飞漫软件十年回顾

2012年04月06日 19:26

北京飞漫软件技术有限公司（飞漫软件）成立于2002年，今年是第十个年头了。飞漫软件的十年，浓缩了嵌入式软件技术在中国的发展历程。本文将回顾飞漫软件的十年历程。回味过去，或许能给我们的未来发展一些启迪。

一、十年回顾

笔者创办飞漫软件的 2002 年，正是嵌入式软件技术在全球开始得到关注的一年。在此之前，2000 年开始，才有嵌入式（embedded）这个领域被专业人士提及。笔者供职过的深圳（蓝点）有限公司，是国内最早专注于嵌入式软件技术的公司。然而，蓝点因为 2000 年的 .com 泡沫而关张大吉，未能坚持到嵌入式软件开始创造市场价值的那一刻。

此后，笔者供职于北京中科红旗软件技术有限公司的嵌入式事业部。当时，该事业部认准了实时工控领域，计划开发一款名为 ControlLinux 的嵌入式实时操作系统。当时，该产品的规划非常宏伟，从内核、基础库到开发工具均有涉及。然而，因为缺乏基本的市场认知以及研发团队能力的不足，该产品无疾而终，该事业部也在笔者离开之后合并到了其他事业部。当然，中科红旗在过后多年，又重新设立了嵌入式事业部——这是后话。

笔者离开中科红旗之后，即筹备创建了飞漫软件。起初，并没有明确的思路来如何经营这个公司。但开源 MiniGUI 的一些用户给了飞漫软件起步的机会，飞漫软件通过定制 MiniGUI 或者开发一些基于 Linux 和 MiniGUI 的外包项目开始创造收入。飞漫软件也逐步壮大，到 2003 年，有了十人左右的团队，并实现了微薄的盈利。

2004 年，《MiniGUI 及其配套开发工具》项目入选科技部中小企业创新基金，并获得了国家和地方政府超过百万元的无偿资助。另外，华为技术也在 2004 年采购了 MiniGUI，从而获得了一笔不小的收入。这两笔资金，足够让飞漫软件继续发展 MiniGUI，并将 MiniGUI 打造成了一个颇有知名度的嵌入式图形中间件产品。公司也随之进一步发展壮大。2005 年初，和大唐移动签署的 TD 手机合作项目，为飞漫软件转向手机行业起到了举足轻重的作用。

2005、2006 年，飞漫软件基本上保持了 30% 的年增长率，积攒了大量的用户基础，也基本确立了以销售软件使用许可（license）为主的业务模式。

2007 年，飞漫软件获得了一笔外来投资，因扩大研发团队而首次出现亏损。2008 年，金融危机的出现，给飞漫软件的发展雪上加霜，不得不通过裁员来获得生存的机会。2009 年，飞漫软件开始获得联芯（大唐移动）支付的 TD 手机使用 MiniGUI 的提成费，从而扭亏为盈；2010 年，飞漫软件继续保持了良好的增长势头，开发了 mDolphin 等浏览器软件，并保持盈利。

然而，2011 年起，Android 系统的飞速普及，为飞漫软件的发展带来了非常大的不确定性。之前，飞漫软件的主要收入来源于 MiniGUI 等产品在功能手机上的许可费以及军工、工业控制等行业客户的许可费。从 2011 年下半年起，因为 Android 的普及以及冲击，大量的功能手机厂商及芯片厂商缩减了在功能手机上的技术投入，飞漫软件的收入也急转直下。在飞漫软件成立九年之际，飞漫软件面临着成立以来的最大的危机。

面对此市场大势，在一些核心员工的倡导下，飞漫软件从 2011 年 6 月起，开始迈向了向移动互联网业务转型的步伐。在 2011 年 10 月之后，陆续发布了面向 Android 平台的领航桌面、领航浏览器等产品。尤其是领航桌面产品，在上线三个月，即达到了100万激活量的骄人战绩，在国内工具类软件中，各项指标排名前 5%。这一来自市场的积极反馈，增强了笔者及团队的信心，飞漫软件转型移动互联网的目标更加坚定。

2012 年，飞漫软件除了服务于联芯、RDA 等手机芯片厂商、军工客户等重点客户获得 MiniGUI 及其相关软件的技术许可费之外，在移动互联网新业务上将近千万元的投入，将从下半年起带来可观的收入。对此，作为创始人，笔者坚信这一天将在不久的将来来到。

二、成功的十年、失败的十年

通过简单回顾飞漫软件的十年，我们能够明显感觉到，飞漫软件创立于嵌入式软件行业萌芽之时，转型于智能手机崛起之时（也就是所谓后 PC 时代的到来）。飞漫软件走过的十年历程，基本浓缩了中国嵌入式软件行业发展的十年。

笔者之所以说这是成功的十年，是因为飞漫软件打造了一个成功的系统级软件，在中国嵌入式软件技术发展的历程中留下了或浓或淡的一笔。使用 MiniGUI 的各类嵌入式设备，不完全统计至少有两亿部。仅华为终端使用 MiniGUI 开发的数码相框类产品，就接近或超过一亿部出货。另外，功能手机方面，总出货量已接近一亿部，而且该数字在未来的几年内，还将保持一定的增长。

然而，因为对国内各行业对软件价值的鄙视，飞漫软件并不能获得和 MiniGUI 这个产品的市场地位相匹配的收入。当然，笔者说是失败的十年，并不仅仅是这个原因，而是因为我们国家的 IT 行业，在后 PC 时代萌芽的十年窗口期中，并没有任何一家企业可以抓住这个机遇，成为苹果、谷歌这样可以在后 PC 时代创造新的生态系统的伟大公司！想想看，在新千年之初，嵌入式软件技术刚刚得到全球关注之时，我们就有 MiniGUI 这样的开源软件，并具有相当的国际知名度，但为什么没有一家企业可以基于这样的软件以及其他的开源软件（如 Linux、Java、WebKit 等），将其打造成一个类似 Android 或者 iOS 这样的系统呢？显然，这样的任务不是一个仅有不多投资的民营企业可以完成的，而是那些手握重金的大佬们去完成。中国的整个 IT 界，应该为这“失去的十年”感到悲哀。因为这样的十年可遇而不可求，下一个这样的十年在哪里？WHO KNOWS?

我们看看在这十年中，作为我们中国的 IT 界之骄傲的一些公司在做什么事情：

\* 华为技术/华为终端。笔者和华为技术、华为终端打了多年交道。这公司作为中国最具代表性的民营 IT 公司，是我们的楷模，他创造了通信业中国民营企业的神话。不得不佩服。然而，大家都知道，华为终端直到今年，才开始逐步从围绕运营商的市场转向直接面向消费者的开放市场。华为的狼性文化注定了这个企业是短视的，看不到未来十年的发展方向，只能是跟随而不是主导。

\* 腾讯、百度、盛大、新浪等互联网企业。这些公司在这个窗口期，其目的就一个：赚现钱！这些企业在未来的十年内，仍然不能成为想苹果、谷歌这样伟大的、可以创造一个新的生态系统的公司。

\* 各类创业公司。这些公司忙于应付各类创业竞赛、写商业计划书、拜访投资方，能拉到钱就是成功，先烧钱再说，哪有什么心思考虑未来十年？

归根结蒂，浮躁的大环境造就了中国 IT 界的现状——既然很多公司可以没有任何道德底线地生存，谁会脚踏实地地去积累？如果这样做，岂不是被人看成傻子？

接下来的十年，不会再有嵌入式软件这个行当了。嵌入式软件将整个被平台化的系统（iOS、Android、Windows）占据，而这些系统平台，全 TMD 是老美的作品！这就是这十年的悲哀！不仅仅是笔者个人的悲哀，也是中国 IT 界的悲哀。不仅仅是飞漫软件的失败，也是中国 IT 界的失败！

三、软件工程管理

在飞漫软件发展各阶段，我们曾采用过多种软件开发管理模型。

以 MiniGUI 为例。最初，基本上是作坊式的小团队，没有独立的质量保证团队。MiniGUI 1.0 到 2.0 的各个版本，基本上出自本人以及当时公司的另外一个主要创始人Snig。那时，基本上没有什么管理，靠的是兴趣和一腔热情编码。

在飞漫软件开始开发一些 MiniGUI 的外围软件时，比如简易浏览器（mSpider）、PMP 方案（mGallery），很自然地想到引入质量保证团队来协助开发团队保证软件的质量。

时间推移到 2008 年，在我们开发 MiniGUI 3.0、mDolphin 等产品时，飞漫软件内部形成了一套严密的、基于瀑布模型的软件开发管理模型和体系，制定了一系列的软件开发管理规范和工作规范。最多时，围绕 MiniGUI 3.0 开发的人员总人数高达 20 人，其中包括产品管理团队（含产品经理、UI设计师等）、开发团队以及质量保证团队。

直到 2011 年 4 月，笔者从未考虑过我们投入 20 人的团队开发 MiniGUI 3.0，到底是不是值得？暂且不说是否脱离了市场，但在长达一年多的开发过程中，层出不穷的缺陷和不停的小版本演进，到底给飞漫软件以及用户带来了什么？

直到 2011 年上半年，飞漫软件和一家老美公司合作开发一个 ACL 项目时，笔者才发现，我们多年来自然而然坚持的一些软件开发管理方法，其实并不是最佳的方法。该项目试图为不同的操作系统引入一个统一的 Android 兼容层，使得标准的 Linux、Windows 或者 RTOS（如 VxWorks）上，能够运行 Android 应用程序，开发过程采用了 SCRUM 敏捷开发模型。本人花了两天的时间阅读了两本有关 SCRUM 开发模型的书，结果得到一个惊人的结论：传统的软件工程思想，其实是一个大大的骗局！

传统的软件工程思想，以“瀑布法”为典型，按照需求分析、设计（又细分为概要设计、详细设计、单元测试设计、测试用例设计）、编码、测试的过程进行管理，不停迭代，直到缺陷数量降低到零，或者缺陷数量从最初的几百个收敛到几个，才认为是形成了可正式发布的版本。但这个过程极其漫长，MiniGUI 3.0 从第一个可发布版本 3.0.2 发展到基本稳定的 3.0.8，跨度居然长达一年半时间。

为了有效实行瀑布法模型，我们制定了详细的过程管理规范，从需求分析（草案）的编写、概要设计、详细设计、单元测试设计到最后的测试用例开发，每一步都要求形成对应的文档，经过评审后进入下一个单元。比如，软件架构师负责分析需求并进行概要设计，高级工程师来编写详细设计文档，经软件架构师审定后进入下一个环节，等等。看起来，一切都那么完美，只要每个人都按照要求和流程来做事，没有达不到的目标。但现在想来，这其中存在如下一些深层次的问题：

\* 文档流于形式。一方面是因为开发人员本身对文档工作存有天然的抵触情绪，另一方面，开发人员并没有受到过如何撰写开发文档的培训。自然而言，文档描述不清、不及时更新等问题就出现了，最后，文档基本上就会流于形式。通常的结果是，需求文档或者概要设计文档写得很好，但详细设计文档、单元测试文档等等，越往后越差。

\* 没有人仔细阅读文档。大部分开发者其实不会仔细阅读文档，他会根据自己对需求的理解进行编码，即使详细设计文档就是他自己编写的，他仍然会给你敲出一份偏离详细设计的代码。

\* 瀑布开发模型，忽视了软件质量的第一保证人是开发者本人这一要求，使得开发人员非常依赖于测试人员，测试人员又抱怨开发者编写的软件充满了缺陷。最后，使得整个开发过程充斥着责任不清和相互的埋怨，大大降低了开发效率，质量也很难得到保证。

然而，如果我们仔细回想 MiniGUI 早期版本的情况，我们会发现，那时，没有专职的质量保证团队或者测试人员，就两三个开发人员，软件的质量仍然相当好。再比如，Linux 内核的开发过程显然也没有采纳瀑布模型，但为什么仍然取得了那么伟大的成果？

如果你仔细想想这其中的奥秘，你就会发现，传统的软件工程思想，是模仿传统工程管理方法，比如建筑工程的管理方法设计出来的。瀑布模型，有其可适用之处，但并不是万能药。在任何一个软件开发中，期望通过传统软件工程方法来进行管理并取得良好效果，基本上属于一厢情愿。

为什么笔者认为传统软件工程思想是一个骗局呢？其一，宣传传统的软件工程方法，为那些靠 CMM 等软件管理标准或者规范吃饭的人给了一个可以赚钱的机会；其二，利用传统的软件工程方法，为那些贩卖项目管理软件的公司一个可以赚大钱的机会；其三，传统的软件工程方法，创造了更多的就业岗位。

自从笔者接触了 SCRUM 开发模型后，我发现它和传统软件工程方法最大的不一样，就是：前者围绕软件本身进行管理，后者围绕流程进行管理。SCRUM 开发模型强调 3C，即 Card（卡片）、Confirmation（确认或承诺）和 Communication（沟通或交流）。该方法去除了一切形式化的东西，比如复杂的文档和流程，让开发过程关注到最终可以交付的软件及其功能的演进上。而且，采纳 SCRUM 开发模型时，它的管理手段非常简单，任何有基本管理素养的人，只要遵照其基本原则和方法，都能做好相应的管理工作。

然而，SCRUM 开发模型的执行过程非常容易走样。很多人仍然喜欢使用电子化的方式来管理项目，比如使用 ScrumWorks 这样的软件，但这其实违背了 3C 原则中的 Card 和 Communication 这两项；很多人非要按照一周、两周或者一个月来划定一个冲刺的目标而不是按照冲刺工作集来确定发布时间；甚至一些管理人员，连燃尽图都懒得画。

这导出了本文的第四个主题。

四、中国软件工程师的特点

先告诉大家我的结论：中国的软件工程师，大致有一半或者更多是不应该从事这个行业的，他们做事随意，缺乏自律和学习精神，也缺乏必要的工程素养。

我们先看看中国软件工程师的来源。

中国几所顶级大学计算机相关专业的毕业生，大多选择了出国或者进入外企、知名国企工作（也有部分进入金融、投资等领域，少数选择创业或者加盟创业团队）。谷歌、百度、腾讯等大型互联网企业以及华为、中兴等大型通信企业，吸纳了这些顶级大学计算机相关专业中优秀的毕业生。但这些毕业生显然是少数，大多数从事软件开发的人员，毕业自二三流大学。

以笔者曾经面试过的应聘者以及很多共事过的软件开发人员为例，笔者得出了上述结论：

\* 教育方面的问题众人皆知，笔者不再赘述。许多来自二三流大学的毕业生来应聘我们公司的职位，甚至还有一些有过专业的职业培训经历，然而，我看不到任何可以录取他们的理由。在我看来，大多数人是因为就业压力大，找不到适合自己的职位，才选择薪水水平相对较高的软件开发职位作为自己踏入社会的第一步。有些人为了加大成功就业的概率，自己掏钱做职业培训，之后再找工作。但问题是，大学里边基本上什么也没学到，怎可能靠几个月的培训就能达到用人单位的要求？

\* 很多软件开发者不明白为什么要有一致的编码风格（coding style）。写出来的代码行文混乱，毫无美感而言。其实，字如其人，敲不出漂亮代码的开发者，也写不出符合要求的文档，而且代码必定错误百出。这些开发者，显然没有经过良好的工程素质训练，缺乏必要的工程素养。

\* 我们公司从 2005 年起利用 Wiki 系统管理内部文档。我发现，许多开发者连基本的 Wiki 标记语言都不能快速掌握。许多情况，照猫画虎就可以的，还会弄得乱七八糟。就一个命名规则，很多人都无法理解命名规则到底有什么意义，非要取“概要设计”这样的主题名称。怎么就不能想想，下个项目的概要设计，难道你也用这个名称？在我看来，这些开发者其实不应该进入这个行业，因为他缺乏必要的计算机科学、软件工程敏感性，他的头脑其实根本不适合做软件开发。

\* 如上一章节所说（SCRUM 开发模型的执行容易走样），包括一些管理者在内，许多软件开发者并不是合格的管理者或者被管理者。他们做事随意，不讲规则，缺乏自律。当然，这主要的原因来自管理者自身，大多数普通的开发者需要一定的管理约束和鞭策，当管理者自身随意、不讲规则，缺乏自律，那整个团队也会这样。这和大多数管理者出身自技术人员有关。

尽管笔者得出上述结论来自于笔者接触过的软件开发者，但相信这些问题也存在于很多企业当中。华为、中兴等大型企业的管理策略，基本上靠流程和人海战术，导致组织越来越庞大，效率越来越低下。这些企业因为已经具备了一定的市场地位，组织的臃肿和庞大并不会带来致命的后果。但如飞漫软件这样的小型企业或者创业团队，如果模仿华为、中兴等大企业的做法，必定要承受昂贵的代价。请各位看官切记！

五、外聘 CEO 之殇

2007 年，飞漫软件吸纳外资从内资企业变更为合资企业。根据外方董事的建议，公司用高薪聘请了一位来自台湾的H姓女性作为合资公司的 CEO，本人改任 CTO。

新聘 CEO 曾有过海外工作经验，主要工作经验是销售管理，来飞漫软件工作，算是第一次担任 CEO。H CEO 显然对第一次担任 CEO 表示出了极大的热情，问我在大陆，她名片上的职位，到底应该是“执行长”呢还是“首席行政长”还是其他的什么名称。我说，就是“首席执行官”，要么就写 CEO，大家都明白。最后，那名片上还是写了个“执行长”——也许“执行长”这个抬头，更加有气势？

H CEO 上任伊始就对公司进行了大刀阔斧的改革，比如，培养人事经理成为项目经理，以高薪吸纳她之前的台湾下属作为海外销售经理等等。同时，H CEO 也积极行动，发挥她的销售专长，去上海、深圳、台湾等地方拜访客户，寻找可能的销售机会。当然，每次出行必然是住四星级以上酒店。在北京，也住的是包月酒店，每月一万的房租。

然而，在其工作三个月之后（2008年元月），公司突然出现了一个离职潮，大量员工提出离职申请。显然，这位 CEO 并不适合飞漫软件这样的小企业。本人不得不提请董事会解雇职这位 CEO。但我们为此付出了极大的代价——成立合资公司引入的资金之一半基本上赔偿给了这位 CEO。这也是 2008 年，除了金融危机的影响之外，飞漫软件不得不裁员的一个另外一个主要原因。

这位 CEO 在被解雇后，在香港注册了一家皮包公司，从我公司采购了一套 MiniGUI，然后改头换面开始当做自己的产品进行销售。当然，笔者根本不在意这点，因为离了飞漫软件，MiniGUI 就是无源之水，你想复制飞漫的业务，那基本上不可能。

这里有个类似的插曲。2009 年的时候，2005 年期间代理 MiniGUI 的一家韩国公司，突然联系我，说我们公司有个前员工弄了个什么软件，想找他代理，还把其技术白皮书发给我了。但其实呢，就是他自己找这个前员工弄的，事情没弄成，反过来到我这里告状——蛮有意思的。

外聘 CEO 这件事情，在外方董事推荐之时，我内心其实不是非常赞同的，但我没有听从自己内心的声音，而选择了盲目的信任。

我记得在确定 OFFER 之前，曾邀请这位女士到我公司，作为双方互相考察之用。我开车去了机场接这位女士。见面之时，我注意到了两个细节：

\* 这位女士脚穿凉鞋，同时还穿着一双袜子。

\* 这位女士在见到我们举着写有她名字的牌子时，眼神掠过一抹非常难以察觉的轻蔑之神情。

之后我和当时的销售总监邀请她吃饭，送她去酒店住下，然后我就扁桃体发炎，高烧到了 39 度（我自打记事起，还没有如此发过烧）。之后的两天，打吊针输液，昏昏沉沉就过去了。出于对外方董事的信任，这考察也就草草走过场，然后就给了这位女士一个按照国际标准执行的 CEO OFFER。

显然，老天爷提醒了我，但我没有听从自己内心的声音，导致这惨痛的教训。各位看官，也请吸取我的教训，一定要按照乔布斯所说，听从自己内心的声音。当然，你面临的问题，也许是根本不知道自己的内心到底发出了什么样的声音，呵呵。

六、最大的经营失误

飞漫软件的过去十年，经历了很多事情。现在回过头来看，最大的经营失误是盲目开发新的软件产品，为此浪费了很多现金。

除了 MiniGUI 之外，飞漫软件曾经开发过很多东西，比如 mEagle、mSpider、mGallery、mDolphin、mStudio，包括后来的 HybridOS 等等。

在这些软件产品当中，给我们带来收入最多的自然是 MiniGUI，除此之外就是 mDolphin。其他的软件，现在看来，根本没有必要开发，因为这些软件脱离了市场需求，自然不会有客户买账。要是不开发这些软件，飞漫软件基本上可以以一个不超过 30 人的规模高效运行，按开发人员计算，人均年收入达到 40 万到 50 万是没有任何问题的。

然而，这些都是马后炮。写出来，是为了给各位看官一些启迪，希望对创业者、中小软件企业的管理者有所启发。

七、通过合作看华为

这里主要讲讲华为这个公司。

我之所以直接提这个公司的名字，是因为我希望这个公司能够有所变革，成为一家像苹果、谷歌等真正伟大的公司。

华为技术在 2004 年的时候以买断形式采购了 MiniGUI 软件，飞漫软件由此获得了在当时可以在北京四环外购买一套小两居住房的现金收入。

2009 年时，华为终端使用 MiniGUI 开发数码相框类的产品，遇到了一些技术问题，找我们公司帮忙。起初我们不同意他们的出价，他们的领导不停给我电话，说了很多好话，说和华为合作机会很多，这次少点，下次多点云云。最后五万块钱的服务费，我同意帮了。我安排了公司最资深的 MiniGUI 专家前去服务，前后两周时间，纯粹就是帮他们个忙。这样的事情很多，华为的人，总是以业界大拿的做派找我们这样的专业小公司，帮这个忙帮那个忙。去年底，海思还找我们帮他们解决浏览器上 Flash 插件的问题，我们帮了。领导说跟华为搞好关系，以后有大大的机会赚钱云云。结果呢，我们从华为系统的企业赚到的钱并没有多少。我现在已经死了从华为再赚钱的心了，所以我爆料给各位看官。

前文已经提到，华为终端采用 MiniGUI 开发的终端产品接近或超过一亿台。

各位看官，你们大概不知道华为技术和华为终端是两个独立的法人企业吧？我也是之后才知道的。也就是说，华为终端使用来自飞漫软件许可给华为技术的 MiniGUI 产品，是未经许可的。我们提出这个问题后，经过了长达半年的唇枪舌战，华为终端不得不在去年上半年补上了 MiniGUI 的许可费。当然，以华为一贯的作风，这个钱没有太多。

华为技术、华为终端也好，这个企业骨子里有股不好的基因，那就是喜欢压榨供应商，对供应商抠门的不行。我的结论是，华为当前充其量就是个“独善其身”阶段，还达不到苹果那样可以创建一个生态系统，从而“兼济天下”的水平！

华为，你未来的路还很长。

八、一些小的教训

作为结尾，我给大家罗列一些十年里边遇到的小的教训，希望各位看官防备：

\* 你永远会遇到一些小人，试图以不道德的方式获取利益。比如本文提到的韩国代理，H姓CEO。这种情况下，不用理会，事实证明小人成不了大事。如果你花更多的精力和他们较真，你将失去更多。

\* 本文提到的 ACL 项目，那美国的公司欠了我们将近 2.5 万美金不付（这公司是个初创公司，没有足够的现金做这个项目，加之 ACL 项目本身前景不妙，他们希望卖给 Intel 在 MeeGo 上用，希望卖给 HP 在 WebOS 上用，但 2011 年上半年，大家都知道，这两个项目终止了，这公司根本没法获得进一步投资）。所以，并不是所有老外公司（就算是老美公司）都那么遵守规则和具有商业道德，你要做的就是，尽量在前期收到足够多的钱，且不要盲目相信他们。

敬以此文纪念飞漫软件过去的十年。

[**Scrum 开发方法的实施**](http://caoyanbao.iteye.com/blog/608849)

**博客分类：**

* [项目管理](http://caoyanbao.iteye.com/category/73090)

[敏捷开发](http://www.iteye.com/blogs/tag/%E6%95%8F%E6%8D%B7%E5%BC%80%E5%8F%91)[软件测试](http://www.iteye.com/blogs/tag/%E8%BD%AF%E4%BB%B6%E6%B5%8B%E8%AF%95)[项目管理](http://www.iteye.com/blogs/tag/%E9%A1%B9%E7%9B%AE%E7%AE%A1%E7%90%86)[CMM](http://www.iteye.com/blogs/tag/CMM)[配置管理](http://www.iteye.com/blogs/tag/%E9%85%8D%E7%BD%AE%E7%AE%A1%E7%90%86)

Scrum是基于过程控制理论的经验方法，倡导自组织团队；其运行框架核心是迭代增量型并行开发，也是“适应性”的软件开发方法。Scrum提供了高度可视化的用于管理软件开发复杂性管理的敏捷项目管理的实践框架或敏捷过程，可以用于对现存软件工程实践的包装，提高软件生产率，改善沟通和合作的方法，使人们协作并注重业务目标。现在Scrum已被众多的软件企业使用，其中不乏有业界知名企业，如Microsoft 、IBM、Google和Nokia等。  
  
作为一名Scrum教练，笔者经常被问到有关Scrum实施以及敏捷开发方面的各类问题，现总结如下，供对此方法有兴趣和有疑问的读者参考。  
  
一问：Scrum的核心特征是什么？  
一答：基于功能开发而组成的多功能、自组织团队；高度柔性的可视化敏捷项目管理自适应框架；以及支持增量并行开发的30天时间盒迭代。  
  
二问：哪类项目可以使用Scrum?  
二答：最初Scrum使用于需求难以预测的复杂商务应用产品的开发，但经过10多年的发展，它被应用于所有领域的软件中，从生命攸关的软件到更为随意的软件，都可以使用Scrum。在使用Scrum时，无需讨论工件是什么以及它们的数量，而是讨论需要严谨到什么程度。作为一个指导原则，由整个Scrum团队来决定正规性的程度，并尽可能地低。当然，这需要有丰富的实践经验来判断。  
  
三问：Scrum团队一定是7个人吗？  
三答：在Scrum中有3个基本的角色：产品所有者Product Owner、开发团队Development Team和ScrumMaster。Scrum团队通常有5～9个成员，典型一个Scrum团队应当有7个成员。但可以由多个团队完成一个项目，即使用Scrum of Scrums实践规则进行拓展项目团队规模：每一个Scrum Team同样有一个代表，通常是Scrum Master，参与Scrum of Scrums会议协调多个Scrum Teams的工作，这些会议类似于Daily Scrum Meeting，但每周召开一次。  
  
四问：看上去Scrum非常简单，可以给我们更简化地总结一下吗？  
四答：是的，Scrum看上去确实很简单，可以把Scrum总结得非常简单：  
  
团队和项目出资人创建一个团队需要做的所有事情的列表。这可以是一个任务的列表或者特性的列表。这就是Product Backlog。  
  
每个月，团队都努力实现列表最顶端的任务，这一部分是他们估计需要一个月完成的工作。他们把它展开成一个详细的任务列表，叫做Sprint Backlog。这个团队承诺在月底向出资人演示或交付结果。  
  
每天，团队都面对面地开5～10分钟的会，彼此更新各自的状态和排除使他们减慢的路障。这个叫Daily stand-up meeting。  
  
指定一个特别的人担任Scrum Master，这个人的任务是排除或安排别人排除在例会上这个团队提到的任何路障。  
  
但是它的实践执行并不简单，需要获得关键的自适应和坚持Scrum核心价值观——承诺、专注、公开、敬重和勇气。  
  
五问：我们认为，坚守一定的Scrum Meeting模式是必要的；但是执行一段时间后觉得很困难，有些人甚至觉得“恶心”，你对此怎么看？  
五答：每天举行15～20分钟左右的Scrum meeting是Scrum和项目的心脏。如果出现这一问题，我估计是软件团队倾向于在现有的项目管理方法下诠释Scrum，没有充分理解自我管理、涌现机制、可视性和评估/适应循环的根本原则。  
  
按照“定义的”参考框架去执行Scrum的实践，忽视了从控制转向授权、从命令转向协作，Scrum Master很可能将“自上次Scrum Meeting会后的一天里我做了什么？”理解为“检查团队成员是否完成上次Scrum Meeting中他所布置的任务”，将“从现在到下次Scrum Meeting的一天我将做什么”理解为“告诉团队人员从现在到下次Scrum Meeting的一天应做什么”，将“在工作中遇到了哪些障碍”理解为“他将审核是否能帮助团队完成目标”。而团队成员把Scrum Meeting理解为按顺序报告工作情况的会议。  
  
坚持以下7个基本原则，将有利于有效执行Scrum Meeting：  
  
1 团队信仰自我管理和支持自我管理。  
  
2 他们作为团队共同承诺Sprint目标。  
  
3 他们认识到沟通的重要性，并且通过Daily Scrum Meeting推动沟通。  
  
4 他们理解和拥抱贯穿整个Sprint周期的必要的日常任务变更，相互依赖的会议规则，每日会议允许团队成员管理和响应变化。  
  
5 团队有一位卓有成效的Scrum Master或得到他们授权的领导来决策和问责。  
  
6 团队认为工作可视化很重要，透明改进团队和组织其他团队之间的关系，从而得到更高层次的信任和协作。  
  
7 团队将Daily Scrum Meeting回顾与其他里程碑相联系使会议尽可能有效。  
  
  
六问：用什么来判断软件团队在真正实施Scrum?  
六答：对这一问题，Scrum创始人之一Jeff Sutherland用诺基亚测试的8个判断条件来判定是否在真正执行Scrum。这8个判断条件是：  
  
1 你们有固定的迭代周期么？你们的迭代周期是否以某个特定的时间开始并以某个固定的时间结束，且迭代周期必须少于6周？（回答否定的则不符合迭代开发原则）  
  
2 在每个迭代周期结束时，你们能提供可以工作的软件么？（回答否定的则不符合迭代开发原则）  
  
3 在迭代开始之前，你们是否需要必须有一个完整细致的需求说明？（回答肯定的则不符合迭代开发原则）  
  
4 是否将测试作为迭代增量开发的一部分，在开发过程中进行测试？（回答否定的则不符合迭代开发原则）  
  
接下来，用4个附加的Scrum规则来判断是否实现了Scrum：  
  
1 你们是否有产品所有者？是不是有人可以代表客户和你们一起工作？  
  
2 如果有产品所有者的话，他们是否能提供待开发的产品Backlog？且此产品Backlog是否按照优先级来排序的？是否估算过开发这些功能的所需时间？  
  
3 团队在开发过程中是否使用了Burndown图来展示工作量变化、跟踪进度、推算团队开发速度？  
  
4 在迭代过程中，是否能保证项目经理不干涉团队工作？  
  
通过以上8点基本上就可以确定，团队是否真正地实现了Scrum。  
  
七问：在Scrum中我怎样去度量团队绩效？  
七答：你可以通过速度去度量团队绩效，即在一个Sprint中将需求转化为软件功能的能力。可以是一个Sprint中完成多少Product Backlog Item（包括功能和非功能需求及其他议题），或者转化为1个合适单位货币如10000完成多少Product Backlog Item。  
  
八问：在Scrum中我怎样去度量个人绩效？  
八答：你不能度量个人绩效，只能度量整个Scrum团队的绩效。Scrum是自我管理的团队，而不是个人组成的组。当然，可能你的软件组织要求这么做，这确实是个棘手的问题，我也没有好的解决方案。对于软件组织这一层面，我建议你首先把度量的焦点聚焦于你生产的软件、真正的开发功能和软件组织用于改进基准和市场价值的能力。而项目这一级别，我建议将完整的个人检查过程简化为3个问题：你对增加组织的价值有什么帮助？你做了什么使客户高兴？你的同事怎么看待你？可以请同事来评估个人贡献，并列出1～10的等级。在Daily Scrum Meeting上，你可以看到谁有贡献，谁没有。  
  
九问：在Sprint期间，如何去修改一个缺陷？  
九答：Scrum团队的目标之一是在发现缺陷的Sprint中就修复它们。在他们逐渐精通采用30天迭代周期以后，尤其是通过对自动化测试的利用，他们能够达到这个目标。当Scrumt团队成员做出对某项编码任务的估计时，这个估计值就包含了用于修复在实现过程中发现的缺陷的时间，否则就应该确定和估计一个独立的任务（“修复缺陷”）即缺陷作为Product Backlog Item处理。我的偏好是只确定一项任务，但是在它通过所有的测试之前不认为已经完成。  
  
后来发现的（或者在发现它的迭代中没有修复 的）缺陷应该按照与Product Backlog一样的方法来对待。应该按照与Product Backlog一样的方法确定缺陷修复工作的优先级，分配到后续的某次迭代中。只要超出一次迭代的范围，就不再有什么缺陷的概念。修复一个缺陷和增加一个功能只是一件事的两种说法。另外，如果现有团队还需要维护现有产品时，则需要提醒软件团队在做计划时拿出专门的时间处理那种需要马上响应的缺陷修改任务。  
  
十问：Scrum的局限性是什么，实施中需要注意什么？  
十答：我们都知道Scrum只是一种敏捷管理的一种实践框架（Framework），任何方法都有其边界和局限性，套用业界流行的一个说法就是“没有银弹”。Scrum为软件开发管理只定义了一个高层次的、易于操作与遵循的非常小的实践集，Scrum避免了说软件团队应该如何开发软件，它坚持认为：人们在自己的工作中和处理问题时，应该像一个成熟的成年人一样，因此它并不涉及具体的软件开发技术和人员沟通、期望管理、问题冲突等管理技能，这些都需要其他相关理论和技能来补充，另外，如同其他项目一样，需要软件团队在其业务领域的专业能力来确保软件项目的成功。  
  
Scrum源于美国软件界，对国内实施强调自组织管理的Scrum需要破除可能习惯于听命行事的组织环境，建立自我约束、自我组织和实现的工作管理方式和组织环境，同时根据Scrum背后的科学原理则可以根据特定的情形进行调整。建议最初时，按Scrum提供的实践框架执行，然后，当积累了丰富实践经验后再根据Scrum提供的避免做什么的说明视实际情形进行调整，到最后，不要在乎自己是否执行Scrum或是其他什么敏捷方法，也就是达到从心所欲不逾矩。  
  
十一问：Scrum中如何看待文档，或是对待写文档的问题有何建议？  
十一答：这一问题屡屡被提及的是有着非常现实背景，软件组织或软件项目都需要一些软件文档，而在Scrum中没有规定具体的做法。我认为软件文档主要的作用在于理解、沟通和管理要构建的软件，需要交付给用户和软件组织为了方便维护软件和知识产权方面考虑用作保存和传递知识的作用，特殊软件需要满足法律法规的要求。  
  
因此，第一、是否需要文档，标准是该文档是否增值，坚持精益原则，消除不必要的浪费，比如如果设计人员和专家坐在一起，就可以免去设计文档，而是在白板上粗略地勾勒草图，然后用照片记录白板上的图或用可打印的白板将其打印出来。第二，假设该文档只是符合规定（法律、审计或是客户等其他原因的强制需求）的需要，则可以考虑安排部分项目时间交给非核心人员编写文档，避免影响软件开发核心工作的开展。第三，对于理解和沟通所要构建软件的关键文档，原则上用自动化的代码文档工具如Doxygen、JavaDoc或NDoc等，在必要时产生符合代码标准的自解释的源代码文档，并创建自文档化的软件UML图模型。另外，用简明扼要的1～2页的软件架构文档，描述软件的架构，为新来的开发者表示出关键构件和接口。第四，不要奢求需求是完美的，设计文档能够与代码同步更新，或项目计划与项目状态完全匹配，需求文档能够满足后续开发需要即可，总之，软件文档符合精益（Lean）、易理解（Mean）和足够（Enough）的要求，确信在当前环境能够严格地保证沟通需要。还有一个有趣的现象是技术人员基本上都讨厌写文档，但需要指出的是编写精益易理解足够的文档也是职业软件开发者的一项基本功，当然同时，也要避免为了文档而文档，它意味着用于项目的每个文档都应该证明是对项目有意义的。  
  
十二问：Scrum与流行的RUP有什么异同？  
十二答：这个问题网上好象有专文论述，这里根据我的自己的实践体会简单地说一下，Scrum与RUP都强调需求导向、迭代增量开发和风险驱动，两者可以结合使用。Scrum与RUP的侧重点不同，其中一个表现Scrum反对预定义的过程和预见性的步骤，RUP会有一些可选的但又需要预先定义的活动，如与需求分析、测试等相关的活动，并有先后顺序。需求管理方面Scrum由Product owner管理，RUP则由需求工程师管理；可视化建模方面，scrum团队自定，通常进行敏捷建模，RUP倡导可视化建模；软件体系结构方面，Scrum软件组织或团队自定，RUP则由软件体系中心确定；迭代周期方面，Scrum迭代周期推荐30天，RUP推荐2～6周。  
  
十三问：一提起敏捷方法，为什么通常将Scrum与XP一并谈论？  
十三答：这大概是XP与Scrum是敏捷方法中被业界采用最为广泛的原因吧。另外，一个值得注意的原因是两者都聚焦于信息价值流和信息沟通，除了迭代长度稍有差别外，大多数Scrum实践与XP是兼容且相互补充，Scrum侧重于项目管理和人，XP有许多工程技术实践，两者相得宜彰。Scrum这种强调项目管理价值与实践而不在乎需求、实现等工程技术用作其他软件开发方法包装器的特征，具有高度适应性和柔性是其他敏捷方法不具有的，因此，它很容易与其他方法进行组合，或者作为其他方法的补充，这一点不是它的弱点，而应当看作其长处。  
  
十四问：我们公司采用CMM/CMMI进行软件过程改进，是否适合实施Scrum？  
十四答：CMM/CMMI与Scrum或敏捷方法之间的关系是常常被人们关心或经常提起的一个有趣的话题。2002年Scrum创始人 各级ken Schwaber与美国卡内基。梅隆大学软件工程研究所SEI的Paulk Mark曾经评估Scrum如何实现CMM的关键过程域，结论是Scrum实践规则可以满足CMM2级全部和CMM3即的大部分关键过程域。事实上，国内外都有利用Scrum通过CMM/CMMI评估的案例。  
  
通常认为CMM/CMMI之间的区别在于：关注目标不同，CMM/CMMI关注组织级，Scrum关注项目级；基础假设不同，CMM/CMMI假设软件开发可预测与可重复，符合统计过程控制，优化放在最后，Scrum假设软件开发是自适应，高度复杂，需要基于过程控制理论的经验方法，优化一开始就进行。  
  
业界有一个流行观点是将Scrum纳入到一个已经通过CMM/CMMI 3或更高级别评估的组织，看起来比将一个Scrum项目改为CMMI更简单。同时，一些实施Scrum或其他敏捷方法的软件组织对CMMI评估没有兴趣，把CMMI评估看作浪费金钱，也就是说，他们不需要CMMI评估来获得合同，但获得这一评估却需要耗费金钱和时间，我的看法是在现实商业环境中的软件组织是否需要组合CMM/CMMI和Scrum主要不在于理论或其背后的软件哲学理念，而在于业务运作环境所需考虑的优先级。假设您需要组合两者并有通过CMMI评估需要，关键在于找到一个理解如何灵活使用CMMI模型又理解Scrum的CMMI主任评估师。  
  
十五问：如何引入和执行Scrum?  
十五答：对于第一个项目建议在一个有执行Scrum经验丰富的专家指导下进行。在试点项目的选择上Scrum的创始人鼓励选择最为困难和关键的项目，我个人建议是如果试点项目团队有强烈意愿并获得高层支持是完全可以的。否则，试点项目应该是构建真实的、具有合理的、客观的商业目标的软件，团队规模6～8人为宜，以避免一开始就使用Scrum of Scrums，项目长度通常2～6个月为宜。在项目启动后，外部的项目管理者和潜在使用Scrum的可以邀请其观摩Daily Scrum Meeting、Sprint Planning Meeting 、Sprint Review Meeting和Sprint Retrospective Meeting。最终，Scrum的实践扩展至软件组织的最高层，每一层都基于团队，高层管理者每月组织一次Scrum meeting就可以了。  
  
  
十六问：Scrum的迭代Sprint周期一定是30天吗？  
十六答：Scrum的目标是在一系列（3～8个）短期的时间框（time box）内交付尽可能多的优质软件。其中时间框被（固定时间间隔）称为Sprint，典型地，其将持续大约一个月30天的时间。但在许多公司已缩短至两个2周或更短。我建议刚开始实施Scrum时Sprint周期最好固定为30天，随着经验积累可以根据实际进行调整期Spint周期为2～4周。  
  
十七问：Sprint的长度取决于哪些因素？  
十七答：绝大多数的迭代增量开发方法推荐1～6周的长度。Scrum推荐其迭代Sprint的长度为30天。确实需要调整时，考虑发布的总时间长度、不确定性的多少、获得反馈的难易、优先级可以保持多久不变时间长度、迭代系统开销、紧迫感产生有多快等因素。这里需要特别指出的是国内有企业一开始就执行2周为Sprint周期的企业，因为其执行短周期迭代开发的基础设施（如软件配置管理和自动化测试的基本实践不扎实）不到位而导致退化为瀑布开发的案例。  
  
十八问：导致Scrum项目失败的主要原因是什么？  
十八答：导致失败的主要原因是软件团队不是自组织团队，团队由项目经理或Scrum Master进行指导和组织。其次，Product Owner或客户不参与每次迭代，不进行需求优先级划分，不参与每次演示，并且不为下一迭代选择具有最高商业价值的项。另外，在迭代期内给团队成员追加新的需求或额外的任务。  
  
十九问：您如何看待Scrum Master认证培训？  
十九答：敏捷联盟推出Scrum Master认证培训即Scrum Certified Master，事实上其敏捷联盟内部对此也有一些争论。我个人认为您如果有机会参加由那些真正理解Scrum的人负责的为期2天的培训自然是件好事。但我认为把它叫做“认证”会带来一些错误的认识，比如国内外都有参加了Scrum认证后就认为自己是Scrum Certified Master，然后在软件组织内执行Scrum最后导致失败的案例。要真正达到通常意义上的认证，至少你必须进入软件团队，用Scrum工作方式几个月的实践才行，倘若没有任何软件开发经历和理解Scrum背后的哲学体系，仅是接受2天的培训而去推广Scrum，后果不堪设想。另外，如果纯粹为了就业或是获得证书什么的，我认为没有必要参加目前还相对昂贵的认证培训，看看类似的认证培训效用就可以知道了。相信任何理性的软件组织都更看重的是实践能力而非仅仅通过2天培训换来的一纸证书。在这里我特别声明我并不反对任何人任何时候参加任何机构举办的Scrum Master认证培训。  
  
二十问：对于刚刚学习和实施Scrum者有没有可以推荐的读物？  
二十答：书籍方面目前主要有由Scrum创始人Ken Schwaber主笔的三本书：《Agile Software Development with Scrum》、《Agile Project Management with Scrum》和《The Enterprise and Scrum》。其中《Agile Software Development with Scrum》还得到了Beedle和Sutherland的协助，国内出了影印版中文名为《敏捷软件开发——使用Scrum过程》。《Agile Project Management with Scrum》出了中文版，中文名为《Scrum 敏捷项目管理》，《The Enterprise and Scrum》也已引入英文版。另外著名技术网站infoQ有2本迷你书。我个人倾向推荐《Agile Software Development with Scrum》。  
  
此外，我个人认为在精力和时间允许的情况下最好对Scrum产生重大影响的一些书籍和文章进行研读，这样可以更深入理解Scrum和领悟其自适应与自组织的精髓

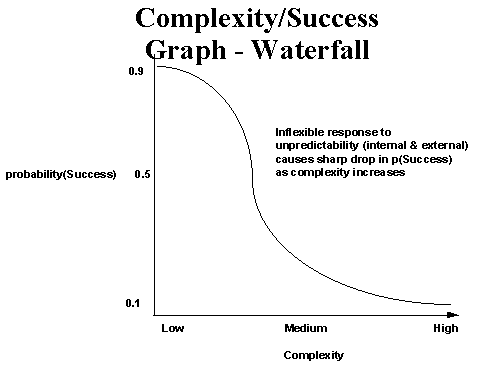
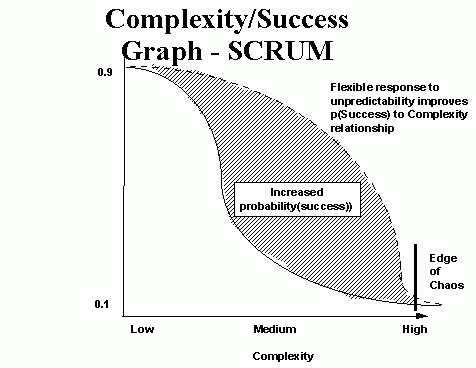
[**敏捷软件开发模型--SCRUM**](http://blog.csdn.net/changemyself/article/details/4217531)

分类： 2009-05-26 15:42354人阅读[评论](http://blog.csdn.net/changemyself/article/details/4217531#comments)(0)[收藏](javascript:void(0);)[举报](http://blog.csdn.net/changemyself/article/details/4217531#report)

[敏捷软件开发模型--SCRUM](http://www.cnblogs.com/Ring1981/archive/2006/09/07/496591.html)

**一 什么是Scrum?**  
Scrum (英式橄榄球争球队), 软件开发模型是敏捷开发的一种，在最近的一两年内逐渐流行起来。  
  
Scrum的基本假设是：  
  
开发软件就像开发新产品，无法一开始就能定义软件产品最终的规程，过程中需要研发、创意、尝试错误，所以没有一种固定的流程可以保证专案成功。Scrum 将软件开发团队比拟成橄榄球队，有明确的最高目标，熟悉开发流程中所需具备的最佳典范与技术，具有高度自主权，紧密地沟通合作，以高度弹性解决各种挑战，确保每天、每个阶段都朝向目标有明确的推进。

Scrum 开发流程通常以 30 天(或者更短的一段时间)为一个阶段，由客户提供新产品的需求规格开始，开发团队与客户于每一个阶段开始时挑选该完成的规格部分，开发团队必须尽力于 30 天后交付成果，团队每天用 15 分钟开会检查每个成员的进度与计划，了解所遭遇的困难并设法排除。

**二 Scrum较传统开发模型的优点**  
Scrum模型的一个显著特点就是响应变化，它能够尽快地响应变化。下面的图片使用传统的软件开发模型(瀑布模型、螺旋模型或迭代模型)。随着系统因素（内部和外部因素）的复杂度增加，项目成功的可能性就迅速降低。  
  
  
下图是Scrum模型和传统模型的对比：  


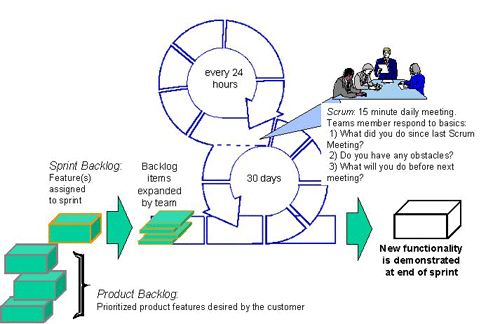
**三 Scrum模型  
  
一) 有关Scrum的几个名词**

backlog: 可以预知的所有任务， 包括功能性的和非功能性的所有任务。  
  
sprint:一次跌代开发的时间周期，一般最多以30天为一个周期.在这段时间内，开发团队需要完成一个制定的backlog,并且最终成果是一个增量的，可以交付的产品。  
  
sprint backlog:一个sprint周期内所需要完成的任务。  
  
scrumMaster: 负责监督整个Scrum进程，修订计划的一个团队成员。  
  
time-box: 一个用于开会时间段。比如每个daily scrum meeting的time-box为15分钟。  
  
sprint planning meeting: 在启动每个sprint前召开。一般为一天时间（8小时）。该会议需要制定的任务是：产品Owner和团队成员将backlog分解成小的功能模块, 决定在即将进行的sprint里需要完成多少小功能模块，确定好这个Product Backlog的任务优先级。另外，该会议还需详细地讨论如何能够按照需求完成这些小功能模块。制定的这些模块的工作量以小时计算。

Daily Scrum meeting：开发团队成员召开，一般为15分钟。每个开发成员需要向ScrumMaster汇报三个项目：今天完成了什么？　是否遇到了障碍？　即将要做什么？通过该会议，团队成员可以相互了解项目进度。

Sprint review meeting：在每个Sprint结束后，这个Team将这个Sprint的工作成果演示给Product Owner和其他相关的人员。一般该会议为４小时。

Sprint retrospective meeting：对刚结束的Sprint进行总结。会议的参与人员为团队开发的内部人员。一般该会议为３小时。

**二）实施Scrum的过程简单介绍**1) 将整个产品的backlog分解成Sprint Backlog,这个Sprint Backlog是按照目前的人力物力条件可以完成的。  
2) 召开sprint planning meeting，划分，确定这个Sprint内需要完成的任务，标注任务的优先级并分配给每个成员。注意这里的任务是以小时计算的，并不是按人天计算。  
3) 进入sprint开发周期，在这个周期内，每天需要召开Daily Scrum meeting。  
4) 整个sprint周期结束，召开Sprint review meeting，将成果演示给Product Owner.  
5) 团队成员最后召开Sprint retrospective meeting，总结问题和经验。  
6) 这样周而复始，按照同样的步骤进行下一次Sprint.  
  
整个过程如下图所示：  
  
  
  
The diagrams in this article are all from web site: [http://www.controlchaos.com](http://www.controlchaos.com/). Thanks very much!

### 敏捷软件开发宣言

我们 正通过亲身实践 以及 帮助他人实践，揭示更好的软件开发方法

通过这项工作 ，我们认为：

**人和交互** 重于 过程和工具

**可以工作的软件** 重于 面面俱到的文档

**客户合作** 重于 合同谈判

**随时应对变化** 重于 遵循计划

虽然右项也有其价值，但我们认为左项更加重要。

### 原则

1. 我们最优先要做的是通过尽早地、持续地交付有价值的软件来使客户满意。

2. 我们欢迎需求的变化，即使到了开发后期。敏捷过程能够驾驭变化，为客户创造竞争优势。

3. 经常交付可以工作的软件 ，从几个星期到 几个月，时间间隔越短越好。

4. 在整个项目开发期间，业务人员和开发人员必须朝夕工作在一起。

5. 围绕斗志昂扬的人构建项目。给他们提供所需的环境和支持，并且信任他们能够完成工作。

6. 在团队内部，最有效率也最有效果的信息传达方式，就是面对面的交谈。

7. 可以工作的软件是 进度主要的度量标准。

8. 敏捷过程提倡可持续开发。出资人、开发者和用户应该总是保持稳定的开发速度。

9. 对卓越技术和良好设计的不断追求有助于提高敏捷性。

10. 简单——尽量减少工作量的艺术是至关重要的。

11. 最好的架构、需求和设计都源于自我组织的团队。

12. 每隔一定时间，团队都要总结如何更右效率，然后相应地调整自己的行为。

### 极限编程实践

完整团队 用户故事 短交付周期 验收测试 结对编码 测试驱动开发

集体所有权 持续集成 可持续的开发速度 开放的工作空间 计划游戏

简单设计 重构 隐喻

### 避免设计的臭味

 僵化性 （rigidity） ——设计难以改变。

 脆弱性（fragility）——设计易于遭到破坏。

 顽固性（immobility）——设计难以重用。

 粘滞性（viscosity）——难以做正确的事情。

 不必要的复杂性（needless complexity）——过分设计。

 不必要的重复（needless repetition）——滥用鼠标进行复制、黏贴。

 晦涩性（opacity）——混乱的表达。

### 设计原则

 单一职责原则（SRP）：一个类应该只有一个发生变化的原因。

 开放封闭原则（OCP）：软件实体应该对扩展开放，对修改关闭。

 Liskov替换原则（LSP）：子类型（subtype）必须能够替换掉它的基类型（base type）。

 依赖倒置原则（DIP）：a. 高层模块不应该依赖于低层模块。二者都应该依赖于抽象。b.抽象不应该依赖于细节。细节应该依赖于抽象。

 接口隔离原则（ISP）：不应该强迫客户程序依赖并未使用的方法。

 DRY：Don’t repeat yourself Principle。通过抽取公共部分放置在一个地方避免代码重复。

 封装变化 （Encapsulate what varies）。

 面向接口编程而不是实现（Code to an interface rather than to an implementation）。

 优先使用组合而非继承（Favour Composition Over Inheritance）。

### 包和组件的设计原则

* 重用-发布等价原则（Reuse-Release Equivalence Principle, REP）：重用的粒度就是发布的粒度。
* 共同重用原则（Common-Reuse Principle, CRP）：一个组件中的所有类应该是共同重用的。如果重用了组件中的一个类，那么就要重用组件中的所有类。
* 共同封闭原则（Common-Closure Principle, CCP）：组件中的所有类对于同一种性质的变化应该是共同封闭的。一个变化若对一个封闭的组件产生影响，则对该组件中的 所有类产生影响，二对于其他组件则不造成任何影响。
* 无环依赖原则（Acycle-Dependencies Principle, ADP）：在组件的依赖关系图中不允许存在环。
* 稳定依赖原则（Stable-Dependencies Principle, SDP）：朝着稳定的方向进行依赖。
* 稳定抽象原则（Stable-Abstraction Principle, SAP）：组件的抽象程度应该与其稳定程度一致。