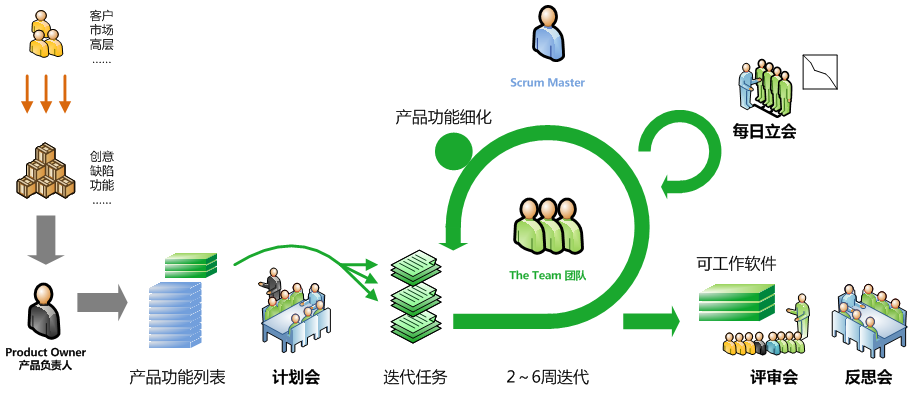
**Scrum敏捷开发过程实战**

**产品级，大团队的敏捷实战方法**



**需求结构化**

**需求描述**

**版本规划**

**迭代计划**

**日常活动**

**团队建设**

与传统灌输理念的培训不同，此实战培训中不只包含“按客户价值进行优先级排序”“利用自组织团队发挥主观能动性”等含糊的指导性思想，更在每个阶段均介绍一种或多种直接可以使用的方法来完成落地。

按照实际项目的开发顺序，培训分为三个环节，其主要内容如下：

* **需求结构化与需求描述**（主要受众为**产品负责人Product Owner、团队骨干**）
  + 将产品愿景转换为可实现的业务需求；
  + 将高层业务需求分解为具备层级结构的需求树；
  + 编写用户故事，面向用户使用场景而非产品功能描述单条需求；
* **版本规划与迭代计划**（主要受众为**产品负责人、Scrum Master，团队骨干**）
  + 在宏观层面上，确认整个产品中所有子系统的优先级，并将其顺序计划到版本和迭代中；
  + 在微观层面上，利用Scrum计划会估算每个迭代中任务的工作量；
* **日常活动与团队建设**（主要受众为**Scrum Master，团队成员**）
  + 日常活动中，利用每日立会、故事板、看板跟进开发进度；
  + 团队建设中，利用自组织团队、松结对编程等方法建立师徒制度，在实际工作中培养队员；
  + 在大型、跨职能团队研发时的团队结构与工作方式
* ***附：敏捷设计与工程实践****（仅出现于3天培训中，主要受众为****团队成员及技术管理者****）*
  + *如果从用户故事经过简单设计得到代码结构*
  + *如何利用用户故事来产生、管理测试用例*
  + *如何利用用户故事来管理变更、缺陷和客户反馈*

课程将围绕每个小组实际工作中各自产品或项目的自身需求展开，通过对其进行结构化、用户故事化、用户建模、模拟计划会估算、设定验收标准等，从而演练Scrum各个环节所需的技能。知识及案例讲解约占70%，实际练习约占30%。

注：本大纲中以一个易于理解的电子商务系统的研发为例，实际应用时可应用于银行、电信、政府、电子商务、互联网社区娱乐、仪器仪表等各种主流行业。

**×××××××××××××××××××××××××第一天××××××××××××××××××××××××××××××**

# 概述

***本阶段培训通过简短介绍，让学员大致了解敏捷开发的历史及其尝试解决的问题。***

* 敏捷开发尝试解决的问题
* Scrum及其历史
  + 产品负责人 Product Owner
  + 产品负责人团队
  + 产品负责人的职责

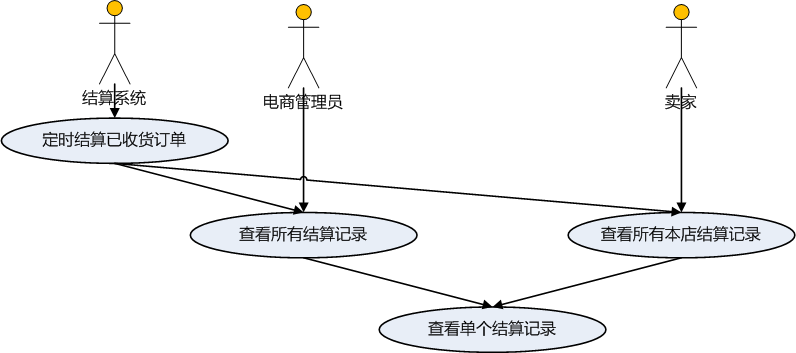
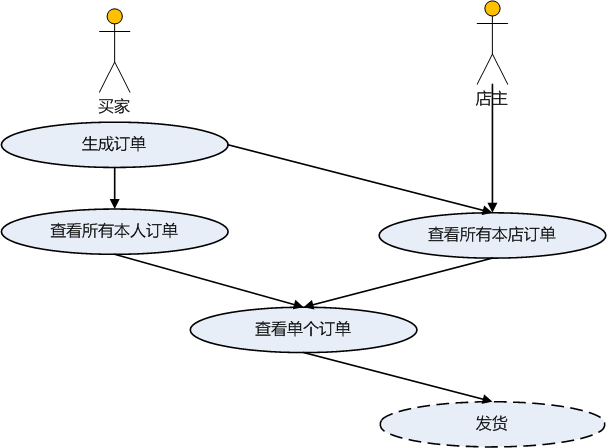
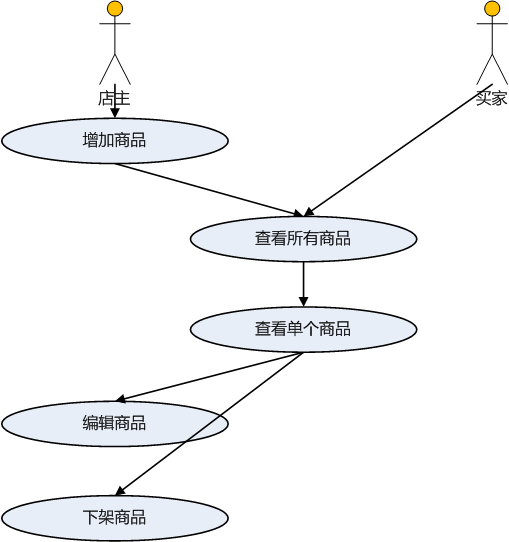
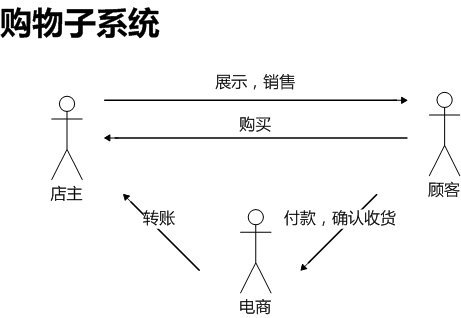
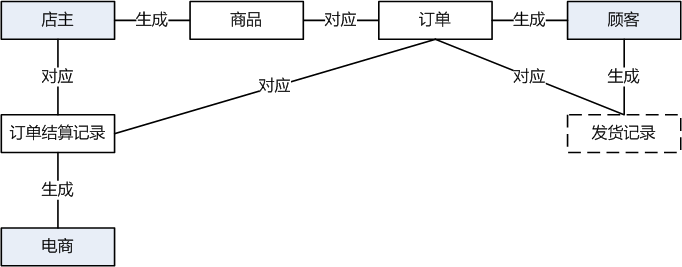
现场演练：分组并推选Product Owner

# 第一阶段：需求结构化与需求描述

***本阶段培训旨在从头到尾打通需求，即从感性的产品愿景分解到可供开发的具体需求条目。***

*传统敏捷开发使用“条目化”的方法来表达需求。但具体分解和描述需求的方法停留在“每个故事交付一个客户价值”“从客户价值角度描述需求”等非常含糊、难以落地的层面上。这导致分析所得的需求个体差别很大，难以作为大型、长期产品研发的正式工程文档。*

*本课程引入了QUML作为分界用户故事的基础，通过三层需求完成从产品愿景到可开发任务的分解：*



\*配图中的三层分解流程见下文分步描述。

*三层需求分别为业务愿景（左上图），业务操作（左下图中的方框，即史诗故事），业务数据（右侧三张图中的椭圆，即用户故事）。*

*这就解决了传统敏捷开发中存在的以下问题：*

1. *最初的产品愿景与割裂的用户故事之间缺少必然联系*
2. *大量用户故事之间缺少清晰的结构*
3. *用户故事颗粒度大小不一*

*此外，这种体系下建立起来的“用户故事树”（需求树）还能：*

1. *直接分配到开发任务中*
2. *直接生成代码结构*
3. *直接用于结构化管理变更、增强、重构、缺陷等*
4. *直接与测试用例匹配*

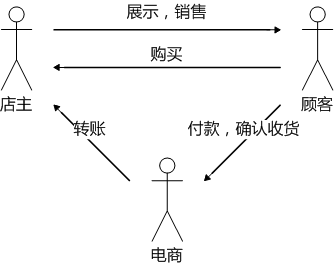
*而为一人年的工作量进行这种需求分析，只需要1小时左右。*

## 第一步：业务愿景——利用“角色-业务图”来支撑产品愿景

愿景（Vision）是用户对产品的核心期望。

培训中使用“角色-业务图”（简称RB图）来表达和落实愿景。

比如在配图中：“购物子系统”核心愿景是“建立一种有保障的网上购物方式”；图中使用“确认收货-转账”的第三方监管业务实现。这样软件开发人员就能得到确切的技术方案，而不是面对描述非常虚的愿景；而技术方案实现后，又能支撑愿景。



有了愿景，产品就不会简单停留在“能用”的状态，而是要积极增加可以实现愿景的功能。

现场演练与指导：建立角色业务图（20分钟）

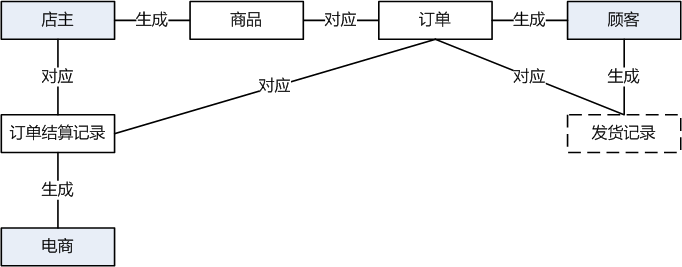
案例分享：RB图详细规则与最佳实践

## 第二步：业务数据——利用“实体-关系图”发掘业务数据

此内容将客户愿景转化为“对某些的业务数据的操作”，从而逐渐进入开发人员可理解的范畴；同时业务数据还是早期功能点估算的核心元素。

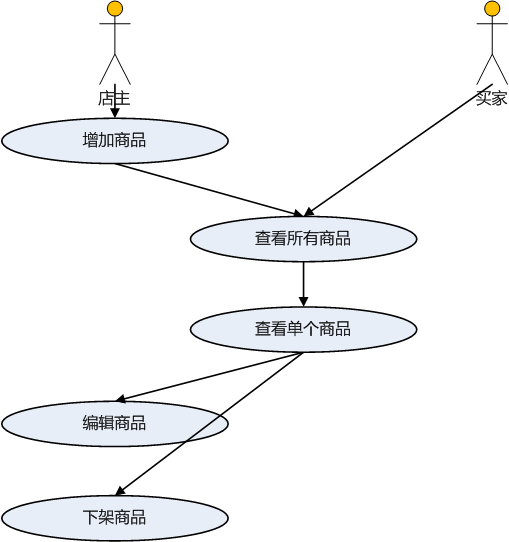
具体分析工具是实体-关系图（简称ER图），而业务数据对应其中的实体（图中方框）。

实体-关系图（教学过程中进行了简化）中分析了实体及其依赖关系，通过适当定义，不但可以保障不会遗漏实体，甚至能直接协助进行早期估算和部分设计工作。



**重要！在敏捷开发中，我们将业务数据作为史诗故事进行开发。**

比如在配图中，所有实体（5个矩形）均包含一组“增删改查”或类似的操作（就是第三步中的用户故事），由此可知此图包含１６５人天左右的工作量／３张数据库主表和２张关系表／５组增删改查操作页面。



现场演练与指导：建立实体关系图（30分钟）

案例分享：ER图详细规则与最佳实践

## 第三步：业务操作——利用“用例-流程图”分析业务操作

借助精益需求建模方法（“用例-流程图”，一种由User Case和状态图结合演进产生的新图形，简称UCF图），找到一个最小的、完备的业务操作集合，作为一次交付所能发布的最新功能集合。在精益开发中，这个集合称之为MVP， Minimum Viable Product最小可用产品。

用例-流程图的“一致性”非常好，即两个不同的分析人员针对同一需求的分析结果，无论用例的数量、名称、乃至排列顺序都惊人地相似。

**重要！在敏捷开发中，我们将业务操作作为用户故事。**

右图是QUML中的“增查查改删”模板中，通过将需求分解为增加-查看所有-查看单个-修改-删除五层，并将不同角色执行的操作放在其正下方（共有操作放在中间），需求分析人员可以迅速而无遗漏地获得所有用户故事。

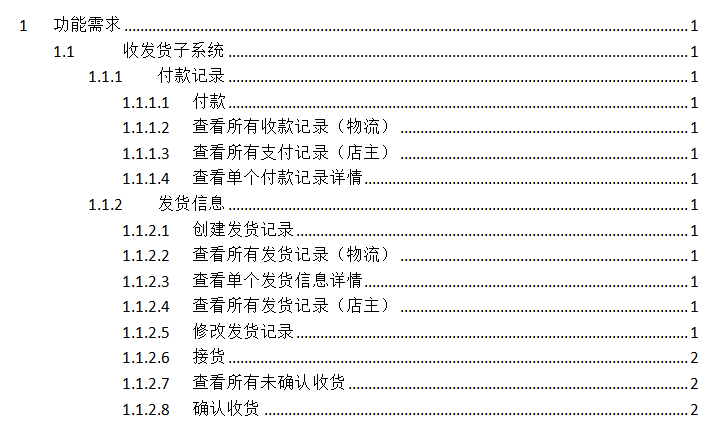
同时，图中由业务逻辑连接的各个业务操作（即椭圆形区域）形成一个MVP，多一个操作则是多余的，少一个则不能完整交付。这对于每个迭代能持续交付至关重要。

现场演练与指导：建立用例流程图（60分钟）

案例分享：UCF图详细规则与最佳实践

## 第四步：需求树——建立结构化的需求

传统用户故事组织方法均呈现“列表结构”，在用户故事数量庞大时（注：每人年大约能完成用户故事50个，外加子故事50～200个），很难看到整个需求的全貌。



培训中，会借助业务愿景-业务数据-业务操作的层次，对需求条目进行结构化表达，形成一棵有层次的需求树。

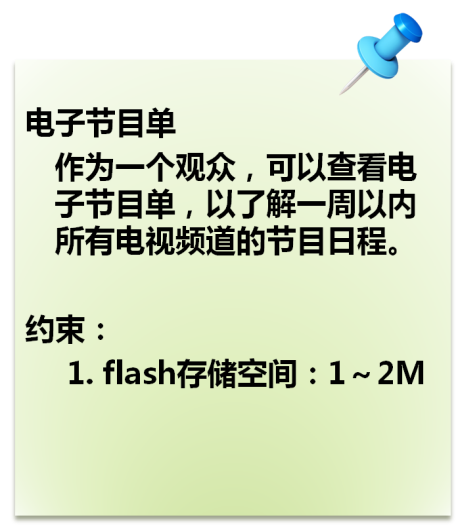
如图，看似是一个很普通的“增删改查表”，但图中的第二至四级目录实际上来自于之前的业务愿景-业务数据-业务操作。这样就很容易从之前的图形化需求形成树形的需求树，其不同层次对应不同尺度的用户故事。

*注：很多业界的敏捷开发工具如Jira都引入了层次化用户故事，但均没有提供层次定义和可操作的分解方法。本培训采用Word作为演示工具，也可对应到具体工具中。*

**×××××××××××××××××××××××××第二天××××××××××××××××××××××××××××××**

## 第五步：用户故事——面向用户价值的需求描述方式

很多软件虽然交付了功能，却不是客户想要的。比如，微博这类的大型系统的管理员，是否会有一个“查看所有用户”这样的功能来管理几亿个用户？如果没有，他怎么知道有哪些用户？如果有，如何避免海量用户造成的信息爆炸？



敏捷开发引入了一种面向客户价值而非产品功能的需求描述方式，将功能放在具体的使用环境中讨论，从而能为客户制作出符合其价值的产品。

现场演练与指导：编写自己的用户故事（30分钟）

案例分享：文字游戏还是价值挖掘挖掘

## 第六步：用户建模——购买决策者/主要使用者

“今年过节不收礼，收礼就收脑白金”。尽管多数收礼者（主要使用者）并不知道脑白金到底包含何种成分，服用后到底有哪些好处，但是确有无数的送礼者（购买决策者）选择购买。



本内容介绍如何区分购买决策者和主要使用者，并面向核心用户编写用户故事。

现场演练与指导：建立自己的用户模型（30分钟）

案例分享：一款年收入12亿元的网络游戏对“所有用户”的理解

# 第二阶段：版本规划与迭代计划

***本阶段以第一阶段生成的各层次用户故事为输入，进行宏观的版本规划和微观的迭代计划。***

*传统敏捷开发缺少版本规划的具体实施方法，“按客户价值优先级进行排序”听起来有道理但却难以实施。尤其是在初期无法获得全部用户故事的情况下，优先级排序非常困难。*

*本培训中的方法可以：*

1. *在开发的初期即可提供颗粒度可控的高层需求（史诗故事）进行排序；*
2. *产品经理根据业界统计数据即可进行版本规划；*
3. *在版本规划的同时自动完成工作量规划，从而准确安排迭代的数量；*

*在每个迭代的计划会上通过“敏捷扑克估算”，借助集体智慧解决个体问题：*

1. *迅速找到最快的解决办法；*
2. *发现高手与新手的差距，并通过讨论弥补差距；*
3. *以10分钟代价提前发现上千行代码的浪费；*

## 第一步：版本规划——项目早期的量化分层规划方法

版本规划涉及到立项时的战略性规划、迭代间的发布规划、随时可能发生的产品升级规划等不同层次。培训中会建立三级规划方法与之对应，分别是业务愿景规划、业务数据规划和业务操作规划。

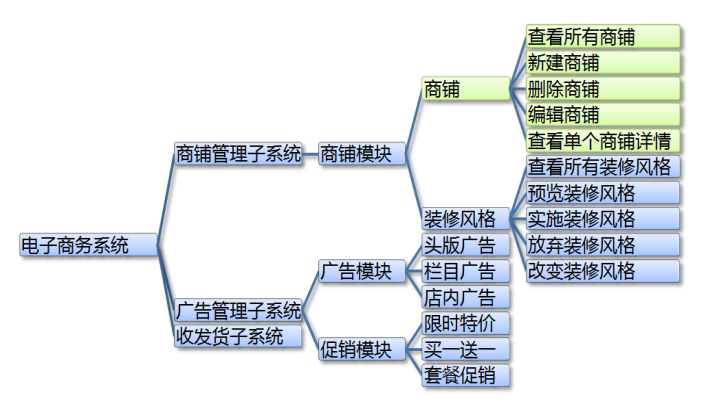


由于业务数据的定义兼容FPA（功能点分析）中ILF（内部逻辑文件）的定义，因此每个业务数据无需知道细节即可按业界数据2人月计算（精确数值为35人天）。

配图中展示了一个电商网站不同阶段规划的情况。左侧业务愿景每个对应4～20人月规模的需求；中间业务数据级别每个对应2人月规模的需求；右上角黑体字则是业务操作，每个对应4～5人天规模。

## 第二步：迭代规划

若一个版本需要在三个迭代后才能完成，那么每个迭代应该完成哪些功能？



本培训中引入了精益创业（Lean Start-up）中MVP（最小可用产品）的概念，介绍如何聚焦于最少的工作量完成一个可以供用户使用并提交反馈的产品。

在完成迭代规划后，产品经理就得到了一个意向性的迭代列表，以及每个迭代中的需求分布情况。接下来在每个迭代开始第一天，需要召开计划会议对详细需求进行讲解和估算。

## 第三步：迭代计划会

本阶段通过讲解计划会的完整过程及背后的思想，并通过实际练习，对之前需求分析阶段获得的用户故事进行估算。



详细内容包括：

**猪与鸡的故事——敏捷计划会背后的分权思想**

如何通过放权提升开发人员个体的积极性。

**产品经理如何讲解故事**

如何从大到小、从整体到局部、从背景到功能地分层讲解用户故事；如何在客户环境中理解需求。

**开发人员扑克估算**

如何让团队成员说出真实的估算值；如何让高手在估算时就能帮助新手；如何通过估算来澄清需求。

讲师在从事开发时，曾将已经完成的、多达4000行代码压缩为55行代码。在实施敏捷估算后，在计划会即可避免这种浪费，而无需等待编码实际结束后才发现。

小游戏：世界□□□□（保密内容）有多高？（演示估算扑克的使用方法，10分钟）

现场演练：我的故事要多少工作量？（使用客户内部开发需求，15分钟）

**×××××××××××××××××××××××××第三天××××××××××××××××××××××××××××××**

# 第三阶段：敏捷日常工作

***本阶段的核心内容是如何在顺利在迭代期内跟踪项目进展。***

*此阶段培训将涉及到传统的每日立会燃尽图、故事板等工具，但同时也会介绍多家公司根据自身情况对敏捷开发活动的改良；*

## 日常活动第一步：每日立会与敏捷生态系统



无数敏捷团队因为每日立会简单易行而将其作为第一个推广的敏捷活动；无数敏捷团队也在无奈中亲自见证了这个简单易行的会议最后如何变得死气沉沉，不得不依靠“迟到罚款箱”来维持。

培训中将介绍每日立会——乃至所有敏捷开发活动——背后的运行原理：**敏捷生态系统**。看似无关的敏捷实践与团队模型之间其实存在着隐形的联系。

## 日常活动第二步：看板

看板是一个通过将所有迭代内工作分为“待开发”“开发中”“开发完毕”三个状态来进行任务可视化管理的工具。

不过，看似简单的看板和燃尽图也有很多背后的故事。

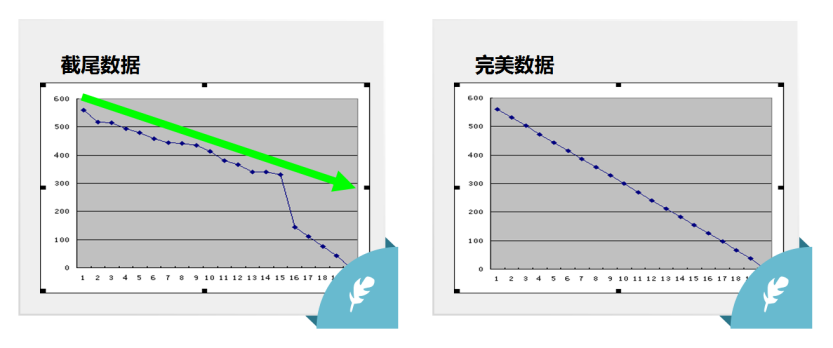


本阶段培训后，学员会发现除了简单地“透明化”之外，看板还能用来确保产品按时发布、防止任务堆积、风险前移等。

启发式演练：如何按需改进看板？

## 日常活动第三步：燃尽图

燃尽图不仅是项目进度的仪表盘，还是分析项目问题的利器。



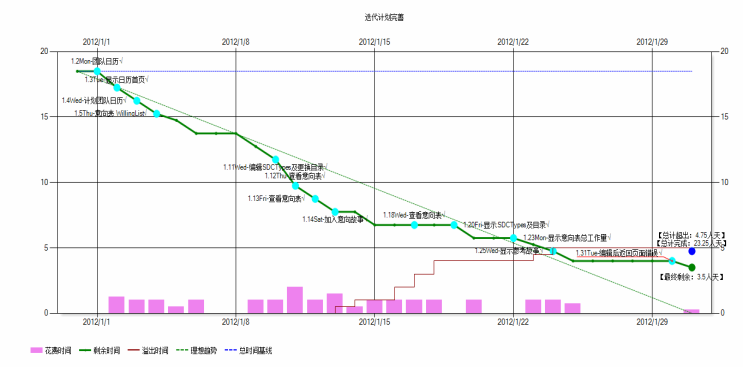
在这个环节，学员们需要动用十八般武器——从第一天到最后一天培训的所有内容——才能分析并驯服失控的燃尽图。

现场演练：分析失控的燃尽图

## 高级敏捷实践

介绍实际企业在推广敏捷时所作的改进。包括：

* NEC的预估会与预评审会制度



* 金山的渐进式评审与跟进人制度
* DSDM方法中的MoSCoW方法
* 当前流行的看板开发
* 面向MVP与持续交付的的可变周期开发

## 场景演练：迭代计划+每日立会+任务分配策略实践

此1小时的练习环节模拟一个4轮的迭代，演示：

* 迭代计划，优先级排序：“必须完成/尽可能完成的工作”
* 每日立会，团队协作，燃尽图：项目进展控制
* 变更管理：在发生意外时如何选择最佳结果

注：此演练仅限于人数低于40人、有充足活动空间的公开课或企业内训。

# 第四阶段：自组织团队建设

***本阶段的核心内容是如何把一个普通团队，打造为跨职能、自组织的敏捷团队，并在建设团队的同时培养个人。***

*很多团队在实施敏捷时采取了“极端敏捷主义团队”，比如团队个体自行估算和领取任务、任何人不过问其他人的工作（为了充分“放权”）……这种听起来很“敏捷”的团队却存在以下问题：*

1. *个体能力差异很大，完成任务的工期、质量可相差10倍以上；*
2. *新手得不到指导，长期无法提高（与之前各自为战的游击队没有区别）*

*此阶段培训还将涉及以下内容：*

1. *大型敏捷开发团队的结构；*
2. *敏捷开发绩效管理——从团队考核到个人考核；*

## 团队建设第一步：自组织团队原理与同行压力

团队认为10个月才能完成的项目，被领导拦腰砍成5个月。领导刚刚离开会议室，队员们哈哈大笑：原来他们早就知道进度会被腰斩，所以提前多估了一倍！



为了提高生产率，公司加强了绩效管理：凡是延期交付的项目全部会被扣奖金！于是，所有团队都为自己的项目预留了100%的额外时间……结果是：一年后，公司除了生产率下降一半之外，什么都没有得到。

领导压力似乎总是失效，那么到底是什么在驱动敏捷团队呢？

本内容会介绍神秘的“同行压力”是如何在计划会、每日立会上驱动团队努力前行的。

## 团队建设第二步：项目经理转型



敏捷开发来了，项目经理该怎么办？

很多公司尝试废弃项目经理，启用Scrum Master，这反而令原来的项目经理成为推广敏捷的阻力。

培训中会介绍项目经理应该如何升级为PM2.0，从而成为符合敏捷开发原则的项目领导者。

## 团队建设第三步：1-3-9团队——大型团队的敏捷方案

当团队超过10人时，众多敏捷开发中未曾想到的问题就浮出了水面。

越来越难以实现众多人员之间的“跨职能”，甚至在纯软环境中分工也越来越难以打破；高手总是认为自己很忙，因而提供帮助的应该是别人；计划会上越来越多的人开始弃权，因为只有个别人感觉自己是潜在的任务执行者……

1-3-9团队创造了一种大团队分组的方法，即通过产品需求树将团队分解为多个Feature Team，独立运行敏捷开发，从而降低了团队的规模。

## 团队建设第四步：松结对编程

项目中的顶尖高手应该做什么？各自为战？安排攻坚？

在敏捷开发“团队协作”的语境中如何使用高手？如果他们为新手提供帮助，如何保证其自身的生产力不会降低？

“松结对编程”提供了一种有别于两个人同时编程的“结对编程”。具体实践包括：高手与新手一起估算任务，在发现由能力造成的工作量分歧后，约定某个时间点高手协助新手，以极短的时间解决新手遇到的困难；每天执行代码审查，确保每一行代码的质量。

## 团队建设第五步：代码审查与编码规范

如何才能做到“只要高手能避免的缺陷，其他人也一样能避免”？

在“松结对编程”中一个重要工作是代码审查，目的是令高手每天都有机会协助新手改善编码质量。高手同时将各种避免缺陷的方法总结为编码规范。这种“避免缺陷”的规范远比“编码整洁”的规范更受欢迎。

本课程的讲师为超过1000人的大型研发企业编写过编码规范，并总结出7条代码审查最佳实践。

# 附：敏捷设计与开发工程实践

注：本内容仅出现在3天课程、敏捷咨询咨询、敏捷CMMI咨询等场景中。

***本阶段旨在依据之前的需求模型，通过极轻量级的设计，即转化为代码结构，从而完成从用户愿景到代码、测试的完整过程。***

*此环节需要结合asp.net mvc或java struts/spring mvc框架。*

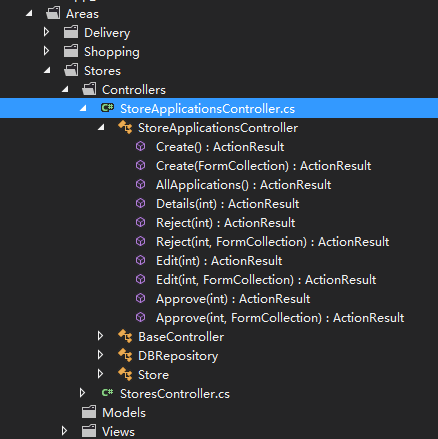
*从实现功能性需求的角度看，设计有两个主要目的：1. 作为从需求到代码的桥梁；2. 以更少的代码完成更多的需求。*

*然而传统设计过程缺少一个步骤化的、一致性的设计方法。不设计，新手无法完成高质量的代码；设计，在代码完成之后几乎没有人再去看设计文档。*

*此章介绍如何使用QUML从需求文档直接产生代码。第一，从QUML已有的RB图、ER图、UCL图，可以迅速得到MVC架构中的Area、Controller和Action，无需任何额外的设计环节；第二，配合QUML中的VCMD代码分层原理，一个Action中的代码会自动被分解到View、Controller、Model、Data四个层次中，即使是初学者也可以掌握。*

## 第一步：从RB/ER/UCL图到Area/Controller/Action（1小时）

MVC架构除了能更好地实现代码分层外，Controller/Action结构还能很好地与树状结构的需求进行分层对应。



在同时应用QUML和MVC时有如下对应关系：

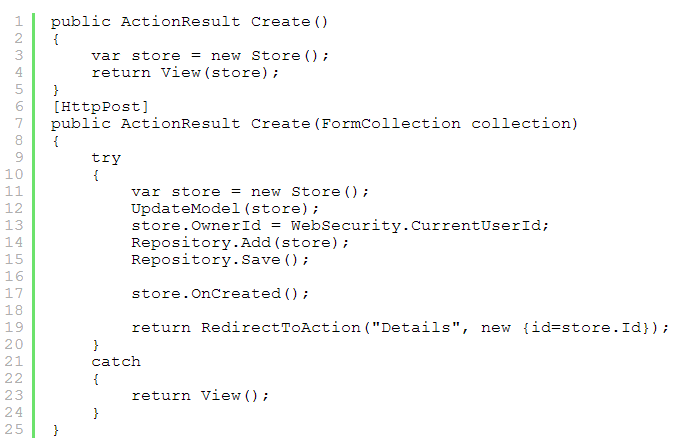
* 子系统=Area
* 业务数据=Controller
* 业务操作=Action

经过这种对应关系之后，只要拿到前文提到的需求树，MVC中的所有Area/Controller/Action均被唯一地确定下来，其数量、名称、顺序几乎不会发生变化。

除了节省大量设计时间之外，这种做法还使得需求与代码建立了意义对应关系，为今后的需求变更贯穿打下了基础。

如右图，从代码结构中即可看出它实现了“商铺管理子系统”（Stores）中的业务数据“建店申请”（StoreApplications）的“创建（Create）”到“批准（Approve）”的所有业务操作。

案例分析：电商系统的MVC代码结构



## 第二步：VCMD代码结构（1小时）

QUML将MVC的代码扩展为四层结构：

* View，仅包含页面布局代码
* Controller，仅包含业务控制代码（做什么）
* Model，仅包含业务逻辑代码（怎么做）
* Data，仅包含数据访问代码

有了这种四层代码规范后，一个方法中的大约200行代码将被分解到至少四个文件中，平均每个仅包含50行代码，从而达到分层设计的目的。

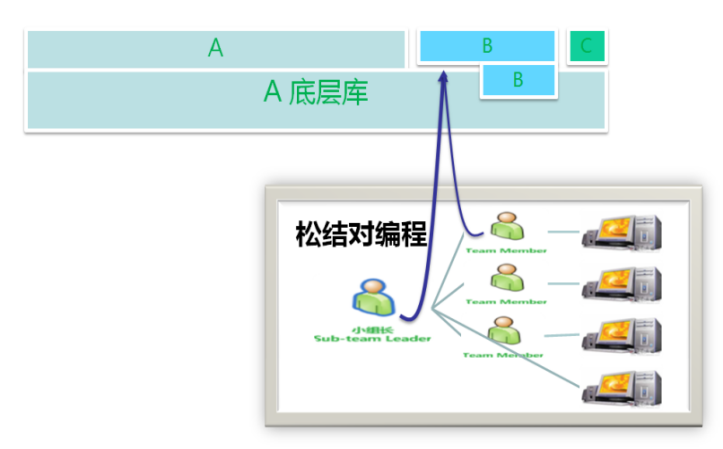
通过3.1中的Area/Controller/Action切分和3.2中的View/Controller/Model/Data切分，不需要任何文档，代码就会被自动切分到12处，多数时候不需要任何设计活动（或仅需要进行一些代码重用和局部设计），即可安排开发。

此方法即使是刚上手的新人也很容易理解和应用，大大降低了设计工作量。

案例分析：电商系统的VCMD代码分层

## 综合应用：L型代码结构（1.5小时）

L型代码结构可以用1/3左右的代码完成相同功能的需求。其中融合了以下工程、管理、团队实践：



* MVC框架
* VCMD分层设计
* 基于需求的团队分工（Feature Team）
* 松结对编程
* 139团队（师徒团队）
* 可复用底层库的建设
* 分层复用与自动化集成
* 代码质量控制

右图中介绍的“L型代码结构”的方法，通过高手/新手的合理搭配，形成业务/底层代码的分离，将最复杂最容易出错的代码由高手封装成可复用库（Model与Data层），而新手只调用可复用库完成业务功能（View与Controller层）。

讲师所在的团队开发的软件产品，通过此种代码结构将代码有效性提高到QSM发布的业界数据的3倍；某银行的大型软件，也已经做到能利用此种结构超过QSM数据的10%。

## 第三步：基于需求树的测试用例/缺陷管理（0.5小时）

传统需求用例的编写往往部分独立于需求文档，很难保证测试覆盖率和测试强度，甚至无法简便地度量两者。



在培训中，通过QUML的需求树，可以有效发现需求；进而依托需求树进而开发出测试用例树，以保证测试用例的完备性。

## 第四步：基于需求树的需求变更管理（0.5小时）

由于缺少一致性的从需求、设计、编码到测试的方法，分析需求变更影响一直是个难题。



但在上述使用3.1～3.4打通需求到测试的全过程后，需求变更可以一致性地、快速地从需求追溯到代码。

图中展示了经过各种变更（增强、重构、技术债务修复、缺陷等）后需求的情况。所有变更均被置于其影响的需求树相应的位置上，变更的实施者很容易由此理解前后文，并评估变更是否可能影响到父、兄弟等需求条目。