SEMIntro：导论：

1、软件工程定义：1.将系统化的、严格约束的、可量化的方法应用于软件的开发、运行和维护，即将工程化应用于软件；2.在 1 中所述方法的研究。

2、软件维护：软件开发完成后，对软件的问题进行解决，增加功能，与开发阶段有很重要的联系。

3、软件工程知识域：软件需求、软件设计、软件构造、软件测试、软件维护、软件配置管理、软件工程管理、软件工程过程、软件工程模型与方法、软件质量、软件工程职业实践、软件工程经济学、计算基础、数学基础、工程基础

Ch1 没有银弹

1、主要思想：没有任何一种单纯的技术或管理上的进展，能够独立地承诺十年内使生产率、可靠性或简洁性获得数量级上的进步；所有大家看到的技术、管理方法都不会给软件开发带来意想不到的效果；软件开发在根本上就是困难的（Brooks 认为**根本困难**是固有的概念复杂性）

2、根本任务：打造由抽象软件实体构成的复杂概念结构

3、次要任务：使用编程语言表达这些抽象实体，在空间和时间限制内将它们映射成机器语言。除非次要任务占了所有工作的 9/10，否则即使全部次要任务的时间缩减到零，也不会给生产率带来数量级上的提高.（硬件的发展使次要任务越来越容易解决）

4、软件项目的现状：常常看似简单明了的东西，却有可能变成一个落后进度、超出预算、存在大量缺陷的怪物。（**软件开发 Vs 硬件开发**：不是软件发展慢，是硬件发展太快）

6、探寻软件产业发展的问题：软件开发中的根本问题：软件特性中固有的困难，规格化、设计和测试这些概念上的结构，而不是对概念进行表达和对实现逼真程度进行验证。次要的问题：出现在目前生产上的，但并非那些与生俱来的困难。一个相互牵制关联的概念结构，是软件实体必不可少的部分，它包括：数据集合、数据条目之间的关系、算法、功能调用等等。这些要素本身是抽象的，体现在相同的概念构架中，可以存在不同的表现形式。尽管如此，它仍然是内容丰富和高度精确的。

7、软件系统中无法规避的内在特性：复杂度、不一致性、可变性和不可见性。银弹与软件的内在特性相悖。Brooks 认为软件开发中困难的部分是规格化、设计和测试这些概念上的结构，而不是对概念进行表达和对实现逼真程度进行验证，那么软件开发总是非常困难的，天生就没有银弹。**1.复杂度**：软件实体很复杂，软件的复杂度是必要属性,不是次要因素。抽掉复杂度的软件实体描述常常也去掉了一些本质属性。复杂度导致沟通困难,影响可靠性,使程序难以使用和扩充,造成很多安全机制状态上的不可见性,引发了管理上的困难。 **2.不一致性**：软件开发面对的复杂度往往是随心所欲、毫无规则可言的，来自若干必须遵循的人为惯例和系统。软件开发面面对的是人，不是上帝；很多复杂性来自保持与其他接口的一致。**3.可变性**：软件实体经常会遭受到持续的变更压力。软件很容易修改（它是纯粹思维活动的产物，可以无限扩展）；软件的变更来自于人们要求扩展、更改功能和硬件的变化，软件与整个社会联成一体，后者在不断变动，它强迫软件也跟着变动。**4.不可见性**：软件是不可见的和无法可视化的；软件的客观存在不具有空间的形体特征。限制了个人的设计过程，也严重的阻碍了相互之间的交流。

9、当年的银弹：Ada 和其他高级编程语（它通过使用更加抽象的语句来开发，降低了机器的次要复杂度）、**面向对象编程**（它们的出现都消除了开发过程中的非本质困难，允许设计人员表达自己设计的内在特性，而不需要表达大量句法上的内容；对于抽象数据类型和层次化类型，它们都是解决了高级别的次要困难和允许采用较高层次的表现形式来表达设计；使得修改局部化，提高了可维护性）、**人工智能**（AI-1：使用计算机来解决以前只能通过人类智慧解决的问题。AI-2：使用启发式和基于规则的特定编程技术）、**专家系统**（专家系统是包含归纳推论引擎和规则基础的程序。专家系统最强有力的贡献是给缺乏经验的开发人员提供服务，用最优秀开发者的经验和知识积累为他们提供了指导）、**图形化编程**（现在的屏幕非常小，像素级别，无法同时表现软件图形的所有正式、详细的范围和细节；软件非常难以可视化）、**程序验证**（能够在系统设计级别、源代码级别消除 bug，可以在大量工作被投入到实现和测试之前，通过采用证实设计正确性，彻底提高软件的生产率和产品的可靠性？程序验证不意味着零缺陷的程序；完美的程序验证只能建立满足技术说明的程序）、**环境和工具**（回报很有限）、工作站（硬件速度的加快；编译速度，开发速度；1986!）、**购买和自行开发**（适用软件）、**需求精炼和快速原型**（详细的技术需求的困难）、**增量开发——增长而非搭建系统**（客户；士气；迭代式开发）、**卓越的设计人员**

10、没有银弹的影响：软件开发本质的认识；软件过程

Ch2 大教堂和市集

自由软件和商业封闭软件：“一种是封闭的、垂直的、集中式的开发模式，反映一种由权利关系所预先控制的极权制度；而另一种则是并行的、点对点的、动态的开发模式。”

1、Fetchmail：Linux 开发模式：1.每一个好的软件的起因都是挠到了开发者本人的痒处 2.好程序员知道该写什么，伟大的程序员知道该重写（和重用）什么（伟大程序员的一个重要特点是建设性的懒惰。他们知道你是因为成绩而不是努力得到奖赏，而且从一个好的实际的解决方案开始总是要比从头干起容易；Linus 并不是从头开始写 Linux 的。相反的它从重用 Minix）**3.计划好抛弃，无论如何，你会的**（你常常在第一次实现一个解决方案之后才能理解问题所在，第二次你也许才足够清楚怎样做好它，因此如果你想做好，准备好推翻重来至至一次）**4. 如果你有正确的态度，有趣的问题会找上你**（你在思考、审视一些你感兴趣的软件时，你会有新的更好的想法） **5.当你对一个程序失去兴趣时，你最后的责任就是把它传给一个能干的后继者**（在开源软件的开发中）**6.把用户当作协作开发者是快速改进代码和高效调试的有效方式 7. 早发布、常发布、听取客户的建议**（尽量早尽量频繁的发布是 Linux 开发模式的一个重要部分，多数开发人过去都相信这对大型工程来说是个不好的策略，因为早期版本都是些充满错误的版本，而你不想耗光用户的耐心；这种信仰强化了建造大教堂开发方式的必要性；Linus 的创新并不是这个(这在 Unix 世界中是一个长期传统)，而而是把它扩展到和他所开发的东西的复杂程度相匹配的地步，而且因为他培育了他的协作开发者基础，比其他任何人更努力地充分利用了 Internet 进行合作，所以这确实能行） **8. 如果有一个足够大的 beta 测试人员和协作开发人员的基础，几乎所有的问题都可以被快速的找出并被一些人纠正**（如果有足够多的眼睛，所有的错误都是浅显的；建造教堂和集市模式的核心区别，**在建造教堂模式的编程模式看来**，错误和编程问题是狡猾的、阴险的、隐藏很深的现象，花费几个月的仔细检查，也不能给你多大大确保把它们都挑出来的信心，因此很长的发布周期，和在长期等待之后并没有得到完美的版本发布所引起的失望都是不可避免的；**以市集模式观点来看，**在另一方方面，我们认为错误是浅显的现象，或者至少当暴露给上千个热切的协作开发人员，让他们来对每个新发布进行测试的时候，它们很快变得浅显了，所以我们经常发布来获得更多的更正，作为一个有益的副作用，如果你偶尔做了一个笨拙的修改，也不会损失太多；两种版本：Linus 也做了一些改进，如果有一些严重的错误，Linux 内核的版本在编号上做了些处理，让用户可以自己选择是运行上一个“稳定”的版本，还是冒遇到错误的险而得到新特征，这个战略还没被大多数 Linux 黑客所仿效，但它应该被仿效，存在两个选择的事实让二者都很吸引人）**9.聪明的数据结构和笨拙的代码要比相反的搭配工作的更好 10. 如果像对待最宝贵的资源一样对待 beta 测试员，他们就会成为你最宝贵的资源。11. 想出好主意是好事，从你的用户那里发现好主意也是好事，有时候后者更好。 12 最有突破和有创新的解决方案常常来自于你认识到你对问题的概念是错误的。** （你在开发中碰壁了，头破血流，反省后才能得到最好的解决方案）**13.“最好的设计不是再没有什么东西可以添加，而是再也没有什么东西可以去掉。”14.任何工具都应该达到预期的用处，但是一个伟大的工具会带来你从来预期不到的用处 15.当写任何种类的网关型程序时，多费点力，尽量少干扰数据流，永远不要抛弃信息，除非接收方强迫这么作！16. 如果你的语言一点也不像是图灵完备的，严格的语法会有好处 17. 一个安全系统只能和它的秘密一样安全，当心伪安全 18.要解决一个有趣的问题，请从发现让你感兴趣的问题开始**（最好的开发是从作者解决每天工作中的个人问题开始的，因为它对一大类用户来说是一个典型问题，所以它就推广开来了）**19.如果开发协调人员有至少和 Internet 一样好的媒介，而且知道怎样不通过强迫来领导，许多头脑将不可避免地比一个好**（自由软件的将来将属于那些知道怎样玩 Linus 的游戏的人，把大教堂抛之脑后拥抱市集的人，这并不是说个人的观点与才气不再重要，而是，自由软件的前沿将属于从个人观点和才气出发的人，然后通过共同兴趣自愿社团的高效建造来扩展）

2、Linus 定律——“如果有足够多的眼睛，所有的错误都是浅显的”；如果有一个足够大的 beta 测试人员和协作开发人员的基础，几乎所有的问题都可以被快速的找出并被一些人纠正。

3、Delhpi 效应（神庙效应）——社会学家在几年前已经发现一群同样内行的(或同样无知的)观察者的平均观点比在其中随机挑选一个来得更加可靠。

4、Brooks 定律——向一个进度落后的软件项目中增加开发人员只会让它更加落后，他声称项目的复杂度和通讯开销以开发人员的平方增长，而工作成绩只是以线性增长，在众多软件项目中，缺乏合理的时间进度是造成项目滞后的最主要原因，它比其他所有因素加起来的影响还要大。

5、“忘我(egoless)的编程”中，Weinberg 观察到在开发人员不顽固保守自己的代码，鼓励其他人寻找错误的地方，软件改进的速度比其他地方有戏剧性的提高。虽然编码是一个人活，真正伟大的工作来自利用整个社团的脑力，在一个封闭项目中只利用自己脑力的人会落在知道怎样创建一个开放的、进化的，成百上千的人在其中查找错误和进行修改的环境的开发人员之后。

6、集市风格的必要的先决条件——1.不能以一个市集模式从头开发软件，我们可以以市集模式测试、调试和改进，但是以市集模式从头开始一个项目是非常困难的。(linux 也不是从头开始的）2.当你开始创建社团时，你需要的是一个有趣的项目。3.协调人要能把从他人那里得到的好的设计重新组织起来 4.市集项目的协调人或领导人必须有良好的人际和交流能力 5.为了建造一个开发社团，你需要吸引人，你所做的东西要让他们感到有趣，而且要保持他们对他们正在做的工作感到有趣，保持他们对他们正在做的工作感到高兴，技术方面对达成这些目标有一定帮助，但这远远不是全部，个人素质也有关系。

7、怎么解释开源与 Brooks 定律的矛盾？用 Internet 沟通代价很小；开源项目的通讯结构是核心开发者、beta 测试人员、协助开发者。Brooks 基于一个前提：每个人都与其他所有人交流。但是在开源项目中，外沿的开发者做的实际上是平行分离的子项目，彼此交流很少；代码变动和臭虫报告都流经项目的核心，只有在小小的核心团体中全面的布洛克成本才有效。

8、结论：也许最终自由软件文化将胜利，是因为商业世界在进化的军备竞赛中不能战胜自由软件社团，因为后者可以把更大更好的开发资源放在解决问题上。

Ch3 开源软件经济学

很多高科技公司投入巨额资金发展开源软件，而通常开源软件本身免费。这些公司并不是放弃资本主义，而是认为这是个好的商业策略。

1、替代物品&互补物品

替代物品是首选商品太贵时会改买的另一种东西。**互补物品**是通常会和其它产品一起购买的产品。当商品的价格下降时互补物品的需求就会增加（当计算机变便宜就会有更多人买，由于计算机要有操作系统，于是操作系统的需求就增加，而操作系统的盈利也水涨船高）

2、为什么公司要支持开源？很多有责任尽量提升股东价值的大型上市公司，投入很多资金支持开放源码软件(通常是负担大型程序团队的开发费用),可以用**互补物的原理**来解释。聪明的公司试图让产品的互补物普及化，产品的需求就会上升而你就可以卖贵一点然后赚更多钱。要让开源软件成为自己产品的互补品而不是替代品。

IBM 已成为 IT 顾问公司，为不同行业提供整体解决方案。企业软件是 IT 顾问的互补物品。

微软的策略：微软的目标是要让 PC 市场普及化。互补物 MS-DOS 的需求也增加了。Xbox 策略：用普及的 PC 硬件而非订制零件。微软 Xbox 另一个策略是使用 DirectX。这里的目标是想普及化视讯芯片，卖出更多获利的游戏。视频游戏产业模式:硬件价格低，依靠软件盈利。为了要让服务器来赚钱 **Sun 的两大策略**：1.推广并发展免费软件让软件普及化；2.推广 Java 及其架构和让硬件普及化。**Sun 和 HP 研究 Gnome**：盒子。为了让软件普及化然后由硬件赚更多钱。

6、开源软件获利方法：1、需要获利空间向其上下游两方面延伸，并且和运营商确定合适的商业模式（可采用我方技术，服务入股，运营商设备网络投入折成股份，根据比重对客户所产生的移动网络费用及使用量来分红）2、针对更加上游客户，提供更多功能，包括类似 GPS 定位，设定 10 米的相隔距离，当双方不知不觉走散时，其移动电话会自动报警等功能，为客户提供更多增值服务，以增加获取利润的空间。

Ch4 人月神话

1、进度总成问题： 在众多软件项目中，缺乏合理的时间**进度**是造成项目滞后的最主要原因，它比其他所有因素加起来的影响还大。**Why?** 1 对估算技术缺乏有效的研究；2 采用的估算技术隐含地假设人月可以互换，错误地将进度与工作量相互混淆；3 由于对自己的估算缺乏信心，软件经理通常不会有耐心持续地进行估算这项工作；4 对进度缺少跟踪和监督。其他工程领域中，经过验证的跟踪技术和常规监督程序，在软件工程中常常被认为是无谓的举动；5 当意识到进度的偏移时，下意识（以及传统）的反应是增加人力。这只会使事情更糟，从而进入了一场注定会导致灾难的循环。（合理的做法是不停更新估算）

2、乐观主义：所有的编程人员都是乐观主义者，所有系统编程的进度安排背后的第一个假设是：一切都将运作良好，每一项任务仅花费它所“应该”花费的时间。**创造性活动**分为三个阶段：构思、实现和交流。不像电器安装，编程基于十分易掌握的介质，编程人员通过非常纯粹的思维活动——概念以及灵活的表现形式来开发程序。正由于介质的易于驾驭，我们期待在实现过程中不会碰到困难，因此造成了乐观主义的弥漫

3、“人月的不合理性”：1. 成本的确随开发产品的人数和时间的不同，有着很大的变化，进度却不是如此。用人月作为衡量一项工作的规模是一个危险和带有欺骗性的神话。它暗示着人员数量和时间是可以相互替换的。2.人数和时间的互换仅仅适用于以下情况：某个任务可以分解给参与人员，并且他们之间不需要相互的交流。这在系统编程中近乎不可能。3.当任务由于次序上的限制不能分解时，人手的添加对进度没有帮助。4.对于可以分解，但子任务之间需要相互沟通和交流的任务，必须在计划工作中考虑沟通的工作量。因此，相同人月的前提下，采用增加人手来减少时间得到的最好情况，也比未调整前要差一些。5.沟通所增加的负担由两个部分组成，培训和相互的交流。每个成员需要进行技术、项目目标以及总体策略上的培训。这种培训不能分解，因此这部分增加的工作量随人员的数量呈线性变化。6.相互之间交流的情况更糟一些。如果任务的每个部分必须分别和其他部分单独协作，则工作量按照 n(n-1)/2 递增。7. 一对一交流的情况下，情况会更加恶劣。所增加的用于沟通的工作量可能会完全抵消对原有任务分解所产生的作用。8. 因为软件开发本质上是一项系统工作，沟通、交流的工作量非常大，它很快会消耗任务分解所节省下来的个人时间。从而，添加更多的人手，实际上是延长了，而不是缩短了时间进度。

4、系统测试：由于乐观主义，通常实际出现的缺陷数量比预料的要多得多。不为系统测试安排足够的时间简直就是一场灾难；直到项目的发布日期，才有人发现进度上的问题；此时此刻的延迟具有不寻常的、严重的财务和心理上的反应。

5、进度安排：1/3 计划 ，1/6 编码 ，1/4 构件测试和早期系统测试 ，1/4 系统测试

6、空泛的估算：还没有可靠的估算技术出现；在基于可靠基础的估算出现之前，项目经理需要挺直腰杆，坚持他们的估计，确信自己的经验和直觉总比从期望派生出的结果要强得多。

7、实例研究——一个软件项目落后于进度：1. 重新安排进度；2. 削减任务。在现实情况中，一旦开发团队观察到进度的偏差，总是倾向于对任务进行削减。当项目延期所导致的后续成本非常高时，这常常是唯一可行的方法。

Ch5 敏捷软件开发

敏捷开发不是银弹。

敏捷软件开发：推进开发迭代；强调面对面交流；将可运行的软件作为评判开发过程的主要标准。

敏捷宣言：个人交互胜于过程和工具；可以工作的软件胜于完整的文档；客户的合作剩余合同的磋商；应对变更胜于遵循计划。

与计划驱动的比较：适应性，短计划 VS 可预期（严格遵循预先计划的需求、分析、设计、编码、测试的步骤顺序进行，变更代价高昂导致难以应对变更），短周期 VS 长周期；可运行软件进行度量 VS 可交付产物进行度量；敏捷仍有严格的规范，与最终产物有关。

敏捷方法特点：迭代式，关注交互交流，减少资源敏感的中间产物

什么情况下使用敏捷（适合处理不可预测的经常变更的需求）：

产品角度：需求不明确，需求变更剧烈，不适合重要的，高可靠性，高安全性；组织角度：三因素：文化，人，交流。组织文化支持协商，员工必须被信任，员工总数少，高技能，习惯由开发人员作出技术决定，有效帮助团队成员相互交流；项目规模：20-40，面对面交流；需求变更：要求需求稳定度

1、软件项目**成功传统观点**：1.成功的:按时完成，费用不超过预算，而且所有特性和功能都符合原先的设计规格；2.不太成功的:已完成并且可以运行，但费用超出了预算，没有如期完成，拥有的特性和功能少于原先的设计规格；3. 失败的:在开发周期的某个时刻被取消了

2、敏捷对于项目成功的观点：为客户创造价值是评价成功的最重要标准，所有软件开发实践都应该以提升项目收益为首首要目标。项目控制没有那么重要。

3、敏捷的本质：是承认软件开发的复杂性。而且承认，这种复杂性，达到了这样一种程度：“无法通过足够充分的前期准备，而消除后续的风险。甚至于，前期准备得越是充分，后续的风险越大”；软件开发具有复杂性和可变性；软件开发需求可变性“实际上具有其合理性”。

4、如何做到敏捷：敏捷并不仅仅是一种可以遵循的具体过程；而是一种软件开发哲学、一组实践、一套互补的原则和一个社区；敏捷软件开发是一种思考软件开发的方式：敏捷宣言和敏捷原则。

5、敏捷价值观、原则和实践：技术是重要的，如果不掘土、种植，你就不是在做园艺。**实践**（站立式会议，持续集成、测试驱动开发、重构）是你天天做的事情，对实践进行详细描述是有用的，它们是明确和客观的。比如在编码前写测试代码，即使你不懂为什么，这种实践仍然是有价值的；价值观是知识和理解的另外一个层次，价值观是在某种处境中我们喜欢或不喜欢某事情的根源，XP 价值观是“沟通,简单,反馈,勇气,尊重”。**价值观与实践的关系**，没有价值观，实践很快会变成生搬硬套，缺乏目的或方向，价值观和实践相结合程序员才能高效的执行实践；实践是价值观的表现；价值观是在一个高层次上的表达，可以以价值观的名义做任何事；实践让价值观清晰可见；实践是清晰明了的；价值观是普适的，理想情况下，工作中的价值观和生活中其他时候是一样的，但实践却差别很大。原则，在价值观和实践之间架起桥梁的是原则；原则是生活中具体领域的指导方针。

敏捷缺陷：缺乏结构和必要文档;初级开发者难以掌握;早期软件设计不够;要求公司文化变更;合同签署方面问题;有时会没有效率;项目开始时无法估算(投标问题);范围剧烈变更的危险(没有详细的需求文档);功能驱动,无法表示非功能需求;缺乏文档,设计;没有达到宣称的效果：大规模提高生产力。

敏捷软件开发人作用的认识:“敏捷型方法”是面向人的而非面向过程，它们试图使软件开发工作顺应人的天性安排，而非逆人，它们强调软件开发是一项愉快的活动，实施敏捷开发的一个关键之处是大家接受一个过程而非强加过程，通常软件开发过程是管理人员决定这种过程经常受到抵制，特别是管理人员已经脱离软件开发过程很长时间了，而接受一个过程需要一种自愿能力，这样大家就能积极参与进来，正由于 IT 行业技术发展变化之快，所以管理层必须信任和依靠当前的开发人员。

6、敏捷宣言遵循的原则：我们最重要的目标，是通过持续不断地及早交付有价值的软件使客户满意；欣然面对需求变化，即使在开发后期也一样；为了客户的竞争优势，敏捷过程掌控变化；经常地交付可工作的软件，像个几星期或一两个月，倾向于采取较短的周期；业务人员和开发人员必须相互合作，项目中的每一天都不例外；激发个体的斗志，以他们为核心搭建项目；提供所需的环境和支援，辅以信任，从而达成目标；不论团队内外，传递信息效果最好效率也最高的方式是面对面的交谈；可工作的软件是进度的首要度量标准；敏捷过程倡导可持续开发；责任人、开发人员和用户要能够共同维持其步调稳定延续；坚持不懈地追求技术卓越和良好设计，敏捷能力由此增强；以简洁为本，他是激励减少不必要工作量的艺术；最好的架构、需求和设计出自自组织团队；团队定期地反思如何能提高成效，并以此调整自身的举止表现。

Ch6 Scrum

1、Scrum 的概念：Scrum 不是构建产品的一种过程或一项技术，而是一个框架，在这个框架里可以应用各种流程和技术。Scrum 能使产品管理和开发时间的相对功效显现出来，以便随时改进。Scrum 框架包括 Scrum 团队及其相关的角色、事件、工件和规则。框架中的每个模块都有一个特定的目的，对 Srum 的成功和使用都至关重要。Scrum 框架包括三个角色（产品负责人、Scrum Master、团队）四个仪式（Sprint 计划会议、每日站会、Sprint 评审会议、Sprint 回顾会议）三个物件（产品 BackLog，Sprint BackLog，燃尽图）

2、Scrum 角色:1.产品负责人代表了客户的意愿。产品负责人编写用户故事,排出优先级,并放入产品订单。2.Scrum 主管促进 Scrum 过程,他的主要工作是去除那些影响团队交付冲刺目标的障碍。Scrum 主管确保 Scrum 过程被按照初衷使用。Scrum 主管是规则的执行者。3.负责交付产品的团队。一个团队通常由 5 至 9 名具有跨职能技能的人(设计者,开发者等)组成,承担实际的开发工作。

3、Scrum 会议:在冲刺中,每一天都会举行项目状况会议。今天你完成了那些工作？明天你打算做什么？完成你的目标是否存在什么障碍？(Scrum 主管需要记下这些障碍)

4、Scrum 文档:1.产品订单(product backlog)是整个项目的概要文档。2.冲刺订单(sprint backlog)要在冲刺中完成的任务的清单。3.燃尽图(burn down chart)在冲刺长度上显示每天进展的图。

5、Scrum 流程:在每一次冲刺(一个 15 到 30 天的周期,其长度由开发团队决定)当中,开发团队创建可用的(可以随时推出)软件的一个增量。每一个冲刺所要实现的功能来自产品订单(product backlog)。那些订单项会被加入一次冲刺由冲刺计划会议决定。 在会议中,产品负责人告诉开发团队他需要完成产品订单中的哪些订单项。开发团队决定在下一次冲刺中他们能够承诺完成多少订单项。 在冲刺的过程中,没有人能够变更冲刺订单(sprint backlog),这意味着在一个冲刺中需求是被冻结的。每一个冲刺完成后,都会举行一次冲刺回顾会议,在会议上所有团队成员都要反思这个冲刺。举行冲刺回顾会议是为了进行持续过程改进。会议的时间限制在 4 小时。Scrum 提倡所有团队成员坐在一起工作,进行口头交流,以及强调项目有关的规范(disciplines),这些有助于创造自我组织的团队。Scrum 的一个关键原则是承认客户可以在项目过程中改变主意,变更他们的需求,而预测式和计划式的方法并不能轻易地解决这种不可预见的需求变化。同样,Scrum 采用了经验方法－ 承认问题无法完全理解或定义,而是关注于如何使得开发团队快速推出和响应不断出现的需求的能力最大化。

6、纸牌估算法： 1.每个团队成员拿到一组卡片；2.产品负责人或者一名团队成员扮演阅读者的角色，他负责阅读需要估算产品 backLog 的条目，并且询问大家是否有疑问；3.团队讨论这个条目；4.当团队理解了这个条目之后，每个团队成员按包括我自己的想法给出估算结果，并且选择对应的扑克出牌，估算结果不能告诉他人，出牌时字面朝下；5.阅读者向大家确认是否都已经确定估算结果，确认后，大家同时展示估算结果 6.团队评估不同的估算结果（一致？有分歧？有什么未考虑？）最终达成一致，若差异不可接受则返回 3；7.返回 2，开始估算下一条目。**估算差异的原因：**需求理解不同（复杂登录 Or 简单登录）；技术不熟悉（无某方面经验的人估算这一领域）**为什么使用这个方法** 1.团队的智慧要高于某一个人的智慧（Delphi 效应）；真正参与工作的人做出的估计要高于其他人做出的估计。**价值：**传统估算通常是一个人在思考，而使用指派估算是，鼓励跨职能团队的多个团队成员参与估算，团队成员可以从不同的视角来思考和分析问题，估算的过程中考虑的更加全面、估算也更加准确 2.在估算过程中，团队对估算的结果进行讨论和评判，在一个高度透明的环境下，估算的结果更加真实和客观。这样也避免了很多时候过于武断，或是拍脑袋做出的决定 3.估算的过程也是一个只是分享和学习的过程，对某一个条目不清楚的成员通过其他成员的阐述会增加对该条目涉及到的要点的认识。

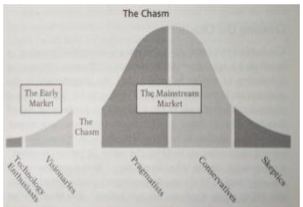
Ch7 高科技市场

1、Life Cycle：早期市场(1,2);鸿沟;保龄球道;龙卷风(3);主流市场(3,4);消亡

2、用户：1.**狂热的技术爱好者**,没有钱,但是有影响力 2.**有远见的人**,期望能够首度使用新特性而提高竞争力,是第一个可以带来真正的 money 的群体 3.**实用主义者**,相信演化而非变革,当产品被证明有效之后才会购买。偏向于向行业领先者购买。4.**保守主义者**,对从科技投入中获利持悲观态度,只会在被迫情况下从事,对价格敏感,怀疑,很多要求。他们的要求很少被满足,一部分原因是他们不希望对额外的服务有所付出,有的只是对高科技的刻薄观点。5.**怀疑者** ,讨厌高科技,喜欢挑战高科技市场,不以之为销售目标

3、1980s 高科技市场策略:从向技术热爱者宣传新产品开始,吸引有远见的人;一旦引起了有远见的人的兴趣,做任何使客户满意的事,使其可以作为实用主义者的好参考;通过实用主义者得到利润(税收)的大部分,以成为市场领导者为目标,制定实际的标准;通过在实用主义者身上得到的成功经验,使产品足够可靠和便宜来满足保守主义者的需求。忽略怀疑论者。

有远见的人 VS 实用主义者	
支持变革	支持进化
反向投资者	遵奉者
打破常规	和大众在一起
听自己的指令	咨询同仁
冒风险	管理风险
以为来的机会为动机	以解决现在的问题为动机



寻找什么是可能的	追求什么是可靠的

4、六个市场阶段:

1.早期市场，客户是狂热的技术爱好者和想抢先使用新产品的有远见的人 2.鸿沟，早期市场变小,主流市场因为解决方案的不成熟还没有接受产品,在特定领域的小规模市场被认可(如果跨越了鸿沟,则进入主流市场,由实用主义者和保守主义者组成)3.保龄球道，在进入主流市场之前,在特定市场被接受的时期,针对有特定需求的消费者,厂商愿意制作特定的有利可图的整个产品。 购买力:终端用户，是消费者的市场。4. 龙卷风,被大规模市场接受.形成 reference point。一个公司成为市场领导者,其他会效仿;购买力:技术团体,IT 部门;实用主义者,在同一时期内作出选择,选择行业领先者。卖方的市场(进入龙卷风的标志:形成产品类别)5.Main Street,市场后期,基础需求已经被满足,目标是充实潜在需求。供大于求,购买力:终端用户,提升满意程度(取决于替代品出现的时间,不可预测)6. End of life。以产品为基础转向以服务为基础。

5、市场领导者的优势:卖的更贵;大量订单,高市场占有率,低成本;低销售成本

6、策略:

（1）跨越鸿沟目的:高科技财富主要来源于主流市场，跨越鸿沟是企业的必然选择。

（2）跨越鸿沟策略：1、完整产品（有远见的人和实用主义者的最大不同在于前者希望尽早使用,愿意购买还不完备的产品而后者只愿意购买完整的产品。整个产品必须包含产品和服务的最小子集以保证目标用户有充足的理由去购买）；2、营销方式：D-Day，诺曼底登录（将大量的精力和资源集中在一个有限的市场空白中,在鸿沟期中只关注销售量的驱动作用会带来致命的后果，口碑对高科技市场非常重要。）

（3）要考虑的问题:1 是否建立了目标用户群体,他们是否已经准备好作为主要的消费群体 2 他们是否有强烈的理由去购买 3 是否可以借助合作者的帮助,交付一个完整的产品来满足用户的需求 4 当前市场是否没有确立的竞争者防止我们成为行业领先者 5 如果占领了当前特定市场后,是否能进入额外的市场

（4）保龄球道:原则:1.不要和在同类产品上强于你的公司竞争,关注最终用户团体对市场发展的影响,而不是技术团体 2.积累资本和信用,以期成为行业领先者 3.尽可能占据多的有利可图的市场 4.不要和已经取得领先地位的竞争者竞争(避免竞争)。选择 segment 的两要素:1.该领域的用户有强烈的理由购买你的产品。2.该领域当前没有别的竞争者提供很好的服务。当进入一个新的 segment 时,尽力投资以保证自己的领先地位。然后转到另一个 segment。选择相对小的 segment,以保证能够取得领先地位。(好处:1 现在赚钱 2 跟着市场领导着走)

（5）跨越保龄球道策略：获取收入；争取市场领导者地位

（6）龙卷风:尽可能的出售产品。不需要定制化,专注于自身发展而非客户。加快产品的开发,发布,安装,使用。注意供应链和质量,确保产品可靠性。攻击竞争者以确保领先地位。

（7）三类角色 Gorilla 领域的市场领头人 Chimpanzees 有较强竞争力的公司;必须在价格,交付,产品特征和服务上做妥协Monkeys 进入市场晚,没有隐形成本,对整体没有贡献,没有研发及市场投资,在龙卷风阶段开始后才进入市场。策略是模仿市场领导者的产品并且低价出售。

（8）主流市场：市场增长不依靠销售基本产品给新的消费者,而是为存在的消费者在基础平台上开发特殊化的扩展 1.The whole product + 1,针对不同用户提供新功能。找到自身投入小但能获得高市场回报的新功能。2.研发部门要和销售部门合作,发现用户在意的新特性。

（9）离开主流市场后:以产品为基础转向以服务为基础。

龙卷风,保龄球道,主流市场的比较:

Bowling Alley（保龄球道）	Tornado（龙卷风）	Main Street
关注商业购买者和最终用户，接近 infrastructure buyer	关注 infrastructure buyer	卖给最终用户
强调投资回报作为购买的强烈理由	忽略 ROI,关注即时部署可靠的基础设施	关注用户使用体验,试图满足他们的个体需求
为了单独的应用修改整个产品	将整个产品做成通用的	产品加上有竞争力的新特性以赢得市场
与 value-added 分发渠道合作,以定制产品符合客户需求	通过低成本,高产量的渠道分配以最大化市场曝光率	继续原来的渠道,关注+1 市场的销售规划
使用基于价值的价格以最大化利润	使用有市场竞争力的价格最大化市场份额	使用增加新特性,在低成本的基础上获取高收益
避免竞争以获得特定市场份额	攻击竞争者以获得主流市场份额	和自己的低成本产品竞争,以获得剩余市场份额
在特定垂直市场定位产品	作为全球范围水平定位产品	定位于特殊市场,基于最终用户的个体喜好

注:手机跨越鸿沟策略:

主要以争取市场份额为主,瞄准一个细分市场,争取在某个细分市场上获得突破成为行业的领导者,这样可以在可以为保龄球道阶段争取一个好的竞争位置。暂时不考虑成本问题,当然也要控制一定的成本范围内,当开发技术和开发成本都不再成为发展的障碍时,用户的关注度就成为下一个争取的目标,这是可以通过免费提供技术服务的方法以吸引客户的眼球。还要继续融资,将资金投放到广告市场,让广大的潜在客户知道和了解这个新兴业务。当市场上客户普遍接受该项服务功能,并且相应的竞争对手纷纷倒下的时候,才可以正式推出新的基于这个基本功能上的收费功能,以获取更大的利润。

7.非连续性创新：任何时候，我们面对的新产品需要我们改变自己一贯的行行为模式，或者需要对我们目前依赖的产品或服务进行行改变时，这种创新在学术上称为“非连续性创新”或“破坏性创新”。相对的创新称为“连续性创新”，指产品正常升级，不需要我们改变行为。

Ch8 颠覆式创新

1、为什么管理良好的企业会遭遇失败?

它们实际上一直管理不善；这些遭遇失败的企业的管理已经做到了极致，但它们在大获成功后做出决策的方式，却最终埋下了它们日后失败的种子。

2、良好的管理：

听取客户的意见；大力投资客户表示希望得到进一步改善的技术；争取更高的利润率；以更大的市场——而不是更小的市场——为目标

3、延续性技术：延续性技术所具有的共同点就是，它们都是根据主要市场的主流客户一直以来所看中的性能层面，来提高成熟产品的性能。这些技术有可能是突破式的，也有可能是渐进式的。大部分技术改进都是延续性技术。最具突破性、最复杂的延续性技术，也很少会导致领先企业失败。

4、破坏性（颠覆性）技术：其性能要低于主流市场的成熟产品，但它们却拥有一些边缘客户所看中的新特性。基于破坏性技术的产品通常价格更低、性能更简单、体积更小，而且通常更便于客户使用。最初应用于非主流市场或新的市场，积累经验并得到足够投资后，提高产品性能，最终占领主流市场。率先进入一个新的市场具有先发优势。领先企业的失败大部分都是破坏性技术导致的。

5、如何应对颠覆式创新？

1、把开发破坏性技术的职责赋予存在客户需求的机构，设立一个独立的、满足小型市场发展的机构。

2、设立一个能够欣然接受较小收益的独立小型机构

3、为失败做好准备

6、为什么不能在市场明确后投入力量进入市场？

因为数据证明先行者有巨大的领先优势，可以用鸿沟理论来解释。

7、为什么采用新技术不能过晚？

IT 人员都非常重视口碑传播，他们经常在一起讨论一个问题，是否到了采用一个新技术的时间；如果太晚采用一个新技术，会使得公司处于不利的竞争地位，个人会失去竞争能力；如果太晚，IT 人员将使得公司陷入高科技产品的生命周期尾端，维护成本会大大增加。