程序，在这里指的是源程序，就是一行行的代码。仔细看过去，它们的确是建立在数据结构上的一些算法。程序还要对数据进行操作，这些数据有些是静态的（例如软件的图标、提示信息），有些是动态的（例如程序生成的随机数字、程序通过网络下载的数据、用户的文字或语音输入等）。但是光有代码和静态数据还是不行，工程师要把它们构建为机器能懂的可执行代码。构建不仅仅是cc和link命令，一个复杂的软件不但要有合理的软件架构（Software Architecture）、软件设计与实现（Software Design, Implementation and Debug），还要有各种文件和数据来描述各个程序文件之间的依赖关系、编译参数、链接参数，等等。这些都是软件构建的过程。

Program here refers to the source, that is, line by line code. A closer look at the past, they do is based on the data structure of some of the algorithms. Programs also operate on the data, some of the data is static (such as software icon, message), some are dynamic (such as a program to generate random numbers, the program through the network to download data, the user's text or voice input, etc. ). But light has a code and static data or not, they want to build a machine engineer can understand executable code. Cc and link command to build not just a complex software is not only a reasonable software architecture (Software Architecture), software design and implementation (Software Design, Implementation and Debug), but also all kinds of files and data to describe the various procedures dependencies between files, compiler parameters, link parameters, and so on. These are the software build process.

软件团队的成员每天都在修改各种源代码，怎么保证软件在修改过程中能不断提高质量，至少要维持以前的质量，不至于崩溃？有些时候，我们要为某个需求写一些特殊功能，不久后又要把这些功能再合并回主要版本。有些程序要配置不同的界面，运行在中文、英文或其他语言的操作系统上；有些程序还有32位版本、64位版本等。这是源代码管理（Source Code Control）的问题—有时候也叫配置管理（Software Configuration Management）。

Members of the software team in modifying various sources every day, how to ensure that the software modification process to continuously improve the quality, to maintain at least the previous quality, will not collapse? Sometimes, we have to write some special features as a requirement, and soon after to make all these features and then merged back into the main version. Some programs you want to configure different interface, running on the operating system in Chinese, English or any other language; some programs there are 32-bit versions, 64-bit version and so on. This is the source code management (Source Code Control) problem - sometimes called configuration management (Software Configuration Management).

我们还有一系列的工具和程序来保证程序的正确性，这些工具流程和程序本身应该更正确，才能保证别的软件的质量。这就是质量保障（Quality Assurance），具体的验证过程叫做软件测试（Testing）。一个软件或者服务要有人买，就得找到顾客，顾客有各种需求，有些靠谱，有些不靠谱；有些容易做到，有些难以做到。软件团队要从需求分析（Re-quirement Analysis）开始，把合适的需求梳理出来，然后逐步展开后续工作，如设计（软件架构）、实现（写数据结构和算法）、测试，到最后发布软件。软件团队的人员也会流动，新的成员要尽快读懂已有的程序，了解程序的设计，这叫程序理解（Pro-gram Comprehension）。

We also have a range of tools and procedures to ensure the correctness of the program, these tools and processes program itself should be more correct, in order to ensure the quality of other software. This is the Quality Assurance (Quality Assurance), the specific verification process is called Software Testing (Testing). A software or service you want people to buy, you have to find customers, customers have a variety of needs, some fly, some do not fly; do some easy, some difficult to achieve. Software team from requirements analysis (Re-quirement Analysis) began to sort out the appropriate needs, and then gradually expand the follow-up work, such as design (software architecture), implement (write data structures and algorithms), testing to the final release software. Software team personnel will flow as soon as a new member of the existing program to read, understand design of the program, called Program Understanding (Pro-gram Comprehension).

软件在运行过程中还会出这样那样的问题，也许我们要时不时给软件打一个补丁，或者维护众多的服务器，团队的新老成员要一起工作，修复各种各样的问题，这叫软件维护（Software Maintenance），或者服务运营（Service Operation）。这一系列过程就是软件的生命周期（Software Life Cycle，SLC），有人得负责软件项目的管理（Project Management）。一个好的软件，即使功能和同类软件区别不大，但是会让人感觉到非常好用。这就是软件的用户体验（User Experience）。用户体验和数据结构、算法没有直接的关系，但是很多非常成功的软件就赢在这个方面。

Software during operation will be out this kind of problem, maybe we need to make a patch to the software from time to time, or to maintain a large number of servers, the new and old members of the team to work together to fix a variety of problems, called Software Maintenance (Software Maintenance), operators or service (service operation). This series of process is the software life cycle (Software Life Cycle, SLC), some people may be responsible for software project management (Project Management). A good software, even if not very different functions and similar software, but it will make people feel very easy to use. This is the software user experience (User Experience). User experience and data structures, algorithms are not directly related, but a lot of very successful software win in this regard.

软件开发过程有什么特别的难题？学者们总结了下面五点：

1. 复杂性（Complexity） 软件可以说是人类创造的最复杂的系统类型。大型软件（操作系统、办公软件、搜索引擎）有超过百万行的源代码，上万个不同的文件。而软件工程师通常一次只能看到30—80行源代码（相当于显示器的一屏），他们的智力、记忆力和常人差不多。软件的各个模块之间有各种显性或隐性的依赖关系，随着系统的成长和模块的增多，这些关系的数量往往以几何级数的速度增长。

Software development process have any particular problems? Scholars have summarized the following five points:

1. Complexity (Complexity) software is arguably the most complex type of system created by human beings. Large software (operating systems, office software, search engine) has over one million lines of source code, tens of thousands of different files. The software engineers usually only see 30-80 lines of source code (equivalent to a display screen), their intelligence, memory and almost ordinary. There are various explicit or implicit dependencies between the various software modules, with the increase in the growth of the system and modules, the number of these relationships often geometric rate.

1. 不可见性（Invisibility） 软件工程师能直接看见源代码，但是源代码不是软件本身。软件以机器码的形式高速运行，还可能在几个CPU核上同时运行，工程师是“看”不到自己的源代码如何具体地在用户的机器上被执行的。商用软件出现了错误，工程师可以看到程序在出错的一瞬间留下的一些痕迹（错误代号、大致的目标代码位置、错误信息），但是几乎无法完整重现到底程序出现了什么问题。

2. invisibility Software engineers can directly see the source code, but the source code is not the software itself. Software at high speed in the form of machine code, also may run simultaneously on several CPU cores, designers are not "see" your source how specifically on the user's machine to be executed. Commercial software error has occurred, the engineer can see the program in the wrong moment left some traces (error code, object code position substantially, the error message), but it is almost impossible to reproduce the complete program in the end what went wrong.

1. 易变性（Changeability） 软件看上去很容易修改，修改软件比修改硬件容易多了。人们自然地期待软件能在下面两种情况下“改变”： a) 让软件做新的事情；b) 让软件适应新的硬件。但是与此同时，正确地修改软件是一件很困难的事情。

3. variability (Changeability) looks very easy to modify the software to modify the software to modify than hardware easier. It is natural to expect the software can be in the following two situations "change": a) let the software do new things; b) allow the software to adapt to the new hardware. But at the same time, the right to modify the software is a very difficult thing.

1. 服从性（Conformity） 软件不能独立存在，它总是要运行在硬件上面，它要服从系统中其他组成部分的要求，它还要服从用户的要求、行业系统的要求（例如银行利率的变化）。

4. Obedience (Conformity) software can not exist independently, it is always on top of the hardware to run, it should be subordinated to the system requirements for other components, it is also subject to the requirements of users, the system requirements of the industry (such as changes in bank interest rates ).

1. 非连续性（Discontinuity） 人们比较容易理解连续的系统：增加输入，就能看到相应输出的增加。但是许多软件系统却没有这样的特性，有时输入上很小的变化，会引起输出上极大的变化。

5. The non-continuous (Discontinuity) It is easier to understand continuous system: increase the input, we will see a corresponding increase in output. But many software systems have no such characteristic, sometimes small changes in input will cause great changes in output.

计算机科学（Computer Sci-ence）这一学术领域可以分为下面这些领域： 计算理论（Theoretical Computing） 信息和编码理论（Information and Coding The-ory） 算法和数据结构（Algorithm and Data Struc-ture） 形式化方法（Formal Methods） 程序设计语言（Programming Language）

Computer Science (Computer Sci-ence) this academic field can be divided into the following areas: computational theory (Theoretical Computing) information and coding theory (Information and Coding The-ory) algorithm and data structure (Algorithm and Data Struc-ture) form method (Formal methods) programming language (programming language)

偏实践的领域： 计算机体系结构（Computer Architecture） 并行计算和分布式系统（Concurrent, Parallel and Distributed System） 实时系统和嵌入式系统（Real Time and Embed-ded System） 操作系统（Operating System） 计算机网络（Networking） 科学计算（Scientific Computing） 安全和密码学（Security and Cryptography）

Partial areas of practice: computer architecture (Computer Architecture) parallel computing and distributed systems (Concurrent, Parallel and Distributed System) real-time systems and embedded systems (Real Time and Embed-ded System) OS (Operating System) computer network ( Networking) scientific computing (scientific computing) security and cryptography (security and cryptography)

人工智能（Artificial Intelligence） 这个领域涵盖了许多相关的领域，如模式识别（Pattern Recognition）、机器学习（MachineLearning）、数据挖掘（Data Mining）、信息提取（Information Retrieval）等。 计算机图形学（Computer Graphics）、计算机视觉（Computer Vision）、 多媒体（Multimedia） 数据库和大规模数据处理（Database and Large Scale Data Processing） 万维网（World Wide Web） 自然语言处理和语音（Natural Language Pro-cessing and Speech） 人机交互（Human Computer Interaction） 软件工程（Software ngineering）

AI (Artificial Intelligence) in this area covers a number of related fields such as pattern recognition (Pattern Recognition), machine learning (MachineLearning), data mining (Data Mining), information extraction (Information Retrieval) and the like. Computer Graphics (Computer Graphics), computer vision (Computer Vision), multimedia (Multimedia) and large-scale data processing database (Database and Large Scale Data Processing) WWW (World Wide Web) natural language processing and speech (Natural Language Pro-cessing and Speech) HCI (Human Computer interaction) software Engineering (software ngineering)

人类文明要向前发展，离不开思考、发现、构建。我曾经在微软亚洲研究院技术创新部工作过七年，我所在的工程团队和很多计算机科学家在不同领域一起做项目，这些项目各有特点：

* Build To Learn：开发软件，构建系统的目的是做进一步的试验，试图发现客观规律或探求某方法的优劣。这些项目经常是科研论文的基础工作。
* Build To Show：为了突出地展现某个技术的作用，开发一些以演示为目的的软件，这些项目很吸引眼球，经常获得新闻报道，但是功能未必全面。
* Build To Serve：为了服务一定范围的目标用户而构建的工具等，有时以公开SDK的形式发布。
* Build To Win：以在市场上赢得用户为目标而构建的软件。这也是种种科学发现、技术突破最好的试金石。

To move forward the human civilization is inseparable from thinking, discovery, build. I used to work at Microsoft Research Asia, the Ministry of technological innovation over seven years, and the engineering team where I do a lot of projects together computer scientists in different fields, these projects have their own characteristics:

o Build To Learn: developing software build system is designed to do further tests to try to find some objective law or explore the pros and cons of the method. These projects are often the basis for the work of research papers.

o Build To Show: In order to highlight the role to show a technology demonstration to develop some software for the purpose of these projects is very eye-catching, often get news, but the function may not be comprehensive.

o Build To Serve: To serve a range of target users and build tools, and sometimes in the form of public release of the SDK.

o Build To Win: to win customers in the market and to build for the target software. This is also a variety of scientific discoveries, technological breakthroughs best touchstone.

一个团队需要一定的流程来管理开发活动，每个工程师在软件生命周期所做的工作也应该有一个流程，这一章会介绍PSP（Personal Software Pro-cess，个人软件开发流程）。

如何能让自己负责的模块功能定义尽量明确，模块内部的改变不会影响其他模块，而且模块的质量能得到稳定的、量化的保证？单元测试就是一个很有效的解决方案。

A team needs some processes to manage development activities, the work of each engineer in the software life cycle should also be made to have a process, this chapter will introduce the PSP (Personal Software Pro-cess, individual software development process). How can responsible for their own module functions defined as clear as possible, change the internal module does not affect the other modules, and modules can be stable quality, quantitative guarantee? Unit testing is a very effective solution.

卡内基梅隆大学（CMU）的能力成熟度模型（CMM和CMMI），是用来衡量一个团队能力的一套模型。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。

Ability Carnegie Mellon University (CMU) Maturity Model (CMM and CMMI), is used to measure the ability of a team set of models. CMU experts for software engineers also have a model, called the Personal Software Process (PSP), PSP, and, like any other methodology, are not achieved overnight.

现代软件产业经过几十年的发展，一个软件由一个人单枪匹马完成，已经很少见了，软件都是在相互合作中完成的。合作的最小单位是两个人，两个工程师在一起，做的最多的事情就是“看代码”，每人都能看“别人的代码”，并发表意见。但是每个人对于什么是“好”的代码规范未必认同，这时我们很有必要给出一个基准线—什么是好的代码规范和设计规范。程序员写的代码是给人看的，还是给机器看的？人也看，机器也看，但是最终是人在看。我们的代码要让“旁观者”看得清清楚楚。

Modern software industry after decades of development, a complete software by one person alone, has rarely met, the software is done in mutual cooperation. Cooperation is the smallest unit of two people, two engineers together, to do most things is the "look at the code," everyone can see "someone else's code," and express their views. But everyone of what is "good" code specifications may not agree, then it is necessary to give us a baseline - what is good coding standards and design specifications. Programmers to write code posters, or to the machine to see? See also, the machine could not see, but ultimately people watching. Our code to make "bystander" to see clearly.

做一个有商业价值的项目，或者在团队里工作，代码规范相当重要。“代码规范”可以分成两个部分：

1. 代码风格规范。主要是文字上的规定，看似表面文章，实际上非常重要。
2. 代码设计规范。牵涉到程序设计、模块之间的关系、设计模式等方方面面的通用原则。

Do a commercial value of the project, or work in a team, code specifications is very important. "Code norms" can be divided into two parts:

1. Code style norms. The main provisions of the text on the seemingly superficial, in fact, very important.

2. Code design specifications. General principles of programming involves the relationship between modules, and other aspects of design patterns.

代码风格的原则是：简明，易读，无二义性。提示：这里谈的风格是一家之言，如遇争执，关键是要本着“保持简明，让代码更容易读”的原则，看看争执中的代码规范能否让程序员们更好地理解和维护程序。

Code-style principles: concise, readable, and unambiguous. Tip: Here to talk about the style of one of the words, in case of dispute, the key to the spirit of "keeping concise, to make the code easier to read" principle, take a look at the dispute in the code specifications can allow programmers to better understand and maintain program.

是用Tab键好，还是2、4、8个空格？ 结论：4个空格，在Visual Studio和其他的一些编辑工具中都可以定义Tab键扩展成为几个空格键。不用Tab键的理由是，Tab键在不同的情况下会显示不同的长度，严重干扰阅读体验。4个空格的距离从可读性来说，正好

行宽必须限制，但是以前有些文档规定的80字符行宽太小了（以前的计算机/打字机显示行宽为80字符），现在时代不同了，可以限定为100字符。

It is to use the Tab key to good, or 2,4,8 spaces? Conclusion: The four spaces in Visual Studio and some other editing tools can be defined Tab key to expand into several spacebar. Tab key is no reason, the Tab key will display different lengths in different situations, severe interference reading experience. From the four spaces from the readability, the just

Line width must be limited, but some 80 lines of characters before the document specified width is too small (previous computer / typewriter display line width of 80 characters), and now the times are different, can be limited to 100 characters.

如果我们做的项目是真实的，有具体而多变的需求，有工期、质量和资源的矛盾，团队成员各自的水平、目标也不一致，那么团队内部不可能没有矛盾。但是，矛盾不是一开始就爆发的，它有自己的生命周期，有不同的发展阶段。谈恋爱、两人合作项目，都有相似的几个阶段，下面我们以生活中的跳交谊舞为例，描述一下两人合作的各个阶段。

If we do the project is real, specific and changing needs, there are contradictions schedule, quality and resources, team members each level, the goal is also inconsistent, so the team can not be without internal contradictions. However, the contradiction is not the beginning of the outbreak, it has its own life cycle, there are different stages of development. Love, two cooperation projects have similar phases, following our life ballroom dancing, for example, describe the various stages of the two cooperation.

1. 萌芽阶段（Forming） 以跳舞为例，两人刚认识时，拘谨而彬彬有礼。这一阶段的现象：两人刚刚互相认识，这时大家都有礼貌，一般交流不少，每个人都想得到对方的接纳，试图避免冲突和容易引起挑战的观点。对即将进行的舞蹈，有不同的期望值，但是双方彼此并不了解。
2. 磨合阶段（Storming） 开始跳舞，开始踩脚。接触之后，才感到手足无措，眼睛不知往哪里看，才能感受到对方原来舞步是这样的……这样的笨拙。
3. 规范阶段（Norming） 跳舞逐渐和谐、合拍，团队成员就很多事情取得了一致。一些成文或不成文的规则逐步建立起来了。男方轻轻的一个手势，女方就知道如何旋转。
4. 创造阶段（Performing） 跳舞二人合而为一，为艺术而舞蹈（大家发出了唏嘘向往之声）。并不是所有的合作都能达到这一阶段，磨合太多后，我们还可能进入“解体阶段”。
5. 解体阶段（Deforming） 散伙，各走各的独木桥，回宿舍抱着板凳跳舞，或者另找舞伴。

1. infancy (Forming) to dance, for example, when the two first met, cautious and courteous. This phase of the phenomenon: They just know each other, then we are polite and generally a lot of exchange, everyone wants to accept each other, trying to avoid conflict and opinions prone challenges. Dance forthcoming, there are different expectations, but the two sides do not understand each other.

2. The run-in phase (Storming) began to dance, beginning step on the foot. After the contact, just feel helpless, his eyes do not know where to look, to feel each other original dance is so ...... so clumsy.

3. specification phase (Norming) gradually harmonious dance, step, team members made a lot of things the same. Some written or unwritten rules and gradually built up. A gesture the man gently, the woman will know how to rotate.

4. The creation stage (Performing) into one dance duo, art and dance (she sent a buzz and desire). Not all cooperation to achieve this stage, running too much, we may also enter the "disintegration of the stage."

5. disintegration stage (Deforming) disband, go their difficult path, back to the dorm holding bench dance, or find another partner.

可以看出，这些团队有共同的特点： 1. 团队有一致的集体目标，团队要一起完成这目标。一个团队的成员不一定要同时工作，例如接力赛跑。（王屋村搬砖的“非团队”成员则不然，每个人想搬多少就搬多少，不想干了就结算工钱走人。） 2. 团队成员有各自的分工，互相依赖合作，共同完成任务。（王屋村搬砖的“非团队”成员则是各自行动，独立把任务完成，有人不辞而别，对其他的搬砖人无实质影响。）

As can be seen, these teams have a common characteristic: 1. The team has the same collective goal, the team to accomplish this goal together. A team member does not have to work at the same time, such as a relay race. (King Estates moving bricks "non-team" members is not the case, how much each person would like to move much on the move, I did not want to leave the settlement wages.) 2. The team members have their own division of labor, dependence mutual cooperation and common tasks . (King Estates moving bricks "non-team" members of their action is independent of the task is completed, some people walked out, no real impact on other people moving bricks.)

瀑布模型（Waterfall Model） 当软件行业还在年幼的时期，它从别的成熟行业（硬件设计，建筑工程）借用了不少经验和模型。在那些“硬”的行业中，产品大多遵循 [分析→设计→实现（制造）→销售→维护] 这个流程。由于在“硬”行业中产品一旦大规模生产，要再返回去修改时就非常困难，甚至是不可能的。因此这个模型描述了单向的、不可逆的生产过程。

Waterfall model (Waterfall Model) When the period of the software industry is still young, it is from other mature industries (hardware design, construction) borrowed a lot of experience and models. In those "hard" in the industry, the products are mostly follow [Analysis → design → implementation (manufacturing) → Sales → Maintain] this process. Since the "hard" once the mass production of products in the industry, returns again to modify it very difficult, if not impossible. Therefore, this model describes the one-way, irreversible processes.

Rational Unified Process 统一流程（RUP）从瀑布模型开始的各种模型都有一个共同点：重计划，重事先设计，重文档表达。这一类的方法中集大成者要算Rational统一流程（Rational UnifiedProcess，RUP）[注释5]。RUP把软件开发的各个阶段整合在一个统一的框架里。要完成一个复杂的软件项目，团队的各种成员要在不同阶段做不同的事情，这些不同类型的工作在RUP中叫做规程（Discipline）或者工作流（Workflow）

Rational Unified Process Unified Process (RUP) various models from the beginning of the waterfall model have one thing in common: weight program, heavy pre-designed, re-expression of the document. This type of approach to be considered a master of the Rational Unified Process (Rational UnifiedProcess, RUP) [Note 5]. RUP all phases of the software development integration in a unified framework. To complete a complex software project, various members of the team to do different things at different stages, these different types of work called the procedure (Discipline) or in the RUP workflow (Workflow)

老板驱动的流程（Boss-Driven Process） 笔者在和中国一些企业的软件开发者交流的时候，听闻不少人提到开发流程事实上是由行政领导主导，或者由公司的老板驱动，我们姑且把它命名为老板驱动的流程（Boss-Driven Process）。

Owner-driven processes (Boss-Driven Process) and the author of some Chinese companies software developers exchange when heard many references to the development process in fact led by the executive leadership or by the company's owner-driven, we shall say it is named for the owner-driven processes (boss-driven process).

渐进交付的流程（Evolutionary Delivery），MVP和MBP 这个流程是史蒂夫·迈克康奈尔（Steve McConnell）在1996年总结的，但是它其实已经很接近现在大家谈论较多的迭代式开发流程。当系统的主要需求和架构明确之后，软件团队进入了一个不断演进的evolution循环中：[开发→发布→听取反馈→根据反馈做改进  
Progressive flow delivered (Evolutionary Delivery), MVP and MBP this process is Steve McConnell (Steve McConnell) in 1996, summed up, but it is already very close and now we are talking about more iterative development process. When the main demand and the architecture of the system clearly, software team into an evolving evolution cycle: [development → News → → listen to feedback to make improvements based on feedback

敏捷开发原则

1. 尽早并持续地交付有价值的软件以满足顾客需求
2. 敏捷流程欢迎需求的变化，并利用这种变化来提高用户的竞争优势
3. 经常发布可用的软件，发布间隔可以从几周到几个月，能短则短
4. 业务人员和开发人员在项目开发过程中应该每天共同工作
5. 以有进取心的人为项目核心，充分支持信任他们
6. 无论团队内外，面对面的交流始终是最有效的沟通方式
7. 可用的软件是衡量项目进展的主要指标
8. 敏捷流程应能保持可持续的发展。领导、团队和用户应该能按照目前的步调持续合作下去
9. 只有不断关注技术和设计，才能越来越敏捷
10. 保持简明—尽可能简化工作量的技艺—极为重要
11. 只有能自我管理的团队才能创造优秀的架构、需求和设计
12. 时时总结如何提高团队效率，并付诸行动

Agile development principles

1. The early and continuous delivery of valuable software to meet customer needs

2. Agile processes Welcome changing requirements, and take advantage of this change to improve the user's competitive advantage

3. regularly publish available software, publishing intervals from weeks to months, can be as short as short

4. Business people and developers in the project development process should work together every day

5. enterprising artificial heart project, the full support of their trust

6. Whether the team inside and out, face to face communication is always the most effective way to communicate

7. The available software is a key measure of the project's progress

8. Agile processes should be able to maintain sustainable development. Leadership, team and users should be able to continue at the current pace continue cooperation

9. Only constant attention to technology and design, in order to more agile

10. Keep simple - as simple as possible workload skills - extremely important

11. The only self-managing teams to create outstanding architectures, requirements and design

12. always summarize how to improve team efficiency, and put into action

敏捷流程的经验教训 这里有一些实践者的经验教训：

1. 敏捷宣言表明的是一些优先级，不必当作圣旨或者教条来争论。
2. Scrum Master不是一个官，而是一个没有行政权力的沟通者，就像微软的PM那样。他/她同时还要在团队中做具体的工作。直接把原来的“经理”变成Scrum Master，大多行不通。
3. 一些项目需要很多暗箱操作和政治角力才能搞定，Scrum会把这些矛盾都摆到明处。这有好处，也有风险。
4. 在复杂的项目里，要让一线团队成员做决定。
5. 创业公司的团队其实经常是运行在Scrum 的模式中（只不过大家太忙，没工夫论证自己到底有多么Scrum）。
6. 在Scrum计划阶段的估计不是一个“合同”，领导们不要把它当成一个合同。估计总是不准的。坚持短期的Sprint，这样即使不准的估计也不会有大的损害。
7. 不要和管理层谈“流程”，他们只关心“结果”。
8. 在大型团队、跨地区的团队，或者复杂项目中，Scrum并没有非常完美的答案，Scrum的创始人也承认这一点。

Lessons agile processes here are some practitioners lessons:

1. Agile Manifesto suggests some priorities, you do not have to be as controversial decree or dogma.

2. Scrum Master is not an officer, but a no executive powers communicators, like as PM Microsoft. He / She would also like to do specific work in a team. Directly to the original "manager" into Scrum Master, most of the work.

3. Some projects require a lot of secret operations and political wrangling to get, Scrum will these contradictions are placed in the daylight. This is good, there are risks.

4. In a complex project, let the front-line team members to make a decision.

5. The start-up companies in fact are often running team in Scrum modes (except everyone is too busy, no time to argue their own in the end how Scrum).

6. In estimation Scrum planning stage is not a "contract", the leaders do not regard it as a contract. Always inaccurate estimate. Insist on short-term Sprint, so that even if there will not be big inaccurate estimate of the damage.

7. Do not talk about management and "process", they only care about "result."

8. In large teams, regional teams, or complex projects, Scrum and no perfect answer, Scrum's founders admit it.

软件团队里除了能写代码、测试代码和画图做设计的成员，还有一类角色，不做上面这些事情但也很重要，我们叫他们项目经理——PM。PM的M就是Manager，但是P有这几种：Prod-uct Manager、Project Manager、Program Man-ager，在不同的行业和公司，他们的作用各不相同。这一章主要介绍微软的项目经理——ProgramManager。Product Manager：产品经理——正确地做产品。目前国内公司大部分PM都是指这个职位。产品经理对一个或多个产品或产品线负责，而互联网产品涉及到这些方方面面：产品定位、市场发展、需求分析、运营、营销、市场推广、商务合作。产品经理横跨这些部门，寻找资源，持续推进产品。随着产品的发展，不同公司，对PM要求会不一样。核心要求是，根据市场和用户需求，协调各部门资源，正确地把握产品定位和方向，解决用户的痛点，持续优化产品

In addition to the software team to write code, test code to do the design and drawing of members, there is a class role do these things above, but also very important, and we call them project managers --PM. PM of M is the Manager, but there are these types of P: Prod-uct Manager, Project Manager, Program Man-ager, in different industries and companies, their roles are different. This chapter introduces Microsoft's program manager --ProgramManager. Product Manager: Product Manager - to do the right product. At present, most of the company PM refer to this post. Product manager for one or more products or product lines is responsible, and Internet products related to these aspects: product positioning, market development, demand analysis, operations, marketing, marketing, business cooperation. Product Manager across these sectors, find the resources to continue to promote the product. With the development of products, different companies, the PM requirements will be different. Core requirement is based on market and customer needs, coordination of various departments resources, product positioning and correctly grasp the direction, to solve customer pain points, continue to optimize our

Project Manager：项目经理——正确地做流程。在某些公司，这个职位与产品经理分开单列。他们对项目流程负责，即项目从立项到上线按时完成。正确地协调团队内部外部，调配各部门资源和时间，有效进行风险管理，保证一个项目顺利按计划结项，是一个项目经理的核心价值

Project Manager: Project Manager - to do the process correctly. In some companies, this position separately with separate product manager. They are in charge of the project process, namely on-line project from the project to be completed on time. Proper coordination of internal and external teams, the deployment of the various departments of resources and time, effective risk management, to ensure the smooth knot according to plan a project, a project manager is a core value of

Program Manager：微软的职位名称。微软产品团队三足鼎立的角色分配就是PM、开发、测试。PM负责除产品开发和测试之外的所有事情。从某种意义上说，是前面两种角色的综合。微软通常有专门的产品策划（Product Planner），他们和市场部门的专职人员一起，负责产品的长期发展和市场推广。

Program Manager: Microsoft's job title. Microsoft product teams are assigned the role of the three pillars of PM, development, testing. PM is responsible for everything except for product development and testing. In a sense, the first two characters of synthesis. Microsoft usually have a special product planning (Product Planner), full-time staff and their market sector together, responsible for long-term development and marketing of products.

随着业务的发展和团队的壮大，下面这两个问题凸显出来：

1. 团队成员之间交流的成本急剧增长
2. 有很多开发和测试之外的事情，需要专人负责

些同学说，我写的代码都不用测试，我真想不到除了开发和测试之外，还有什么事情可做。我们看看微软公司有哪几类PM。 有做功能设计的PM；有些功能或产品需要深入掌握各个计算机科学分支的专业知识才能做好。例如Visual Studio中的各种计算机语言、框架、TFS的项目的项目经理，SQL Server、Windows Server、Azure、Bing Search核心算法等团队的PM 有些PM需要对商业和客户有很强的了解能力，例如Office办公软件的PM 有些PM需要具备广泛的经验和知识面，以及商业拓展能力，例如互联网MSN部门的PM 有些是驱动流程的PM，例如推动几百人的团队完成一个版本的开发，又如保证WindowsPhone在能在几十种不同硬件上发布 也有专门深入某一领域的PM，例如负责软件的国际化/本地化（Globalization/Localiza-tion） 还有和研究人员合作，琢磨如何将前沿技术引入主流产品，做技术转化的PM

With the business development and growth of the team, the following two questions stand out:

1. The sharp increase in the cost of communication among team members

2. There are many things to develop and test other than the person responsible needs

Some students said, I do not have to write code to test, I do not think in addition to development and testing, as well as what to do. We look at what Microsoft has several types of PM. There do functional design of PM; some features or products require a deep understanding of the various branches of computer science expertise to do a good job. Such as Visual Studio in a variety of computer languages, frameworks, project manager TFS project, SQL Server, Windows Server, Azure, Bing Search algorithm core team of some PM PM needs of business and the customer has a strong ability to understand, for example, Office office software some PM PM requires extensive experience and knowledge, as well as business development capabilities, such as MSN Internet division of PM and some are driven processes PM, for example, hundreds of people pushing to complete a version of the development team, and if the guarantee WindowsPhone release can also have specialized in a particular area of ​​the PM depth in dozens of different hardware, such as internationalization / localization (Globalization / Localiza-tion) as well as software and researchers responsible for cooperation, wondering how cutting-edge technology into the mainstream products, do technical transformation PM

成为一个合格的PM，需要哪些能力呢？

1. 观察、理解和快速学习能力PM要能够在一个新的领域中很快上手。PM要能理解用户，能站在用户的角度上考虑问题，观察发现用户不善于表达的需求，体察团队成员的言外之意，倾听老板/客户/利益相关人的弦外之音。要有能够理解别人的处境、心理、动机的能力——同理心。一个PM平时或许能玩转很多高技术的工具，但是当工作需要时，他/她能突然把自己变成一个完全不懂技术的菜鸟用户，从用户的角度来看问题。
2. 分析管理能力每天项目中发生的事情千头万绪，PM要能够分析出重点，找到优先级，做判断、做决定……
3. 一定的专业能力如果一定要说专业能力的话，PM的专业就是理解和表达，你能否理解不同人的心理、需求和言外之意？你能否借助文字、图表、草图，甚至代码来清晰准确地表达自己的想法？PM 要始终能满怀激情地向用户兜售产品，向团队兜售希望。PM通常也能写代码，能玩转Excel、PPT、Visio、甘特图，会PS，有文字功底，写的博客有人爱读，反正，总得有几招绝活吧！不用说还要有大量的阅读，对IT行业、用户心理、社会都要有广泛的了解。
4. 自省的能力。一个PM做第一个项目时可以拍脑袋定工期，拍胸脯打包票，最后拍屁股走人（谁没年轻过呢），但是失败之后要有自省和自我改进的能力。在生活中能不能锻炼PM的能力呢？当然可以，比如装修房子、组织一个大型活动、带自己的孩子、帮邻居家带三天孩子，等等。

Become a qualified PM, which need capacity?

1. observation, understanding and fast learning ability PM to be able to quickly get started in a new field. PM to be able to understand the user, to the user's perspective to consider the issue, observed that implication needs of the user inarticulate, understand the team members, listening to the boss / client / stakeholder overtones. We must be able to understand other people's situation, mental ability motivation - empathy. Usually a PM might be able to get along a lot of high-tech tools, but when work needs, he / she can suddenly transform itself into a completely non-technical novice user, from the user's perspective.

2. Analysis of daily management of a multitude of things happening in the project, PM to be able to analyze the priorities, find the priority, to make a judgment, decision ......

3. Be sure of professional competence If anything, professional competence, then, PM is the professional understanding and expression, you can understand the different human psychology, needs and implication? Can you help of text, graphics, sketches, and even the code to clearly and accurately express their thoughts? PM should always be able to enthusiastically sell the product to the user, we want to sell the team. PM also usually write code, to Fun Excel, PPT, Visio, Gantt charts, will be PS, have writing skills, writing blog someone to love to read, anyway, must have learned the recipe for it! Needless to say, there are also a lot of reading on the IT industry, user psychology, society must have a broad understanding.

4. introspective ability. You can set a PM racking our brains to do the first project period, each of the offerings promise last shot ass leave (who was young it did not), but after the failure to have the ability to self-reflection and self-improvement. In life can exercise ability PM it? Of course, such as decorating the house, organize a large-scale activities with their children, with three days to help a neighbor child, and so on.

在一个项目中，PM的具体任务是什么呢？他们的任务是：

1. 带领团队形成团队的目标/远景，把抽象的目标转化为可执行的、具体的、优美的设计；
2. 管理软件的具体功能的生命周期（需求/设想/设计/实现/测试/修改/发布/升级/迁移/淘汰）；
3. 创建并维护软件的规格说明书，让它成为开发/测试人员及时准确的指导，而不是障碍；
4. 代表客户和用户的利益，主动收集用户反馈，预期用户新的需求。协调并决定各种需求的优先级；
5. 分析并带领其他成员对缺陷/变更需求形成一致意见，并确保实施；
6. 带领其他成员确保项目保持功能/时间/资源的合理平衡，跟踪项目进展，确保团队发布令客户满意的软件；
7. 收集团队项目管理和软件工程的各种数据，客观分析项目实施过程中的优缺点，推动项目成员持续改进，从而提振士气。

In one project, the specific tasks PM What is it? Their task is to:

Objectives 1. Lead the team formation team / vision of the abstract goals into actionable, concrete, beautiful design;

2. The specific functions of the software life-cycle management (demand / idea / design / implementation / test / edit / publish / upgrade / migration / phase-out);

3. Create and maintain the software specification, let it become a developer / tester timely and accurate guidance, rather than an obstacle;

4. The representative of the interests of customers and users, the initiative to collect user feedback, the new user demand is expected. Coordinate and determine the needs of priority;

5. Analysis and led other members of the defect / change needs a consensus, and ensure implementation;

6. lead other members to ensure the project hold function / time / resources in a reasonable balance, track project progress, to ensure customer satisfaction team released software;

7. The various data collection team project management and software engineering, an objective analysis of the advantages and disadvantages of the project implementation process, project members to promote continuous improvement, boosting morale.

其实，计算机软件的用户界面（User Interface，UI）和用户体验（User eX-perience，UX）是一个有着丰富内容的学术领域，软件工程师们在长期工作中也积累了很多相关的经验

诺尔曼进一步阐明了设计的三个层次，以及对应的产品特性：本能（Visceral）层次的设计——外形行为（Behavior）层次的设计——使用的乐趣和效率反思（Reflective）层次的设计——自我形象、个人满足感、回忆三个层次的因素相互交织，共同影响了用户体验。大部分软件工程师主要关心的是“使用的效率”，这只是用户体验设计的很小的一部分。

In fact, computer software UI (User Interface, UI) and user experience (User eX-perience, UX) is a country with rich content of academic fields, software engineers work in the long term and has accumulated a lot of relevant experience

Norman further clarified the three levels of design and the corresponding product characteristics: instinct (Visceral) level design - shape behavior (Behavior) level design - fun and efficient use of reflection (Reflective) level design - self-image, personal satisfaction, recall three levels of factors are intertwined, and the combined effect of the user experience. Most software engineer is primarily concerned with "use efficiency", this is just a small part of the user experience design.

1. 尽快提供可感触的反馈系统状态 要有反馈，等待时间要合适。现在程序发生了什么，应该在某一个统一的地方清晰地标示出来。一个目标用户能够只靠软件的主要反馈来完成基本的操作，而不用事先学习使用手册。系统的反馈可以是视觉的、听觉的、触觉的（例如手机振动）。但是要避免简单重复的提示。
2. 系统界面符合用户的现实惯例（Familiarity，Avoid Surprise） 与用户沟通，软件系统要使用用户语言而不是开发者语言，所用的概念要贴近生活实际，而不是用学术概念或开发者的概念。我们说的生活实际，最好是目标用户的实际生活体验。例如，给病人使用的网络挂号系统，就不宜使用只有医务工作者才熟悉的术语和界面（最坏的结果是使用软件工程师才熟悉的术语和界面，而医护人员和病人对此很不熟悉）软件的反馈要避免带给用户惊奇——例如，在用户没有期待对话框的时候，软件从奇怪的角度弹出对话框，或者给用户提示“找不到对象”。
3. 用户有控制权 操作失误可回退，要让用户可以退出软件（很多软件都没有退出菜单，这是导致用户反感的一大原因）。用户可以定制显示信息的多少，还可以定制常用的设置。
4. 一致性和标准化 在软件中，对同一事物和同类操作的表示用语，各处要保持一致。例如，某词典软件有“帮助用户收集生词并且背诵生词”的功能。这个功能要有明确一致的称呼，不能混杂着叫“单词本”、“生词本”、“Word List”、“Word Book”、“单词文件”……等等。
5. 适合各种类型的用户 我们的软件要为新手和专家提供可定制化的设计。一些操作方式，如快捷操作，用户可以自行调整。我们还应该为存在某些障碍的用户（色弱、色盲、盲人、听力有缺陷的用户、操作键盘鼠标不方便的用户等）提供一定程度的便利。对于长期使用某个软件的用户，软件应该能适应用户的使用习惯，让用户越用越顺手，最后产生感情上的好感和忠诚度。
6. 帮助用户识别、诊断并修复错误 软件的关键操作要有确认提示，以便帮助用户及早消除误操作。要注意使用朴素的语言来表述错误信息。错误信息需要给出下一步操作提示（我现在出错了，那下一步怎么办）。必要时提供详细的帮助信息，并协助用户方便地从错误中恢复工作。让所有的用户都可以通过电子邮件或者表单来提交反馈意见。有些程序用一对简单的笑脸/哭脸符号来鼓励用户提交反馈，这也是很好的办法。
7. 有必要的提示和帮助文档 不需要文档，用户就能使用自如，当然更好，必要时还可以提供在线帮助。如果软件和用户的工作相关（而不是简单的游戏），那么基本的提示和帮助文档还是很有必要的，而且也要提供便利的检索功能。文档要从用户的角度出发描述具体步骤，并且不要太冗长。有些软件在首次启动时会通过图示或动画展现某些新功能的用法，或引导用户进行一些基本的设置（例如第一次使用输入法时，让用户选择候选词的个数、字体大小，等等）。这些都是不错的方法。在PC桌面软件时代，软件团队总是要等到项目的稳定阶段才开始写“帮助文档”，因为之前的软件界面和功能还有很多变化，然后要很快写好，才能和软件一起发布。在互联网时代，离线的帮助文档进步到“联机帮助”网页；在大量带宽和活跃的用户社区帮助下，我们可以看到用户创造的如何高效使用各种软件的视频。这应该给软件团队很多启发——如何能用好各种形式的“帮助文件”

1. To provide tactile feedback as soon as possible to have the system status feedback, the waiting time to the right. The program is now what is happening, it should be clearly marked in one unified place. A target user is able to rely on the main feedback software to perform basic operations without first learning to use the manual. Feedback system can be visual, auditory, tactile (for example, the phone vibrates). However, to avoid repetition of simple tips.

2. system interface match the user's real practice (Familiarity, Avoid Surprise) to communicate with users, the software system to use the language of the user rather than the developer language, the concept used to be close to real life, rather than the concept of academic term or developers. We are talking about real life, real life experience is the best target user. For example, the network used by the patient registration system, it should not be used only to medical workers are familiar with the terminology and interface (the worst results were obtained using software engineer was familiar with the terminology and interfaces, and the medical staff and patients of this very familiar) to avoid feedback software gives the user a surprise - for example, the user does not expect the dialog box, the software at odd angles from the pop-up dialog box or prompt the user to "find the object."

3. The user has control over operating errors can be rolled back, let the user can exit the software (some software does not exit the menu, which is leading a major cause of resentment user). Users can customize the amount of information displayed, you can customize common settings.

4. The consistency and standardization in the software, for a term representing the same thing and the same operation, to be consistent throughout. For example, a dictionary software "to help users gather and recite the words words" function. This function must be clear and consistent call can not be mingled with is called "the word of this", "Glossary", "Word List", "Word Book", "word document" ...... and so on.

5. suitable for all types of users of our software to provide customized design for novices and experts. Some mode of operation, such as fast operation, the user can adjust. We should also exist certain impaired users (color weakness, color blindness, the blind, hearing impaired users, inconvenient keyboard mouse user, etc.) provide a degree of convenience. For long-term use of a software user, the software should be able to adapt to the user's habits, allowing users to more easily use more, and finally generate goodwill and loyalty emotional.

6. help users identify, diagnose and repair errors in the software key operations have confirmation prompt, the user as soon as possible in order to help eliminate misuse. Note that the use of plain language to express the error message. Error Message need to give prompt the next operation (I'm wrong, then the next step how to do). It provides detailed information to help when necessary, and to help users easily restore your work from your mistakes. So that all users can submit feedback via e-mail or form. Some programs use a simple pair of smiley / sad face symbol to encourage users to submit feedback, which is a good solution.

7. necessary tips and help files do not need to document, users can freely use, of course better, if necessary, can provide online help. If the software and the user's work-related (rather than a simple game), then the basic tips and help documentation is necessary, but also offers a convenient search function. Documents from the user's point of view describe the specific steps and not too lengthy. Some software will show when you first start the usage of some new functions by way of illustration or animation, or guide the user through some basic settings (for example, when you first use the input method, allowing users to select the number of candidate words, font size, and many more). These are a good way. In the PC desktop software era, software team always have to wait until the stabilization phase of the project began to write "Help documentation" because the previous software interface and features, there are many changes, and then you want to quickly write, to publish and software together. In the Internet era, offline help document progress to "Online Help" page; in large amounts of bandwidth and active user community help, we can see how efficient use of various software user-generated video. This software should give the team a lot of inspiration - how to use all sorts of forms of "help file"

在我们熟悉的计算机和IT领域，所有我们看到的“酷”的东西，都是几代人、许多团队前赴后继持续创新的结果。就像拼图一样，很多聪明人都模糊地看出了最终图像，都在一块一块地拼接，往往拼好最后一块的人得到了最大的荣誉。但是没有前人的积累，没有自身扎实的功力，就没有“最后一块”等着大家去拼。另一个推论是——不要一开始就想着找到并拼对所有的拼图块，以为能够打造一个巨大的创新。彼得·德鲁克（Peter Drucker）说过： Those entrepreneurs who start out with the idea that they’ll make it big – and in a hurry – can be guaranteed failure.

We are familiar with computer and IT field, we see all the "cool" things that are several generations of people, many teams waging continuous innovation results. Like a jigsaw puzzle, a lot of smart people are vaguely see the final image in a piece of splicing often fight the last piece of good people has been the greatest honor. But there is no accumulation of their predecessors, did not own a solid skill, there is no "last piece" waiting for you to spell. Another corollary is - do not start thinking to find and fight all the puzzle pieces, that can create a huge innovation. Peter Drucker (Peter Drucker) said: Those entrepreneurs who start out with the idea that they'll make it big - and in a hurry - can be guaranteed failure.

当然，这时也必须认识到危险所在。优秀团队有时会骄傲自满，团队成员自我感觉太好，过分亲近也可能导致过度利己，不重视与别的团队合作，不重视客户需求等。最近IT行业的故事一再验证了“伟大的公司离破产只有十八个月”这一规律，优秀团队的领导切莫等闲视之。

Of course, this time must also recognize the danger. Excellent team sometimes complacent, team members feel good, too close may also lead to excessive self-interest, not importance to cooperation with other teams, we do not pay attention to customer needs and so on. Recent IT industry has repeatedly verified the story of the "great company from bankruptcy only eighteen months," the law was not to be taken lightly outstanding team leader.