# 技术吐槽

## 稳定性和效率关系

2012-8-8

线上运行系统有一个矛盾的点，就是如果要对其进行修改，就会引入潜在的问题，进而影响系统的稳定性，当然这只不过是一种潜在的风险。而这种潜在风险的高低有一些影响因素。

1. 对现有系统的熟悉程度
2. 对修改技术的掌握，比如重构技术等（重构技术对于修改软件来说非常重要，如果有非常高超的重构技术，那么在对系统不是很熟悉的情况下依然能修改出来高质量的系统，也就是说高的重构技术可以弥补对系统的不熟悉）。
3. 现有系统的架构，如果现有系统的架构比较好，耦合性很低。那么新修改就大大降低了影响范围。
4. 流程的规范程度（开发和测试）。
5. 开发人员的责任心和细心程度（在所有的工作中，一定首先考虑“人是不可靠”的这一前提，然后在这一前提之下做好其他相关工作。这样人如果是可靠的，那么事情会处理的非常顺利。当然事情还是人做出来的，所以最后一条我引入了人的责任心和细心这一因素。还是要相信有一些人可以干好非常细致和繁琐的活而不粗心）

某天听到两个人在吵这件事情，一个激进派觉得现有系统的一些缺陷影响系统的性能，需要进行修改。而另一个属于保守派的人则认为，你的修改可能会引起系统的不稳定，导致系统出现故障。为修改这一个问题不划算。两个人争的脸红鼻子粗。

对于任何一个系统来说，如果要修改，则制定一个稳定性和高效率之间的一个可接受比例，这个比例的制定根据对系统的了解情况和系统的质量来决定，根据以往的经验可以做一个估值，比如我一个修改能提高了50%的效率或者性能，但是这一块的修改会影响到30%的核心，那么就应该考虑修改。经过规范的流程和优秀的技术可以将影响降低很多。

所以，结论就是，在保守（稳定性）和激进（效率）之间根据自己系统的情况设定一个估值比例（改还是不改的估值）。当超过这个比例之后以稳定性为重而考虑放弃高效率，如果在这个可允许的比例之内，则不用害怕影响稳定性而显得非常保守，以至于不敢对系统进行效率上的优化和改进，这样日积月累终将会导致系统虽然很稳定，但是已经没法使用了。

其实在上面的这个估值过程中，还有三个至关重要的因素，一个是领导是否属于保守者，另一个是用户是否属于保守者（固步自封者），还有一个就是真出问题了责任能否担的起。

## 重构时机

2012-8-9

在对项目的业务和代码不是很熟悉的情况下，在添加功能或者修改代码的时候，尽量遵守和之前代码一样的风格和逻辑，即使以前的代码风格和逻辑很不好，此时不管你有多么大的重构冲动，还是应该忍耐一下。但是你可以做一个重构标记，作为一个任务待定。切忌装13而贸然标新立异的改动，记住即使风格混乱，只要混乱的非常一致，那也不错，因为风格没有对错，只有是否一致。如果你需要修改的代码你觉得很不爽，但是人家全部的代码都是这种风格，那么你就需要好好考虑一下。你的改进也许使得代码变得真正的混乱和不一致了。

正如一排走方阵的队伍，如果大家都迈脚迈错了（或者大家都顺拐了），那么对于整个队伍来说还是整齐的。当然观众可以通过这种“错误”的逻辑规则来理解这种方式。但是突然队伍里面来了一个人，他觉得大家都走错了，或者这种走方阵的方式觉得太变态，太out，太没有技术，自己用这种方式都觉得丢人。这时候他自己修改了之前的这种出脚的方式（比如之间都是先出左脚，而他修改为先出右键以显得更为先进和牛逼或者“正确”），这样就出现了有迈左脚有迈右脚的情况。这种不一样的情况出现之后，观众看到的就是混乱和无法理解，或者规则就会变得更加负责和说明文档变得更加厚实。

那么对代码的熟悉如何量化？1，不熟悉此段的业务逻辑，2，不了解此段代码的全局调用关系。

如果此段代码相对独立，调用关系单一，或者你掌握了全部的调用关系，那么即使业务逻辑你不是很熟悉，那么也可以利用重构理论在不改变现有代码逻辑和实现的情况下进行代码结构和风格的重构。

软开心法一||声明

经过考虑，准备将两年前开始写的一本给自己看的书连载出来。这本书主体已经完成，但是会不断的扩展。暂时的名字如题目所示《软开心法》——即软件开发心法。面向软件开发人员，内容涵盖比较广泛，当初名字也是改过多次，在这里暂时以此为名，如有改动再行说明。

本系列的内容最初来源于在一个技术论坛上的连载，当时受版主所邀，开始在论坛写设计模式，所以最初的公开文章应该是在求实bbs论坛。不过在那期间部分文章我也在自己的baidu博客上公开过一些。但不管怎么样，这几个地方的文章所有权都归本博主，本博主保留对此文章的一切权利。

由于本系列都属于技术文章，而技术不是我发明的，所以文章所述都是自己对已经存在技术的理解和心得，还有一些是从事技术过程中的经验体会。当然有些观点可能是出自某个技术书籍或者文章，这些我都会在下一篇说明。

文章中涉及到的代码已经公开在google code和github，你可以通过搜索dprun来得到，C#的代码已经完成，所以基本停止开发。我希望能有对应的java版本的代码，虽然差距不大，也不影响阅读和理解，有了总是有好处的。但是由于时间有限没有完成，当然如果你希望加入开发，那么你可以在github上fork此项目，修改之后可以告诉我，如果我觉得可以会合并你的版本。或者在google code上加入开发组，也可以直接联系博主。此代码通过GPL协议发布，当然也遵守google code和github的相关协议和约定。

关于本系列的传播与发布说明，本系列文章全部是原创文章，你可以转载，但是必须注明，如果有其他用途可以通过联系本博客的主人来说明情况，禁止以营利为目的的公开发表和侵犯本博主对文章的所有权。

最后如果你有什么问题可以和我讨论，或者直接留言，或者通过邮件方式（email可以在本博客的左侧“个人简介”或者顶部的“关于我”中看到）。我非常希望批评和讨论，因为这是推动技术和进步的最好方式。

所有已经发表的，都算公开了，这样如果你想要pdf版本的，可以通过邮件免费索取。当然我也会不定期的将公开部分以pdf文档的形式上传提供下载。

软开心法二||参考文献及推荐阅读

我的任何表达都不会离开自己的技术背景，而文献列表在很大的程度上能表现出来写作的技术背景。所以我在声明之后首先将参考文献列示出来，因为有些表达或者思想我也不能明确来自哪本文献的哪个部分，但是坦白的说，这些文献确实影响了我，有些甚至深入了骨髓，以至于一些时候我所写的都不过是对文献原文转述或者类似表达而已。

下面我将列出本系列文章的主要参考文献，因为这些都是在写作的时候手头会翻的，或者遇到问题会查阅的。我不得不说，这些列表中的书籍都是很经典的，基本都是我认真阅读过至少一遍以上的。你即使不读本书也希望能读一读这个参考文献列表中的书籍，因为更多的时候我觉得这个列表是我的推荐阅读列表。我希望每一个计算机相关专业或者软件开发的人员都能读一读。

下面参考文献列表，名称和版本我是以我书桌上的为准，是否有最新的你可以去查。

[1] Michael Blaha,James Rumbaug.UML面向对象建模与设计.北京：机械工业出版社，2006.

如果你想了解UML及面向对象方面的知识，那么这本书非常不错，前面是UML的基本知识，后面对于软件工程和面向对象的讲解非常不错，由于看真本书之前已经了解了UML的知识，所以这本书对我最大的影响是其中关于面向对象和软件工程思想的一些见解。

[2] Erich Gamma,Richard Helm,Ralph Johnson,John Vlissides.设计模式——可复用面向对象软件设计.北京：机械工业出版社，2000.

真本书是业界所称的GOF的宝典，设计模式的源头书籍，使用C++讲解，是一本博士论文，所以还是比较晦涩的。

[3] 阎宏.Java与模式.北京：电子工业出版社，2002.

相比上面的一本设计模式书籍，这本非常厚的书籍很详细，代码用java实现，例子很生动。

[4] Bruce Eckel.Java编程思想.北京：机械工业出版社，2007.

这本书是java方面不错的一本书，我接触java的时候用java大全入门，然后就看的这本书，书籍很不错，坐着也是C++编程思想的作者，对于面向对象的理解比较透彻。

[5] Christian Nagel,bill Evjen,Jay Glynn.C#高级编程.北京：清华大学出版社，2006.

因为我的书籍中代码实现使用C#，所以有一些语言语法上的东西，我会从这本书中查找，对于C#的学习者来说，刚开始通读一下本书是一个不错的开始。

[6] Pressman，R.S.软件工程——实践者的研究方法.北京：机械工业出版社，2007.

这本软件工程书籍很不错，我们当初使用的教材就是这本，在工作中我也不断的看这本书，从实践者的角度来解析的软件工程的实用性。书籍中的很多工具和建议都非常不错。

[7] Joshua Kerievsky.重构与模式.北京：人民邮电出版社，2010.

这本书英文是refactoring to patterns，从英文上可以看出来，是重构到模式，汉语的翻译会影响对本书的理解。本书是通过对现有的代码重构，可以引入设计模式来重构已有的代码。对于理解设计模式来说，非常有帮助的一本书。但是里面只有部分的模式。

[8] Matin Flow.企业应用架构模式.北京：机械工业出版社，2004.

Martin Flow是我的偶像，如果想从宏观上来理解软件架构，针对企业级的软件架构发法，这本书写的很不错，强烈建议读一下，不同时间读就会有不同感受的一本书。

[9] Matin Flow.重构——改善既有代码的设计.北京：中国电力出版社，2006.

这本书我一直认为是做软件开发必读的，他会教你如何写代码，如何写高质量的代码，看了之后你就觉得，写代码也是一门艺术。

[10] Dino Esposito, Anrea Saltarello.Microsoft .NET企业级应用架构设计.北京：人民邮电出版社，2010.

对于想了解.Net技术架构的人来说，这本书应该读一读。

[11] Jeffrey Richter.CLR Via C#.北京：清华大学出版社，2010.

这本书不用说了，如果你是搞C#或者.Net开发的，你应该多次仔细的看一下本书，不然你会好长时间也不能进入.Net美好的大门。

[12] Donald Norman，梅琼译.设计心理学.北京：中信出版社，2011.

这是Norman的书，感觉和计算机没有说明关系，但是我要说的是，每个人都应该看一下这本书，并且能理解其中的精髓。理解了这本书之后你的脾气就不会那么大了，并且能够在各种地方用到，包括写代码，设计，接口调用等。

[13] Alan Cooper,Robert Reimann,David Cronin,刘松涛译. About Face 3 交互设计精髓.北京：电子工业出版社，2008.

针对计算机的交互设计，作者是cooper的创始人，交互设计之父，vb之父。他对于交互设计和计算机的理解比较深入，做程序应该看一下这本书，不要做一个“不解风情”的开发人员。

[14] 张亮.细节决定交互设计成败.北京：电子工业出版社，2009.

这本书简单易读，如果觉得上面的两本不好读，那么这本也非常不错。我非常强调用户交互，并且倡导将交互理论引入代码设计中。并且用户交互不仅针对计算机，对于用户和工业设计来说，都可以看一下。

[14] Scott W.Ambler,Pramodkumar J.Sadalage著,王海鹏译.数据库重构.北京：机械工业出版社，2007.

因为我专门针对重构技术进行了研究，而本系列文章中也有对此的描述，所以传统的认为重构只能对于应用程序，但是这本书教你如何对数据库进行重构，虽然看着像雷池，但是技术是可以减少风向以达到可行性的，如果你是重构的粉丝，那么你一定要看此书，不然你对重构的理解就比较狭隘。

[16] Eric Freeman,ElElisabeth Freeman等.Head First设计模式.北京：中国电力出版社，2007.

这本书我是最初了解设计模式的时候看过的，第一次看的是GOF的设计模式，看过之后稀里糊涂的，因为是学生并且GOF的书本来不适合入门，但是这本书对我印象很深刻，如果你想了解设计模式，并且觉得GOF的书籍太晦涩，那么这本你可以看看。

[17] 梁爱虎.SOA思想，技术与系统集成.北京：电子工业出版社，2007.

这是一本关于SOA方面的实战性书籍，讲述了SOA的应用。

[18] Connolly,T.M,何玉洁等译.数据库设计教程.北京：机械工业出版社，2005.

脱离了SQL的数据库，如何设计，面向对象和对现实的理解如何转化成高效的数据库，这本书你应该看一下，否则你永远不明白数据库设计的奥秘，并且会阻碍你的进步。

[19] Andrew S.Tanenbaum,陈向群等译.现代操作系统.北京: 机械工业出版社，2005.

操作系统是计算机的基础，这本书对于操作系统的讲解很不错，并且操作系统中的一些理论和原理对于普通程序的设计来说非常都是有借鉴意义的。比如我就常常使用“鸵鸟算法”，如果你不知道什么是“鸵鸟算法”，那么你该看看这本书。

[20] David LeBlanc, 程永敬等译.编写安全的代码.北京: 机械工业出版社，2005.

这是一本教你如何写代码的书籍，如果说《重构》教你写好看的代码，那么这本书就教你如何写安全的代码，如果你想提高自己的代码功夫，那么这本书还是得看一下。不然你不会知道代码中的白名单和黑名单是啥意思。

这些书籍是我在写这本书的时候直接参考过的，当然还有很多没有直接参考，但是凭借自己的记忆和理解，在本书中也都有体现，不一一列举了，但是我看过的一些书籍都在豆瓣的读书列表中有维护，你可以在豆瓣上查找用户Stallman（没错，这是我从偶像Richard Stallman的名字里面摘出来的）的读书列表。当然这个列表也不全，因为开始维护的时候之前读的一些也没有全部维护进去。

所以这篇作为声明的补充部分，作为对相关作者作品的尊重。

三 说明

三 目录

四