**软件工程师的能力及发展**

**介绍:  个人软件流程,**

**职业发展，**

**个人绩效的衡量和提高,**

**软件开发是科学, 工程, 手艺, 或是艺术**

软件工程, 它包括了什么呢?

|  |
| --- |
| 软件工程包括了开发，运营, 维护软件的过程中有很多技术, 做法, 习惯, 和思想。软件工程把这些相关的技术和过程统一到一个体系中, 叫 “**软件开发流程**”，软件开发流程的目的是为了提高软件开发, 运营, 维护的效率，以及用户满意度, 可靠性，和软件的可维护性。 |

软件开发流程不光指团队的流程,  软件团队是由个人组成的。在团队的大流程中, 是每一个具体的个人在做开发，测试，用户界面设计，管理，交流等工作。因此, 个人在软件团队中也有个人的流程。

[](http://images.cnblogs.com/cnblogs_com/xinz/201110/201110220010104911.png)

个人的劳动成果有序地组织起来, 就是团队的流程。这里说的“有序”, 并不是“无争论”,  在大部分成功的软件团队模型中,  各个角色（开发, 测试, 项目管理等）考虑问题的出发点是有区别的, 不同意见的冲突在所难免, 一个好的团队流程能把冲突的积极方面 (各自尽力把自己的工作做好，说服别人) 释放出来,而避免消极方面 (因为冲突而产生的消极，抵触情绪等)。

[](http://images.cnblogs.com/cnblogs_com/xinz/201110/201110220010135308.png)

我们用足球作一个比喻:  足球中有没有个人流程?  当然有,  职业球队对于运动员有很严格的要求: 例如:

体能, 技术, 意识, 斗志

具体技术有传接，盘带，射门, 定位球, 跑位, 等。一些特定的角色（守门员）还有独特的技术要求。

[](http://images.cnblogs.com/cnblogs_com/xinz/201110/201110220010211084.png)

足球的团队有没有流程? 当然有:

阵型, 配合, 临场应变

足球队有不少 “阵型” (442, 433, 451和它们的各种变体, 等等) 和打法 (南美，欧洲，技术，力量, 小快灵, 抢逼围, 两翼齐飞,  全攻全守, 等等).

[](http://images.cnblogs.com/cnblogs_com/xinz/201110/20111022001029514.png)

尽管有这么多理论,  足球的每一次盘带, 传球,  跑动, 射门，扑救，都是单个运动员完成的。  如果单个运动员的技术, 体能不行, 无论是什么阵型用处都不大，有些阵型反而会起反作用,  例如, 让体力弱的球队去打全攻全守。

软件也是这样

软件系统的绝大部分模块都是由个人开发或维护的。在软件工程的术语中, 我们把这些单个的成员叫做Individual Contributor (IC).

IC 在团队中的流程是怎么样的呢? 我们以开发人员为例:

•    **理解问题或任务**

**•       提出多种解决办法并估计工作量**

**•       其中包括寻找以前的解决方案，因为很多工作是重复性的 – 例如实现某些类似的web页面。**

**•       与相关角色交流解决问题的提案, 决定最终方案**

**•       执行, 把想法变成实际中能工作的代码**

**•       修复缺陷,  对结果负责**

每个人的工作质量直接影响最终软件的质量。

如果你是病人, 你希望你的医生是下面的那一种呢?

a)  刚刚在书上看到你的病例, 开刀的过程中非常认真严谨, 时不时还要停下来翻书看看…

b)  富有创新意识, 开刀时突然想到一个新技术, 新的刀法,  然后马上在你身上试验…

c)  已经处理过很多类似的病例, 可以一边给你开刀, 一边和护士聊天说昨天晚上放的 《非诚勿扰》的花絮…

d)  此医生无正式文凭或医院, 但是号称有秘方, 可治百病。

e) 还有这一类, 给你开刀到一半的时候, 出去玩去了, 快下班的时候, 他们匆匆赶回来, 胡搞一气, 给你再缝好, 打了很多麻药，就把你送出了院, 说“治好了”！

事实上, 很多软件项目就是用 a) b) 这样的方法搞出来的。当然也有一些人走 d) 这条路。 e) 这一种可以在很多大学生软件大作业上看到例子。

如果我可以选择, 我要选 c) 那样的医生。

**软件工程师的成长**

一个游戏玩家, 如何成长呢?   在游戏中, 可以打怪, 挖宝, 买装备, 完成任务, 买卡换积分,  等等。。。

那一个刚入行的初级软件工程师如何成长呢?  我认为成长有下面几种:

**1.       知识:  对具体技术的掌握, 动手能力**

例如: 对Java, C/C++/C#, 诊断/提高效能的技术,  对device driver, kernel debugger 的掌握；对于某一开发平台的掌握。

**2.       经验: 对问题领域的知识和经验的积累 (例如: 对于医疗行业的了解, 对于金融行业的了解)。**

第一点和第二点, 也可以比较容易地检测出来。随着经验的增长, 一个工程师可以掌握更广泛，更深入的技术和问题领域的知识。

**3.       通用的软件设计思想,  软件工程思想的提高**

这一方面就比较虚，什么是好的软件设计思想, 什么是好的软件工程思想? 一个工程师开了博客, **转发**了很多别人的文章, 这算有思想么? 另一个工程师坚持任何设计都要画 UML 图, 这算有思想么?

**4.       职业技能 (区别于技术技能)**

职业技能包括: 自我管理的能力; 表达和交流的能力; 与人合作的能力; 把任务按质按量完成的执行力; 这些能力在IT 行业和其它行业都很重要。

**如何衡量, 证明自己的能力?**

问: 你是职业软件工程师么?

答: 是。

问: 你觉得你“职业”到哪一个程度?

答: 嗯, 我在一个能发工资的地方上班, 靠我的软件技术挣钱, 所以我应该是相当职业的。

问: 像职业篮球队员那样职业?

答: 差不多吧。

问: 职业篮球队员都有很详细的记录说明， 例如，下面的网页说明了一个职业篮球队所有队员的详细情况:

<http://china.nba.com/stats/teams/teamStats/1610612745_2010_2_00.html>



出场次数, 场上时间, 命中率, 篮板, 助攻,  抢断,  盖帽,  失误,  犯规,  得分, 罚球命中率等。

作为一个职业软件工程师, 你有类似的数据说明你所有职业活动和成绩么?

答: 嗯…  唯一的数据是，我的 “上场时间”  还是挺长的, 而且**经常打加时赛**。

什么样的数据能说明一个工程师的技术和能力呢? 衡量能力有哪些参数? 没有量化的指标, 就谈不上衡量和比较。我们还是看看搬砖的伙计们，关于工作量, 他们有:

         有多少块砖？

         要搬多远?

他们也有简单的指标衡量工作质量:

         多快搬完?

         搬的过程中损坏了多少块砖?

**软件开发的工作量和质量怎么衡量呢? PSP认为有下列4 个因素:**

a)       项目/任务有多大？

说明项目的大小, 一般用代码行数 (Line Of Code, LOC) 来表示；也可以用功能点 (function point). **一个重要的指标是: 你在实际产品中写了多少代码, 不包括空行/注释行/单字符行** .

b)       花了多少时间?

可以用小时, 天，月，年来表示。一组人所花费的时间可以用 (人数\*时间) 来表示，例如某项目花费了10个人·月。

c)       质量如何?

交付的代码中有多少缺陷?  交付有两个定义,

         在 Code Complete “代码完成” 的时候, 交付给测试人员

         交付到顾客那里去 (在软件交付的时候)。

可以用缺陷的数量来除以项目的大小。  例如 5 bugs / KLOC，  意味着每千行程序有5个缺陷。

也有人用试图用“re-work”来表示质量, 例如：这1000 行代码, 从开始写到最后发布, 一共修改了200 行·次。  另一组代码, 从开始写到最后发布, 一共修改了 50 行·次。那我们做改得少的代码最初质量高– 因为 re-work (返工) 的次数少。

 re-work 只是表明在软件开发过程中花费的时间，re-work 的多少并不和最终的质量成正比关系。软件开发过程很大程度上是一个探索和实验的过程, 不同的 re-work 能帮助工程师深入了解项目的各个难点,尽早交付原型, 找到最优解决方案，等等. 因此 re-work 是有价值的。当然, 如果一个程序员为了一个简单的问题而不断地re-work, 其工作效率就不是太高– 这可以用时间花费来考虑。

d)       是否按时交付?

软件/任务是否按时交付？这个看似简单, 其实也有讲究。例如, 当我们衡量一个程序员在一段时间内的交付情况时, 我们是用简单的平均值呢, 还是用方差来表示? 考虑这个例子:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 我们来看看下面两个程序员的比较 (Al 和Bob).  他们在两次项目中各自完成3 个任务。平均值显示,  Al 完成任务的时间从 10 天提高到了 7 天; Bob 从10 天提高到了 8 天.  看起来是 Al 更好?   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Assignment** | **Al’s estimation** | **Al’s delivery** | **Bob’s estimation** | **Bob’s delivery** | | 1,2,3 | 10 days | 5, 10, 15 | 10 day | 5, 10, 15 | | 4,5,6            (3 months later) | 7 days | 1, 9, 11 | 8 days | 7,8,9 | |  |  | Average = 7 |  | Average=8 | |  |  | StdDev=5.3 |  | StdDev=1 |     在第一轮的工作中,  Al 和Bob 都完成了3项估计为 10 天的工作, 都各自花了 5， 10， 15 天.  在三个月之后,  Al 和Bob 接受了另外三项任务,  Al 的估计都是 7 天, 他花了 1， 9， 11 天。平均是7天时间。  Bob 估计是8 天, 用了 7， 8， 9 天时间.   从总用时来看, Al 的平均时间比Bob 快一天, 似乎应该是稍稍优秀一些, 但是从标准方差 (Standard Deviation) 来看, Al 的方差是5.3，而Bob是1.  显然Bob 比Al 的交付时间要稳定得多。在团队工作中, 稳定，一致的交付时间是衡量一个员工能力的重要方面。    杰克·韦尔奇也讲过:  Standard deviation is key to the quality of service, and key to build trust. |

**职业发展**

问: 你学计算机最喜欢什么?

答:  写程序多爽啊。

问: 那工作以后怎么发展呢?

答: 有更多的时间写程序, 而且还有工资, 太爽了。

几种方法:

         PSP

         考级

         Steve McConnell  Construx

         Corporate Career Model

         Pragmatic Approach

**职业成长– 考级之路:**

在中国有: 计算机等级考试 (<http://sk.neea.edu.cn/jsjdj/index.jsp>)  还有: 全国计算机技术与软件专业技术资格考试 (<http://www.rkb.gov.cn/>  ) 具体情况参见网站。

 此类考级有这样的好处:

         国家认证, 有一定的权威性和通用性

         任何人都可以参与

也有这样一些局限性:

         是一种答题/评分为主要方式的考试形式，没有面对面的口试。

         考试中每个人单独行动, 不能考量团队合作能力。

         要考虑到通用性和稳定性，考题内容相对滞后，部分内容相当滞后。

同时, 很多公司也提供了针对自己产品的职业认证项目 (Certified Program).  例如, 微软公司有 Microsoft Certified Professional (<http://en.wikipedia.org/wiki/Microsoft_Certified_Professional>); Oracle 公司有 Oracle Certification Program (<http://en.wikipedia.org/wiki/Oracle_Certification_Program>).  获得了相应公司和行业的认证, 工程师就可以更容易地获得相应的工作，合同机会。

一些行业协会也有自己的认证项目, 例如IEEE (电气电子协会) 就提供了一些职业认证服务。Link:<http://www.ieee.org/education_careers/education/professional_certification/index.html>

**职业成长– Personal Software Process**

大家可能都听说过CMU 的CMM 团队成熟度模型,  这是用来衡量一个团队的能力。 CMU 的专家们针对软件工程师也有一套模型:  叫 Personal Software Process (PSP)

PSP 和任何其他方法论一样, 也不是一蹴而就的。下图显示了CMU  Personal Software Process 的各个版本的内容 (来源: CMU PSP 网站)。红字标出了每个版本新增的内容:

|  |  |  |  |
| --- | --- | --- | --- |
| ***PSP0*** | ***PSP0.1*** | ***PSP1*** | ***PSP1.1*** |
| Planning  Development  •       Design  •       Coding  •       Test  Postmortem | Planning  Development  **Coding Standard**  Design  Coding  Test  **Size Measurement**  Postmortem  **Process Improvement Plan** | Planning  **Estimate**  Development  Coding Standard  Design  Coding  Test  **Test Report**  Size Measurement  Postmortem  Process Improvement Plan | Planning  **Estimate**  Development  Coding Standard  Design  Coding  Code Review  Test  **Record Time Spent**  Test Report  Size Measurement  Postmortem  Process Improvement Plan |

|  |  |
| --- | --- |
| ***PSP2*** | ***PSP2.1*** |
| Planning           Estimate  Development           **Design Review**           Coding Standard           Design           Coding           **Code Review**           Test  Record Time Spent  Test Report  Size Measurement  Postmortem  Process Improvement Plan | Planning           Estimate  Development           **Analysis**           **Design Spec**           Design Review           Coding Standard           Design           Coding           Code Review           Test  Record Time Spent  Test Report  Size Measurement  Postmortem  Process Improvement Plan |

用最新的版本 (PSP 2.1) 来看看一个软件工程师在接到一个任务之后应该怎么做:

|  |  |
| --- | --- |
| ***PSP2.1*** |  |
| Planning           Estimate  Development           Analysis           Design Spec           Design Review           Coding Standard           Design           Coding           Code Review           Test      Record Time Spent  Test Report  Size Measurement  Postmortem  Process Improvement Plan | 计划           估计这个任务需要多少时间  开发           分析需求           生成设计文档           设计复审 (和同事审核设计文档)           代码规范 (为目前的开发制定合适的规范)           具体设计           具体编码           代码复审           测试（包括自我测试，修改代码，提交修改）  记录时间花费  测试报告  计算工作量  事后总结  提出过程改进计划 |

PSP  的特点:

•       不局限于某一种软件技术 (如编程语言),  而是着眼于软件开发的流程,  这样不同应用的工程师可以互相比较。

•       不依赖于考试, 而主要靠工程师自己收集数据, 然后统计提高。

•       在小型，初创的团队中,  高质量的项目需求很难找到，这意味着给程序员的输入质量不高，在这种情况下,  程序员的输出 (程序/软件) 往往质量不高, 然而这并不能全部由程序员负责。

•       PSP 依赖于数据

•       需要工程师输入数据, 记录工程师的各项活动, 这本身就需要不小的代价。

•       如果数据不准确或有遗失, 怎么办?  让工程师编造一些?

•       如果一些数据不利于工程师本人 (例如: 花很多时间修改缺陷), 我们怎么能保证工程师能如实地记录这些数据呢?

•       PSP的目的是记录工程师如何实现需求的效率,  而不是记录顾客对产品的满意度。工程师可能很高效地开发出一个顾客不喜欢的软件, 那这个工程师还是一个优秀的工程师么?

**职业成长– Steve McConnell 版本:**

Steve McConnell 创立的公司  (Construx Software) 为员工提供了下面的成长路径。

知识：把相关的软件知识分为十大知识领域。

能力：一个工程师对这些知识的掌握分为如下四个阶段:

入门 (Introductory)； 熟练 (Competency); 带头人 (Leadership); 大师 (Mastery)

职业成长级别 (Professional Development Ladder):

把工程师分为6个级别 ( 9 – 15) 每一个工程师属于一个技术等级。一个工程师要从一个级别升到另一个级别, 需要在各方面达到一定的要求。例如, 要达到 level12 ，工程师必须在三个知识领域达到“带头人”水平。  例如要到达“工程管理 (知识领域) 的熟练 (能力)”水平，工程师必须要做到以下几点:

         阅读: 4-6 个经典文献的深入分析和阅读

         工作经验: 要参与并完成6 个具体的工作

         课程：要参加3 个专门的课程

有些水平还要求工程师获得某种专业证书，在工业界，教育界教课, 发表论文等等。

**职业成长– 大公司版本:**

微软公司针对软件工程师的职业发展也有很完备的规划和支持。这方面的资料比较多, 这里简单地以软件开发工程师为例说明一下。  下面的解释部分来自于 Eric Brechner 的书– Hard Code ([link](http://www.amazon.com/Wrights-Hard-Code-Microsoft-Practices/dp/0735661707/ref=sr_1_1?ie=UTF8&qid=1319208448&sr=8-1))

SDE (初级软件开发工程师)

                入门。在学校里学到了一些技能, 但是还没有在实践中得到充分锻炼。

SDE II  (中级软件开发工程师)

独立。可以写别人交给你的任何东西。如果你不懂, 你知道去问谁。

Senior SDE (高级软件开发工程师)

小组领导。影响着3 – 12 个工程师, 或者是他们的行政领导; 或者是他们的技术带头人。怎么显示领导力呢? 这有几个例子:

         你当过新员工的导师么?  他们后来都尊重你的种种教诲么?

         你是否成为别人的榜样?  (写的代码, 做的设计, 别人可以拿来重用)

         你在招人方面是否有心得, 并言传身教, 让大家都认识到面试的重要性和各种技巧。

         你是否创立/改进/推动一些流程,  而且这些流程不需要你亲自参与，也能流传下去?

         在和别的角色 (例如, UX/PM/QA) 打交道的时候, 你往往都能赢得别人的支持,而不是和别人争执, 抱怨不休?

Principal SDE （首席软件开发工程师）

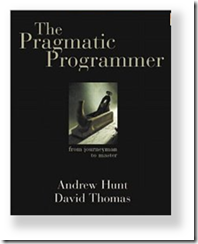
团队领导。影响着12 人以上的一个大团队, 成为影响团队成败的关键人物。

Partner SDE, Distinguished Engineer, Technical Fellow

影响力扩大到整个机构，甚至工业界。

**职业成长– Pragmatic 版本 - 从小工到专家**

IT业界的不少专家也对于程序员的成长提出了不少好的建议.  例如:

[](http://images.cnblogs.com/cnblogs_com/xinz/201110/20111022001036193.png)

这本书的副标题是“from journeyman to master”，从打下手的小工到大师。

它的网页上列出了三十多条建议, 大家可以好好对照一下。

<http://pragprog.com/the-pragmatic-programmer/extracts/tips>

另外有用的经验之谈还不少:

<http://samizdat.mines.edu/howto/HowToBeAProgrammer.html>

**其他话题**

**工程还是艺术**

软件开发是一门工程,  还是一门艺术? 你如何衡量艺术家? 如何衡量创造能力?

如果是一门工程, 那工程师要守规矩; 如果是一门艺术, 那艺术家要创新。

•       写诗歌最多的人是谁?

•       最有创造力的诗人是谁?

一些最有影响力的作家, 他们的作品都非常少，甚至只有一本, 例如:

•        “Gone with the Wind”   by [Margaret Mitchell](http://en.wikipedia.org/wiki/Margaret_Mitchell);

•       <红楼梦> by 曹雪芹  (这一本都没写完!)

另外,  优秀的作品往往并不符合所有”好”的标准。  例如，找出下面这首词中重复的字:

|  |
| --- |
| •      念奴娇 · 赤壁怀古  - 苏轼   大江东去，浪淘尽，  千古风流人物。  故垒西边，人道是，三国周郎赤壁。  乱石崩云，惊涛裂岸，卷起千堆雪。  江山如画，一时多少豪杰。   遥想公谨当年，小乔初嫁了。  羽扇纶巾，谈笑间，樯橹灰飞烟灭。  故国神游，多情应笑我，早生华发。  人生如梦，一樽还酹江月。 |

出现了三遍的有:  江，人;   出现了两遍的有:  国，生，千，故，如。这符合“好词”的标准么?

南宋人俞文豹评价道:

*今人看人文字，未论其大体如何，先且指点重字。*

软件设计工程师们在做代码复审的时候，是看“重复字”的多少, 还是程序的艺术性?

**角色问题**

成熟的产业的一个标志是产业内部的分工比较明确。  软件产业的分工明确么?  我们用当前比较火的房地产业做个比较, 在几个角色中, 我们可以看到“软件工程师”可以应付相当于房地产行业的五个角色。有人说 -  那我找5个种不同的人来承担这五个任务就好了。但是在软件项目中, 这5种人未必能互相沟通, 反而不如同一组人一直工作下去。

|  |  |  |
| --- | --- | --- |
| **房地产企业** | **初级软件团队** | **成熟的软件团队** |
| 房地产老板  •       投标，买地 | 软件工程师 | 投资人 |
| 建筑设计师  •       设计建筑 | 软件工程师 | PM, UX |
| 土木工程师  •       设计土木结构 | 软件工程师 | 高级软件工程师 (架构师) |
| 建筑工人  •       具体建设 | 软件工程师 | 软件开发工程师 |
| 工程监理  •       监管质量 | 软件工程师(测试) | 软件测试工程师 |
| 装修工人 | 软件工程师 | 软件开发工程师 |
| 楼房销售 | 软件工程师 | PM, 产品经理 |
| 楼房中介 |  |  |

**从个人到伙伴，到团队**

上面讲的都是个人的，静态的, 独立工作的能力。

在两人合作的环境中，两个人如何互动，给对方建设性的意见，推动工作进展?

在团队的动态合作中，如何衡量一个人在一个团队中的的绩效呢?