敏捷软件开发方法简介
——以极限编程为例

宋扬

北京大学地球与空间科学学院

2003年5月18日

主 要 内 容

- 敏捷方法的含义
- 软件过程的比较
- 极限编程（ eXtreme Programming， XP）简介
  - 准则
  - 法则
  - 活动
  - 实践
  - 讨论和应用实例

“敏捷”的含义

轻巧、机敏、迅捷、灵活、活力、高效……

- 敏捷过程很容易适应变化并迅速做出自我调整，在保证质量的前提下，做到文档、度量适度。
- 适用于各类软件企业

敏捷方法产生的背景

现代软件的

- 复杂性
  - 软件越来越复杂
- 可变性
  - 需求越来越多变
- 一致性
  - 过程越来越规范

软件开发过程敏捷化趋势

据国际著名咨询机构Cutter Consortium对全球200位IS/IT经理所做的调查——

3个占优的重载方法：

- 51% Rational Unified Process
- 27% CMM
- 26% ISO 9000

大约50%的被调查者预计到2003年其50%以上的项目会使用敏捷方法；14%的被调查者认为其所有的项目会使用敏捷方法。

From THE DECISION IS IN: AGILE VERSUS HEAVY METHODOLOGIES, VOL. 2, NO. 19, by Robert Charette,Senior Consultant, Cutter Consortium

敏捷价值观

- “注重个人及互动胜于过程和工具”
- “注重可用的软件胜于详尽的文档”
- “注重客户协作胜于合同谈判”
- “注重响应变化胜于恪守计划”

www.agilemanifesto.org

《敏捷宣言》12条原则

1.最优先的目标是通过尽早地、持续地交付有价值的软件来满足客户。

2.欢迎需求变化，甚至在开发后期。敏捷过程控制、利用变化帮助客户取得竞争优势。

3.频繁交付可用的软件，间隔从两周到两个月，偏爱更短的时间尺度。

4.在整个项目中业务人员和开发人员必须每天在一起工作。

5.以积极主动的员工为核心建立项目，给予他们所需的环境和支持，信任他们能够完成工作。

6.在开发团队内外传递信息最有效率和效果的方法是面对面的交流。

7.可用的软件是进展的主要度量指标。

8.敏捷过程提倡可持续发展。发起人、开发者和用户应始终保持稳定的步调。

9.简化——使必要的工作最小化的艺术——是关键。

10.持续关注技术上的精益求精和良好的设计以增强敏捷性。

11.最好的架构、需求和设计产生于自我组织的团队。

12.团队定期地对运作如何更加有效进行反思，并相应地调整、校正自己的行为。

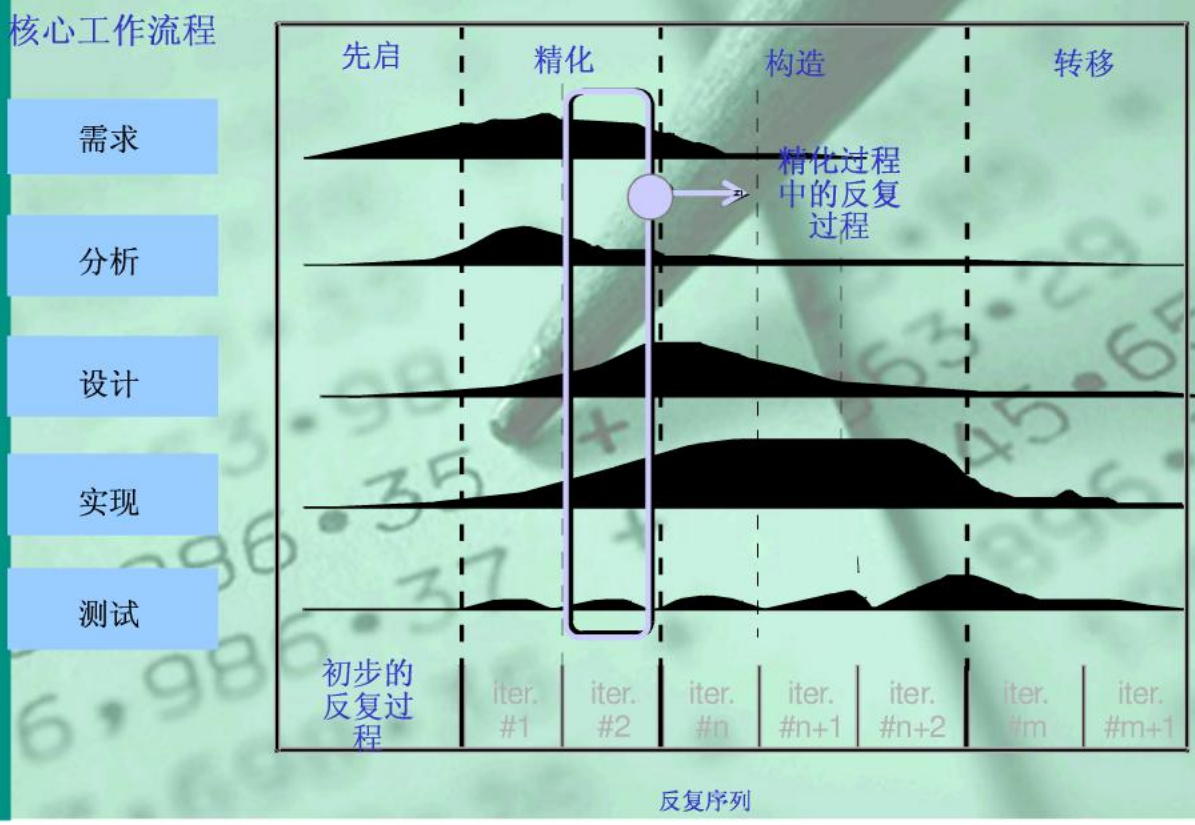
代表方法

- 瑞理统一开发过程：Rational Unified Process
- 敏捷建模：Agile Modeling
- 极限编程：eXtreme Programming
- 自适应软件开发：Adaptive Software Development
- 水晶方法体系:Crystal

etc.

RUP（Rational Unified Process）

核心工作流程



RUP是Rational公司的改进过程的规范，它被设计成一种由用例驱动的、以体系结构为中心的软件开发过程，它以迭代的方式前进，通过执行工作流程递增地产生结果。

它的主要四个阶段是先启、精化、构建和转移，五个核心工作流程为需求、分析、设计、实现和测试。

由于RUP是一种框架，你可以以不同的方式来使用它，如象非常传统的“瀑布” 式开发方式，或敏捷式，如dX。你可以把用得轻捷灵便，也可把它弄成繁文缛节。这取决于你如何在你的环境中对它裁剪运用。

XP到RUP的映射

	先启	精化	构建	转移
需求	用户素材	小型发布	接纳测试	测量
分析	CRC卡片	迭代计划	任务计划、迭代编程	接受背后
设计	系统隐喻	单元测试	重构	持续集成
实现	编码标准	简单设计	集体代码所有权	运行所有测试

- CRC卡片：Class-Responsibility-Collaborator

Agile Modeling

- AM是一种最近才出现的软件思想
- AM是一种轻方法论
- XP实践既给了AM灵感，也是AM的一种具体实现

AM核心原则

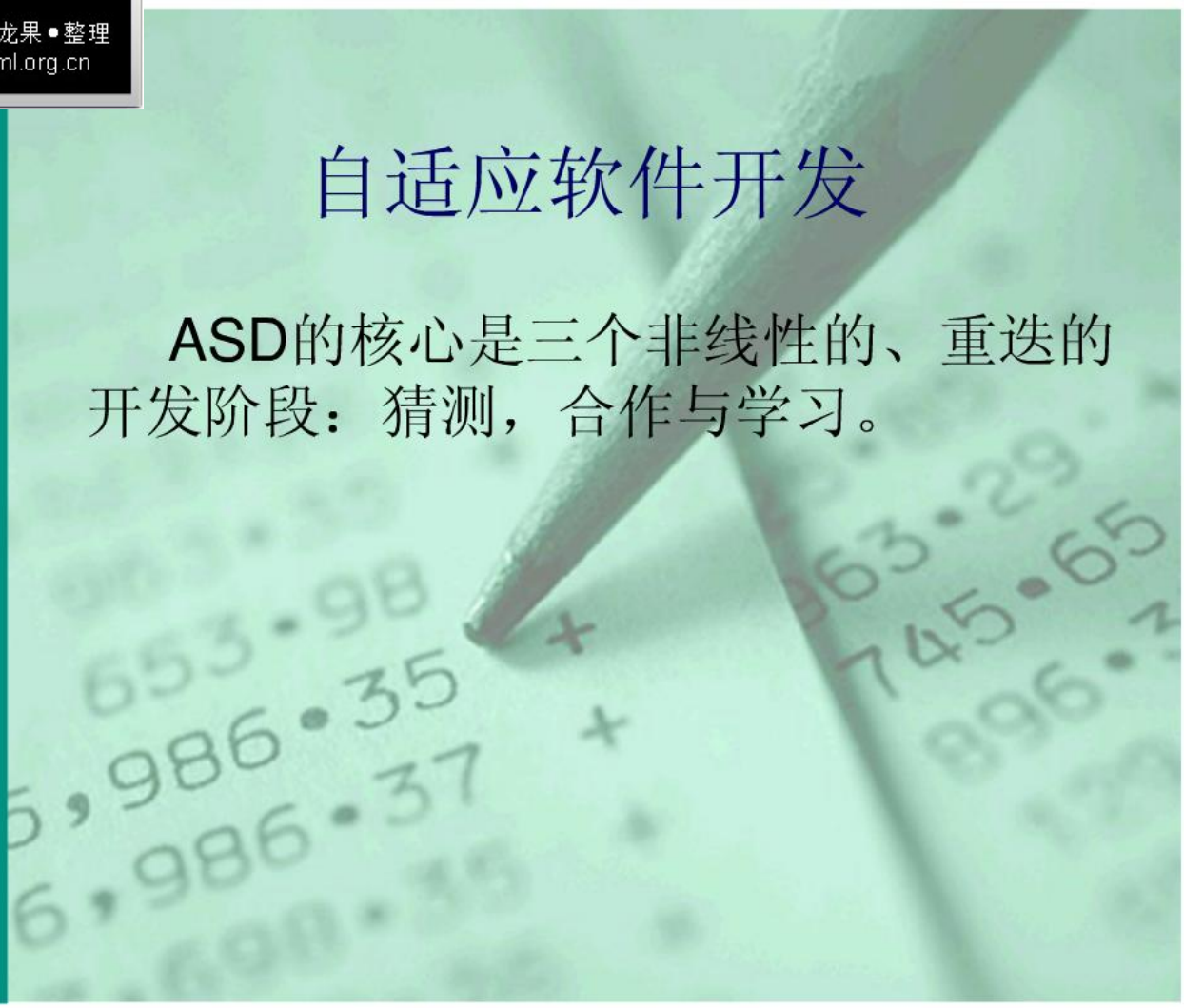
- 主张简单
- 拥抱变化.
- 你的第二个目标是可持续性。简单的说，你在开发的时候，你要能想象到未来。
- 递增的变化
- 令投资人的投资最大化
- 有目的的建模
- 多种模型
- 高质量的工作
- 快速反馈
- 软件是项目的主要目标
- 轻装前进

AM补充原则

- 内容比表示更重要
- 三人行必有我师
- 了解你的模型
- 了解你的工具
- 局部调整
- 开放诚实的沟通

自适应软件开发

ASD的核心是三个非线性的、重迭的开发阶段：猜测，合作与学习。



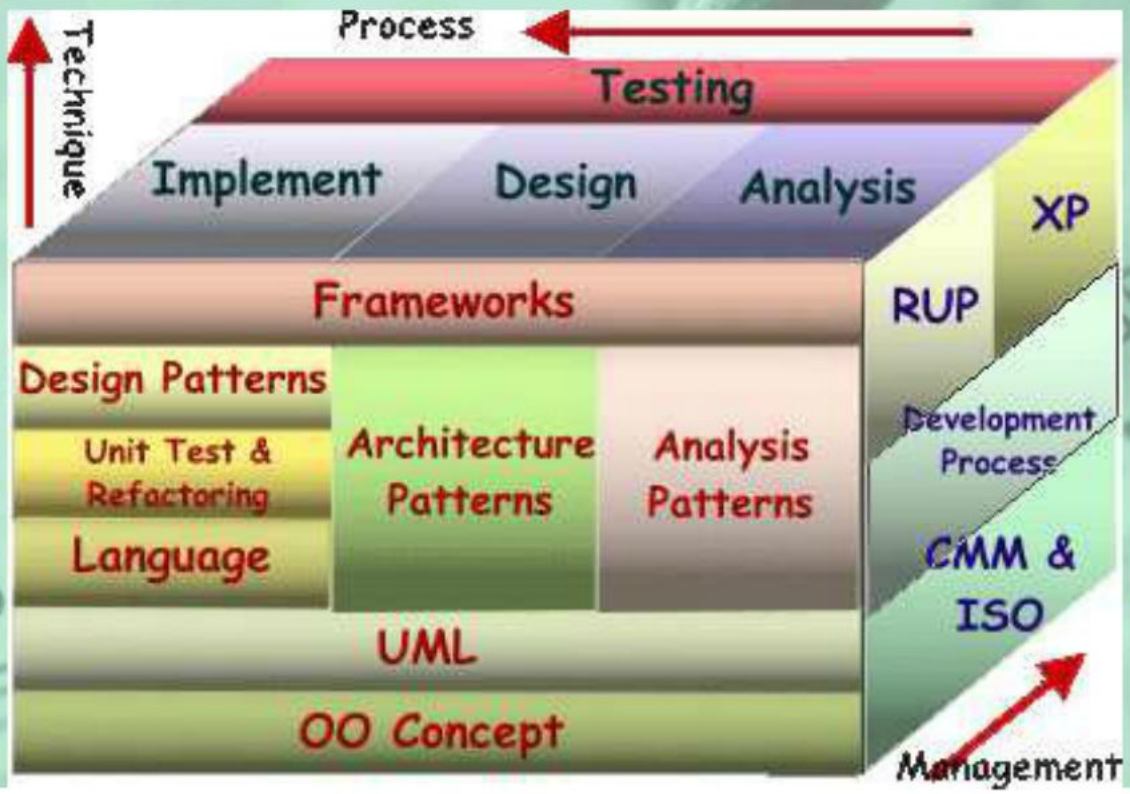
水晶方法体系

水晶方法体系与XP一样，都有以人为中心的理念，但在实践上有所不同。水晶方法体系考虑到人们一般很难严格遵循一个纪律约束很强的过程，因此，与XP的高度纪律性不同，水晶方法体系探索了用最少纪律约束而仍能成功的方法，从而在产出效率与易于运作上达到一种平衡。也就是说，虽然水晶系列不如XP那样的产出效率，但会有更多的人能够接受并遵循它。

ISO9000

- PDCA循环，即由计划（PLAN）、实施（DO）、检查（CHECK）、处理（ACTION）这四个密切相关的阶段所构成的工作方式
- 持续改进

面向对象软件工程概念模型



极限编程（ eXtreme Programming ）
——轻量级敏捷软件开发方法

极限编程（XP）是一种全新而快捷的软件开发方法。XP团队使用现场客户、特殊计划方法和持续测试来提供快速的反馈和全面的交流。这可以帮助团队最大化地发挥他们的价值。

- XP诞生了大概有5年
- XP是以开发符合客户需要的软件为目标而产生的一种方法论
- XP是一种以实践为基础的软件工程过程和思想
- XP认为代码质量的重要程度超出人们一般所认为的程度
- XP特别适合于小型的有责任心的、自觉自励的团队开发需求不确定或者迅速变化的软件

XP准则

——XP软件开发是什么样的

- 沟通
- 简单
- 反馈
- 勇气

有益的潜在补充准则:

- 尊重
- 谦逊

沟通

XP认为项目成员之间的沟通是项目成功的关键，并把沟通看作项目中间协调与合作的主要推动因素。

简单

XP假定未来不能可靠地预测，在现在考虑它从经济上是不明智的，所以不应该过多考虑未来的问题而是应该集中力量解决燃眉之急。

反馈

XP认为系统本身及其代码是报告系统开发进度和状态的可靠依据。系统开发状态的反馈可以作为一种确定系统开发进度和决定系统下一步开发方向的手段。

勇气

代表了XP认为人是软件开发中最重要的一个方面的观点。在一个软件产品的开发中人的参与贯穿其整个生命周期，是人的勇气来排除困境，让团队把局部的最优抛之脑后，达到更重大的目标。

表明了XP对“人让项目取得成功”的基本信任态度。

XP的法则

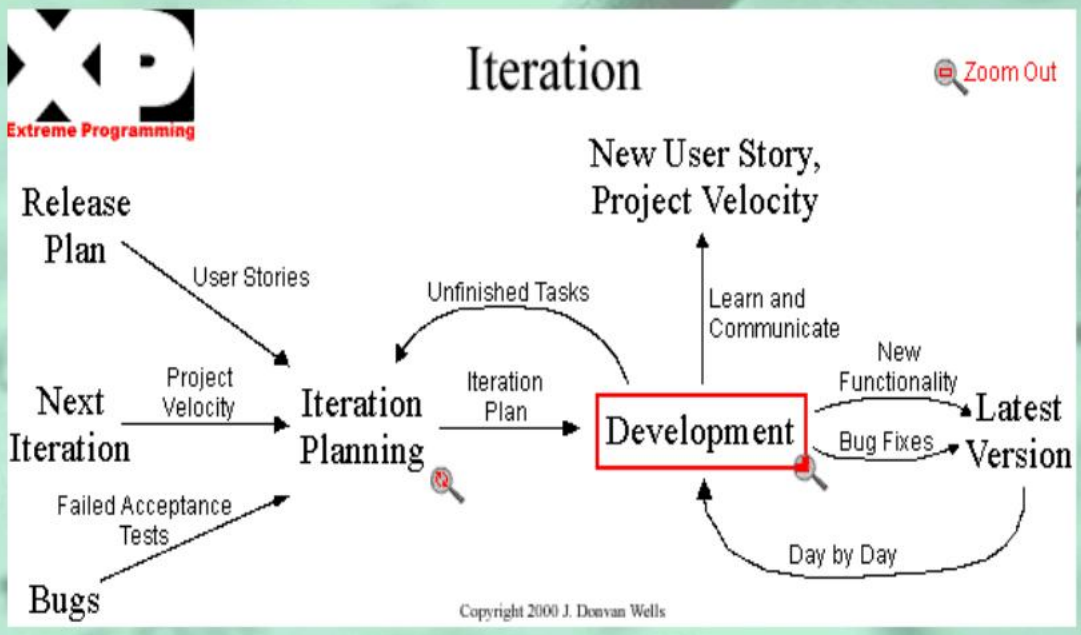
——一项实践在XP环境中成功使用的依据

- 快速反馈
- 假设简单性
- 递增更改
- 提倡更改
- 优质工作

快速反馈

XP提倡尽可能早地、迅速地每天反馈，让编程人员始终把注意力放在最重要的软件功能上，促使系统快速演进。

XP迭代过程



假设简单性

XP试图把注意力集中在能工作的尽可能最简单的实现上；另一方面，可以根据给定的项目资源条件，最优地分配项目资源。

递增更改

XP认为首次更改就尝试重大更改绝对不会成功，提倡以重构概念为基础做小改动，用期望的功能逐步增强系统。

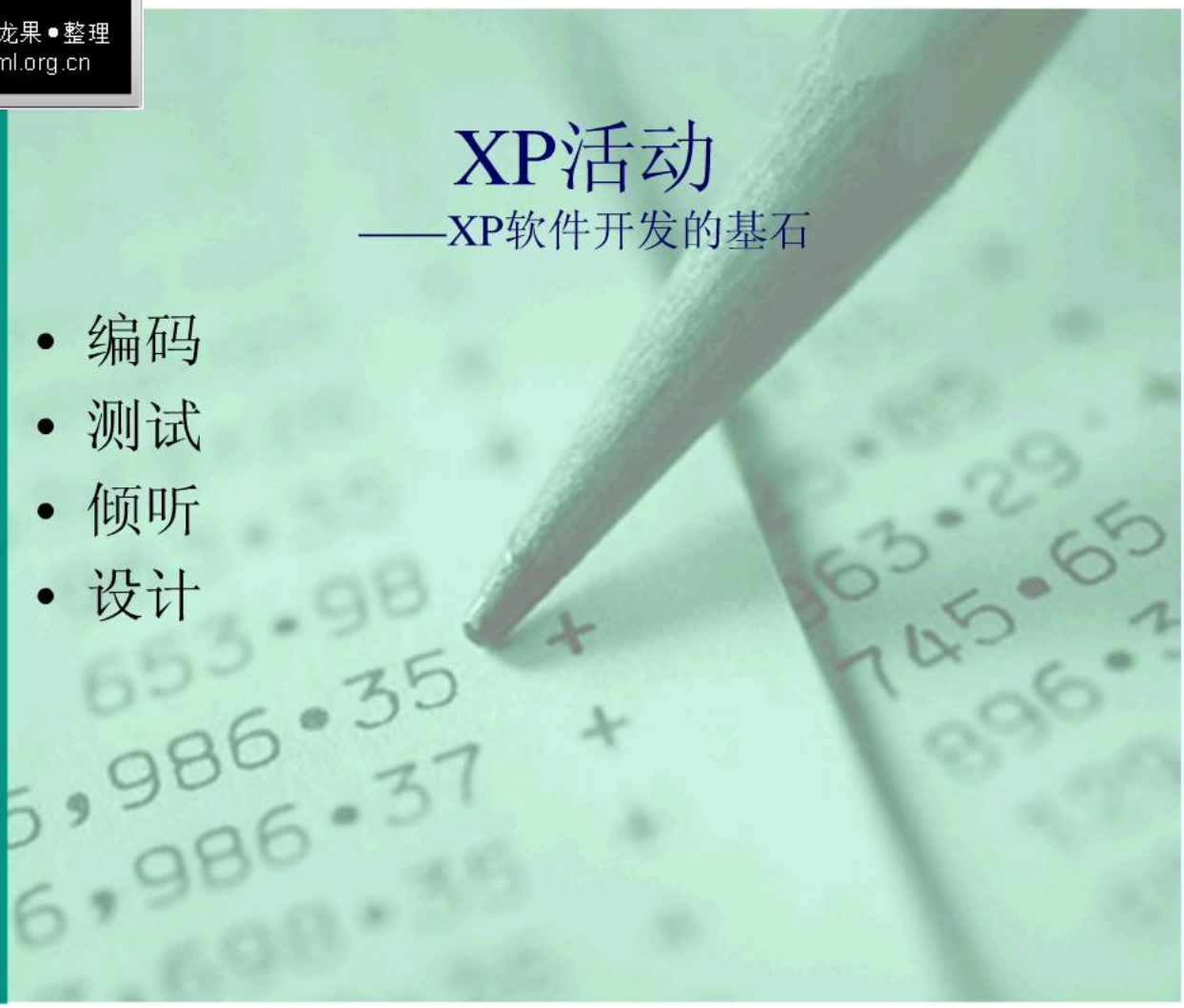
提倡更改

最佳策略是在实际解决最重要的问题的前提下保留最多选项的那一个，在交付最需要的东西上保留选择余地。

优质工作

XP活动

——XP软件开发的基石



- 编码
- 测试
- 倾听
- 设计

编码

作为一种轻量级方法论，XP明确放弃了系统建档和分析以外的任何外在活动。分析保留为一种相当简单，但是在和客户的日常沟通中发生的持续活动。文档则明确不予鼓励，所以编码成为XP最主要的活动。

测试

为了确保编写好的代码能实践工作，XP提倡编写大量测试来检查代码是否正确。

倾听

XP编程人员倾听客户和其他编程人员的需求和意图。

设计

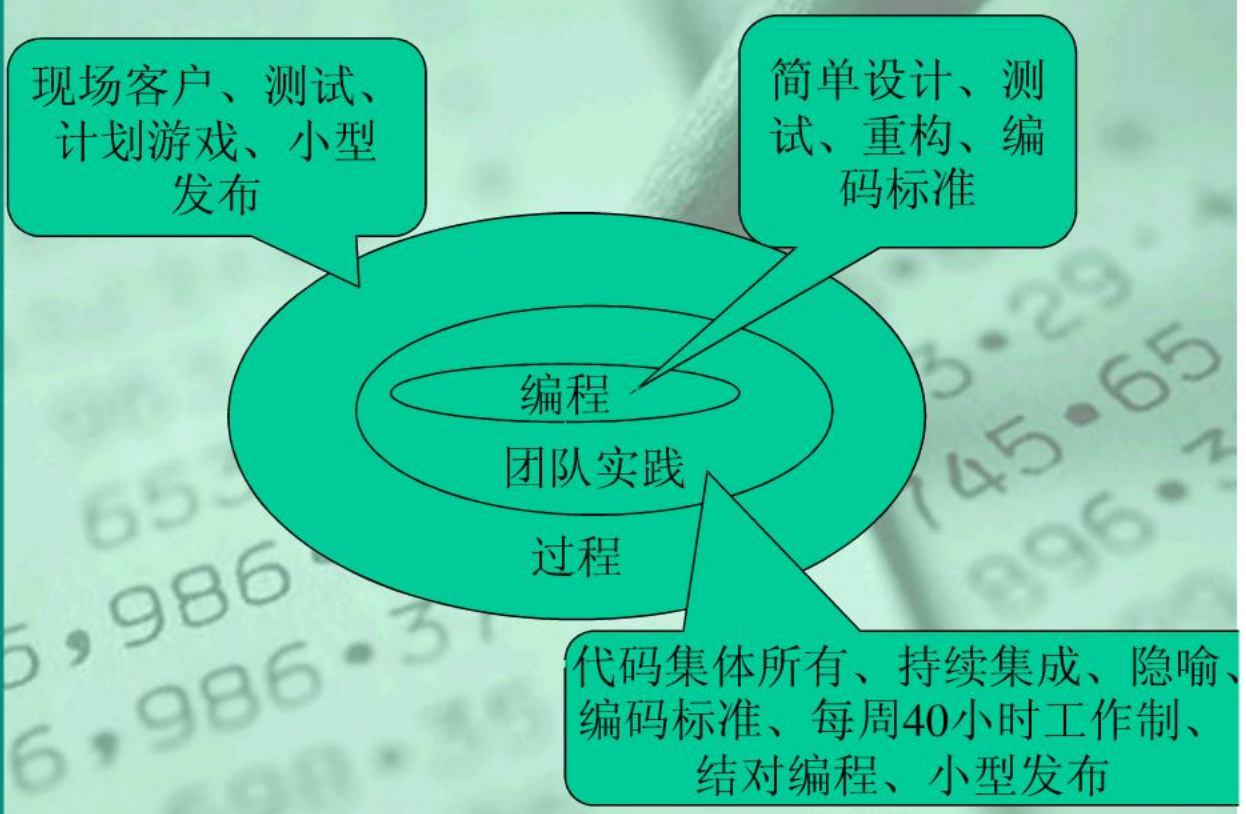
从日常的编码中返回来进行一些一般性设计，小的设计成为XP编程人员日常事务的一部分。

XP的实践

——项目成员用户成功执行XP活动的技术

1.现场客户（On-site Customer）
2.计划游戏（Planning Game）
3.系统隐喻（System Metaphor）
4.简单设计（Simple Design）
5.代码集体所有（Collective Code Ownership）
6.结对编程（Pair Programming）
7.测试驱动（Test-driven）
8.小型发布（Small Releases）
9.重构（Refactoring）
10.持续集成（Continuous integration）
11.每周40小时工作制（40-hour Weeks）
12.代码规范（Coding Standards）

XP层次结构



1 现场客户

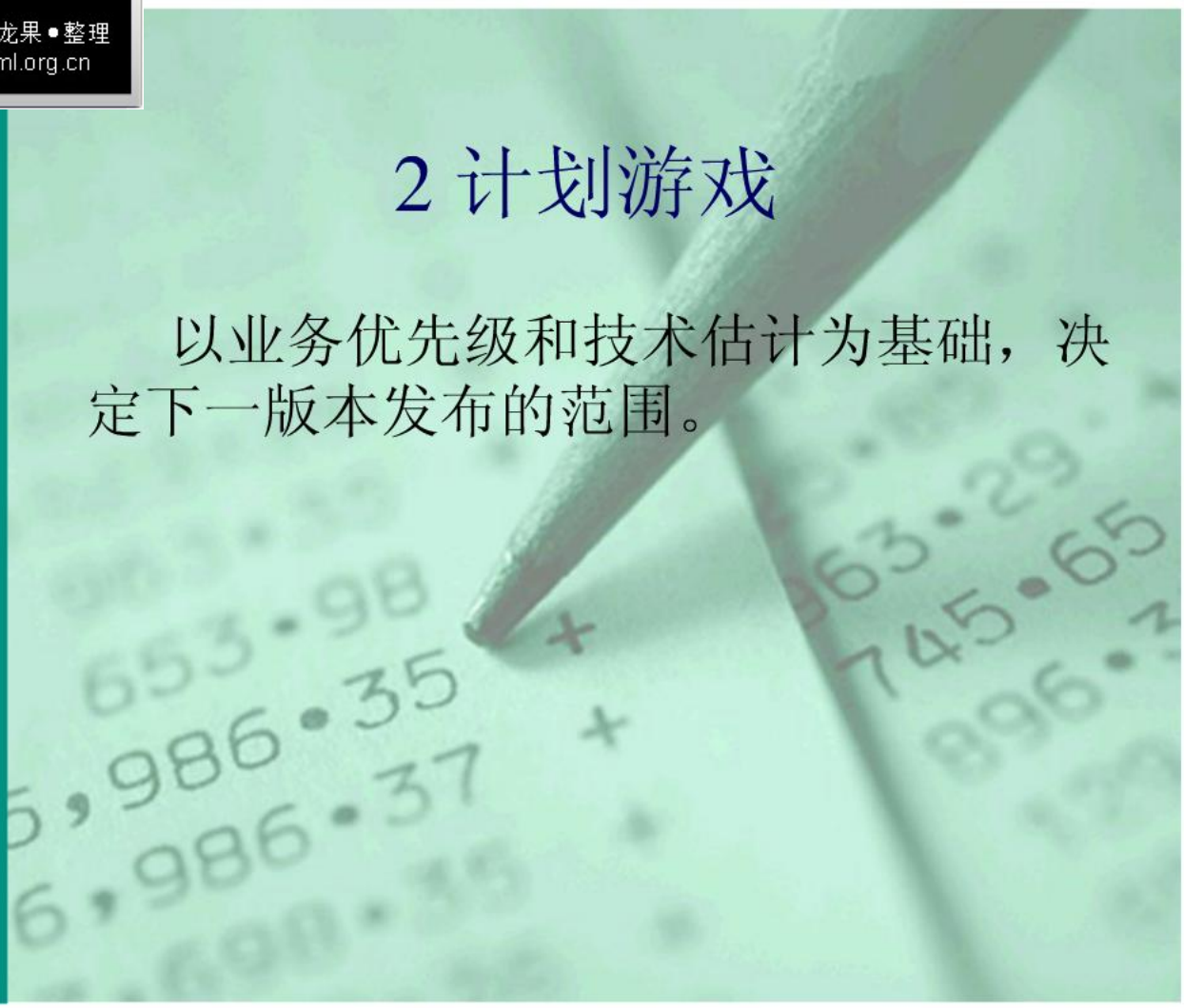
始终在开发团队中有一位客户。

现场客户的工作：

- 回答问题
- 编写验收测试
- 运行验收测试
- 指导迭代
- 接受版本

2 计划游戏

以业务优先级和技术估计为基础，决定下一版本发布的范围。



3 系统隐喻

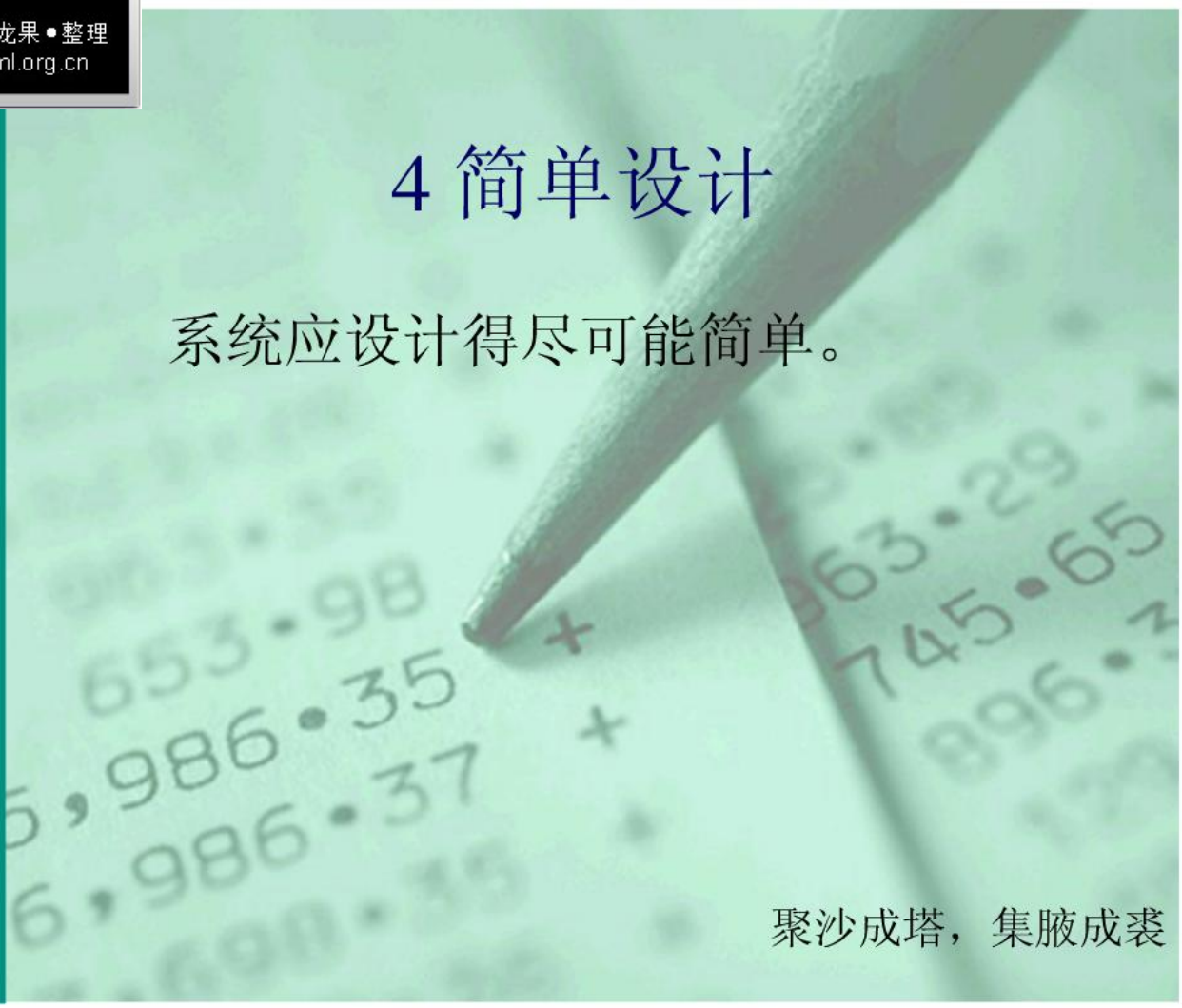
在XP中，隐喻是一种概念框架并提供名称的描述系统，类似于其他方法中的体系结构（或体系结构基准）。

- 共识
- 共享的术语空间

例子：Windows风格的界面、网上购物站点的购物车

4 简单设计

系统应设计得尽可能简单。



聚沙成塔，集腋成裘

5 代码集体所有

整个团队拥有所有代码。任何人都可以更改他们需要更改的部分。没有唯一对代码有所有权的人。

题外话一

编程的乐趣(F P. Brooks)

- 创造的快乐
- 开发对他人有用的东西
- 整体过程的魅力
- 持续学习的快乐
- 来自于易于驾驭的介质上工作

编程的快乐在于它不仅满足了我们内心深处进行创造的渴望，而且还唤醒了每个人内心的情感。

题外话二

编程的苦恼(F P. Brooks)

- 来自追求完美
- 来自由他人设定目标、供给资源、提供信息。
- 陷入琐碎的重复性劳动
- 持续学习的快乐
- 无用功

这，就是编程，一个许多人痛苦挣扎的焦油坑以及一种乐趣和苦恼共存的创造性活动。

6 结对编程

结对编程是让两个人共同设计和开发代码的实践。结对者是全职合作者，轮流执行键入和监视；这提供了持续的设计和代码评审。

不是两个人做一个人的事情。

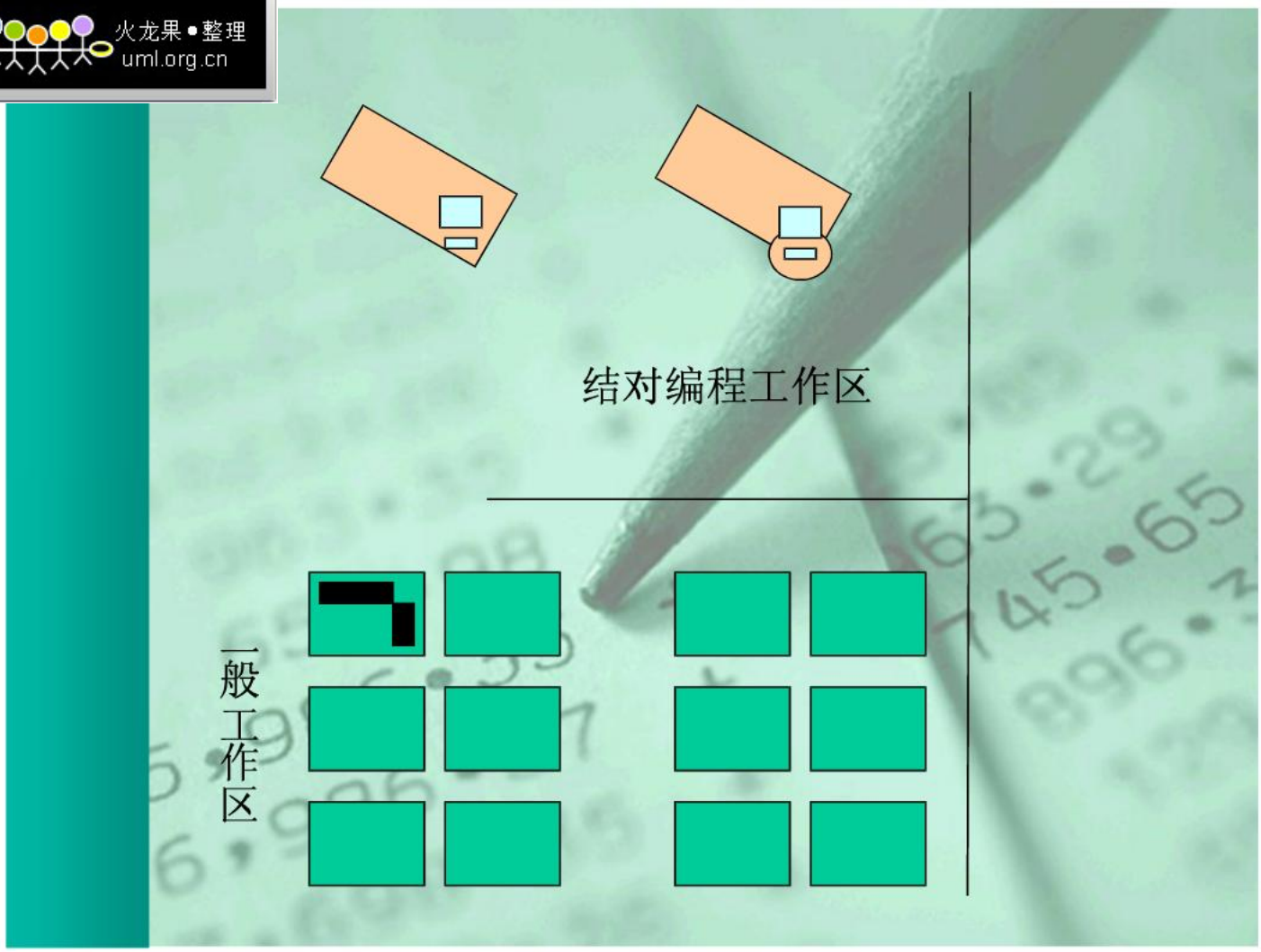
积极影响

- 经济性
- 满意度
- 提高设计质量：分享不同的先验知识、理解和角色
- 持续复查
- 问题解决更快：集思广益和配对接力
- 学习：耳濡目染
- 团队建设和沟通
- 有利于人员和项目管理

卡车问题

一个项目组集体外出，不幸被卡车撞上。有多少人受伤使项目不得不停止？

最坏的情况是一个！



学到的经验

- 程序员和设计人员\协调人结对编程更有效。
- 键盘输入效率！
- 自愿结对编程

我们行业的主要问题实质上更侧重于社会学而不是科学技术。

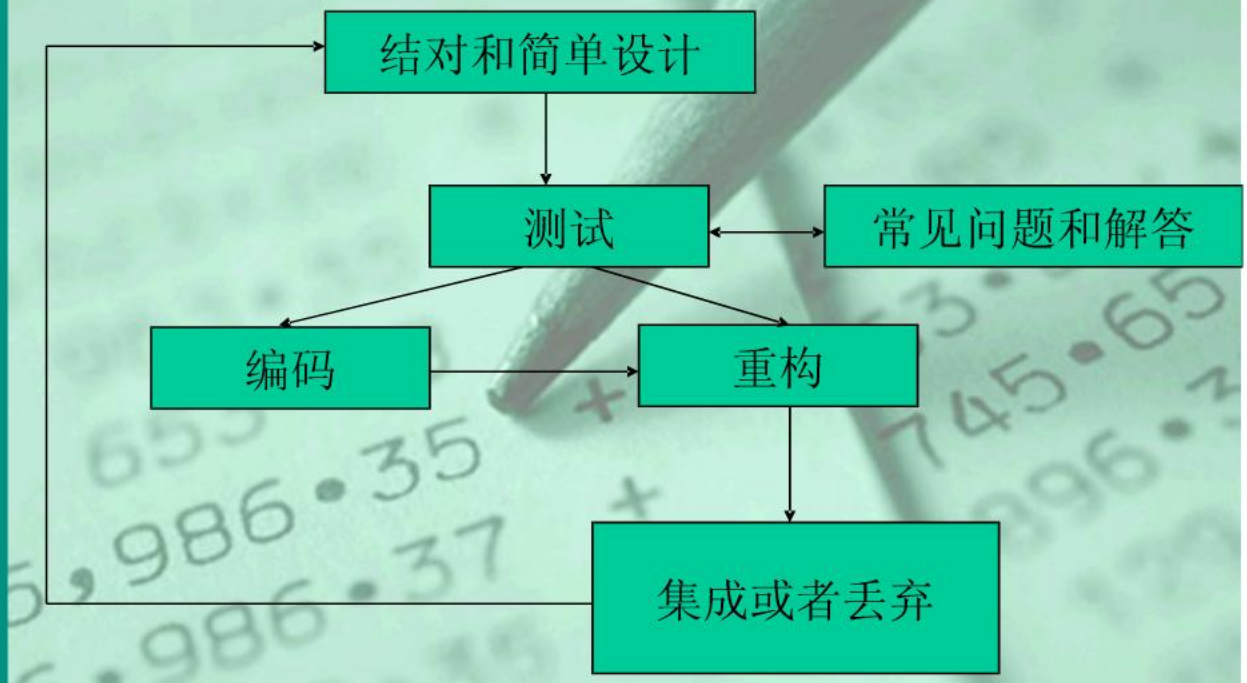
——《人件》

9 重构

重构是XP的一个重要组成部分。所谓重构是指在不改变代码外在行为的前提下对代码做出的修改，以改进代码的内部结构。重构是一种有纪律的、经过训练的、有条不紊的代码整理方法，可以将整理过程中不小心引入错误的可能性降到最低。从本质上说，重构就是在代码写好之后改进它的设计。

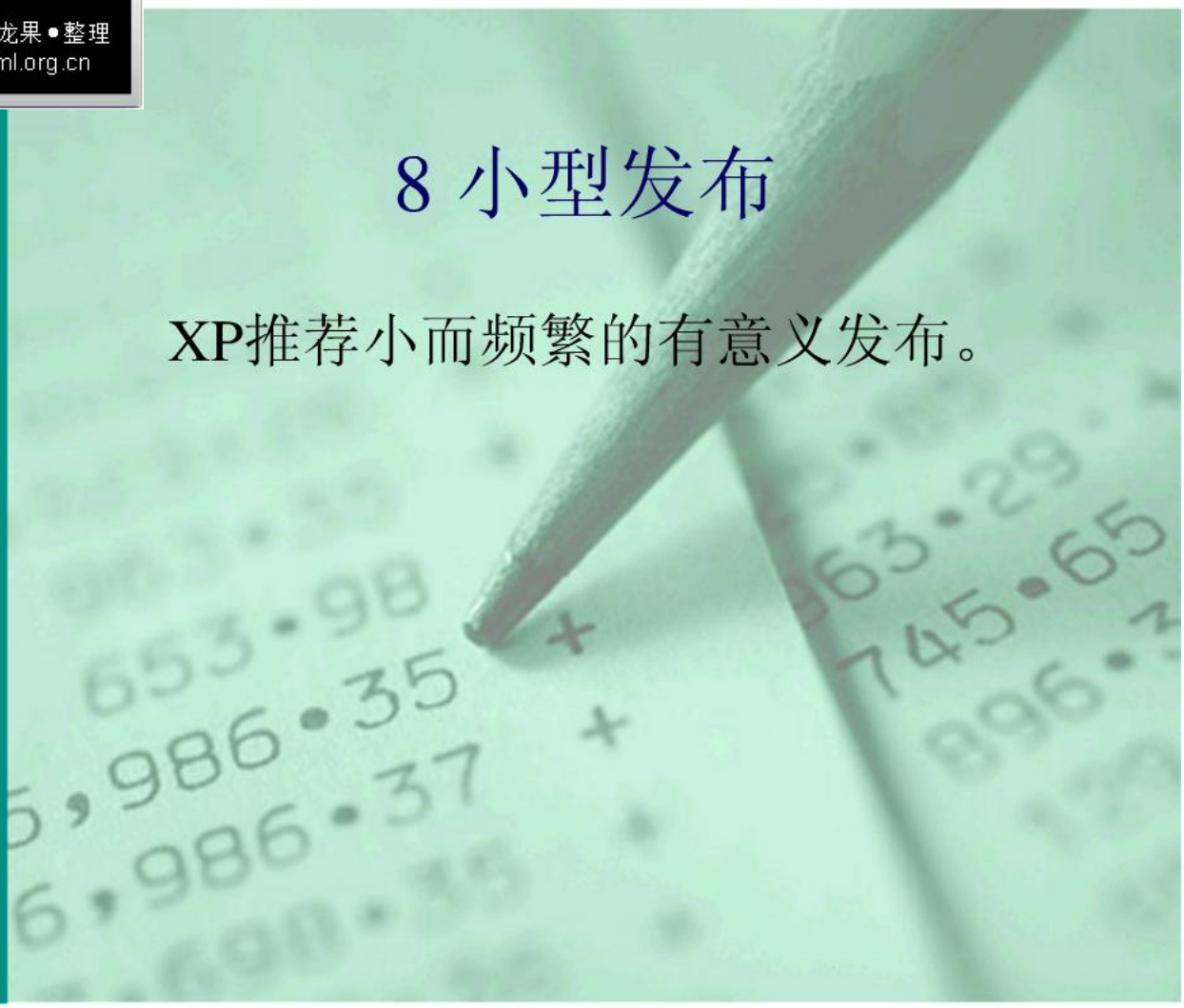
重构的节奏：重新推理、小的更改、重新推理、小的更改、重新推理…

7 测试驱动



8 小型发布

XP推荐小而频繁的有意义发布。



10 持续集成

持续集成的思想是任何时候只有一项任务完成，就集成新代码，构造系统并测试。持续集成是每日构建\每晚构建的一种极限形式，是XP的重要基础。

每日构建\每晚构建是将一个软件项目的所有最新代码取出，从头开始编译、链接，用安装软件包将链接好的程序安装好，运行安装后的软件，使用测试工具对主要功能进行测试，发现错误并报告错误的完整过程。

让开发人员在第一时间了解到软件的错误，并迅速排除错误，是每日构建\每晚构建最重要的目标之一。

每日构建\每晚构建必须出日志和报告，并发布构建结果的有关信息，最好能够使用自动化工具发出电子邮件通知。

每日构建是项目的心跳。如果一个项目的心跳停止了，这个项目就死亡了。

Treat the daily build as the heartbeat of the project.

If there is no heartbeat, the project is dead.

作用

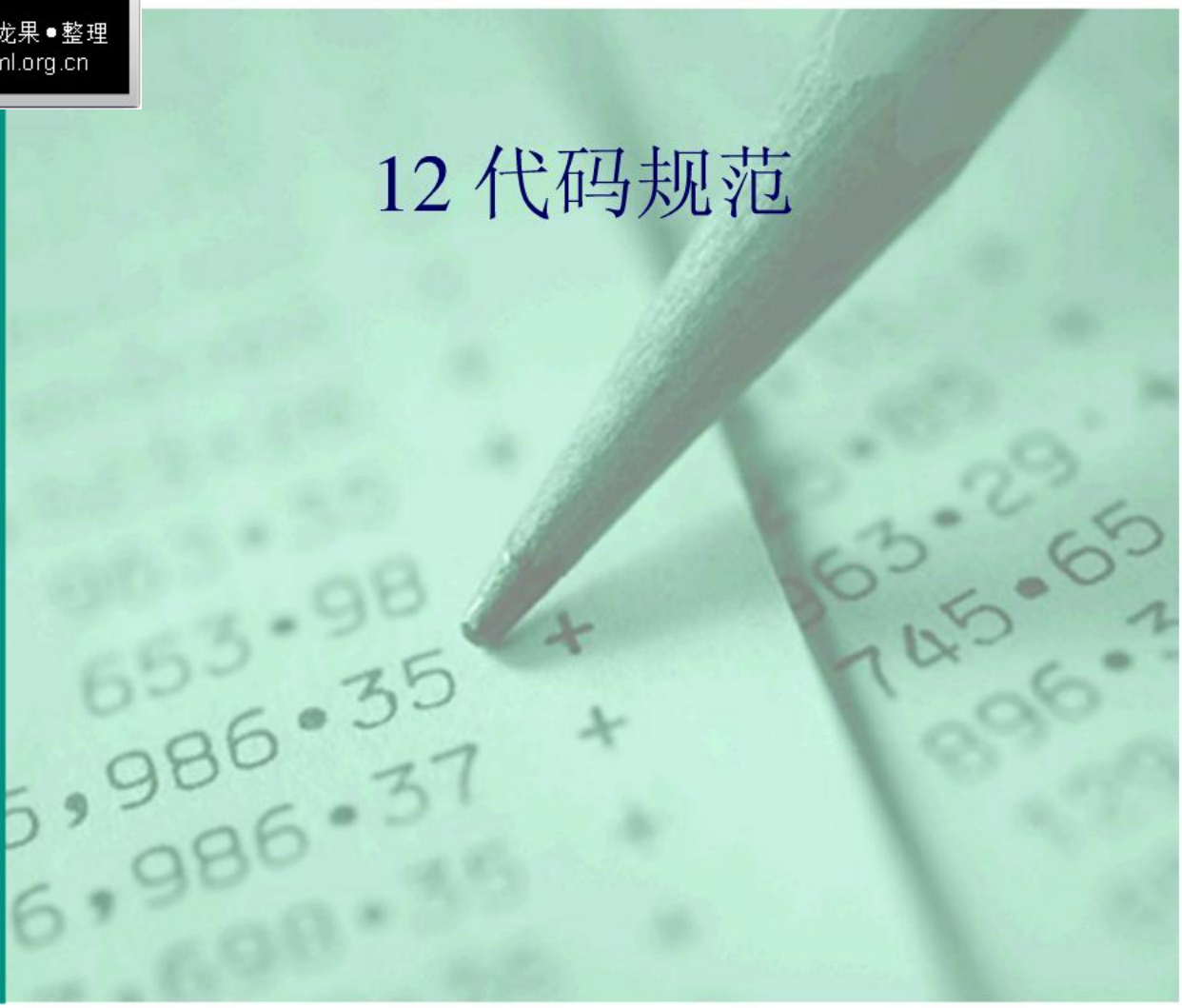
- 降低集成风险
- 加强错误诊断
- 降低不确定性
- 加快开发速度
- 增强团队合作
- 对项目参与者是重要激励

11 每周四十小时工作制

在这里40是一个概数，不是确数。

- 如果能够努力地工作8小时，超过这个时间后就不适于有效地工作了——8小时燃烧
- 再学习
- 你无法改变时间，但是可以改变你的任务。

12 代码规范



XP讨论：

XP和软件设计

XP提倡循序渐进的软件设计方法，以避免在前置设计中花费巨大的精力。

但这不是回到编码加修正（Code and Fix）的开发方式。

不变的只是愿望，变化才是永恒。

如果你聆听代码，好的设计就会出现。

计划设计

初始设计不可能面面具到

受人员变动影响

设计人员和编码人员之间存在协同问题

设计人员和编码人员之间存在技能差异

不能适应变化的需求

柔性设计

缺乏整体性

设计可能不可控

潜在的缺陷可能会引起其它缺陷

修正缺陷的代价可能会呈指数增长

对设计和编码人员要求更高

XP中对柔性设计的支持

- 简单设计
- 测试先行
- 持续集成
- 重构

简单的价值

- 效益
- 轻装上阵

做可能有效的最简单的事情

Do the Simplest Thing That Could Possibly Work

你将不会需要它

You Aren't Going to need it

XP衡量简单的标准

- 通过所有的测试
- 代码体现所有设计意图：Clever Code
- 避免重复
- 类或方法的数量最少

做且仅做一次

Once and Only Once

不要自我重复

Don't Repeat Yourself

模式\反模式和XP

- 模式
- 反模式
- 对初学者
- 重构的目标之一—>模式
- 选择使用模式的时机
- 模式和简单性

UML和XP

最好的UML图也不是产品！

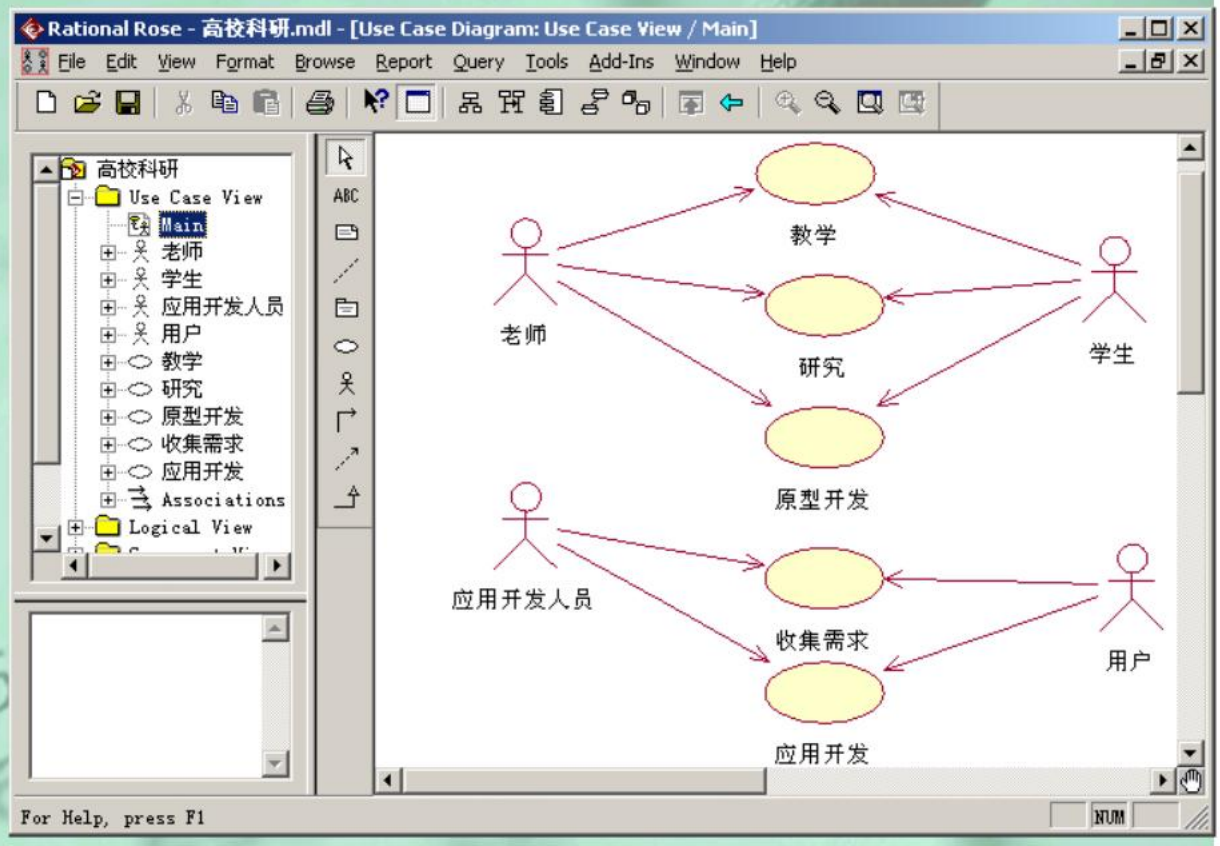
XP应用之一：

XP在高校科研中的应用

高校研究室\组人员的一般构成：

- 一名以上的教授\副教授
- 博士生
- 硕士生
- 附属工作人员
- 临时教学或研究的学生
- 临时合作人员

高校科研用例图（理工类）



高校科研项目中的一些问题

高校软件（原型、项目）往往显示很差的软件工程质量：

- 需求不明确
- 缺乏客户参与
- 人员的持续波动或突变
- 没有一个适用于高校的质量体系
- 总是处于人员培训阶段
- 研发过程不连续，积累困难
- 往往没有激励机制

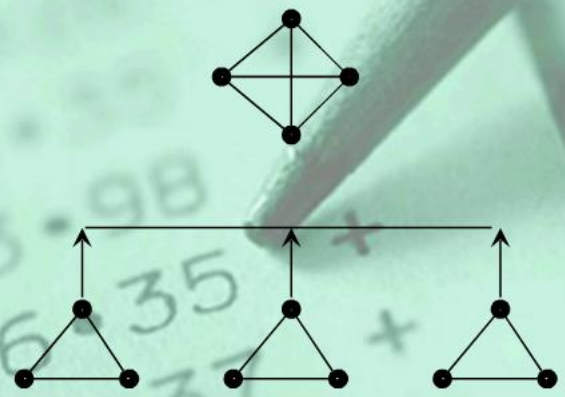
高校中的XP

- 首选结对编程\学习：指定核心；知识流动的不确定性；吸引力随着团队规模的减小而下降。
  - 高年级<—>低年级，不太有效
  - 高年级<—>高年级
  - 教师<—>学生，不太有效
- 持续集成
- 代码集体所有
- 重构
- 设计和文档重要性上升

XP应用之二：

XP在大型项目中的应用

——层次结构的XP



- 用清晰而且微小的接口分割系统
- 每个子项目\子课题的目标是整个项目的一个子目标
- 递增改进
- 有一个集中管理机制
- 充分利用辅助手段进行交流

Q&A

——根据5月27日报告后的问答整理

1 敏捷软件方法适合中国吗？

虽然敏捷软件开发方法体系的形成主要基于国外的软件开发经验，体系的创建人和归纳人也是外国人，但是绝大多数内容国内都可以借鉴，可能某些内容不能照搬，在报告中没有特别区分这些内容。同时敏捷软件开发方法本身还在不断发展和完善，国内也可以结合国内的实际情况对其发展做出贡献。

2 ISO9000体系的文档量比较重，在实际中如何操作？

ISO9000文档数量比较多，国标中主要文档有14个，但是这些文档工作也是穿插在软件开发过程的各个周期，比如代码开发前后都是文档工作量较大的阶段。

3 XP和传统软件开发过程的差异？

XP来自于传统软件开发过程，比如XP的12个实践在传统软件开发过程中都有，但是XP把它们有机地组合了起来，取长补短，最大限度地发挥了它们的作用。另一方面，XP和其他敏捷方法和传统方法有很多不同，比如对人的作用，传统方法只是把人作为一个螺丝钉，没有很好地调动开发者的主观能动性。同时敏捷方法和开源软件有很深的关系，所以在许多地方带有自下而上的色彩，重构就是其中的典型代表。

4 XP有和ISO等其他传统方法一样的具体操作规范吗？

目前敏捷方法还没有具体的操作性强的规范，更多的是一种指导架构和思想，就像AM。

5 采用传统开发流程的软件组织如何引入XP?

这方面XP的领导者如Kent Beck都有很多建议，在XP网站也有很多经验介绍。