格调很低，但胜在大家都能轻松的看懂。写代码不能太傲娇，写注释也不要太傲娇，目的是让你的搭档或者接手者，更轻松的理解，让她/他少加班。

**七、代码结构**

代码结构要清晰。有按照功能划分的，有按照UI结构划分的。还有公用工具类，有数据管理，有主逻辑控制。不管用哪种思想，有序的代码结构，可以让每个人感觉很干净。好比日本的收纳整理技巧让很多小资推崇，无非就是干净、整洁、便于管理。

而且，还有一个重要的好处：代码结构表现出来的其实是——程序的一个模块\逻辑思想——让大家工作在不同的区域。

**八、代码风格**

代码风格统一！好比一家人，有叫Tom的，有叫安东尼的，还有叫流川枫、石破天、圣杰夫拉斯基，无所适从。理论上，看一个函数，就能从名称上区分哪些是成员变量，哪些是局部变量，哪些是全局静态值。

除了命名统一外，还有一行代码最大的宽度，函数的连续调用长度等，头文件的包含风格，也最好有一个约定。类的出现时间，创建人名，最好也加上，看起来没用，但到了追踪问题时，就能看出时间线的好处。

**九、安全与逆向**

这是针对Android说的，还有PC插件也需要考虑。Android上首先要防止被别人逆向，我成功逆向并重新打包过有第一位和第二位的竞品。这似乎有点不可思议，但确实做到了。加固+混淆+代码判断，最好都有。

安全上，可以看金刚扫描的漏洞，逐一修改就行。公司很多工具很好用的！

**十、开发效率**

开发效率可以用这些方式提升：

1 . 构建公用工具类，方便大家使用

2 . 使用开源的一些包，例如ORM思想的数据库等

3 . 可以很快的找到问题。开发中，找bug的时间，往往是很多的。我用的方法有3个： 使用try catch； 拦截所有crash到我指定的地方；超多的Log，Log有统一的控制开关。

4 . 借力：数据上报用灯塔，崩溃上报用bugly，公司KM上很多经验，拿过来用。

**十一、安装包体积**

1 . TINY压缩图片

2 . 删除无效的资源文件

**十二、UI渲染效率**

UI是用户的第一感觉；UI快并稳定，第一感觉就不会差太多；管理好内存，基本管理好了一半crash；管理好UI，等于管理了人机交互感受。

UI上的开发是：渲染效率与渲染效果的平衡。