Shanghai Jiao Tong University

软件过程与管理

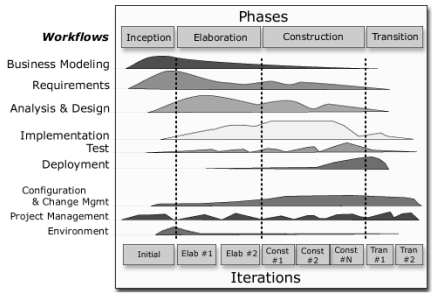
Module: Software Process
软件开发过程

上海交通大学软件工程中心

本节内容

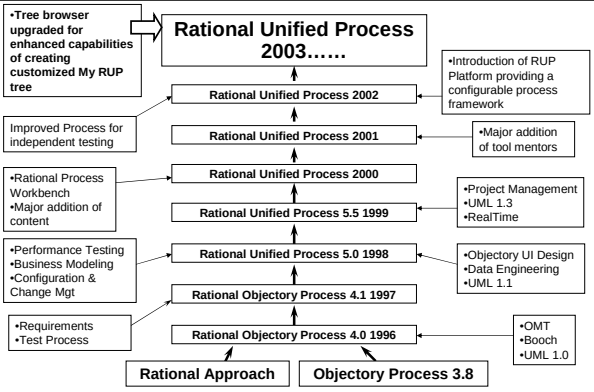
- 统一软件过程 RUP
- 敏捷过程 Agile Process
  - XP
  - SCRUM
  - 微软产品开发过程MSF
- 选择和实施软件过程
- 评估软件过程

统一软件过程 RUP

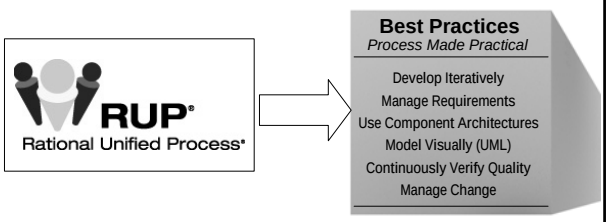


RUP是一个风险驱动的、基于UML和构件式架构的迭代、递增型开发过程。

RUP的历史



RUP 蕴涵了最佳实践准则



RUP和UML的关系

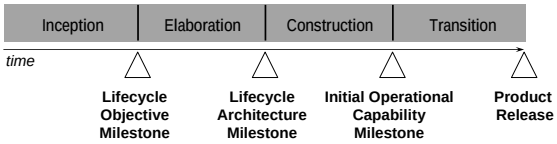
- Rational Unified Process was developed hand-in-hand with the UML.
- Many artifacts in Rational Unified Process have a UML representation.
- Rational Unified Process also includes guidelines for UML concepts.

RUP 的结构

RUP is organized:

- By time
  - Phases and iterations
    阶段和迭代
- By content
  - Disciplines
    规范

RUP按时间组织



- Inception - **Define the scope of project**
- Elaboration - **Plan project, specify features, baseline architecture**
- Construction - **Build the product**
- Transition - **Transition the product into end user community**

每个阶段结束是一个大的里程碑

Inception Phase: Objectives

- Establish project scope and boundary conditions
- Determine the use cases and primary scenarios that will drive the major design trade-offs
- Demonstrate a candidate architecture against some of the primary scenarios
- Estimate the overall cost and schedule
- Identify potential risks (the sources of unpredictability)
- Prepare the supporting environment for the project

Inception Phase: Evaluation Criteria

- Stakeholder concurrence on scope definition and cost/schedule estimates
- Agreement that the right set of requirements has been captured and that there is a shared understanding of these requirements
- Agreement that the cost/schedule estimates, priorities, risks, and development process are appropriate
- All risks have been identified and a mitigation strategy exists for each

Milestone: Lifecycle Objectives (LCO)

Elaboration Phase: Objectives

- Define, validate and baseline the architecture as rapidly as is practical
- Baseline the vision
- Refine support environment
- Baseline a detailed plan for the Construction phase
- Demonstrate that the baseline architecture will support the vision at a reasonable cost in a reasonable period of time

Elaboration Phase: Evaluation Criteria

- Product Vision and requirements are stable.
- Architecture is stable.
- Key test and evaluation approaches are proven; major risk elements have been addressed and resolved.
- Iteration plans for Construction phase are of sufficient detail and fidelity to allow work to proceed, and are supported by credible estimates.
- All stakeholders agree that current vision can be met if the current plan is executed to develop the complete system, in the context of the current architecture.
- Actual resource expenditures versus planned expenditures are acceptable.

Milestone: Lifecycle Architecture (LCA)

Construction Phase: Objectives

- Complete the software product for transition to production
- Minimize development costs by optimizing resources and avoiding unnecessary scrap and rework
- Achieve adequate quality as rapidly as is practical
- Achieve useful versions (alpha, beta, and other test releases) as rapidly as possible

Construction Phase: Evaluation Criteria

The evaluation criteria for the Construction phase involve the answers to these questions:

- Is this product release stable and mature enough to be deployed in the user community?
- Are all the stakeholders ready for the product's transition into the user community?
- Are actual resource expenditures versus planned still acceptable?

Milestone: Initial Operational Capability (IOC)

Transition Phase: Objectives

- Achieve user self-supportability
- Achieve stakeholder concurrence that deployment baselines are complete and consistent with the evaluation criteria of the vision
- Achieve final product baseline in a rapid and cost-effective manner

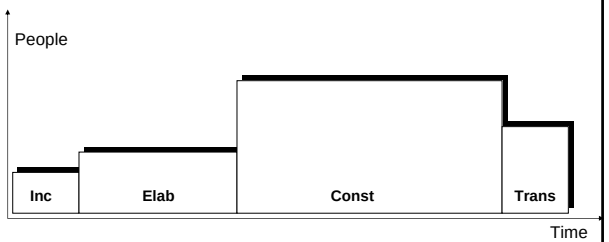
Transition Phase: Evaluation Criteria

The primary evaluation criteria for the Transition phase involve the answers to these questions:

- Is the user satisfied?
- Are actual resources expenditures versus planned expenditures acceptable?

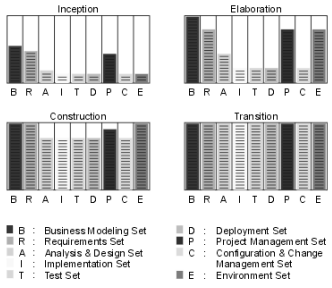
Milestone: Product release ("GA")

Typical Effort and Time Percentages by Phase



	Inc	Elab	Const	Trans
Effort	5%	20%	65%	10%
Time/Schedule	10%	30%	50%	10%

Lifecycle Evolution of Artifacts



Information set evolution over the development phases.

With the iterative approach, artifact sets mature over time.

RUP 按内容组织

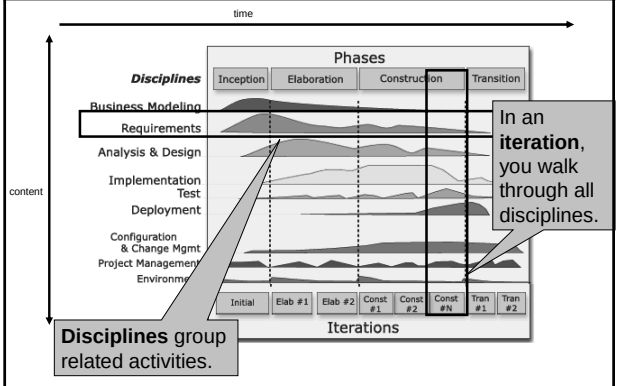
RUP content is organized into **disciplines**.

A **discipline** is a collection of activities that are all related to a major 'area of concern'.

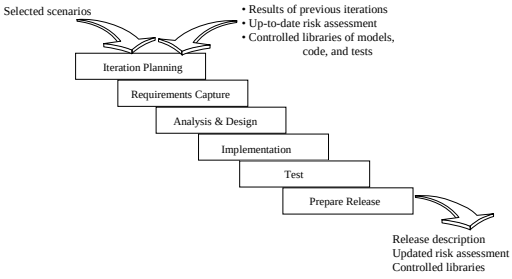
The disciplines are:

- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment
- Configuration & Change Management
- Project Management
- Environment

两者结合: 迭代方法

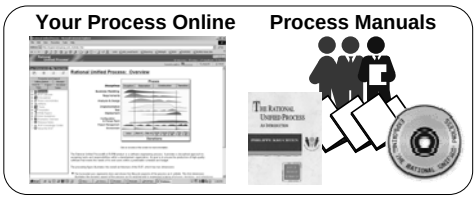


一个迭代周期：一个小的瀑布模型

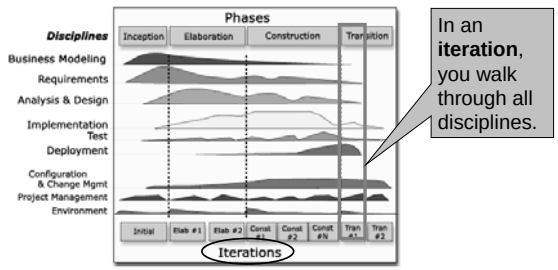


Rational Unified Process

- Captures and presents best practices
- Is a **Web-enabled**, **tailorable** knowledge base that enables efficient development of quality software
- Is continuously improved with regular upgrades
- Contains templates, guidelines, and online Help
- Promotes common vision and culture
- Mentors successful use of tools

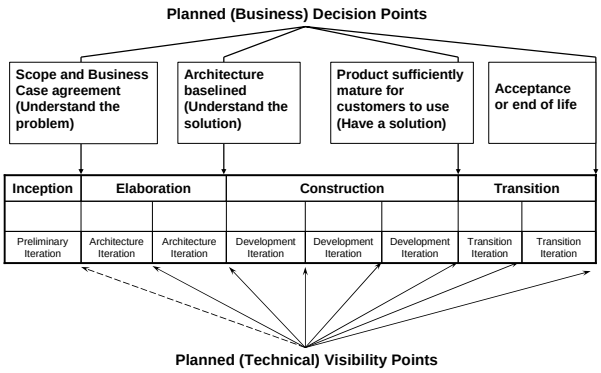


What Is an Iteration?

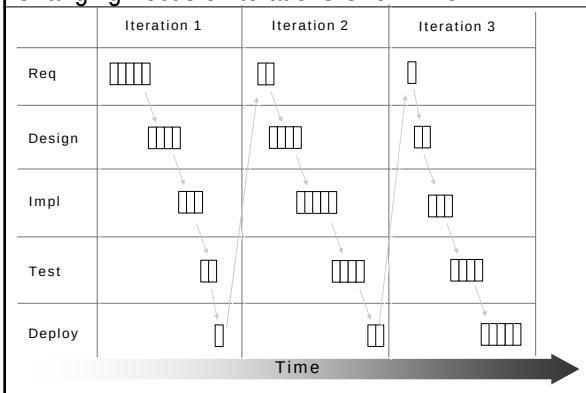


Iteration: A distinct sequence of activities with a baselined plan and evaluation criteria resulting in a release (internal or external).

Changing Focus of Phases Over Time



Changing Focus of Iterations Over Time



Duration of an Iteration

- An iteration begins with planning and requirements and ends with an internal or external release.
- Ideally an iteration should run from two to six weeks, depending on your project size and complexity.
- Factors that affect duration of an iteration:
  - Size, stability and maturity of organization
  - Familiarity with the iterative process
  - Size of project
  - Technical simplicity of project
  - Level of automation used to manage code, distribute information, perform testing

Number of Iterations

- Rule of thumb: Use 6 ± 3 iterations

Phase	Low	Medium	High
Inception	0	1	1
Elaboration	1	2	3
Construction	1	2	3
Transition	1	1	2
Total	**3**	**6**	**9**

Conditions that Increase the Number of Iterations

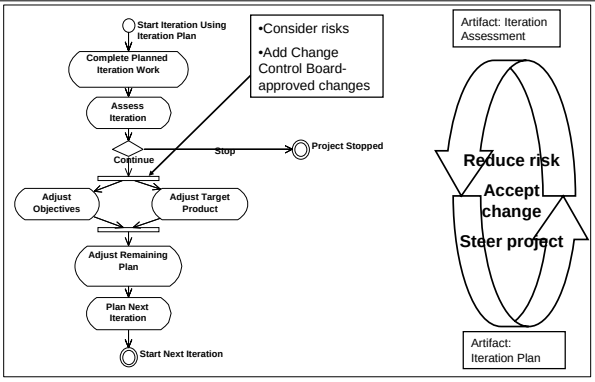
Inception
- Working with new functionality
- Unknown business environment
- Highly volatile scope
- Make-buy decisions

Elaboration
- Working with new system environment (new architectural features)
- Untested architectural elements
- Need for system prototypes

Construction
- Lots of code to write and verify
- New technology or development tools

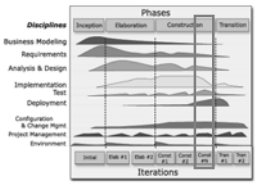
Transition
- Need for alphas and betas
- Conversions of customer base
- Incremental delivery to customers

One Iteration



Artifact: Iteration Plan

A time-sequenced set of activities and tasks, with assigned resources, and containing task dependencies. A fine-grained plan, one per iteration.



Iteration Plan Example

- Shows timeframes and resources by discipline

Outline of an Iteration Plan

1. Objectives
2. Scope
3. References
4. Plan
   - 4.1 Iteration Activities
   - 4.2 Iteration Schedule
   - 4.3 Iteration Deliverables
5. Resources
   - 5.1 Staffing Resources
   - 5.2 Financial Resources
   - 5.3 Equipment & Facilities Resources
6. Use Cases
7. Evaluation Criteria

Iteration Schedule section for Requirements discipline

Requirements	40 days	Tue 12/1/98	Mon 1/25/99	
Develop Vision	25 days	Tue 12/1/98	Mon 1/4/99	System Analyst
Elicit Stakeholder Requests	4 days	Tue 1/5/99	Fri 1/8/99	System Analyst
Manage Dependencies	26 days	Tue 12/1/98	Tue 1/5/99	System Analyst
Capture a Common Vocabulary	10 days	Wed 12/23/98	Tue 1/5/99	System Analyst

Artifact: Iteration Assessment

The Iteration Assessment captures the result of an iteration, the degree to which the evaluation criteria were met, lessons learned, and changes to be done.

Concepts That Drive Iterative Development

- Some important concepts that affect iterative development are:

  1) Early mitigation of risk

  2) Early baselining of architecture

  3) Use of objective metrics

Tailoring RUP For Your Project: A Simple Picture

MyRUP
- Personalize your view of the RUP Web site on your desktop.

RUP Builder
- Import plug-ins
- Make checkmark selections to define various RUP configurations
- Define your choice of views for RUP configurations
- Publish RUP Web sites from configurations

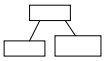
RUP Organizer
- Map content files to your process model to create a completed plug-in.
- Export completed plug-ins to RUP Builder

RUP Modeler
- Design your plug-in

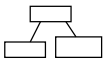
What You Get:

Your personalizable RUP Web site

= Plug-in X + part of Plug-in Y + parts of RUP base = your configuration



Process model + content files = completed plug-in



Process model

Examples of Available Plug-Ins

Shipped with RUP Builder:
- Microsoft.NET
- J2EE
- IBM Websphere App. Server
- BEA Weblogic Server
- User Experience
- Formal and informal artifact resources

Downloadable from RDN
- XP
- Creative Design
- Asset-based Development
- Etc…

本节内容

- 统一软件过程 RUP
- 敏捷过程 Agile Process
  - XP
  - SCRUM
  - 微软产品开发过程MSF
- 选择和实施软件过程
- 评估软件过程

Orthodox Methodologies 传统的软件开发方法

- 目的
  - Seek to avoid chaotic “code & fix” approach
  - Impose a disciplined process on software development
  - Goal is to make development more efficient and *predictable*
  - Strong emphasis on “planning” inspired by engineering disciplines
- 特点：重载(Heavyweight)
  - Emphases Process 强调过程
  - Emphases Documents 强调文档
  - Heavy Burdens for Developers 开发人员负担过重

New Challenge in Software Development

互联网的迅猛发展和经济全球化

Fast-Moving Markets
快速的市场进入时间
Fast-Changing Requirements
快速变化的需求
Fast-Developing technologies
快速发展的技术

Most developers and managers try to sidestep or manipulate the bureaucracy & complexity of the process

So, Ask...

- If design is good, why not make it everyone’s job?
- If simplicity is good, why not use the simplest design that supports the currently desired functionality?
- If architecture is good, why not have everyone work at defining and refining the architecture continuously?
- If short iterations are good, why not make iterations really short (hours and days) instead of weeks and months?
- If requirements, design, and code reviews are good, why not do it all the time?
- If testing is good, why not do it all the time… even customers?
- If integration testing is good, why not do it several times a day?

敏捷过程的产生

- 针对上述问题，产生了一系列轻载(Lightweight)方法。
- 2001年2月，新方法的一些创始人在美国犹他州成立Agile 联盟 (www.agilealliance.org)

XP Scrum FDD Crystal DSDM ASD Lean Development dx

What Is Agility?

- Quickness – act rapidly
- Lightness – do the minimum needed
- Nimbleness – adapt to change

敏捷过程的含义

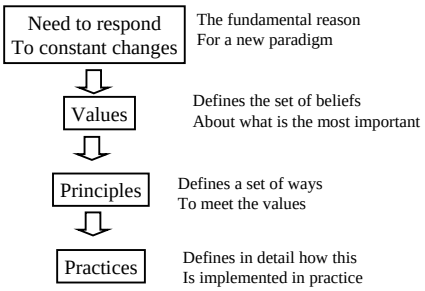
- 敏捷过程很容易适应变化并迅速做出自我调整，在保证质量的前提下，实现企业效益的最大化。
- 敏捷过程在保证软件开发有成功产出的前提下，尽量减少开发过程中的活动和制品，Just enough
- The most immediate difference is that they are less document-oriented, usually emphasizing a smaller amount of documentation for a given task.
- In many ways they are rather code-oriented: following a route that says that the key part of documentation is source code.

敏捷过程的核心理念

- 基于适应而非预测
  - **Agile方法通过快速、短迭代式的开发，不断产出和演化可运行软件，根据用户的反馈信息作适应性调整，然后进入下一轮快速短迭代式开发**
- 以人为导向而非过程导向
  - **努力营造诚信、开放的组织氛围，根据项目中信息流通的具体情况，按高内聚、松耦合的原则，将项目组划分为若干个小组（每个小组以不超过10人为宜，组员均在一个工作间内工作），通过小组内各种渠道的沟通，来减少中间制品的工作负担，提高应变能力**

--Martin Fowler "New Methodology"

Agile Thinking Explained



Value--敏捷宣言

- Individuals and interactions over Processes and Tools
  较之于过程和工具，更注重人及其相互作用的价值
- Working software over Comprehensive documentation
  较之于无所不及的各类文档，更注重可运行的软件的价值
- Customer collaboration over Contract negotiation
  较之于合同谈判，更注重与客户合作的价值
- Responding to change over Following a plan
  较之于按计划行事，更注重响应需求变化的价值

Principles--敏捷过程的12条指导原则 （1）

- 在快速不断地交付用户可运行软件的过程中，将使用户满意放在第一位
- 以积极的态度对待需求的变化（不管该变化出现在开发早期还是后期）
- 以几周到几个月为周期，尽快、不断地交付可运行的软件供用户使用
- 在项目中，业务人员和开发人员最好能一起工作
- 以积极向上的员工为中心建立项目组，给予他们所需的环境和支持，对他们的工作予以充分的信任
- 在项目组中，最有用、最有效的信息沟通手段是面对面的交谈

Principles--敏捷过程的12条指导原则 （2）

- 测量项目进展的首要依据是可运行的软件
- 高度重视可持续开发
- 项目发起者、开发者和用户应能始终保持步调一致
- 应时刻关注技术上的精益求精和设计的合理，这样能提高软件的快速应变力
- 简单化（尽可能减少不必要工作的艺术）
- 最好的框架结构、需求和设计产生于自组织的项目组
- 项目组要定期对其运作情况进行反思，提出改进意见，并进行相应的微调

Agile方法的实践效果

- **我预言XP对当今时代的作用可与CMM在八十年代和九十年代初的作用相媲美**
  -- Tom DeMarco, Cutter Trends Report

新方法在实践中取得了巨大的成功

•IONA公司的Obix技术支持小组在采用了XP方法后，软件生产率提高了67%

• SPG(software productivity group) 的Capers Jones则称，SCRUM方法可提高生产率6倍

敏捷过程的适用范围

Martin Fowler认为：新方法不是到处可适用的

适合采用敏捷过程的情况：
- 需求不确定、易挥发
- 有责任感和积极向上的开发人员
- 用户容易沟通并能参与
- 小于10个人的项目团队

Agile与CMM

- CMM更注重质量，Agile更注重生产率
- CMM强调过程的可观测性，Agile强调可观测的结果（可运行软件）
- CMM注重管理和过程，Agile注重技术和效率
- CMM注重组织，Agile注重个人
- CMM无所不包，Agile有明确的适用范围
- 它们都包含了一些软件工程的好的实践（Practices)

结合点在哪里，如何Just Enough?

本节内容

- 统一软件过程 RUP
- 敏捷过程 Agile Process
  - XP
  - SCRUM
  - 微软产品开发过程MSF
- 选择和实施软件过程
- 评估软件过程

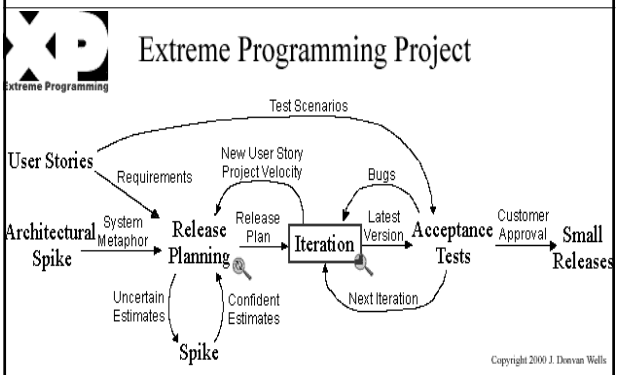
极限编程（XP）

- 由Kent Beck、Ward Cunningham、Ron Jeffries等人提出反响最大、最为完善的敏捷过程方法。
- 最初实践于1997年Crysler公司的C3项目(Smalltalk开发）
- 适用于10人以下项目组、开发地点集中的场合
- 广泛用于需求模糊和挥发性强的场合

www.extremeprogramming.org

XP的价值观

- 沟通（Communication）
  - 项目相关人员之间充分、多渠道（最好面对面）的沟通
- 简化（Simplicity）
  - 在系统可运转的前提下，做最简洁的工作
  - 在开发中不断优化设计，时刻保持代码简洁、无冗余
- 反馈（Feedback）
  - 强调各种形式（如小交付、短迭代等）的反馈
- 勇气（Courage）
  - 面对压力做正确的判断并敢于付诸行动（如敢于丢弃设计不良的代码，疲惫时立即休息等）



XP关键实践

- 现场客户（On-site Customer）
- 计划博弈（Planning Game）
- 系统隐喻（System Metaphor）
- 简化设计（Simple Design）
- 集体拥有代码（Collective Code Ownership）
- 结对编程（Pair Programming）
- 测试驱动（Test-driven）
- 小型发布（Small Releases）
- 重构（Refactoring）
- 持续集成（Continuous integration）
- 每周40小时工作制（40-hour Weeks）
- 代码规范（Coding Standards）

现场客户

- 随时能联系客户是XP方法的基本要求之一
- XP的所有阶段都要求客户的强参与
  - 需求的调研
  - Release的反馈
  - 参与测试...
- 最好有客户派员直接参与开发组
  - 有时，派开发组进驻客户现场

计划博弈

- XP要求结合业务和技术情况，快速确定下一次发布的范围。在项目计划的4要素（费用、时间、质量和范围）中，由客户选择3个，而程序员可以选择剩下的1个。
- 通常客户从业务角度确定项目范围、需求优先级和开发进度，开发人员则做出具体的成本和技术估计。
- XP强调简短和突发性的计划，有时只用几个小时甚至几分钟就能完成，而且可以随时按需进行多次计划。

系统隐喻

- XP通过一个简单的关于整个系统如何运作的隐喻性描述（story）来指导全部开发。
- 隐喻可以看作是一种高层次的系统构想，通常包含了一些可以参照和比较的类和模式，它还给出了后续开发所使用的命名规则。
- XP不需要事先进行详细地架构设计。

简化设计

- 需求是会经常变化的，因此设计不能一蹴而就而应该是一项持续进行的过程。
- Kent Beck认为对于XP来说，简单设计应该满足以下几个原则：
  - 成功执行所有的测试；
  - 不包含重复的代码；
  - 向所有的开发人员清晰地描述编码
  - 以及其内在关系；
  - 尽可能包含最少的类与方法。

集体拥有代码

- 认为开发小组的每个成员都有更改代码的权利，所有的人对于全部代码负责。
- 代码全体拥有并不意味者开发人员可以互相推委责任，而是强调所有的人都要负责。如果一个开发人员的代码有错误，另外一个开发人员也可以进行BUG的修复。
- 没有人会成为变更的瓶颈。

结对编程

- 由两名程序员在同一台电脑上结成对子共同编写解决同一问题的代码。
- 通常一个人写代码，另一个人同时负责保证代码的正确性和可读性，比如编写单元测试程序、进行代码走查。
- PP可以看作是一种非正式的持续进行的同行评审（peer review）。

测试驱动

- 先测试，再编码；代码未动，测试先行
- 强调“测试先行”。在编码开始之前，首先将测试写好，而后再进行编码，直至所有的测试都得以通过。
- 注：测试的自动化。

小型发布

- 强调在非常短的周期内以递增的方式发布新版本，从而可以很容易地估计每个迭代周期的进度，便于控制工作量和风险； 同时，也可以及时处理用户的反馈。
- 用户在发布后两个工作日内，向项目小组提交“用户接收测试报告”，由项目经理评估测试报告，将有效的BUG提交并分配给相应的开发人员。项目小组应该在下一个迭代周期结束前修复所有用户提交的问题。

重构

- 强调代码重构在其中的作用，认为应该经常进行重构。
- 重构是指在不改变系统行为的前提下，重新调整、优化系统的内部结构以减少复杂性、消除冗余、增加灵活性和提高性能。
- 重构不是XP所特有的行为，在任何的开发过程中都可能并且应该发生。

持续集成

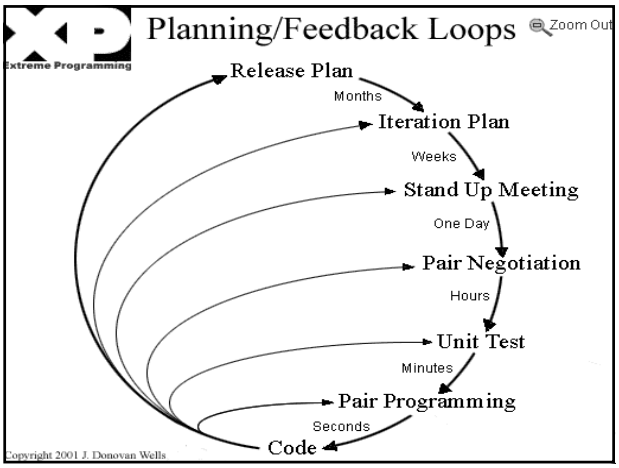
- 开发人员应不断地将代码集成到代码库中，几小时一次，绝不超过1天
- 每个人需要在最后的版本上工作
- 持续集成能够在早期避免或发现一些兼容性问题。“现在付钱还是以后付更多的钱？”

每周40小时工作制

- 不加班，不熬夜
- 超时工作会吞噬开发组的精神和热情，加班不得连续超过两周，否则反而会影响生产率
- 利用版本计划会来改变项目的范围和时间要求
- 项目进度拖延时通过增加资源来改进也不是推荐的方法

代码规范

- 所有代码必须采用统一标准以便理解。
- 代码就是文档。
- 强调通过指定严格的代码规范来进行沟通，尽可能减少不必要的文档。



XP与RUP的共性

- 基础都是面向对象方法（取代传统的结构化方法）
- 都重视代码、文档的最小化和设计的简化
- 采用动态适应变化的演进式迭代周期（取代传统的瀑布型生命周期）
- 需求和测试驱动
- 鼓励用户积极参与

XP与RUP的区别

- XP以代码为中心，编码和设计活动融为一体，弱化了架构的概念。
- RUP过程通常以架构为中心，细化阶段的主要目的就是构造出一个可运行的架构原型，作为将来添加需求功能的稳固基础。
- XP不包含业务建模、部署、过程管理等概念。
- RUP适合各种规模的项目，XP只适用于小团队。

本节内容

- 统一软件过程 RUP
- 敏捷过程 Agile Process
  - XP
  - SCRUM
  - 微软产品开发过程MSF
- 选择和实施软件过程
- 评估软件过程

XP和Scrum

- XP 注重 敏捷工程实践
- Scrum 注重 敏捷管理和组织

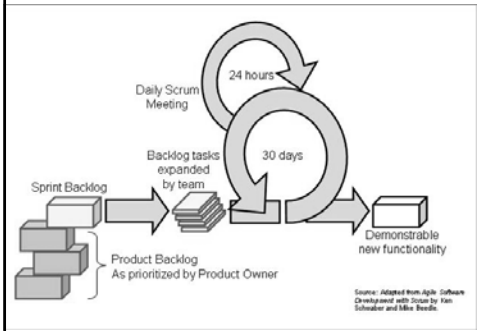
Scrum过程的来历

- 1994年由Ken Schwaber和 Jeff Sutherland 提出
- Scrum一词来源于橄榄球运动，意为两队并列争球，争球过程是迅速、有适应性、自组织的。
- Scrum过程的核心：
  - 一个体育队加小队长,全体团队负责拿球向前冲
  - 团队成员能够独立地、集中地在创造性的环境下工作

Scrum的核心准则

- Team Empowerment 自我管理
  - The project team is divided into self-managing multi-function units called Sprint Teams consisting of up to six or seven people. The team is empowered to use whatever development methods or tools they think best to prepare their deliverables
- Iterative Development 迭代开发
  - The project deliverables are built over several iterative development cycles, each adding additional features, and each resulting in demonstratable results: working code, written documentation, viewable designs, etc.

Scrum过程框架



Sprint
周期为30天的迭代

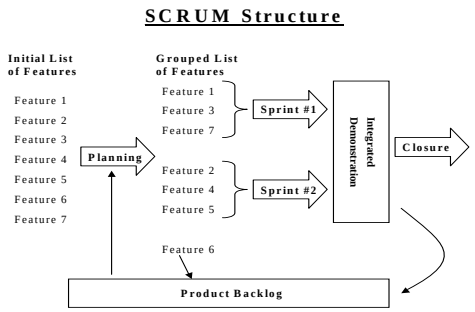
Backlog
待办事项表（功能和非功能需求清单）

Daily Scrum
每日15分钟简会

Scrum Players

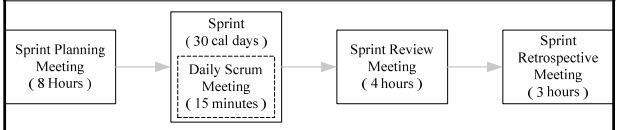
- Scrum 项目组
  - Product Owner
    - Adds items to product backlog list
    - Set priorities
  - Team
    - Determines sprint list (determine what can be done)
    - Develops software
  - ScrumMaster （相当于传统的项目经理角色）
    - Responsible for Scrum process
    - Removes impediments
- Other interested parties
  - Stakeholder
    - Funded project, will use it, affected by it
    - Requests enhancements
  - IT Management
    - Manpower allocation
    - Budgets & Billing
  - Senior Business Management
    - Best use of corporate resources

Scrum Project Structure



By Kevin Aguanno. (C) 2002 Element K Journals.

一个Sprint



- Sprint Planning meeting (two 4 hour segments)
  - Product owner & Team: identifies backlog items to be delivered in 30 day sprint / **Team commits**
  - Team plans details in Sprint Backlog
- Daily Scrum Meeting (15 minutes)
  - Team: Review prior day, identify impediments, plan day
- Sprint Review Meeting (4 hours)
  - Team: **demonstrates completed functionality**- to Product Owner and other interested parties
- Sprint Retrospective meeting (3 hours)
  - Team: how to improve next sprint or make more enjoyable

Daily SCRUM Meeting

- This is a daily meeting attended by all team members for a Sprint. The call is generally very brief (15 mins.), the purpose of which is to update the project manager and other team members on progress and to raise any issues that the project manager or other team members can assist with.
- Three questions are answered by each participant:
  - What did I do in the last 24 hours?
  - What do plan to do in the next 24 hours?
  - What obstacles are standing in my way?

Sprint的规则

- 限期30个连续日历日
- 团队将利用这段时间构建为stakeholder提供重大效益的功能，并保证它能交付
- 在Sprint期间，其他人不可向团队下达通知、指令、评论和方向指示，团队完全进行自我管理。
- Sprint计划会议上，团队选择并承诺实现Sprint Backlog。在Sprint过程中，任何人无权改动Sprint Backlog,直至Sprint结束。
- 若Sprint发生异常，ScrumMaster可非正常终止Sprint。
- 团队若认定无法在Sprint内兑现Sprint Backlog承诺，可与产品负责人协商从当前Sprint中删除部分条目。若删除条目过多，导致Sprint失去价值，ScrumMaster应非正常终止Sprint。
- 团队若认定可在Sprint内超额处理Sprint计划的Sprint Backlog，可与产品负责人协商，添加额外的条目。

SCRUM的控制手段

- SCRUM提出了八个控制项（Controls）用于开发过程的调控，其中风险控制是首要的手段。
  - Backlog
  - 对象/构件
  - Packets
  - 变动（Changes）。实施一个Backlog项时，对相应Packet的改动。
  - 难点（Problems）。实施一个变动时所必须解决的技术难点。
  - 问题（Issues）。涉及到整个项目或在Backlog项分解到Packet之前须解决的问题。
  - 措施（Solutions）。对问题或难点的解决，通常会导致变动。
  - 风险（Risks）。影响项目成功的风险，应持续跟踪评估并相应做出调整。

Scrum的好处

- 通过“限定时间”避免过分追求完美
- 通过“增量交付”改进工程实践
- 通过“放权”和“自组织”激发创造力，更好地处理复杂问题，提升员工满意度

本节内容

- 统一软件过程 RUP
- 敏捷过程 Agile Process
  - XP
  - SCRUM
- ☼ 微软产品开发过程MSF
- 选择和实施软件过程
- 评估软件过程

Microsoft 软件过程是...

The right content for the right person at the right time

1. An agile software development process
2. A formal software development process
3. Framework for customers and partners to implement custom software development process

MSF (Microsoft Solution Framework) 4.0 过程模型

- MSF for Agile Software Development
  - 意在更加灵活，在设计上是重复的
  - 注重测试与原型、较短的开发循环与持续整合
- MSF for CMMI Process Improvement
  - 注重严谨的开发流程
  - 旨在获得 CMMI Level 3 Compliance

Microsoft 在2005年发布

准则(Principles)

- 与客户做伙伴 Partner with Customers
  - 客户的确认通常可以发现现实的和虚构的商业价值间的不同。理解你的方案中陈述的价值， 并有效的同客户沟通这一点是成功的关键因素。
- 鼓励开放式交流 Foster open communications
  - 为了使成员的个人效率最大化，使工作效率提高，成员必须能被充分、有效的得到信息。
- 为了共同的远景工作 Work toward a shared vision
  - 分享远景可以确保所有的成员认同他们可以通过这个项目获得什么。由于每个决定都不是武断做出的，而是基于大家认同的远景做出的，所以可以提高合作性。
- 质量是每个人每天的工作 Quality is everyone's job every day
  - 质量需要错误预防机制和解决方案的验证共同来保证。利用一些方法，如代码分析和相互审核，来防止问题，同时尽可能的多测试来发现问题。所有的角色都对防止问题和验证问题负责。

准则(Principles) cont.

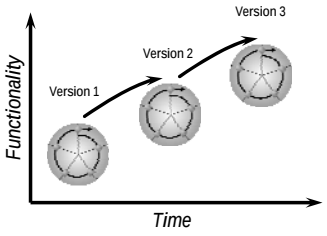
- 保持敏捷，准备改变 Stay agile, adapt to change
  - 一个组织越在技术领域中寻求最大化的商业影响，承担的风险越大。新的领域总是未知的，并容易随着探索和试验发现的新方法和新需求而改变。在一个改变中的环境做出确切的决定是不切实际的，并容易导致失败的结果。
- 使发布成为习惯 Make deployment a habit
  - "Teams that learn to ship, ship!"，这对风险管理、目标偏差消除、和价值流促进是十分重要的。通过持续的发布和测试，甚至是早期的原型，都能让整个团队掌握如何让发布顺利地按预期进行。
- 价值流 Flow of value
  - 一个项目就是实现一组价值的过程。应把不能给客户增加价值的行为降到最低，因为这些是浪费。使用迭代来定期递交和发布一组客户价值，消除目标偏差以促进价值流。同时通过持续提高开发人员的生产率来提高实现价值的能力。

微软软件开发的成功经验（best practice）

- 以bug数据库为基础来保证质量
- 产品技术开发部门的反馈
- 以产品功能规格书为标准
- 完成视觉效果的设计visual freeze
- 源程序树的锁定 lockdown source tree
- 源程序树的分叉fork source tree
- 产品编码完成code complete
- 先自尝其果
- 可用性测试
- 战斗组和战斗会议war meeting
- Component team bug triage
- Context Driven Testing
- 金主碟
- 测试人员签字，同意产品发布
- Using Versioned Releases
- Daily Builds
- Check-in
- Risk Management

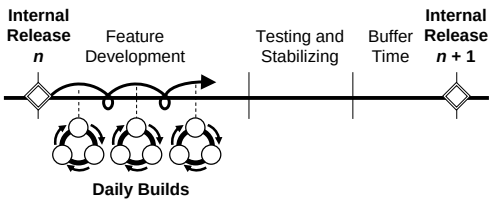
Using Versioned Releases

- Force closure on project issues
- Set clear and motivational goals with all team members
- Manage the uncertainty and change in project scope
- Encourage continuous and incremental feature delivery
- Enable shorter time to market

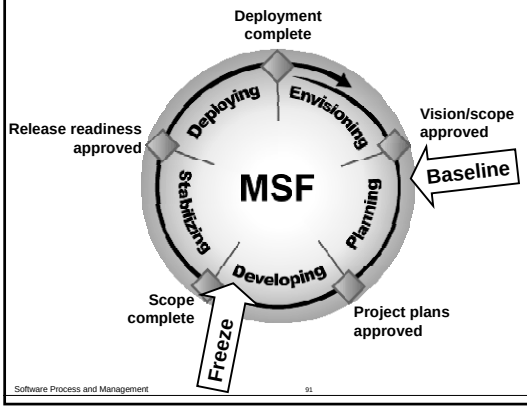


Daily Builds and Internal Releases

- Daily builds lead to internal (alpha releases)

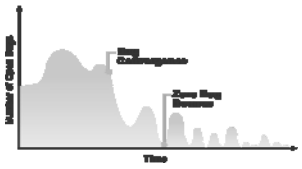


Baseline Early, Freeze Late



Context Driven Testing

- Testing that is acceptable on one project may be criminal on another
- Testing Targets
- Trend Analysis
- Metrics:
  - % Code Coverage
  - Bugs Found
  - % Code Churn
  - % Tests Passed
  - % Scenarios Tested



Risk Management

Identifying, analyzing, and addressing risk proactively

To manage risk proactively		
Anticipate problems	vs.	Fixing them when they occur
Address root causes	vs.	Addressing symptoms of the cause
Prevent and minimize risk through mitigation	vs.	Reacting to consequences
Prepare for consequences to minimize impact	vs.	Reacting to a crisis
Use a known and structured process	vs.	Using an ad-hoc process

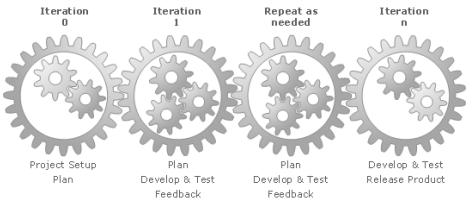
Minimize Surprises

MSF for Agile Software Development

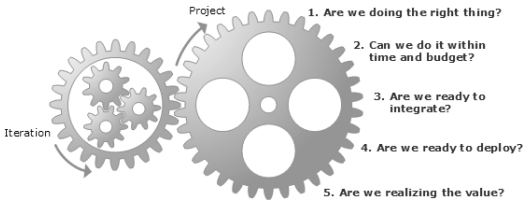
- First agile process that considers the whole software lifecycle and the full software team.
- Iterative and incremental
- Scenario-driven
- Small teams
- Quality of Service requirements
- Risks
- Utilizes a context-driven testing approach (based on test metric thresholds)

迭代(Iteration)

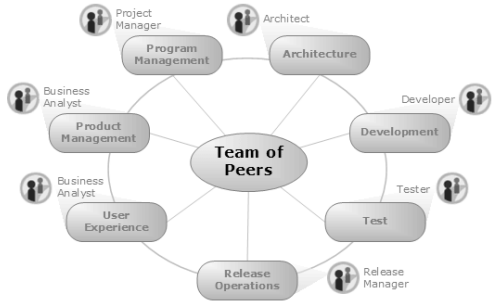
- Product definition, development, and testing occur in overlapping iterations.
- Small iterations allow you to reduce the margin of error in your estimates and provide fast feedback.
- Each iteration should result in a stable portion of the overall system.



管理(Governance)



角色(Roles)



Business Analyst Project Manager Architect
Developer Tester Release Manager

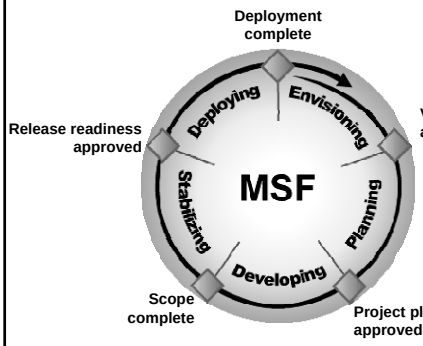
工作项(work item)

- 5 work items to assign and track work
  - Scenario
  - Quality of service requirement
    - Security, performance, and user experience
  - Task
  - Bug
  - Risk

MSF for CMMI Process Improvement

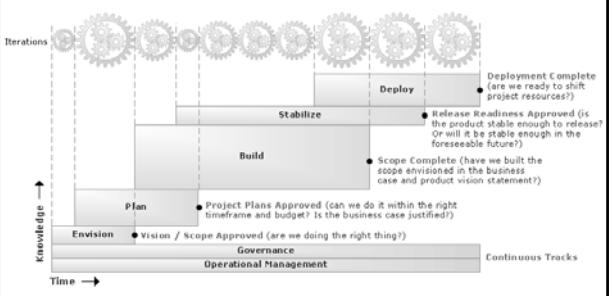
- First agile CMMI process
- Help organizations operate at Capability Maturity Model Integration (CMMI) level 3, a standard defined by the Carnegie Mellon Software Engineering Institute (SEI)
- Elaborates on the MSF for Agile Software Development process
  - More work items
  - Extensive reporting
- Does not replace process improvement infrastructure

过程模型



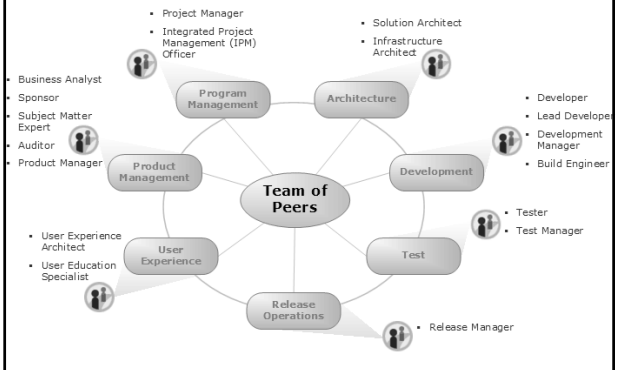
- Milestone-based
- Iterative
- Flexible

管理(Governance)

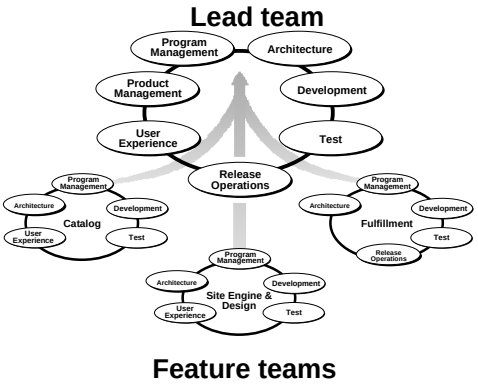


5 phases / milestones

角色(Roles)



大项目的团队



工作项(work item)

- 7 work items to assign and track work
  - Task
  - Change Request
  - Risk
  - Review
  - Requirement
  - Bug
  - Issue

每个阶段的主要活动(Activity)

本节内容

- 统一软件过程 RUP
- 敏捷过程 Agile Process
  - XP
  - SCRUM
  - 微软产品开发过程MSF
- 选择和实施软件过程
- 评估软件过程

最佳过程?

"One Size does not Fit All"

不同的项目需要不同过程

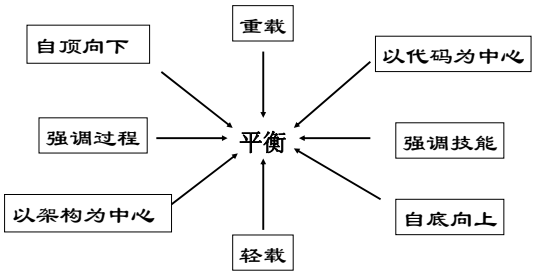
- **具体环境：项目、产品（军用、民用等）、资源、团队、文化、地域（集中、分布等）……**
- **层次：组织过程、项目过程、团队过程、个人过程**
- **业务目标**
- **开发类型：新产品、重用、COTS、维护、服务、产品线……**

过程选择原则

- 越大的团体需要越大的过程。
- 越关键的系统（未发现的缺陷将产生更严重的灾害）在构建的正确性方面需要越多的透明度（更大的密度）。
- 在过程大小或密度上相对小的增加会引起项目成本相对大的增长。
- 最有效的沟通（交换意见）方式是面对面的互动，例如在白板前的讨论。

——Alistair Cockburn, Selecting a Project's Methodology

成功之道——掌握平衡



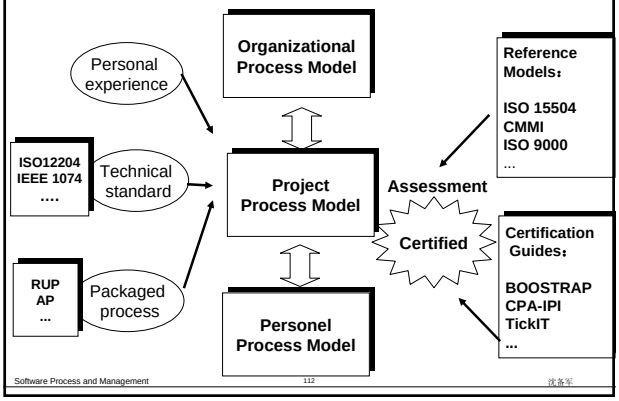
软件开发过程的制定步骤

- 借鉴RUP、敏捷过程等推荐过程，定义适合本机构的软件过程标准
  - 不同的项目类型，可能定义不同的软件过程
  - 软件过程需不断改进
- 在项目中，对本机构的标准软件过程进行裁减和细化，制订项目软件过程，再编制项目计划

本节内容

- 统一软件过程 RUP
- 敏捷过程 Agile Process
  - XP
  - SCRUM
  - 微软产品开发过程MSF
- 选择和实施软件过程
- 评估软件过程

软件过程评估



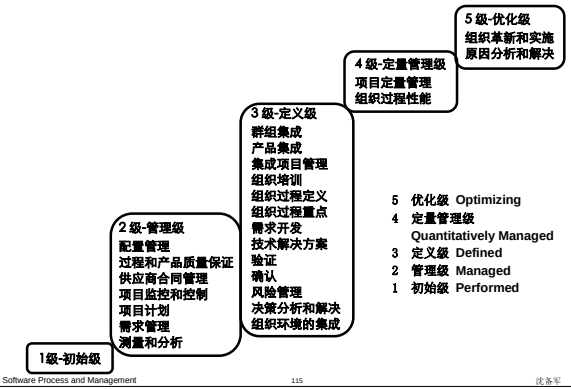
过程参考模型

- CMMI
- ISO 15504
- ISO 9000
- ISO 20000
- …

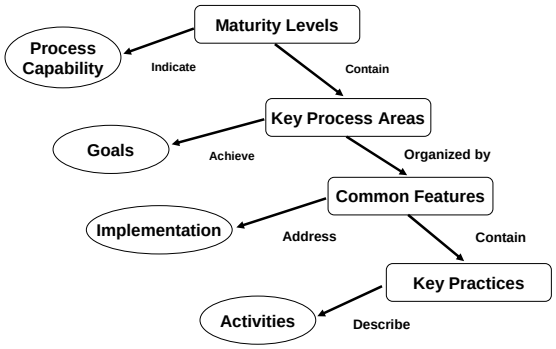
CMM 发展历史

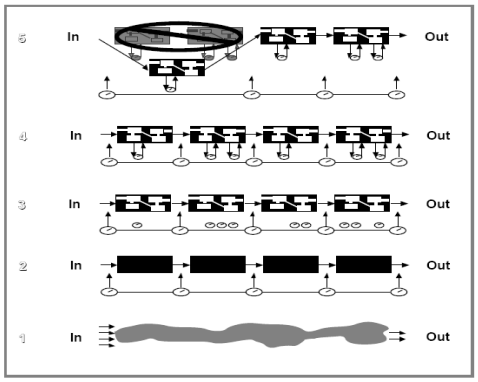
- 20世纪70年代中期，美国的DoD发现，70%的失败项目是因为管理不善而引起，于是委托CMU/SEI开发CMM (Capability Maturity Model)。CMM 模型在理论上基于20世纪30年代Walter Shewart 的统计质量控制原理。
- 1987年 CMM初版
- 1991年 CMM1.0
- 1993年 CMM1.1
- 2002年集成多种标准，提出新版本CMMI 1.1（CMM Integration）
- 2006年 CMMI 1.2

CMMI 成熟度



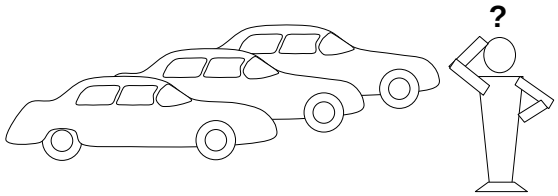
CMMI 结构





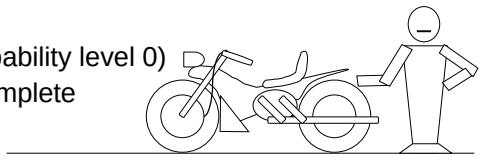
A Management View of Visibility Into the Software Process at Each Maturity Level

Example - Buying a Vehicle for Family Transportation



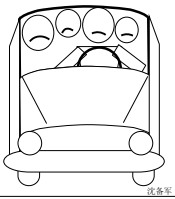
Ineffective Approach

- Go to “Vehicles R Us” and, on a whim, buy a really neat motorcycle!
- Result: Goal of obtaining family transportation is not met.
- (Capability level 0) Incomplete



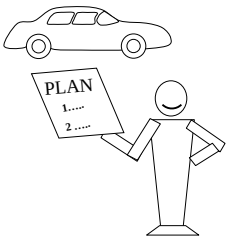
Most Basic Approach

- Go to “Vehicles R Us” and let a salesman talk you into the first thing that will hold your family.
- Result: Goal of obtaining family transportation is met, but how deliberate was this approach? If this worked this time, could you predict similar success the next time?
- (Capability level 1) Performed



A More Deliberate Approach

- Develop a plan for your approach. Allocate the time for the task. Educate yourself on how to do it. Follow the plan. Have your spouse “keep you honest” on following your approach.
- Result: Goal of obtaining family transportation is met. Approach may be repeated the next time.
- (Capability level 2) Managed

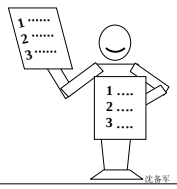


Turning this into a Business

- You get so good at this, you develop an approach to be used by other families. You can tailor your approach to different family finances and needs. You improve your approach to stay in business.
- Result: Goal of obtaining family transportation is met for a variety of situations. Can repeat success in different contexts.
- (Capability level 3) Defined

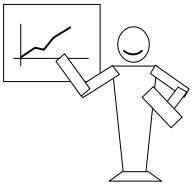
Using Numbers to Run Things

- You get really good at this. Building on what you have, you collect numbers and use them to run all important aspects of the business.
- Result: Goal of obtaining family transportation is met in a variety of situations. Management of the business is quantitative rather than intuitive.
- (Capability level 4) Quantitatively Managed



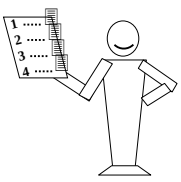
Using Numbers to Improve Things

- You get really good at this and continue improving. You use your numbers to run and improve all important aspects of the business.
- Result: Business goals are met. Culture of continuous improvement driven by numbers is the “way we do business.”
- (Capability level 5) Optimizing



A Common Theme: Learn from the Experience

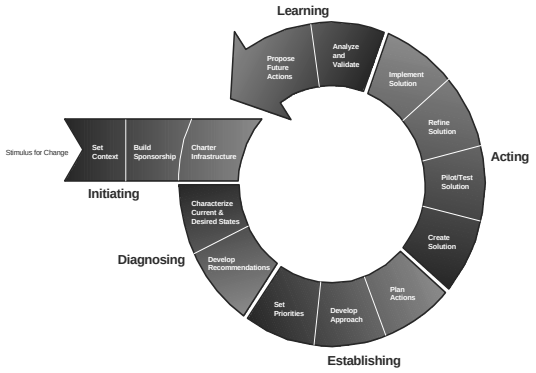
- At each step determine
  - What went well?
  - What could be improved and how?
- Make improvements in the process.
  - Use measures to help.



Remember

- CMM/CMMI model is not a process.
- CMM/CMMI model shows what to do, NOT how to do it or who does it.

软件过程改进的IDEAL℠ 模型



Using CMMI Models with the IDEAL Model 1

- Initiating phase
  - CMMI models can assist an organization in understanding how to build sponsorship and in developing the infrastructure for improvement.
- Diagnosing phase
  - The Standard CMMI Appraisal Methodology for Process Improvement (SCAMPISM) provides a yardstick for appraising processes based on CMMI.

Using CMMI Models with the IDEAL Model 2

- Establishing phase
  - CMMI process areas focus the process improvement teams.
- Acting phase
  - CMMI models provide guidance for defining or improving processes.
- Learning phase
  - Lessons learned are documented and are the basis for revision of an organizational approach.

软件开发过程操练

- 计划开发一个学生在线选课系统，估计10000行代码，5个月开发周期，从5/1开始开发，学校希望9/1学生能正式使用本系统。
- 请设计本项目的软件过程。