**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

**3、软件工程VS计算机科学：**目标（在资源的限制条件下构建满足用户需求的软件系统**VS**探索计算和建模方法，并改进）、产品、进度和时间表、关注（如何为用户实现价值**VS**软件本身运行原理（时间空间复杂度、算法正确性））、变化程度、需要的其他知识（**SE：**相关领域知识**CS：**数学）

**4、软件工程VS计算机程序设计：**1.软件工程存在于各种应用中，存在于软件开发的各个方面。而程序设计通常包含了程序设计和编码的反复迭代的过程，它是软件开发的一个阶段；2.软件工程力图对软件项目的各个方面做出指导，从软件的可行性分析直到软件完成以后的维护工作。

**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.软件开发中的根本任务**：打造由抽象软件实体构成的复杂概念结构。**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）

**3.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。

**4.软件系统的四个内在特性**：**复杂度**：软件实体可能比任何由人类创造的其他实体都要复杂，因为没有两个软件部分是相同的；数字计算机本身就比人类建造的大多数东西复杂；软件实体的扩展不仅仅是相同元素的重复添加，而必须是不同元素实体的添加。**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；很多复杂性来自保持与其他接口的一致。**可变性**：软件实体经常会遭受到持续的变更压力；软件可以很容易地进行修改——它是纯粹思维活动的产物，可以无限扩展；软件的变更：人们要求扩展、更改功能，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。**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;这限制了个人的设计过程，也严重地阻碍了相互之间的交流;UML

**e.g.** UML通过建模和规约，提高了软件系统的一致性；通过绘制模型解决了不可见性；集成了面向对象的概念，在软件有持续变更压力的情况下，提高了可维护性；提供了与高级语言等价的图形标记，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。

**5.当年的银弹：Ada和其他高级编程语**（这类语言出现最大的回报来自出现时的冲击，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）**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）

**Ch2大教堂和市集**

**1.大教堂模式：**封闭的、垂直的、集中式的开发模式；**市集模式：**并行的、点对点的、动态的开发模式。**“Linus定律”：**如果有一个足够大的beta测试人员和协助开发人员的基础，几乎所有的问题都可以被快速地找到并被一些人纠正。**“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。**“Delphi效应”**：一群相同专业的（或相同无知的）观察者的平均观点比在其中随机挑选一个来得更加可靠。

**2. 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）

**3.集市风格的必要的先决条件：**不能以市集模式从头开发一个软件，可以以市集模式测试、调试和改进。但开始创建社团时需要演示的是一个诺言，你的程序不需要工作地很好，它可以很粗糙、很笨拙、不完整和缺少文档、但它不能忽略的东西是要吸引大家卷入一个有趣的项目。

**4. 交流与沟通：**在《人月神话》一书中，Fred Brooks观察到程序员的工作时间是不可替代的：在一个误了工期的软件项目中增加开发人员只会让它拖得更久，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，这个说法被称为“Brooks定律”，被普遍当作真理，但如果Brooks定律就是全部，那Linux就不可能成功 。

**5、Linux与Internet：**1.Linux是第一个有意识的成功的利用整个世界作为它的头脑库的项目， Linux的孕育和万维网的诞生相一致并不是一个巧合，而且Linux在1993-1994的一段ISP工业大发展和对Internet的兴趣爆炸式增长的时期中成长起来，Linus是第一个学会怎样利用Internet的新规的人。2. Linux世界的行为更像一个自由市场或生态系统，由一大群自私的个体组成，它们试图取得(自己)最大的实效，在这个过程中产生了比任何一种中央计划都细致和高效的自发的改进的结果3. Linux黑客取得的最大化的“实际利益”不是经典的经济利益，而是无形的他们的自我满足和在其他黑客中的声望，(有人会说他们的动机是“利他的”，但这忽略了这样的事实：利他主义本身是利他主义者的一种自我满足的形式)

**6、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。

**7、结论：**也许最终自由软件文化将胜利，不是因为协作在道德上是正确的或软件“囤积居奇”在道德上是错的(假设你相信后者，Linus和我都不)，而仅仅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。**Netscape的策略：**让web浏览器普及化。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。

**4.互联网开发模式：**把用户当作协助开发者，尽量早尽量频繁的发布，充分利用Internet进行合作的开发模式。**互联网商业模式**：整合传统商业类型，连接各种商业渠道。???

**百度的商业模式：**主要业务是网络搜索服务，以竞价排名作为主要营销模式

**Google的商业模式：**在搜索业务中增加竞价机制和按点击量付费机制，但广告排名顺序并不是单靠花钱买来的，而是综合考虑广告受欢迎程度，引入点击率因素来决定排序的变化。同时，其业务以搜索为核心，分别向电子书、新闻和移动终端渗透。

**Ch4跨越鸿沟**

**1.破坏性创新：**会改变消费者一贯的行为模式，或者需要对目前依赖的产品或服务进行改变的创新。

**2.当行业领先企业面临破坏性创新**，非连续性创新产品进入市场后，会受到新技术热衷者和有远见者的欢迎，但随后就会陷入鸿沟，销售萎靡。因为这时早期市场变小，而大众市场却仍然不能接受不成熟的新产品。

**3.早期市场：**这是⼀个激动人心的时期，因为消费者是那些新技术热衷者和有远见者，成为⾸首批购买新产品的消费者。**保龄球道：**产品已经在大众市场里有了立足之地，但还没有被整个市场所接受。**龙卷风暴：**这是一个大众接受时期，因为此时的大众市场已经过渡到了使用新产品的阶段。**主街：**这是一个市场发展繁荣的时期。因为社会基本接受新的产品，而下一步的目标则是更好的挖掘潜力。

**4.跨越鸿沟策略：**完整产品+营销方式（D-Day,诺曼底登陆）**完整产品：**能够保证满足目标消费者需求的最少量产品(或服务)。未找到并验证完整产品：和竞争对手一起努力，找不到一起死亡；已经找到并验证完整产品：消灭所有竞争对手，争取市场领导地位。**营销方式：**将大量的精力和资源集中在一个有限的市场空白中。

**5.未跨过鸿沟的产品：**蓝光光盘和HDDVD。上网本……

**6.为什么采用新技术不能过晚：**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；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。

**7.为什么在利基市场，不马上进入龙卷风暴？**对于很多消费者而言，他们可以继续使用旧的产品，虽然他们知道新的产品的优势，但没有非常强的理由来购买；虽然产品跨过鸿沟，对于部分消费者而言是完整产品，但还没有向整个市场证明。**策略：**获取收入，争取市场领导者地位。

**8.市场领导者对高科技市场的重要作用：**实用主义者不愿意承担风险，他们重视秩序和稳定性。市场缺乏清晰的领导者意味着不稳定。实用主义者看到新产品的价值，并且希望将来能够采用，但没有市场领导者，市场没有标准、架构。⼀旦市场领导者出现，在一个自由市场上，秩序将自然出现。

**9.为什么第三方厂商愿意支持市场领导者？**通常市场领导者的市场份额较大，为第三方厂商提供了⼀个巨大的市场。

**10.第三方厂商和消费者：**当第三方厂商出售产品和服务时，完善了市场领导者的产品；客户通过购买第三方厂商产品，使得他们在市场领导者产品上的投资更加有价值，从而推动更多的购买市场领导者产品。

**11.市场领导者的优势：**有权利将同样的产品以更高的价格出售，获取超额利润，甚至它的产品有缺陷；由于出货量大，单位成本低；由于实用主义者愿意购买市场领导者的产品，市场领导者的营销成本很低

**12.为什么会形成龙卷风暴？**IT部门，实用主义消费者总是需要新技术，而且他们既不能过早采用新技术，也不能过晚采用新技术；过早意味着风险！

**13.大猩猩：**一家公司成为了市场领导者；**黑猩猩：**一到两家公司成为了市场领导者的竞争者；**猴子：**其他占有少量市场份额的企业。

**14.两种组织架构模型：**复杂系统（complex systems）：客户多为大型企业。客户基础数量少，交易次数极少，但平均价格高，客户需求定制化。大量交易（volume operations）：客户多为消费者。企业的核心技术不需要为顶尖的高科技，而是符合大众市场的一般标准品。客户群以百万记，个别客户不是焦点，单次交易金额少，但频率高，总数目多。

**Ch4创新者的窘境**

**1.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。

**2.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。（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。）

**破坏性技术：**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。

**价值网络：**⼀种大环境，企业正是在这个大环境下确定客户需求，并对此采取应对措施，解决问题，征求客户的意见，应对竞争对手，并争取利润最大化；价值网络决定了资源分配，成熟企业以客户为导向的资源分配和决策流程，决定了企业内部高层管理者无法真正实现企业转型，主导力量来自机构以外的力量。

**3.应对颠覆性创新：**把开发破坏性技术的职责赋予存在客户需求的机构；设立⼀个能够欣然接受较小收益的独立小型机构；为失败做好准备。

**4.为什么不能在市场明确后投入力量进入市场？**有些领导者不希望自己去尝试找到新的市场，而是希望在别人找到后进入。数据证明先行者有巨大的领先优势！！！可以使用鸿沟理论来解释先行者的领先优势。

**5.成功使用破坏性创新理论的公司：**英特尔、新罕布什尔南方大学、salesforce.com。

**6.其他应对破坏性创新时需要考虑的问题：**怎样判断出某技术是否具有破坏性？它的性能曲线日后有可能与主流市场需求相交汇；新产品的市场到底在哪里；应该采取什么样的产品技术和经销策略。

**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（每个人都知道我是否参加了早上的站立式会议，但我是否重视沟通并不是那么明确和具体的）；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、敏捷软件开发宣言：**我们一直在时间中探寻更好的软件开发方法，身体力行的同时也帮助他人。由此我们建立了如下价值观：个体和互动高于流程和工具，工作的软件高于详尽的文档，客户合作高于合同谈判，响应变化高于遵循计划。也就是说尽管右项有其价值，我们更重视左项的价值。

**7、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。

**Ch6CMMI**

**1.请描述以下本学期CMMI-DEV模型的五个不同成熟度等级的特征，并在此基础上试论述伴随着每个等级的提升（例如，1级升2级，2级升3级等等）其过程改进的终点和出发点在哪里?**

**初始级（Initial）：**过程通常是随机的、混乱和无序的；**已管理级（Managed）：**组织中的项目确保需求得到管理，过程已经计划、执行、度量和控制；**已定义级（Defined）：**建立标准的，且不断改进的工作方式；**量化管理级（Qualitatively Managed）：**过程得到测量和控制；**持续优化级（Optimizing）：**关注过程改进。

1->2：从随机、混乱、无序的阶段，建立尊重质量的纪律文化，使得流程在项目范围内得到遵守。

2->3：建立一个贡共享文化，建立组织流程责任体系、培训体系以及知识共享体系。

3->4：建立一个客观的文化，采用定量管理方法进行度量和控制。

4->5：建立主动文化，发现问题解决问题，并发动组织变革。（起点：analyses are concerned with addressing special causes of process variation； measurements are analyzed quantitatively to establish objectives and requirements for product quality, service quality, and process performance。终点：analyses are concerned with addressing common causes of process variation measurements are used to select improvements and innovations estimate the costs and benefits of the improvements and innovations measure the actual cost and benefits of the improvements and innovations。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Process | Project | Engr. | Support |
| 5 | OPM |  |  | CAR |
| 4 | OPP | QPM |  |  |
| 3 | OPD,OPF,OT | IPM,RSKM | RD,TS,PI,VER,VAL | DAR |
| 2 |  | PP,PMC,REQM,SAM |  | CM,PPQA,MA |

**OPD:** Organizational Process Definition

**OPF:** Organizational Process Focus 组织过程专注

**OT:** Organizational training 组织培训

**IPM:** Integrated Project Management集成项目管理

**RSKM：**Risk Management 风险管理

**PP:** Project Planning 项目规划

**PMC:** Project Monitoring and Control 项目监控

**REQM:** Requirements Management 需求管理

**SAM:** Supplier Agreement Management 供应商协议管理

**CM:** Configuration Management 配置管理

**8.（1）开源软件：谬误1：**开源软件是完全无法盈利的。**原因：**任何开源软件条款（附加条款除外）都没有规定不得将代码或软件用于盈利，而仅仅是需要在使用的过程中承担一点责任。有一大波软件公司是纯粹靠开源软件盈利的。**谬误2：**开源软件就是在鼓励抄袭等未授权的使用。**原因：**开源软件也反对未经授权的非法使用，用户在使用开源软件时必须在法律和开源协议的范围内。

**（2）软件测试：谬误1：**软件开发完成后才进行测试。**原因：**软件测试不应只是软件项目的收尾工作，而应该在软件生命周期的每一阶段中都包含测试。在软件项目的每个阶段，都需要进行不同目的和不同内容的测试活动，以保证各个阶段工作产品输出的正确性。**谬误2：**测试是测试人员的工作，和开发人员无关。**原因：**开发和测试是相辅相成的过程，在项目开发过程中，一般由开发人员针对模块进行白盒测试，这是早期的测试。后期对于测试人员所发现的缺陷，开发人员应根据优先级来进行修复，针对开发人员的修改，测试人员还要进行再测试和回归测试工作。**谬误3：**测试是为了证明软件的正确性。**原因：**测试无法证明软件是正确的，只能证明软件无法按照既定的规格和标准执行。测试的目的是尽可能的发现错误。

**（3）敏捷开发：谬误1：**敏捷是反文档的。**原因：**在实施敏捷的时候，需要在团队内明确哪些知识是必须显性的，这些知识可以通过文档交流。哪些知识是可以隐性的，这些知识则完全可以通过口头的方式进行交流，以达到沟通的最佳效率。**谬误2：**敏捷是“一个”过程。**原因：**敏捷不是一个过程，是一类过程的统称，它们有一个共性，就是符合敏捷价值观，遵循敏捷的原则。

**Note:**

**1.** **模块验收标准：哪些模块可以接收，哪些要拒绝**

待集成的组件必须要有完整的接口描述；必须是从配置库中导出的代码；必须配有相应的测试报告，证明经过测试并有修正

**2.客户需求：**客户提出的问题。**产品需求：**根据问题设计产品方案。

**3. Verification:**你设计出的产品和确定的解决方案是否一致，解决方案正确的被实现。

**Validation:**你根据解决方案做出产品，那产品是否解决了用户需求？