软件过程管理

-Ch.5 软件过程的技术管理

闫波
北京理工大学 计算机学院

yanbo@bit.edu.cn

软件过程的技术管理

yanbo@bit.edu.cn

为了解决软件问题，重要的第一步就是将整个软件开发任务看做一个可控的、可度量的以及可改进的过程。

—瓦特·汉弗莱（Watt Humphrey）

本章提纲

yanbo@bit.edu.cn

5.1 软件过程的技术架构

5.2 软件过程的问题分析和决策方法

5.3 软件过程的技术路线

5.4 知识传递

5.5 软件过程管理工具

5.1 软件过程的技术架构

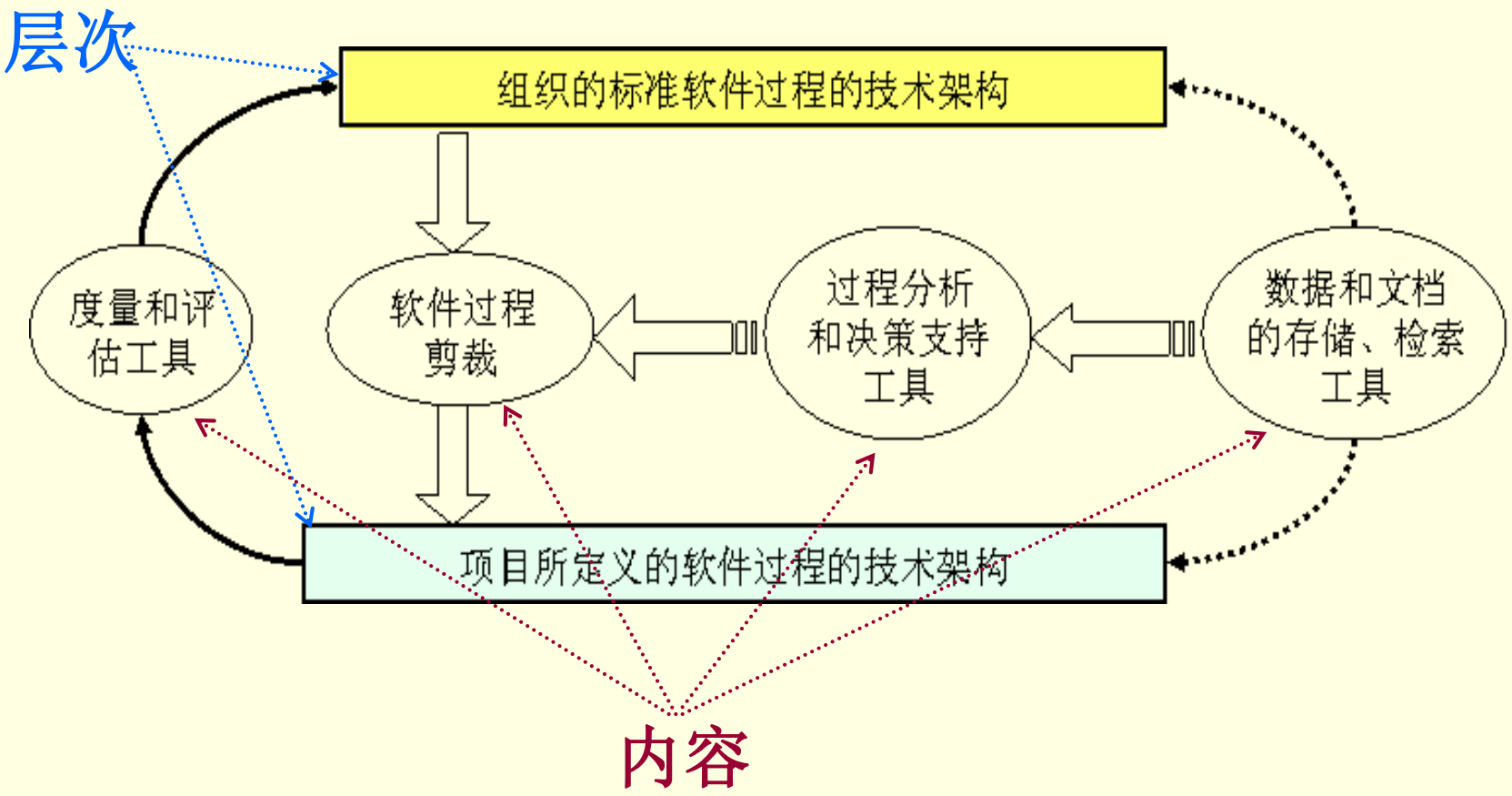
yanbo@bit.edu.cn

5.1.1 过程技术架构的层次和内容

5.1.2 软件过程资源的管理

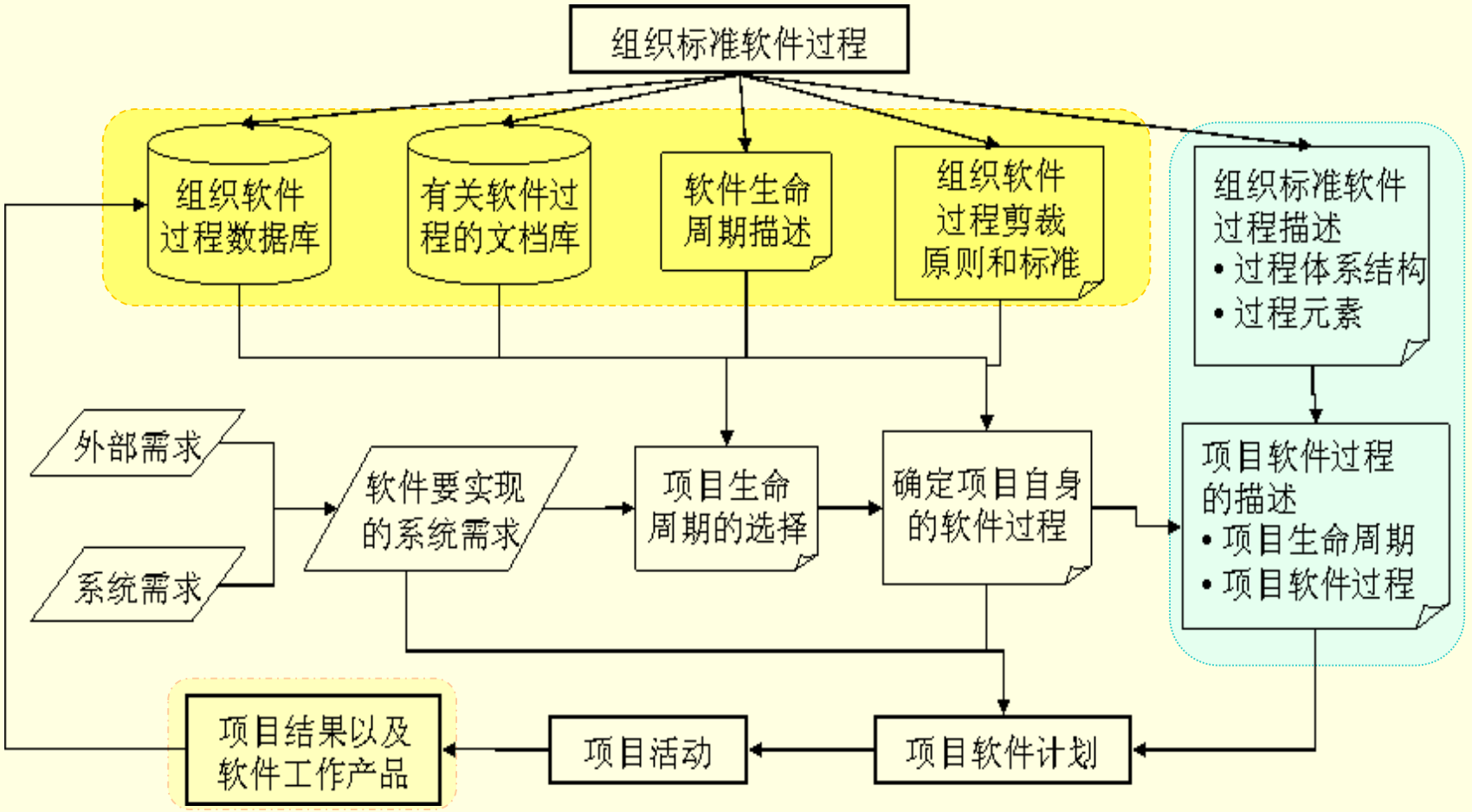
5.1.1 过程技术架构的层次和内容

yanbo@bit.edu.cn



5.1.2 软件过程资源的管理

yanbo@bit.edu.cn



5.2 软件过程的问题分析和决策方法

yanbo@bit.edu.cn

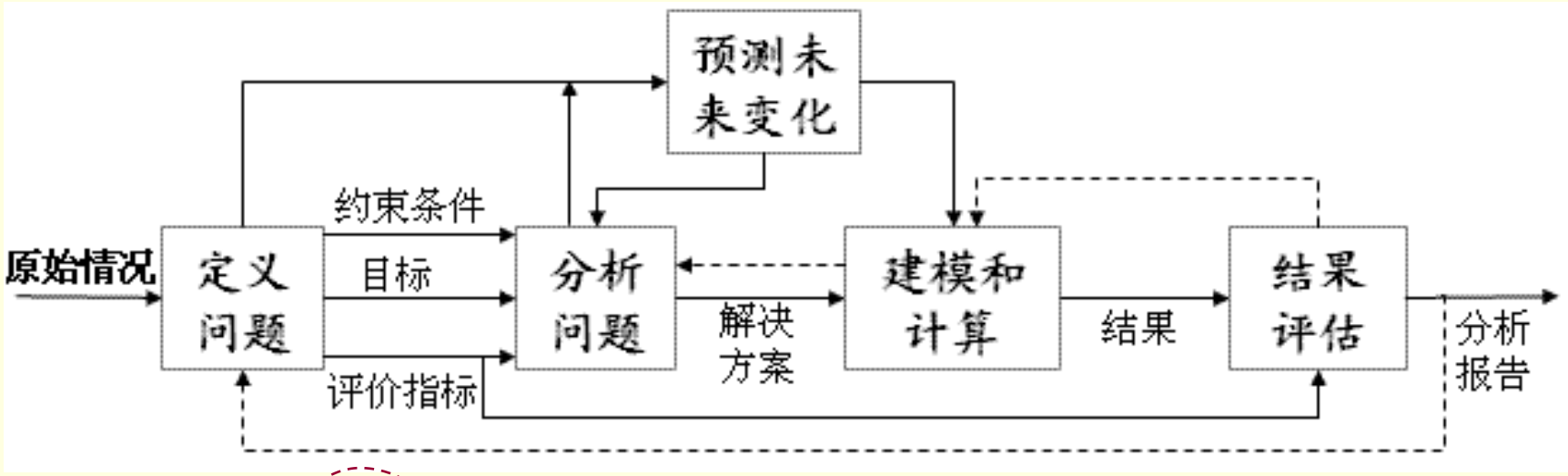
5.2.1 过程问题解决的系统方法

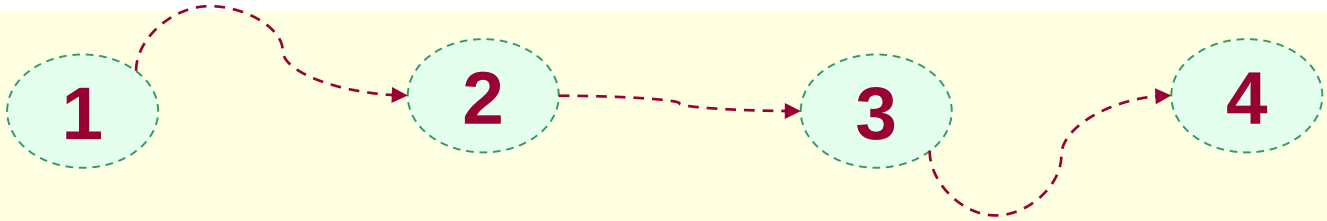
5.2.2 原因分析和缺陷分析

5.2.3 决策分析与决定

5.2.1 过程问题解决的系统方法

yanbo@bit.edu.cn





5.2.2 原因分析和缺陷分析

yanbo@bit.edu.cn

- 在开发周期的每个阶段实施根本原因分析（root cause analysis），为有效开展缺陷预防活动提供依据。
- 通过制订原因分析计划、选择缺陷分析数据而找出原因、实施建议措施、评价变更的效果、记录数据等多个环节，最终完成这一活动。
- 经常使用的工具有：数据库系统、过程建模工具、统计分析包。

5.2.3 决策分析与决定

yanbo@bit.edu.cn

- 选择决策技术和结构层次，制订决策分析与决定的计划；
- 建立作为决策基础的评价准则；
- 建立并运用决策分析指导原则，确定推荐的候选方案；
- 选择评价方法，对照准则评价候选方案；
- 选择解决方案 ；

5.3软件过程的技术路线

yanbo@bit.edu.cn

5.3.1 软件项目过程的技术解决流程

5.3.2 技术解决计划的建立和实施

5.3.3 开发设计

5.3.4 编程和单元测试

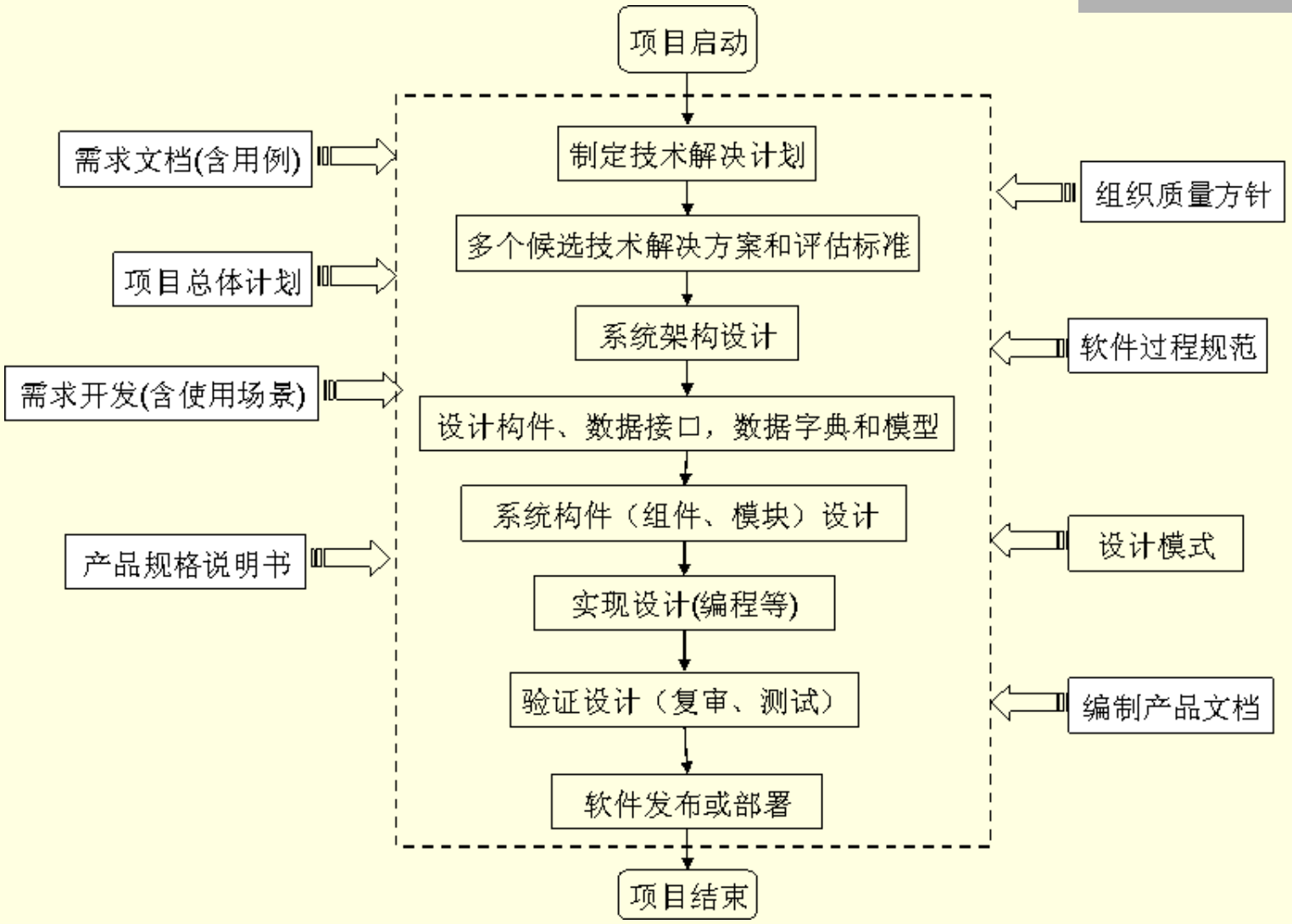
5.3.5 验证、确认与测试

5.3.1 软件项目过程的技术解决流程

yanbo@bit.edu.cn

- 制订技术解决计划。
- 系统定义、候选方案和评估准则。
- 系统操作概念和使用场景。
- 系统架构设计。
- 系统构件的详细设计。
- 实现设计——完成编程和单元测试。
- 通过复审、测试完成对系统的验证。
- 软件发布或部署。
- 软件的操作和维护。

技术解决流程示意图



yanbo@bit.edu.cn

5.3.2 技术解决计划的建立和实施

- 建立并维护技术解决的组织方针，反复进行产品构件的选择、产品和产品构件的设计以及产品构件设计的实现、验证工作。
- 设计技术路线，确定技术路线中关键的难题和初步的解决办法。
- 根据项目的规模以及财力，确定技术解决人力资源、硬件资源和技术解决工具。
- 技术解决方案准则应该包含对软件生命周期设计问题的处理。
- 为每个候选解决方案拟订产品运行和用户交互作用的时间场景。
- 应充分考虑新技术所带来的风险，要计划好一些应急的措施或备用的成熟的技术。

技术工具

yanbo@bit.edu.cn

- 设计规范工具。
- 仿真程序和建模工具。
- 原型设计工具。
- 场景定义和管理工具。
- 需求跟踪工具。
- 交互式文档编制工具。

5.3.3 开发设计

yanbo@bit.edu.cn

1. 系统定义
2. 设计标准和准则的属性
3. 设计方法
4. 产品构件设计
5. 设计文档

- 通用软件设计规范。
- 用户界面标准。
- 设计安全标准。
- 技术限制。
- 设计容差。
- 部件标准

1. 原型设计方法。
2. 基于信息隐蔽原则的Parnas设计方法。
3. 结构化设计方法。
4. 问题分析法（PAM）
5. 面向对象的设计方法。
6. 面向数据结构的软件设计方法 （Jackson方法和Warnier方法）
7. 面向构件的设计

5.3.4 编程和单元测试

- 主要的编程思想
- 推荐的编程方法
- 编程准则和规范
- 单元测试方法
- 代码重构

•测试驱动编程

•采用成熟而先进的方法

•遵守已定义的编程准则和规范

•经常进行代码互为审查、走查

•……重构

•结构化程序设计

•面……

•自……

•软……

•使……

•结构化、模块化。

•清晰……设计。

•……变量命名约定……

•……在使用语言的特定规则，

……指针使用、内存释放。

•可……结构。

•软件构件的结构和分层。

•足够的注释行。

•语句覆盖测试。

•分支覆盖测试。

•条件覆盖测试。

•谓词覆盖测试。

•路径覆盖测试。

•边界值测试。

•特殊值测试。

5.3.5 验证、确认与测试

- **验证（verification）**是指验证或检验软件是否已正确地实现了产品规格书所定义的系统功能和特性，验证过程提供证据表明，软件相关产品与所有生命周期活动的要求相一致。
- **确认（validation）**是为了保证所生产的软件可追溯到用户需求的一系列活动，确认过程提供证据，表明软件是否满足客户需求，并解决了相应问题。
- **测试（testing）**是为了发现软件的缺陷，减少产品质量的潜在风险。测试是实现验证活动和确认活动的最有效的手段和途径。

V&V

yanbo@bit.edu.cn

- Verification：Are we building the product right?
  是否正确地构造了软件？即是否正确地做事，验证开发过程是否遵守已定义好的过程规范。
- Validation： Are we building the right product?
  是否构造了正确的软件？即是否正在做用户真正所需要的产品

详细 比较见表5-4

5.4 知识传递

yanbo@bit.edu.cn

- 纵向传递是一个具有很强时间顺序性的接力过程，指软件产品和技术知识从需求分析阶段到设计阶段、从设计阶段到编程阶段、从开发阶段到维护阶段、从产品上一个版本到当前版本的知识传递过程。
- 横向传递是指软件产品和技术知识在不同团队之间的传递过程
- 知识传递的有效方法

5.5 软件过程管理工具

yanbo@bit.edu.cn

5.5.1 需求管理工具

5.5.2 面向对象的分析设计工具

5.5.3 配置管理和变更管理工具

5.5.1 需求管理工具

- IBM-Rational RequisitePro
- Telelogic DOORS
- 青铜器RDM（IPD+CMMI+Scrum一体化研发管理解决方案）
- Borland Caliber
- Teambition
- Dragonfly
- OSRMT
- ……

5.5.2 面向对象的分析设计工具

yanbo@bit.edu.cn

- IBM-Rational Rose是面向对象技术分析设计工具的代表，是可视化的建模工具
- **面向对象技术分析设计工具很多，如表5-5所示，** 其中SVG是W3C的一种图形矢量标准，可以在网上快速加载矢量图和UML图，强大的事件及脚本功能，也使得UML图具有更强的交互性和更为丰富的表达能力。

5.5.3 配置管理和变更管理工具

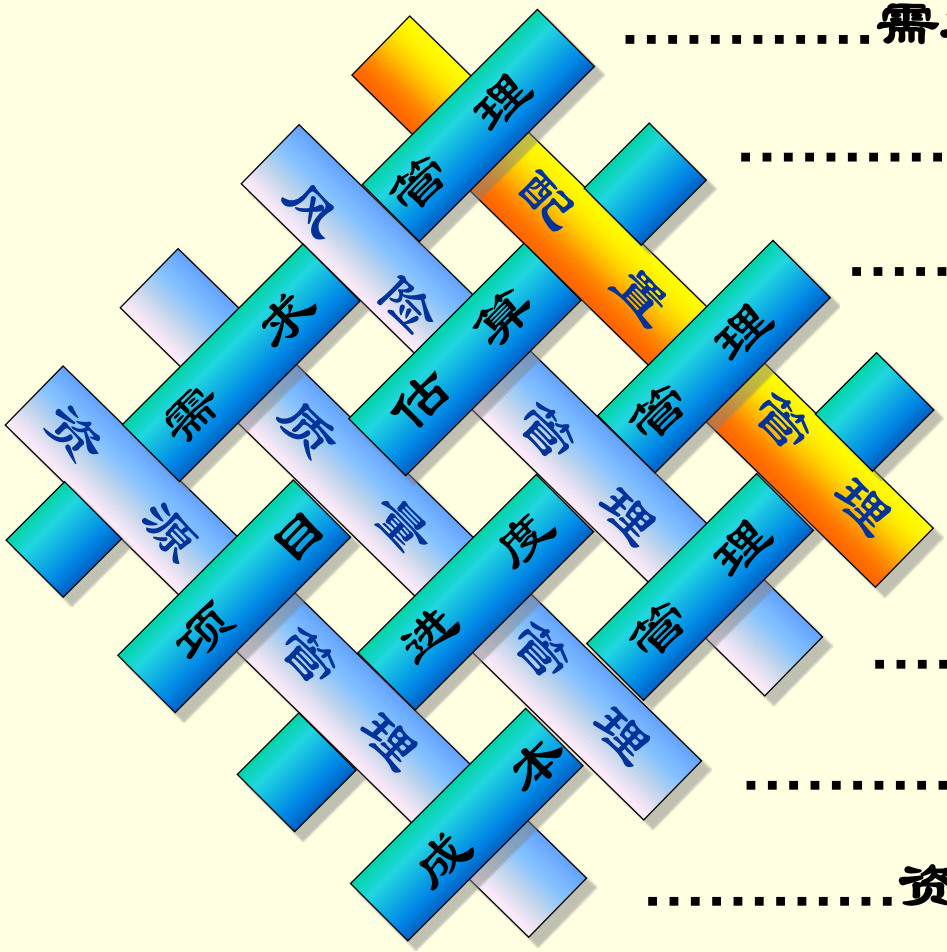
yanbo@bit.edu.cn

- **配置管理**的主要工作包括通过创建软件配置管理库、定义配置项（包括需求、分析设计模型、代码、文档、测试用例、测试数据等）以及建立和维护软件的基线。
- **变更请求管理**的主要工作包括控制和记录配置项内容的变更，建立和维护一个系统并使其追踪和管理变更请求及问题报告。

(1) 开源工具CVS（Concurrent Versions System，并发版本系统）是网络透明的版本控制系统。
(2) IBM-Rational ClearCase。
(3) 青鸟软件配置管理系统（简称JBCM系统）是基于构件复用的配置管理系统。
(4) IBM-Rational ClearQuest是需求变更管理工具。

配置管理-软件项目管理的关键内容

yanbo@bit.edu.cn



一个问题例子

yanbo@bit.edu.cn

软件出问题了...

我马上解决这个问题
(忘了变更登记)

结果...

yanbo@bit.edu.cn

仍然有问题...

我已经改过了》_《

另一些可能的情况

yanbo@bit.edu.cn

问题	现象
找不到软件	我知道我已经写好了，但是不知道放哪儿了
丢失连接	原来还是好好的，但是现在它指向的代码已经不见了
相互覆盖代码	开发人员对相同的代码做了不同的修改，互相覆盖
无法返回	新的修改比原来的更差，但是无法撤回到原来的情况
文档丢失	落下一份没有页码的文档
文档区分不清	落下两份没有标题的文档，哪份是哪份？
版本不清	客户报告了错误，该给他哪个补丁呢？

为什么需要配置管理？

系统的需求

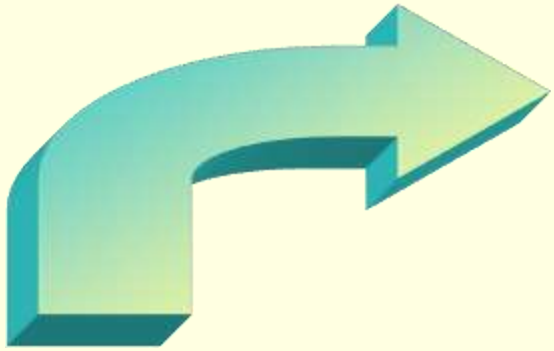
增长900%

然

为什么需要配置管理？

yanbo@bit.edu.cn

从软件开发的规模看

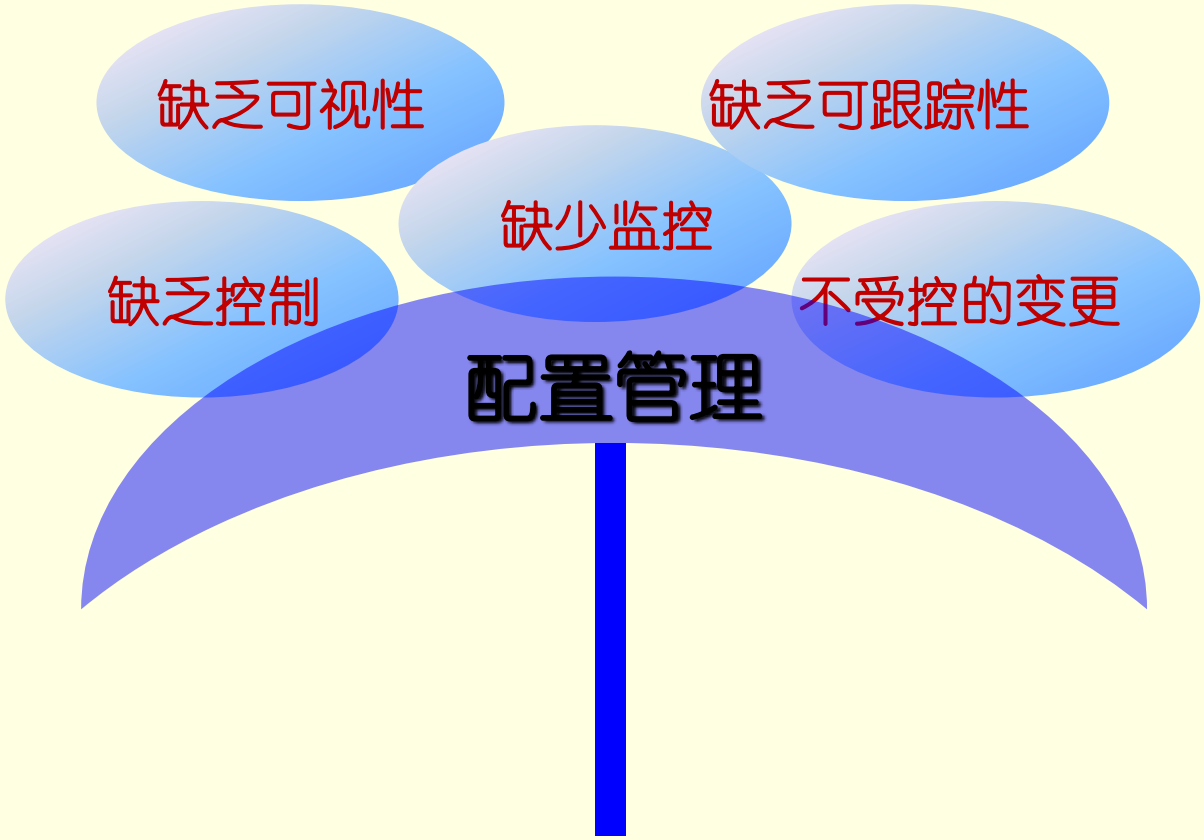


现今:
规模: 超过25,000,000行源代码
人员: 上千名程序员
时间: 大约持续五年
花费: $500,000,000
开发地点: 世界的不同角落

在软件开发早期:
规模: 1000机器代码行以内
人员: 1个程序员
时间: 很少超过一个月
花费: <= $5000
开发地点: 1处

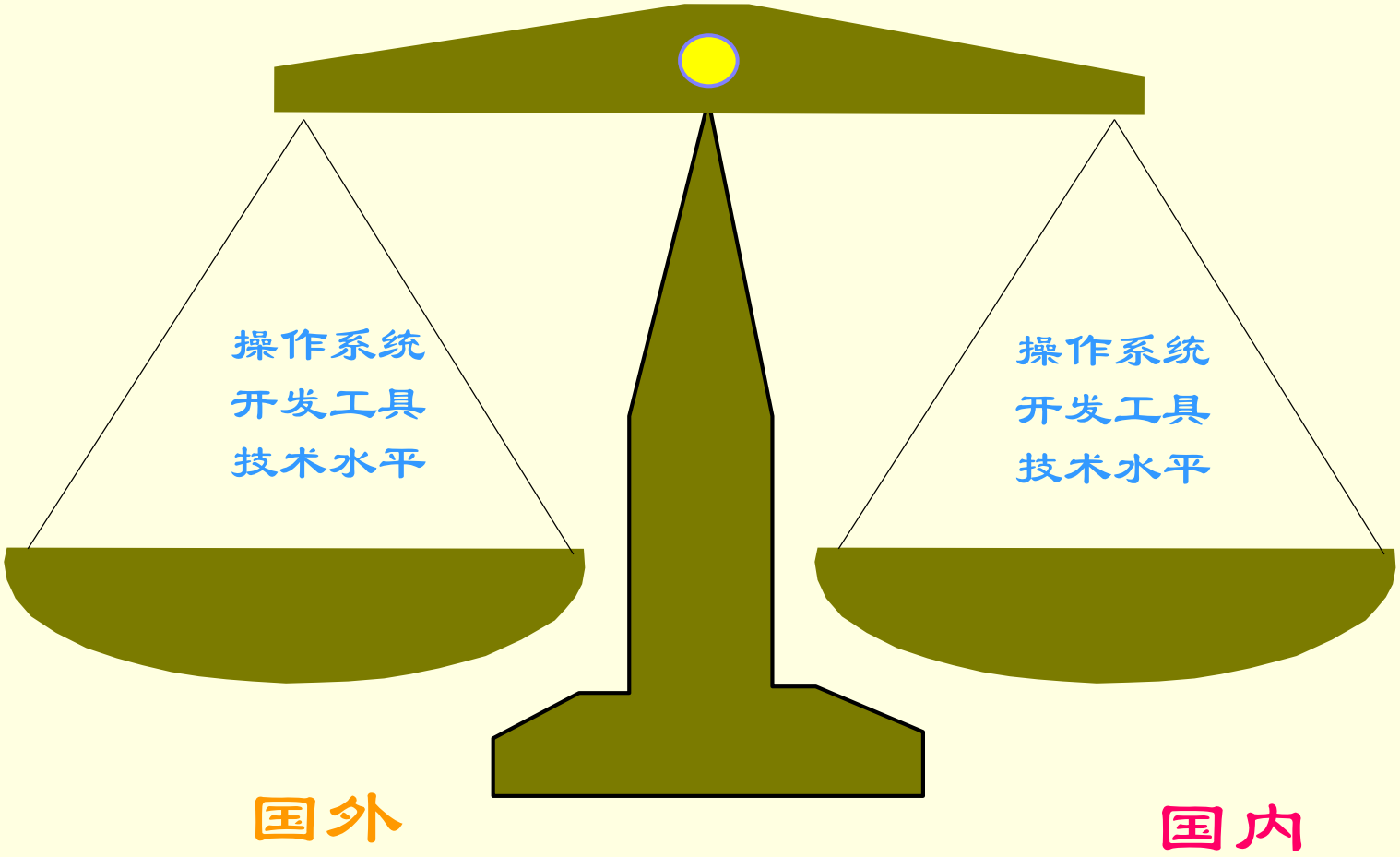
软件开发中的一系列普遍问题

yanbo@bit.edu.cn



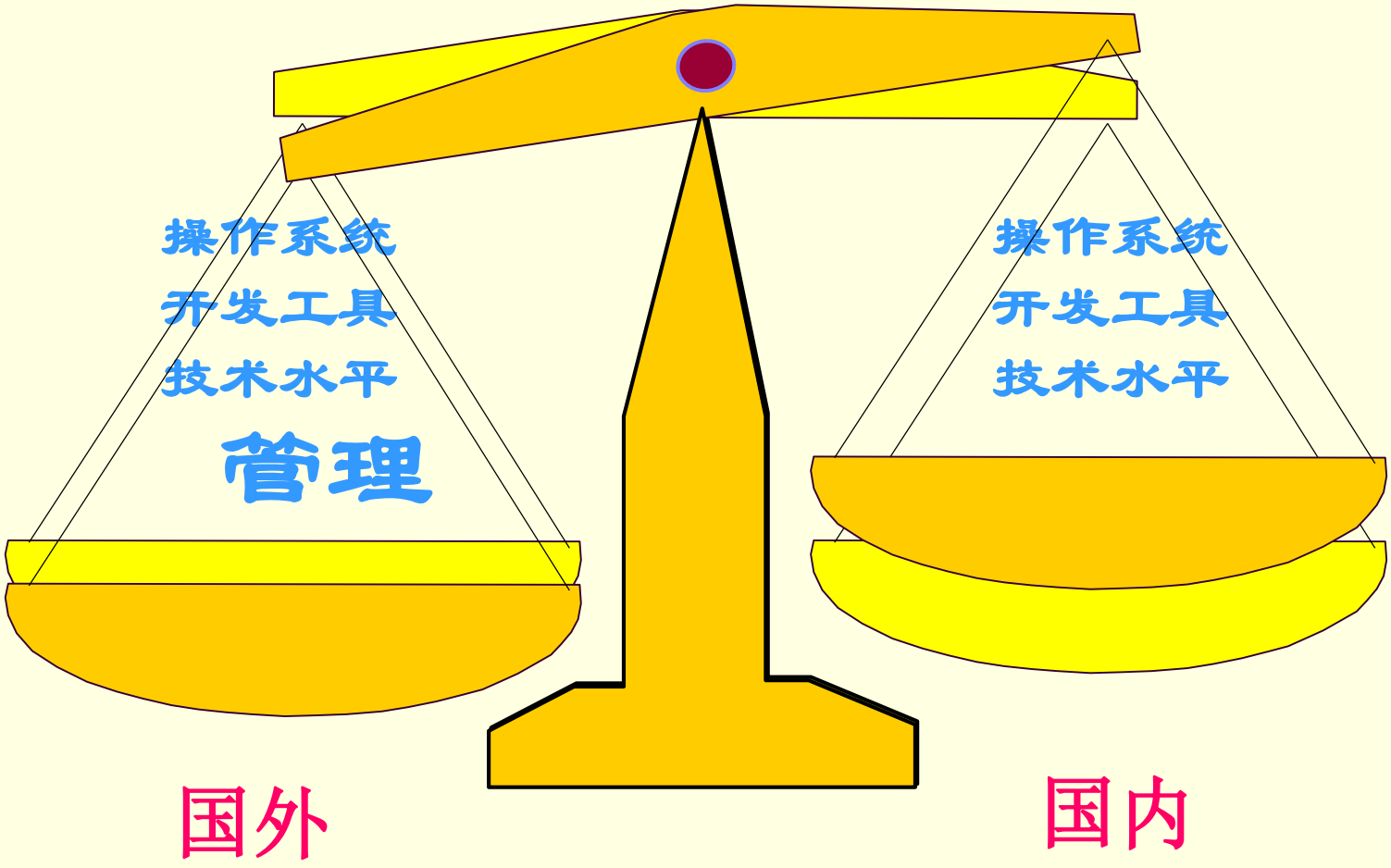
与国外的软件开发相比

yanbo@bit.edu.cn



缺乏规范的管理

yanbo@bit.edu.cn



配置管理给我们带来了什么

yanbo@bit.edu.cn

好处	问题
提高软件开发生产率	为配置管理设置专门的人员，并且要有一些软硬件环境支持，都增加了开发的成本
降低软件维护费用	
确保构建正确的系统	
更好的质量保证	
减少缺陷	对于简单系统的开发没有实际的意义
使软件开发依赖于过程 而不是依赖于人	

yanbo@bit.edu.cn

的一个主要问题是持

变化，以最有效的手
的目标。

信息系统项目相关信息（文档）

yanbo@bit.edu.cn

- 指某种数据媒体和其中所记录的数据。
- 软工：文档用来表示对活动、需求、过程或结果，进行描述、定义、规定、报告或认证的任何书面或图示的信息。

信息系统项目相关信息（文档）种类

yanbo@bit.edu.cn

- 非正式文档、正式文档；
- 项目周期：开发文档、产品文档、管理文档；
- 可研报告；项目开发计划；软件需求说明书；数据要求说明书；概要设计说明书；详细设计说明书；数据库设计说明书；用户手册；操作手册；模块开发卷宗；测试计划；测试分析报告；开发进度月报；项目开发总结报告；

文档管理的规则与办法

yanbo@bit.edu.cn

1 文档书写规范；
2 图表编号规则；
3 文档目录编写标准；
4 文档管理制度；

配置管理

yanbo@bit.edu.cn

- 通过技术与行政的手段对产品及其开发过程和生命周期进行控制、规范的一系列措施和过程。
- 对变更加以控制
- PMBOK/ISO9000/CMM的重要组成部分

软件配置管理

yanbo@bit.edu.cn

- 软件配置管理（Software Configuration Management），又称软件形态管理、或软件构建管理，简称（SCM）。界定软件的组成项目，对每个项目的变更进行管控（版本控制），并维护不同项目之间的版本关联，以使软件在开发过程中任一时间的内容都可以被追溯，包括某几个具有重要意义的数个组合。
- 贯穿于整个软件生命周期，它为软件研发提供了一套管理办法和活动原则。软件配置管理无论是对于软件企业管理人员还是研发人员都有着重要的意义。

软件配置管理

yanbo@bit.edu.cn

- 三个方面的内容：
  - **VersionControl-**版本控制
  - **ChangeControl-**变更控制
  - **ProcessSupport-**过程支持
- 关键活动包括：配置项、工作空间管理、版本控制、变更控制、状态报告、配置审计等。

软件配置管理

- 常用的软件配置管理工具主要分为三个级别：
  - Rational ClearCase，Perforce，CA CCC/Havest
  - Merant PVCS
  - Microsoft VSS，CVS
- 常用的开源免费的软件配置管理工具有：SVN、GIT、CVS。

配置的概念源于硬件

yanbo@bit.edu.cn

C5

C1

C2

C3

C4

List of Parts

C1	Tire
C2	Tire
C3	Tire
C4	Tire
C5	Engine

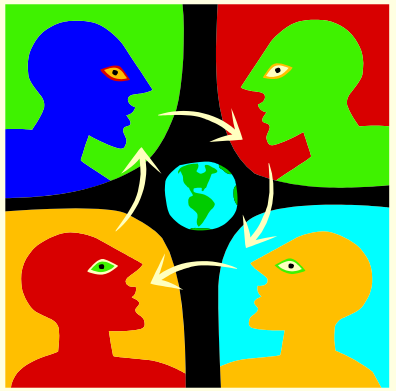
问题：怎样组装和卸装一辆汽车？

问题: 制造和维护产品需要做什么？

软件配置管理的发展

yanbo@bit.edu.cn

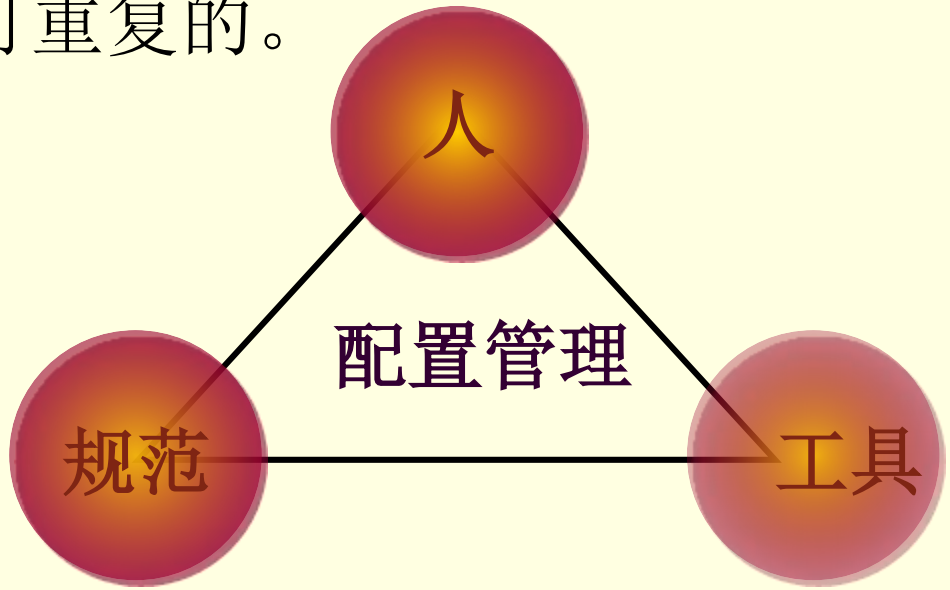
- 配置管理最早在美国的国防工业中被提出。
- 1962年，美国空军发表了有关配置管理的标准-AFSCM375-1。这是第一个配置管理的标准。
- 随着计算机程序越来越复杂和难于管理，软件项目团队越来越大和分布广泛，SCM的概念被大多数软件组织接受和实施。



软件配置管理的概念

yanbo@bit.edu.cn

- 软件配置管理SCM，通过一套工程规范，在整个软件生命周期中跟踪、记录软件，保证全部变更都记录在案，并保证软件的当前状态是已知的和可重复的。



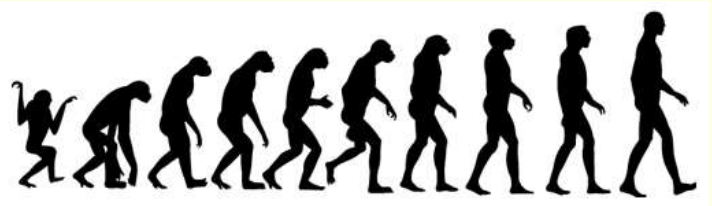
软件配置管理（SCM）

yanbo@bit.edu.cn

- 指一套管理软件开发和软件维护以及各种中间软件产品的方法和规则。

——————

- 记录软件产品的演化过程。
- 确保软件开发者在软件生命周期中的各个阶段都能得到精确的产品配置。
- 最终保证软件产品的**完整性、一致性、追朔性、可控性**。



软件配置管理（SCM）

yanbo@bit.edu.cn

- 配置：源程序、数据文件、设计文档、用户文档，及其组织关系。
- 管理这些部件的产生、修改、提取、发布，以保证产品的正确性/完整性/一致性。

如果没有软件配置管理，将有什么坏处？

- 最大的麻烦是工作成果被覆盖。如果不采用配置管理软件来保存历史版本，在同一个文件上修改内容，保存之后，新的内容覆盖了老的内容。
  - 多数情况下新的内容比老的内容好，覆盖了也没关系。但是总有不少意外，新程序是错误的，而老程序却是对的，无法恢复。
  - 为了避免成果被覆盖，很多人采用最原始的手工管理版本的方式，例如给文件加后缀“-01”、“-02”以表示版本。天长日久，工作目录下就会有一堆带数字后缀的文件，而且也忘记了数字后缀代表什么内容，管理起来非常麻烦。

使用软件配置管理，将有什么好处？

yanbo@bit.edu.cn

- 最直接的好处是工作成果的所有版本都被保留，不会丢失也不会被覆盖。
  - 如今硬盘存储空间价格低廉，用于保存历史版本的存储空间的成本可以忽略不计。如果你保存了工作成果的100个历史版本，哪怕99版本都是“垃圾”，只有一个版本里有“黄金”，那也值了。
  - 间接的好处是，项目的所有工作成果被完整地保留下来，这是企业的知识财富，可以被人们很好地分享利用。而且减少了人员辞职造成的损失。

变更

yanbo@bit.edu.cn

- 软件项目进行中面临的一个主要问题是持续不断的变化。
- 有效的项目管理能够控制变化，以最有效的手段应对变化, 不断命中移动的目标。

配置管理的商业理念

- 企业的商业需求决定了配置管理的力度，不必追求完美无缺的配置管理，而是让开发团队“恰好够用”就行，并将为配置管理所付出的代价控制在预算之内。
- 富有成效的配置管理的特征：
  - 任何项目成员都要对其工作成果进行配置管理，应当养成良好的习惯。不必付出过多的精力，最低要求是保证重要工作成果不发生混乱。
  - 配置管理规范应当清晰明了，便于执行，不必在细节方面要求太多，不给项目人员添加过多的负担，不使人厌烦。
  - 选择配置管理工具应当综合考虑价格、易用性和功能因素，而不是购买最先进的工具。令人满意的工具通常是价格低廉、简便易用、功能恰好够用。

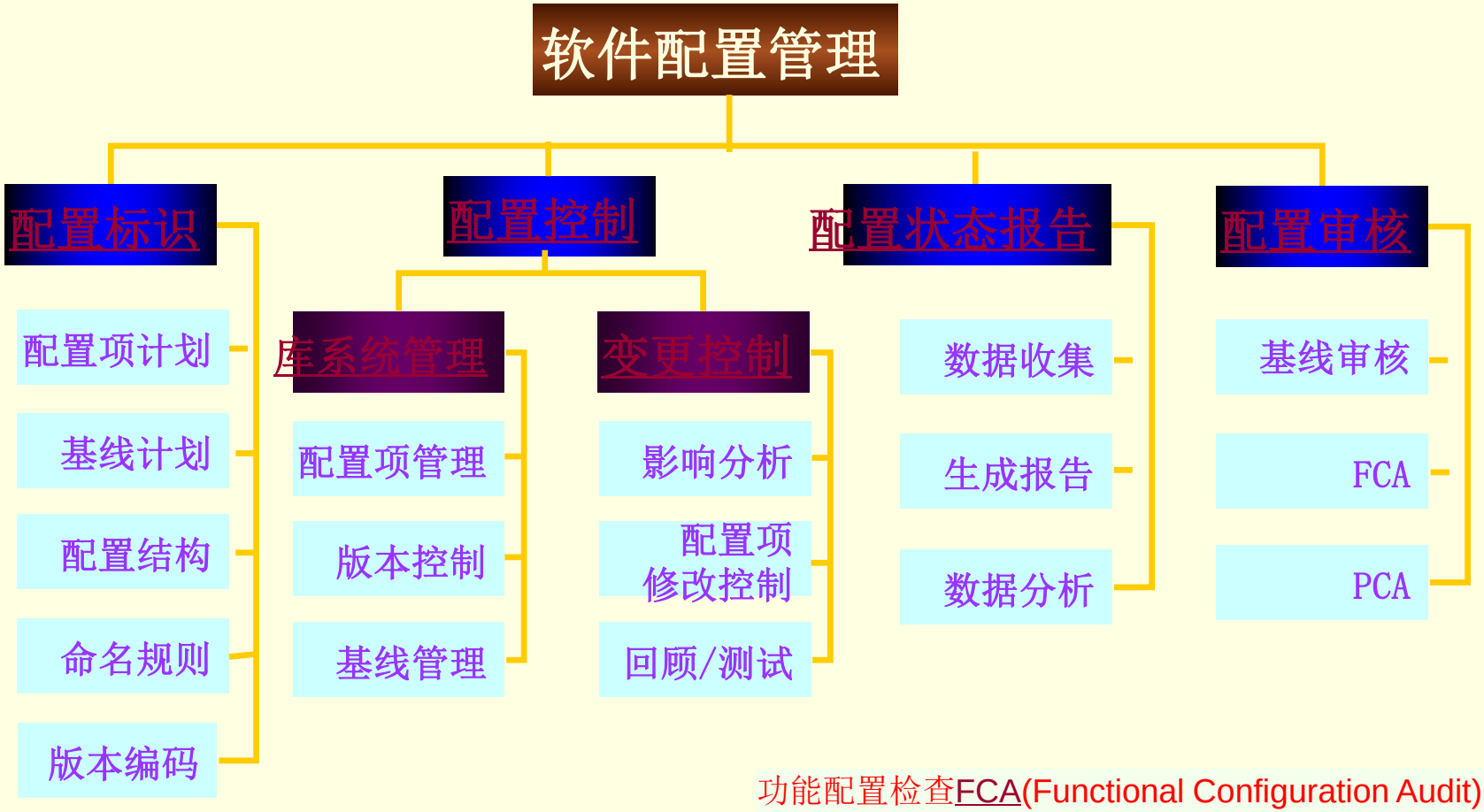
配置管理的商业理念

yanbo@bit.edu.cn

- **CMM/CMMI对配置管理过程域论述得十分清楚详细，假设完全按照CMM/CMMI的要求执行的话，你可以得到100分（满分）的配置管理成绩。**
  - **出于商业利益考虑，我们不向往100分的成绩，因为代价太高了。更愿意付出前者的30%左右代价获取60－70分（及格）的成绩，这样最划算。**
  - **70－100分的配置管理成绩对于大部分商业软件而言没有多少意义，那属于锦上添花，如果没有足够的精力的话，那么就以最低的代价达到及格分数就行了。**

软件配置管理的功能表

yanbo@bit.edu.cn



功能配置检查FCA(Functional Configuration Audit)
物理配置检查PCA(Physical Configuration Audit)

软件配置项

yanbo@bit.edu.cn

- 配置项
  - 产品配置项是指一个产品在其生命周期各个阶段所产生的各种形式和各种版本的文档、计算机程序、部件以及数据的集合。该集合中每一个元素成为该产品的一个配置项。**Configuration Item**
    - 属于该产品组成部分的工作成果；
    - 属于项目管理和机构支撑过程域产生的文档；

配置项与基线

yanbo@bit.edu.cn

- 配置项（Configuration Item）
  - 配置项是处于配置管理之下的软件或硬件的集合体。这个集合体在配置管理过程中作为一个实体出现。例如：项目计划，软件配置管理计划，设计文档，源代码，测试数据，项目数据，用户手册，等等。
- 基线（Baseline）
  - 已经通过正式复审和批准的某规约或产品，它因此可以作为进一步开发的基础，并且只能通过正式的变更控制过程来改变。

配置识别

yanbo@bit.edu.cn

- 识别将置于配置管理之下的配置项和有关的工作产品。识别产品的结构、产品的构件及其类型，并为其分配唯一的标识符。提供存取控制。同时找出需要跟踪管理的中间产品，并维护其关系。

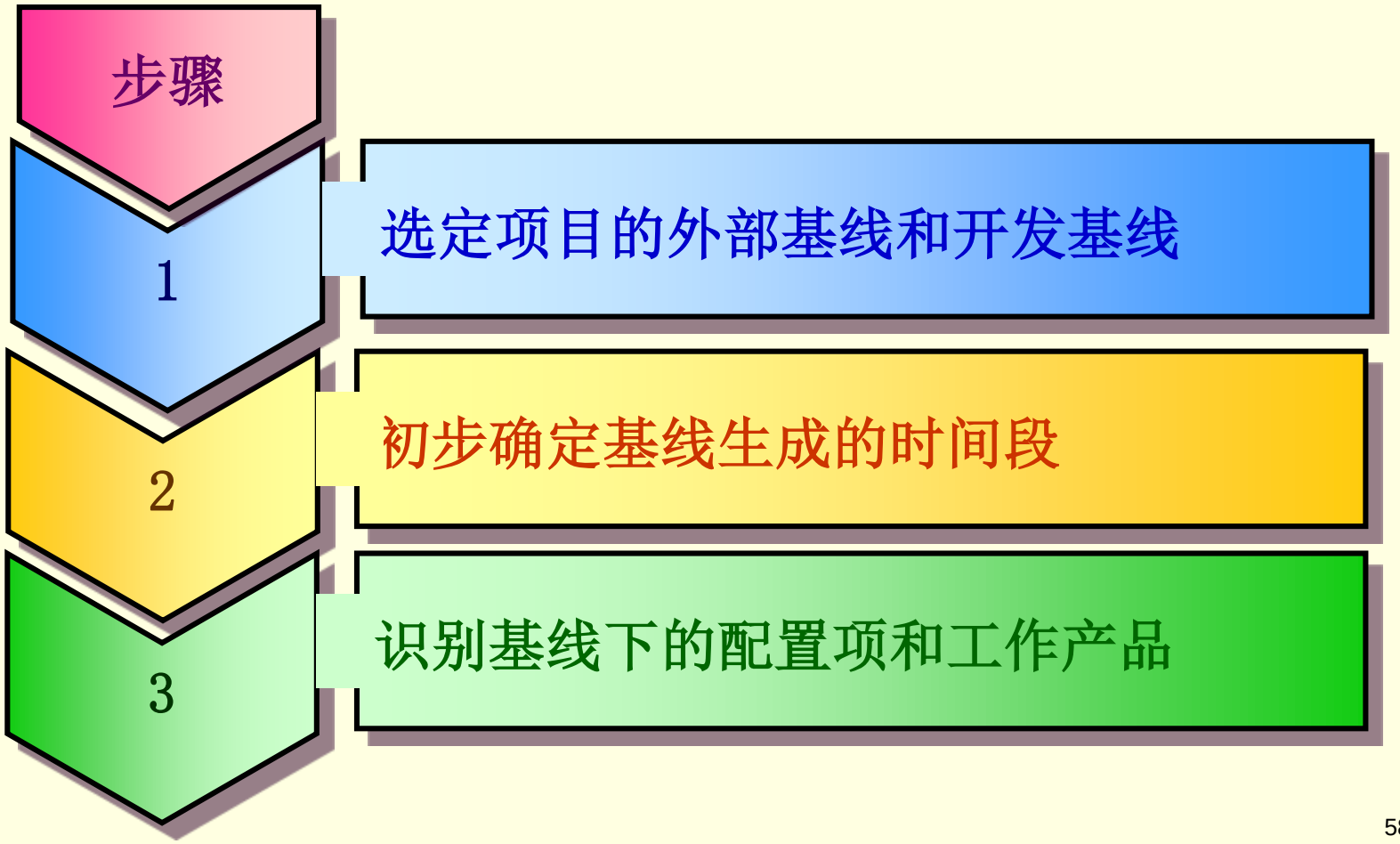
配置项标识

- 识别产品的结构、产品的构件及其类型，并为其分配唯一的标识符。提供存取控制。同时找出需要跟踪管理的中间产品，并维护其关系。

yanbo@bit.edu.cn

定义项目生存周期中的基线

yanbo@bit.edu.cn



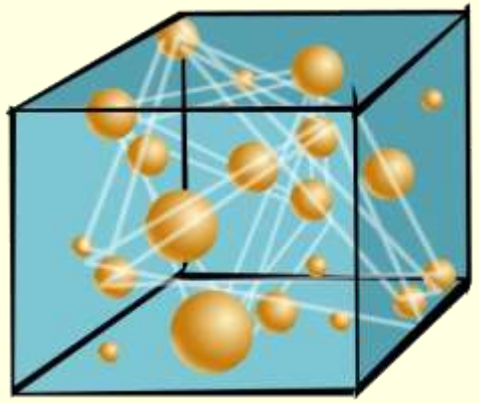
项目基线

基线	描述说明
功能基线	经评审和批准的文档，描述了系统的功能特性和证实这些规定的功能已实现所要求的验证。
指派基线	描述待开发软件所要实现功能的文档(又称需求基线)
开发配置	定义在软件开发进程中不断演化和累积的配置。 （描述了在设计，编码和测试任一阶段的配置。）
产品基线	最初批准的描述配置项全部必要的功能和物理特性的文档； 经产品验收测试验证所选择具有的功能和物理特性 对软件，包括电子媒体上的软件代码和确保代码能再生和维护所要求的其它项（如，软件工具及文档）。

配置控制

yanbo@bit.edu.cn

- 在配置项的配置标识和基线正式确立之后，对其更改进行系统管制的过程。
- 变更控制系统记录每次变更的相关信息（变更的原因、变更的实施者以及变更的内容等）。
- 查看这些记录信息，有助于追踪出现的各种问题。记录正在执行的变更的信息，有助于做出正确的管理决策。



变更管理

- “变更”是信息系统的最普遍的特点。
- 项目变更是指在信息系统项目的实施过程中，由于项目环境或其他原因而对项目产品的功能、性能、架构、技术指标、集成方法、项目的 范围基准、进度基准和成本基准等方面做出的改变。
- 配置管理的主要任务是对变更加以有效控制和管理，防止软件在多变的情况下失控。

项目变更的不可避免性

yanbo@bit.edu.cn

- 用户
- 开发人员

- 无序变更可能导致的后果：
  - 基准失效；项目干系人冲突；资源浪费；项目执行情况混乱；

配置库

yanbo@bit.edu.cn

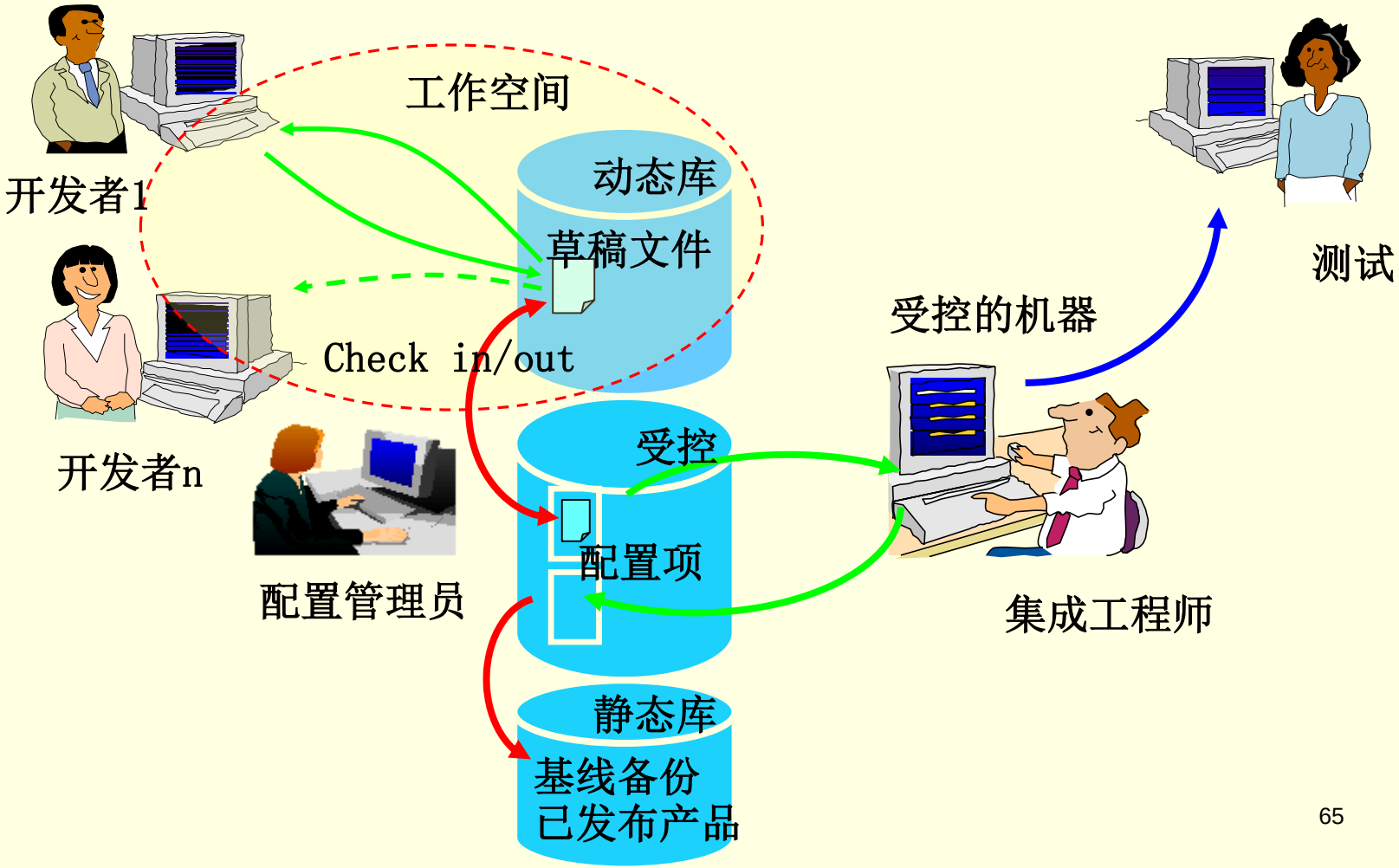
- **Configuration Library**
- 作用：
  - 记录与配置相关的信息；
  - 利用库中信息评价变更后果；
  - 从库中提取配置管理过程的管理信息；

关于软件配置库的概念

- 动态库（开发库、程序员库、工作库）
  - 开发周期的某个阶段，存放与该阶段工作有关系的信息
- 受控库（主库、系统库）
  - 开发周期的某个阶段结束时，存放做为该阶段产品及其相关的信息，配置管理对其中的信息进行管理，也称配置库
- 静态库（软件仓库、软件产品库）
  - 存放最终产品的软件库
- 备份库

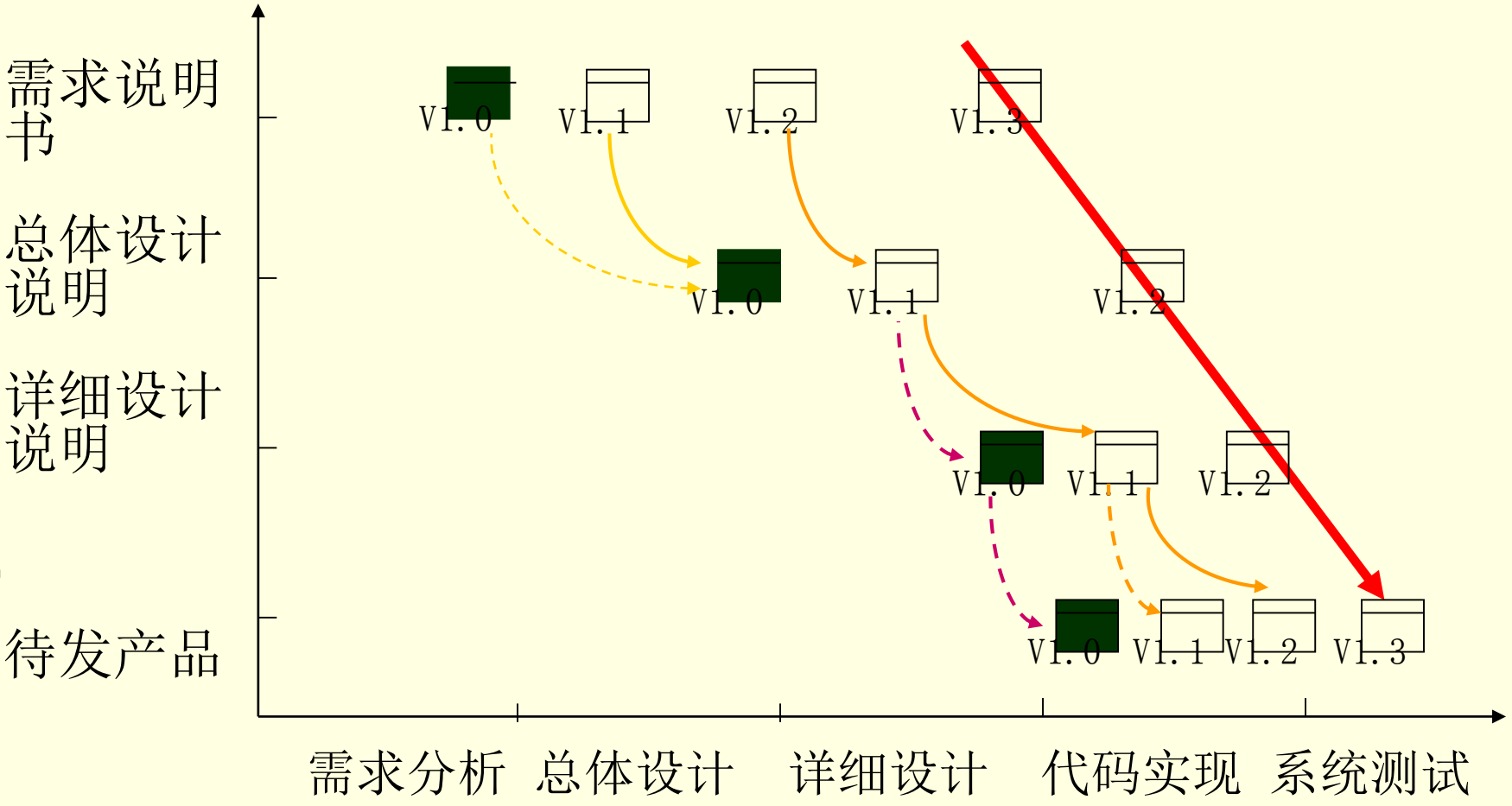
建立配置管理的库系统

yanbo@bit.edu.cn



基线的演进

yanbo@bit.edu.cn

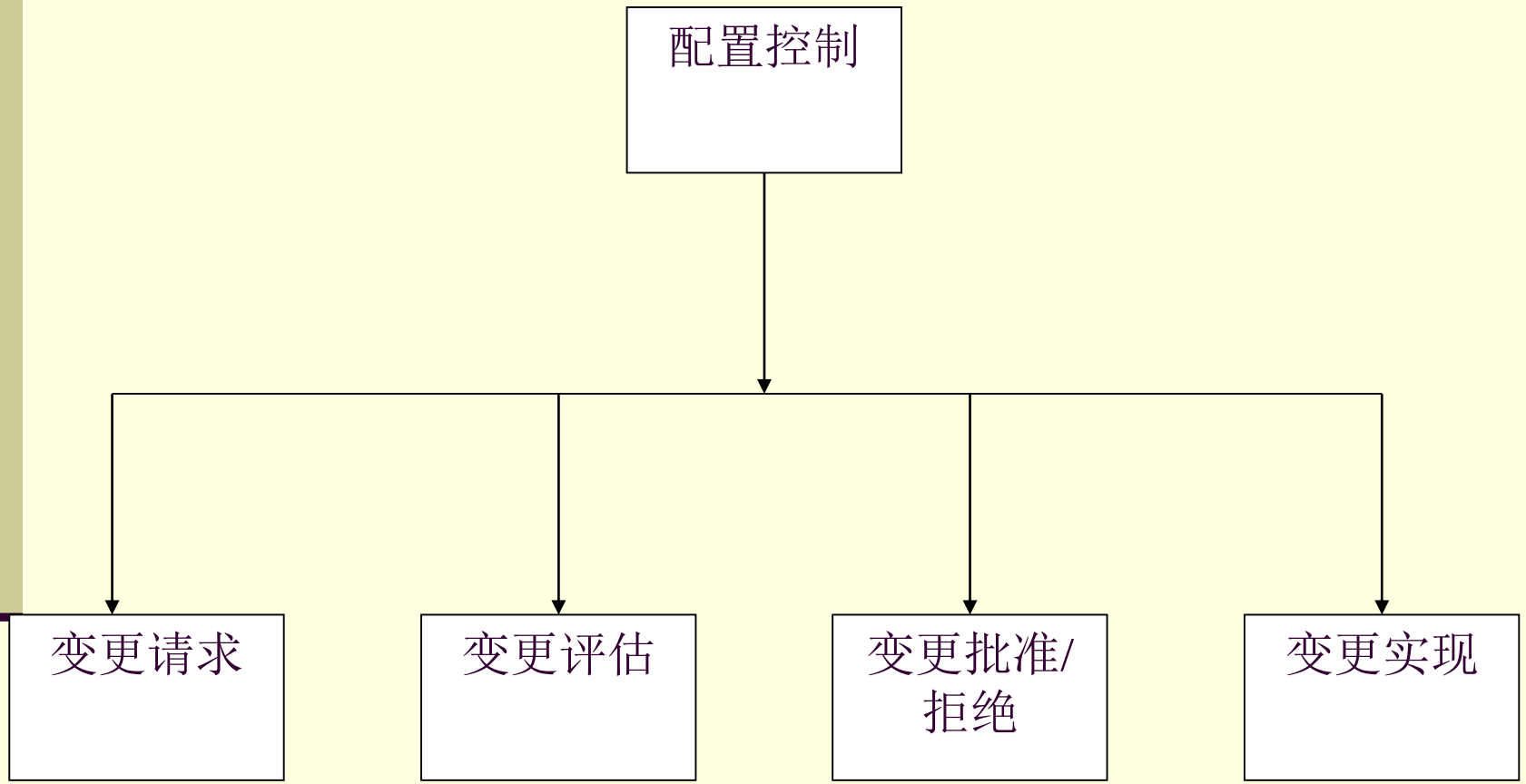


变更控制 (Change Control)

- 配置控制通过建立一个有序的变更控制过程来确保。
  - 对每项变更的影响给于适当的考虑；
  - 对任何基线化的配置项的更改经过批准；
  - 批准的更改得以实施；
  - 记录必要的测量信息。

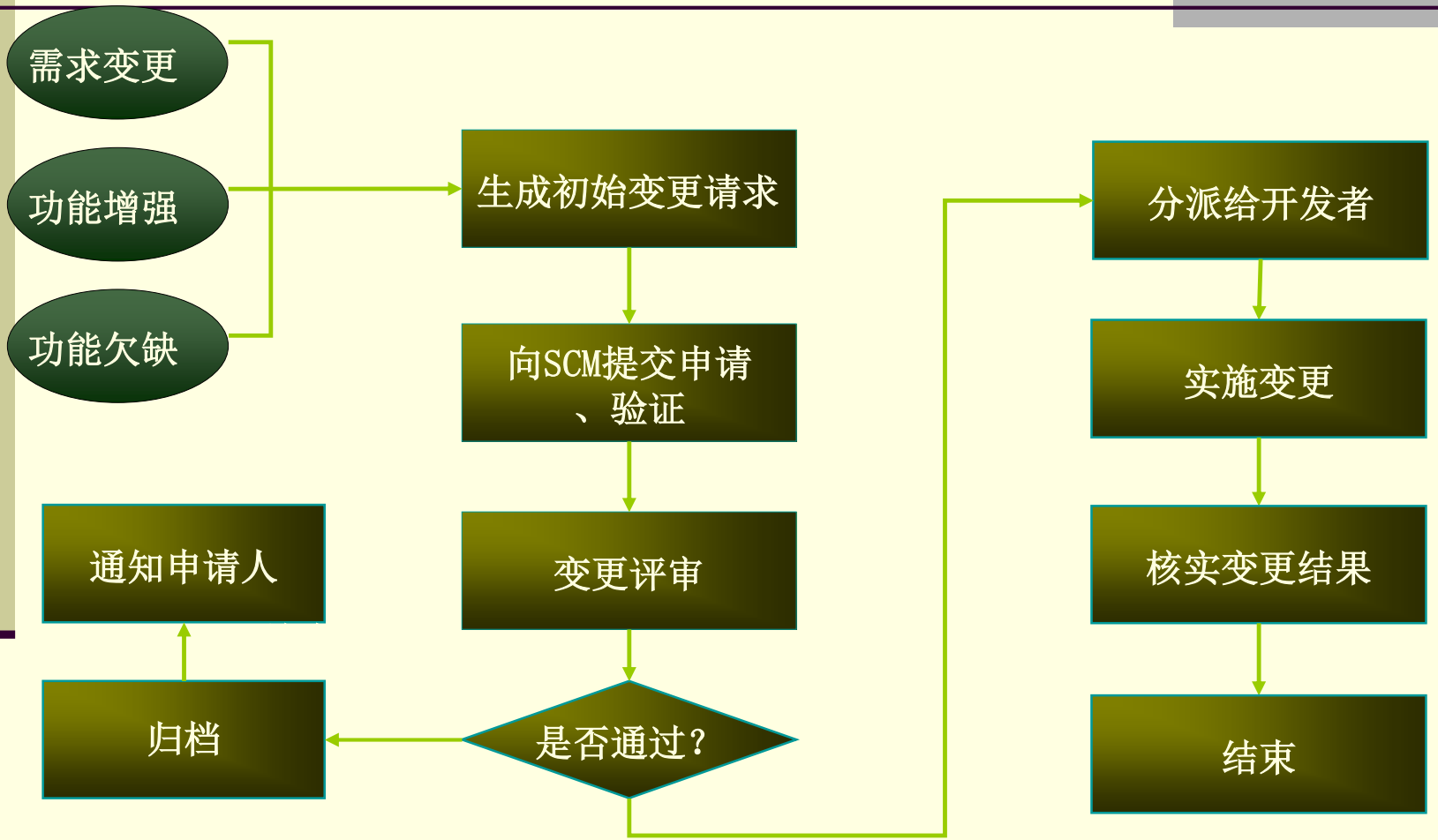
基线变更系统

yanbo@bit.edu.cn



变更控制过程

yanbo@bit.edu.cn



提出变更

- 识别变更需要，对受控的配置项的修改提出一个**变更请求(Change Request - CR)**。
  - 变更请求－*对软件变更需要的描述*。
  - 变更控制过程通过**CR**的流动来实现。
  - 对基线配置项的任何修改都必须与某一**CR**相关。

变更请求表单

yanbo@bit.edu.cn

采购变更请求	*变更申请ID: 021*

变更配置项: 采购清单 **版本: v1.2**
变更描述: 增加采购项
变更原因: 由于项目的进度原因，需要增加采购项
优先级: 2 **版本所有人:** 施铮 **日期:** 2005－5－5

变更 通过 **责任人签字:** 王卫红 **日期:** 2005－5－6
变更所费时间 (以天记): －10
评价: 为了节约时间计，增加采购项，应避免因采购引起风险导致进度拖延

变更执行记录: 采购清单 **版本: v1.2**
变更实施描述 (或评论):变更合理，予以批准

实施人: 刘波 **日期:** 2005－5－6
实施验收责任人签字: 王卫红 **日期:** 2005－5－6

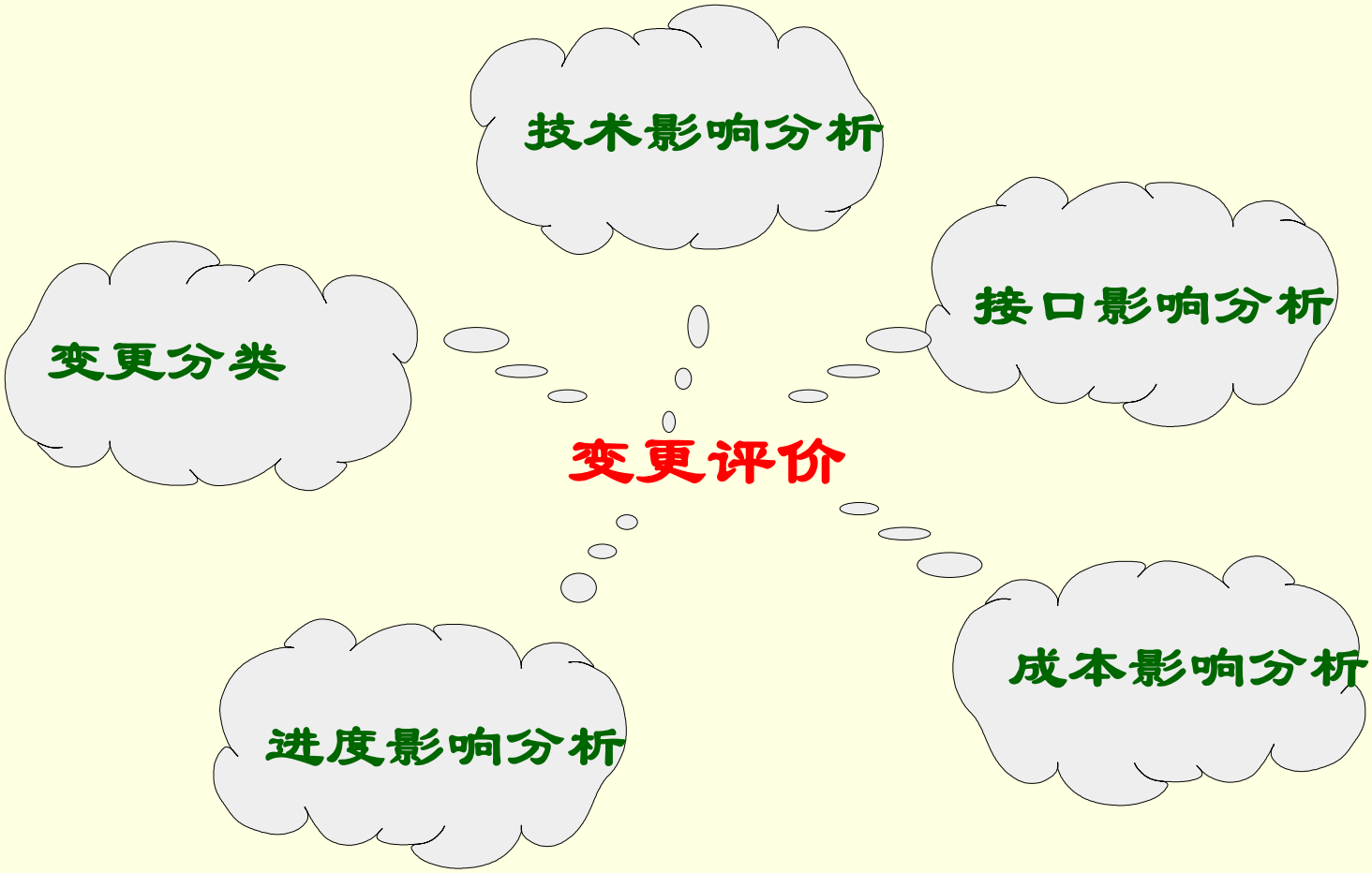
接受变更

- 项目必须建立接收提交的变更请求并进行跟踪的机制。
  - 指定接收和处理变更请求的责任人；
  - 确认变更请求；
    - 检查变更请求的内容是否清晰，完整，正确；
    - 包括：已存在的重复请求，或误解。
  - 对变更请求赋予唯一的标识符；
  - 建立变更跟踪纪录。

yanbo@bit.edu.cn

变更评价

yanbo@bit.edu.cn



变更处置

yanbo@bit.edu.cn

变更请求及附件提交SCCB进行评审并决策。

处置结果	说　明
批　准	指派实施；指派验证； 更新版本；指派发布；
不批准	请求的变更没有必要； 不可行； 有更好的替代解决；
推迟决定	要求提供附加信息； 返回评价，要求给予进一步的分析； 等待必要的内或外协商；

变更处置

- 对每一项批准的变更请求，由SCCB指定变更的完成日期(due date)
  - 尽快；
  - 给出明确的实施期限；
  - 要求得到相应配置项的明确的版本。

变更实施

- 项目(软件)经理负责管理已批准变更请求的实施；
- 软件经理
  - 标识工作包 **(work packages)**；
  - 安排工作进程和实施责任人；
  - 监控进度和质量；
- 实施责任人
  - 检出 **(check out)** 变更项；
  - 实施更改并记录更改信息；
  - 提交验证。

变更验证

- 对已实施的变更必须在配置项/单元不同层次上加以验证。
- 验证包括：
  - 审查、(同级)评审或走查(inspection, peer review or walkthrough)；
  - 重新运行测试计划中规定的测试；
  - 或对测试计划增添相应的附加内容；
  - 进行回归测试。
- 验证实施后，验证组织提交验证结果及必要的证据；
- 将通过验证的配置项检入*(check in)*受控库，记录配置信息。

结束变更

yanbo@bit.edu.cn

- 结束变更的准则：
  - 经验证表明变更已正确的实施；
  - 变更未产生非预期的副作用；
  - 有关的代码、文档和数据项已全部更新并已纳入受控库。
- 配置管理员职责
  - 必要时将原基线备档，建立新的基线；
  - 完成配置记录；
  - 关闭CR，并通知变更提请人(originator)；

配置状态报告

yanbo@bit.edu.cn

- **Configuration Status Reporting**
- 有效的记录和报告管理配置所需要的信息，目的是及时、准确的给出配置项的当前状况，供相关人员了解，以加强配置管理工作。

陷入泥潭的项目

案例：

某软件公司为资金雄厚的电力公司开发一套电网计费系统，该公司要求系统分两期完成。开发公司签订一期合同后即组建项目组开始软件开发。项目组采用瀑布模型，预计经过系统分析、设计、编码、集成、测试，最后把合格产品交付给用户。该公司配备了简单的软件配置管理系统，能进行版本管理，但不能对整个过程实现系统化支持。

陷入泥潭的项目

项目启动后，开发人员到用户现场进行调研，搜集了系统的基本需求。但是由于涉及的系统和用户角色较多，短时间内难以全面了解系统需求。分析人员本着边做边看的想法，将基本需求描述下来交给设计人员，期望需求文档中的一些疑点在后续过程中得到补充和完善。设计人员针对需求形成设计文档，并交给编程人员编码。与此同时，分析和设计人员手中都有各自生成的文档，并在编码阶段进行了多次修改。

陷入泥潭的项目

没有人能确定这些修改是否及时传递给了相关人员。编程人员加班按照设计文档进行编码，保证模块编译通过，在自己的实验环境中基本能运行后，将代码检入到代码库中。

全部模块编码完成后，本来预计2周的集成联调花费了2个多月才完成。原因是系统输入数据的格式和模块接口在编码过程中发生了改变，许多模块要大量改写，部分开发人员在改写时用错了代码版本，引起严重的代码错误。

陷入泥潭的项目

联调成功后，留给测试的时间不多了，测试人员简单进行基本功能测试后，由PM将第一期产品按合同规定日期交付用户。

交付后，由于问题太多，项目组忙于应付用户上报的产品缺陷，而其中有些缺陷应归于第二期的功能需求。项目组卷入无休止的功能细节的讨价还价会议中，项目组内部也为缺陷产生原因争论、推诿。由于很多严重错误得不到解决，用户扬言第一期工程问题如果延期半年

陷入泥潭的项目

yanbo@bit.edu.cn

得不到解决将不签第二期合同，并且不再委托该公司任何其他项目。在内忧外困下，很多开发人员感觉压力大、前途渺茫、工作效率低下、工作状态差，纷纷离职。项目陷入内外交困的泥潭中。

缺乏完善的CM系统对项目的影响

1. 不能准确界定项目需求；

   需求的不断变化；需求基线建立困难；

2. 不能有效管理源文件版本；

   前后不一致；

3. 多版本并行开发困难；

4. 无法有效的管理和跟踪变更；

缺乏完善的CM系统对项目的影响

yanbo@bit.edu.cn

5. 不能及时了解项目的进展状况；

项目随意性大；PM无法及时了解开发过程；

6. 无法开展规范化的测试工作；

开发和测试人员无法顺利的分工与合作；

7. 缺乏客观的产品交付依据；

缺乏需求-设计-编码-测试的可跟踪性。

缺乏完善的CM系统对项目的影响

8. 对软件版本的发布缺乏有效的管理；

产品发布时无法标注版本；客户特定问题版本无法重现。

9. 软件复用率低下；

共用组件难以维护；

10. 不能有效的促进开发团队和开发过程的提高；

无法动态收集开发过程的数据。

ClearCase/ClearQuest

yanbo@bit.edu.cn

动变更集；

ClearCase/ClearQuest

- ClearCase UCM模型
  - 标识工件，并将工件存入安全的版本库中；
  - 控制并记录对工件的变更；
  - 保持稳定、一致的工作空间；
  - 支持工件组件的并行开发；
  - 及早集成、经常集成；
  - 保证软件**Build**可重现；
  - 记录记并追踪变更请求；

ClearCase UCM模型

yanbo@bit.edu.cn

面向特定客户技术规格

BUG报告功能

新功能追加需求

新平台支持需求

统一变更管理流程

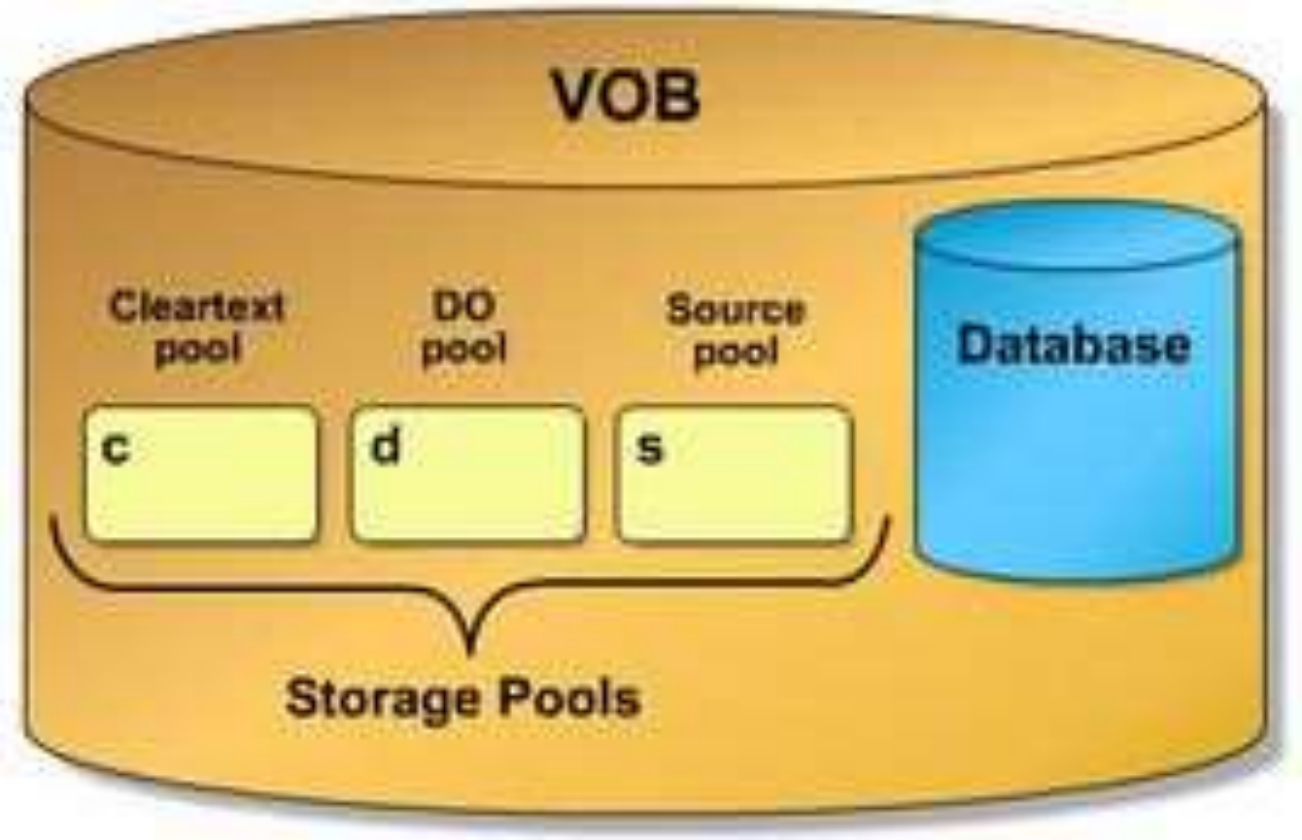
变更后的系统

UCM中项目和相关对象

yanbo@bit.edu.cn

PVOB—项目—流—活动

VOB：



字储

系统

元，

版本

所

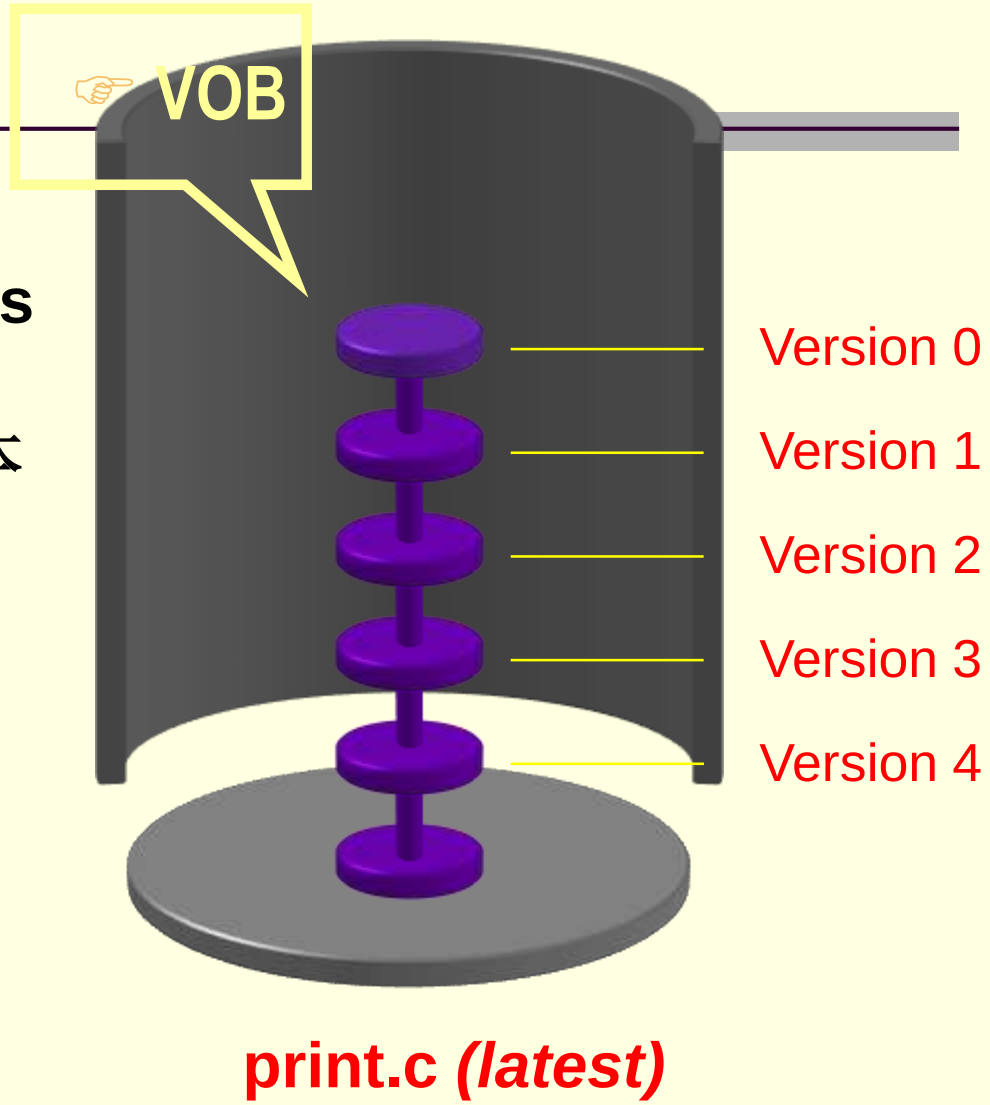
对，

VOB基本概念

yanbo@bit.edu.cn

VOB 存储库
---Version Object Bases
作用
---存储所有历史上的版本
---防止随意变更
存储内容

- 需求
- 模型
- 二进制文件和源代码
- 测试脚本
- **Bitmaps & JPEGs**
- **Html 文件CGI 脚本**



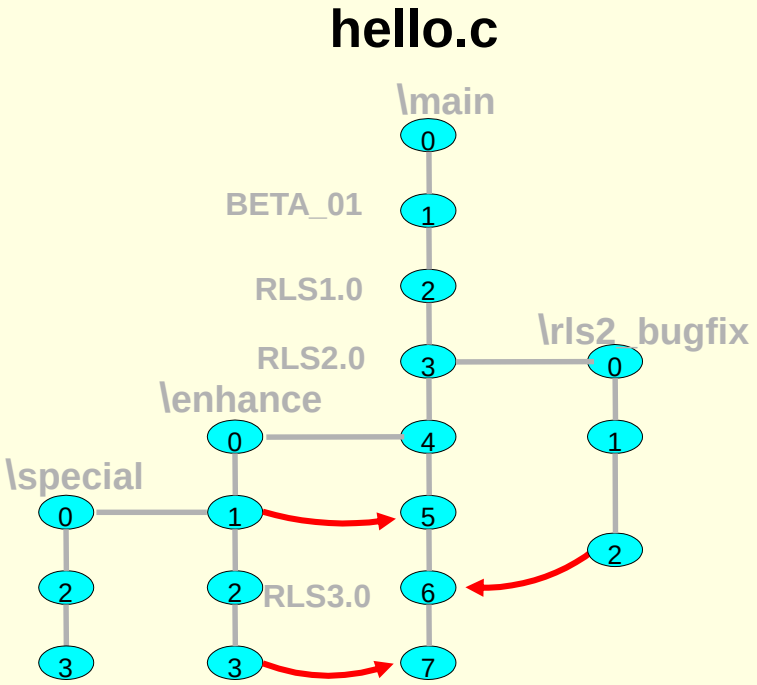
元素

元素是受ClearCase控制并包括一组版本信息的对象；

任何文件系统中存放的对象都可以作为一个元素放在ClearCase系统中进行版本控制；

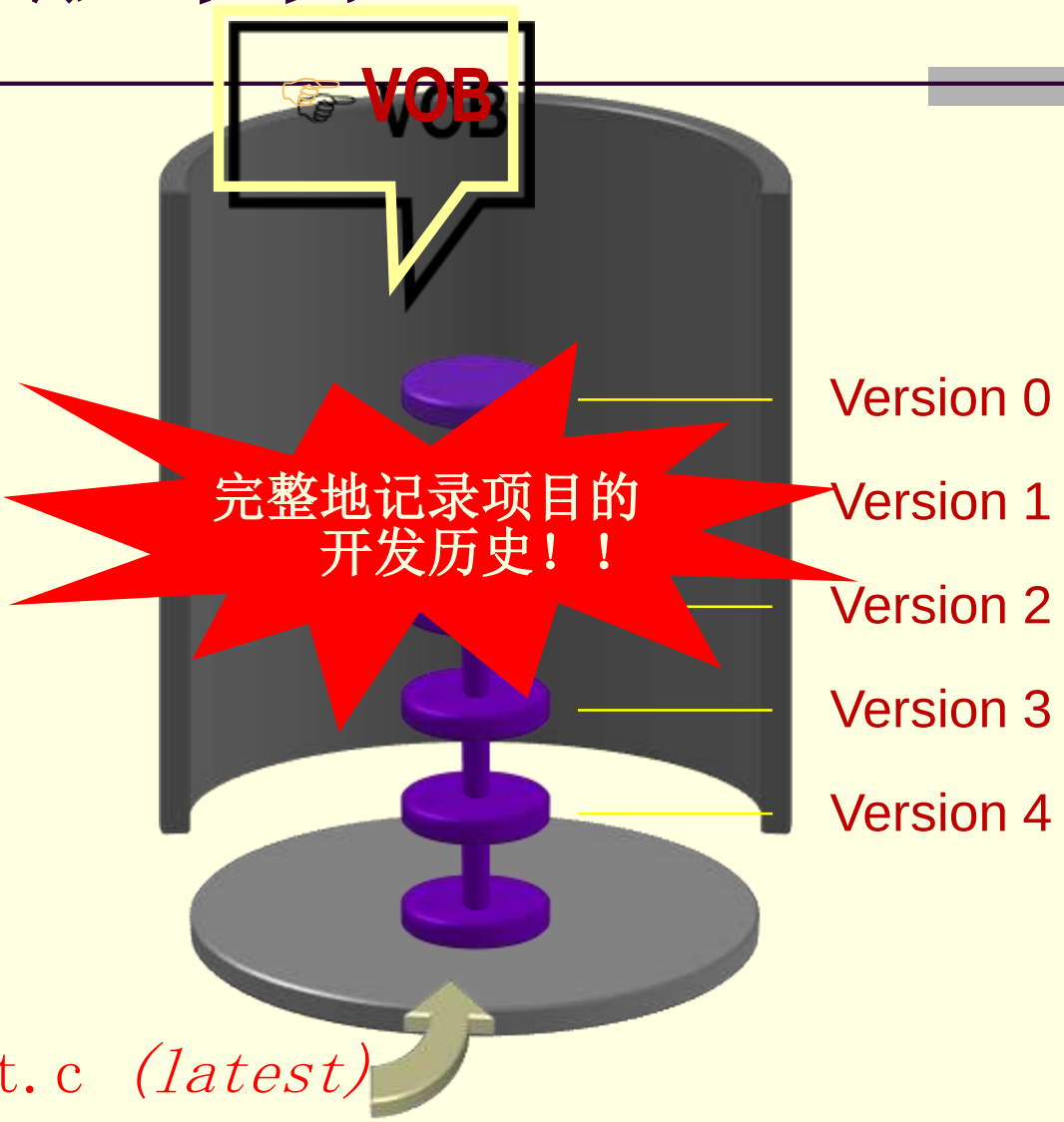
例如:

- Source files
- Directories
- Binary files
- Object libraries
- Documents

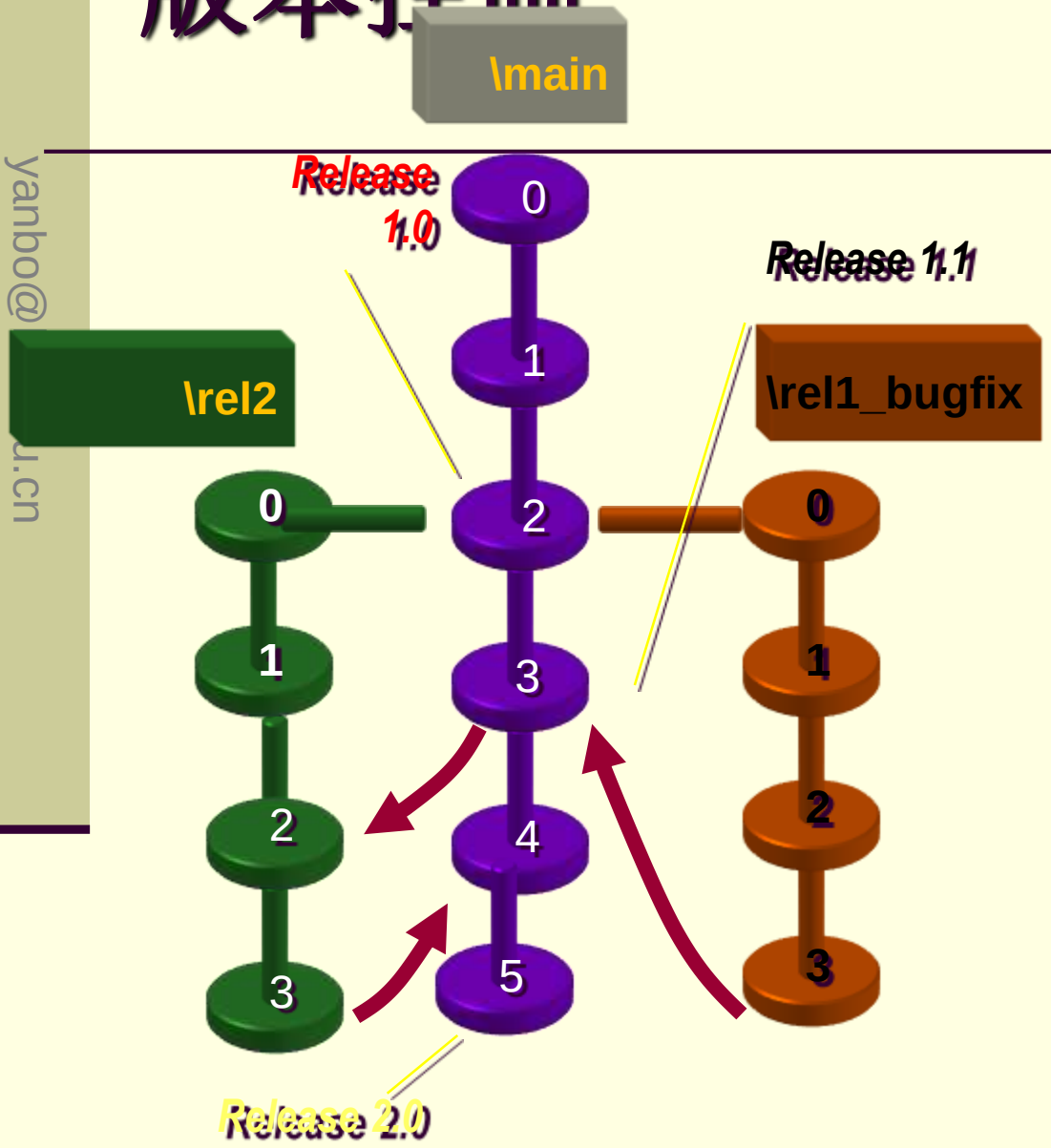


版本、版本树

yanbo@bit.edu.cn



版本控制

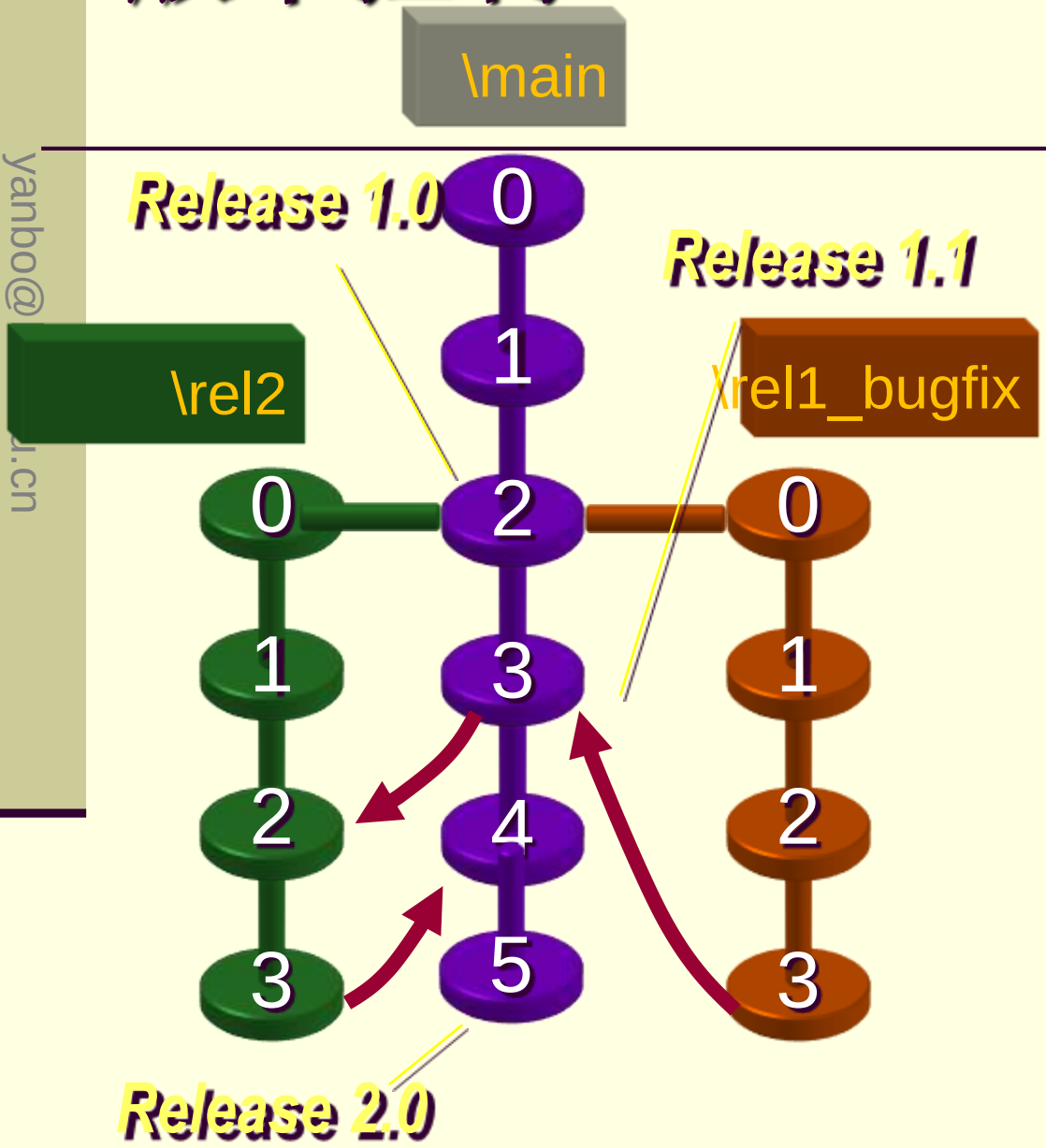


- 控制任何文件的版本
- 可对目录和子目录进行版本控制

- 分支和归并功能
- 采用版本树结构

- 文本比较
- 丰富的注释和版本报告信息

版本控制

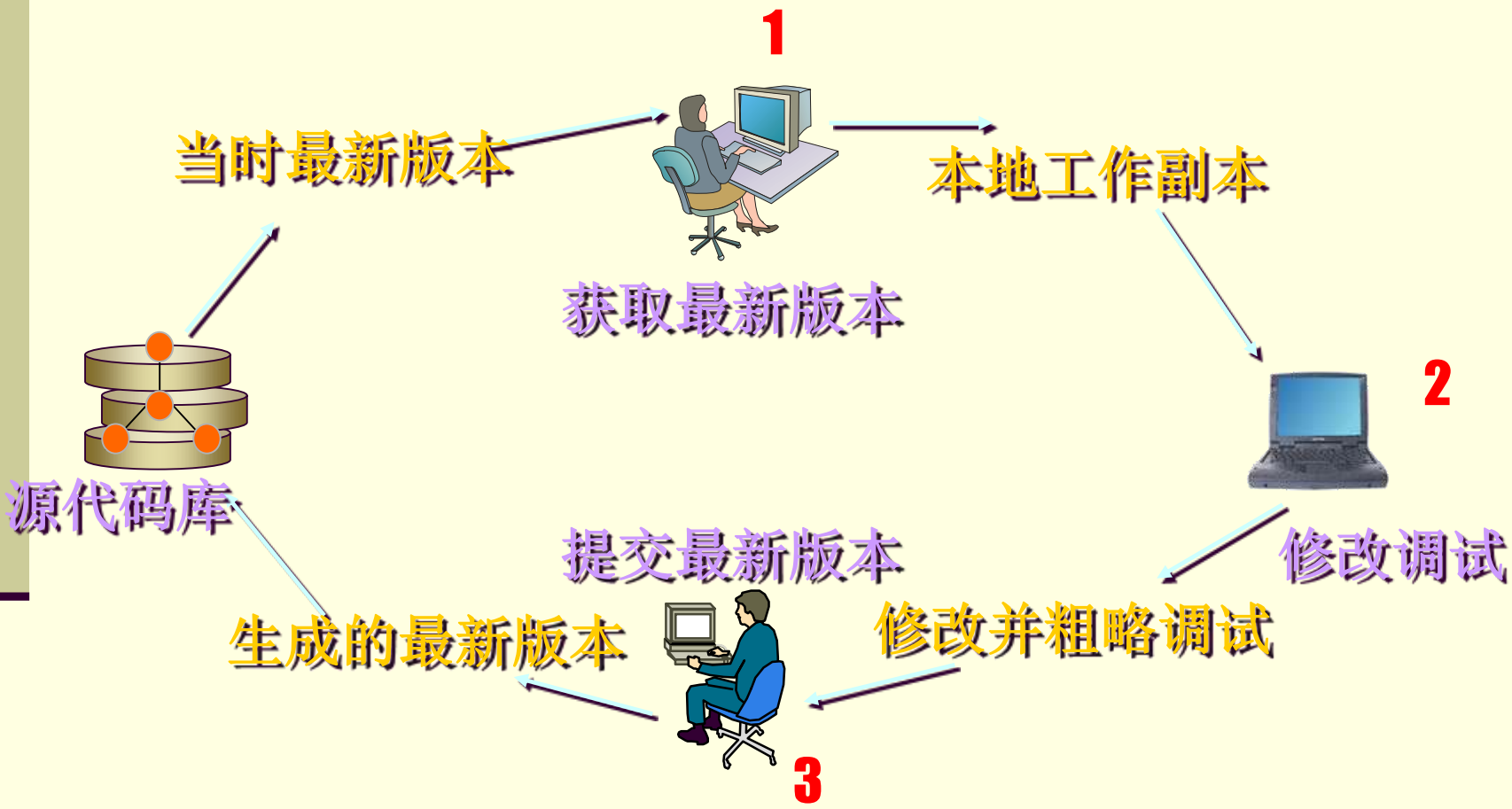


长方形表示一个分支；圆形表示检入的时间排序的版本号；箭头表示从一个分支到另一个分支的变更回归（归并）；“发布版本1.0/1.1”是这个版本上的标签。

目录是元素，也是版本对象。ClearCase对目录也进行版本管理。为了能在前一个版本中修复BUG，或者从新版本退回到旧版本，就有必要恢复一个旧的版本。目录被修改，在检入的时候，也要进行记录。还可以借助目录机制，重建或构造软件系统的前一个版本。

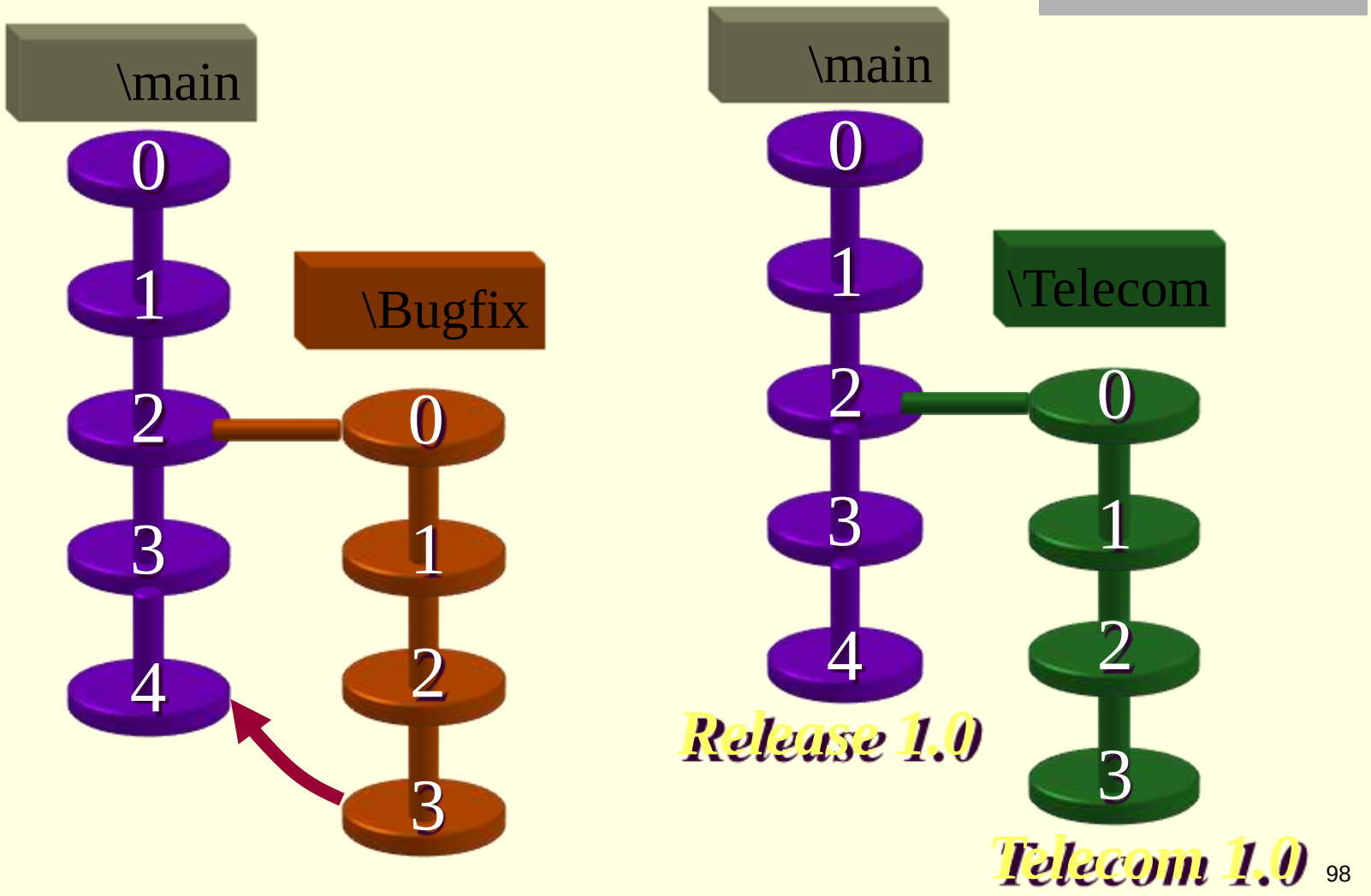
版本控制

yanbo@bit.edu.cn



并行开发的版本控制

yanbo@bit.edu.cn



ClearCase 并行开发

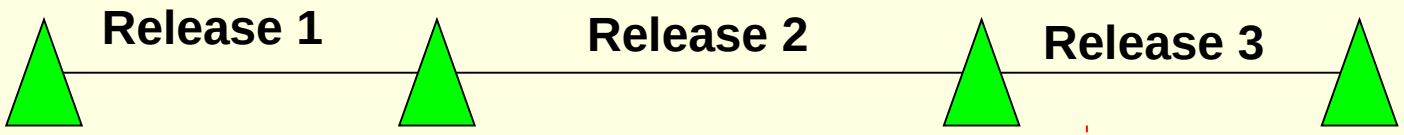
yanbo@bit.edu.cn

- 确保修改过的错误不会再出现；
  - 归并功能能够找到所有未归并的文件；
- 减少集成时间；
  - 对于一个项目可以节省几个月，对于一个建立(Build)可以节省几天；归并的工作能跨越多个存储库；
- 在多个平台上同时发布；
  - 归并功能可以同时归并32个文件；
- 避免传送一个缺陷的修改到多个版本；
  - 可以在任何方向上建立分支和进行归并；
- 不会拖延产品投放市场的时间；
  - 使高风险的工作隔离在单独的分支上；

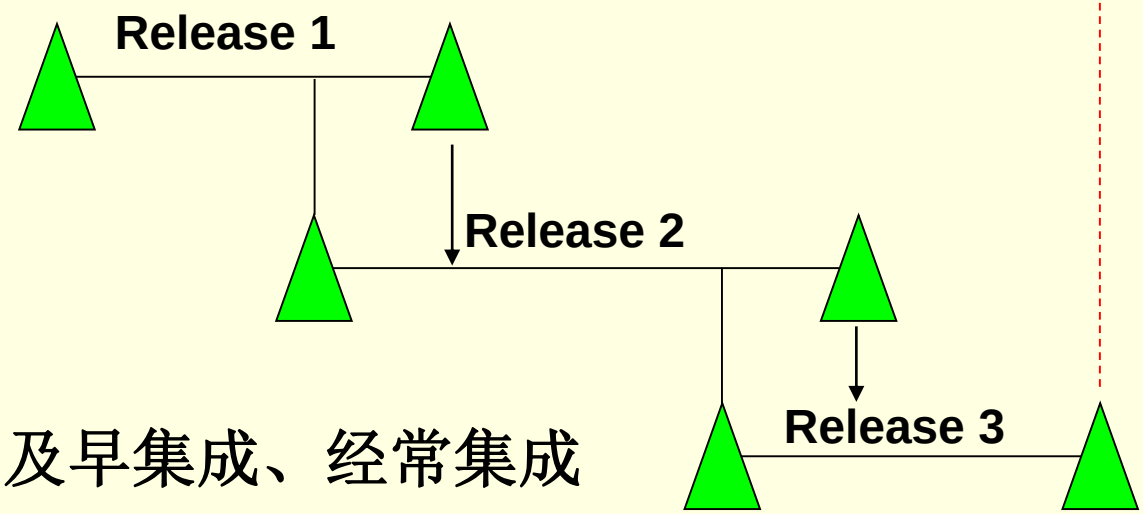
并行开发的必要性

yanbo@bit.edu.cn

串行开发



并行开发



及早集成、经常集成

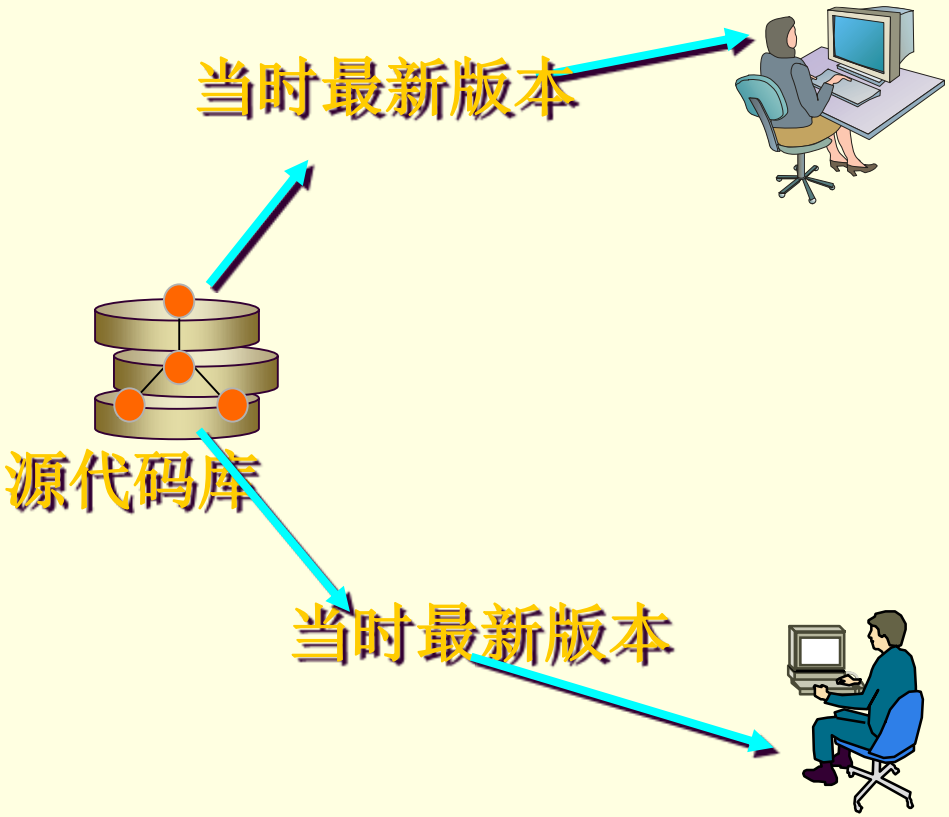
缩短产品上市期

思想-并行开发支持

- 为实现并行开发，配置管理系统需要提供灵活的分支机制和工作空间管理。
- 创建分支的过程实际上就是一个建立副本的过程，针对每个发布分别建立相应的分支，分支之间具备相对的独立性，这样不同的发布就可以在各自的分支上并行进行开发，在适当的时候，分支之间可以进行合并，从而实现将**Release 1** 中后期开发的功能合并到 **Release 2** 中。

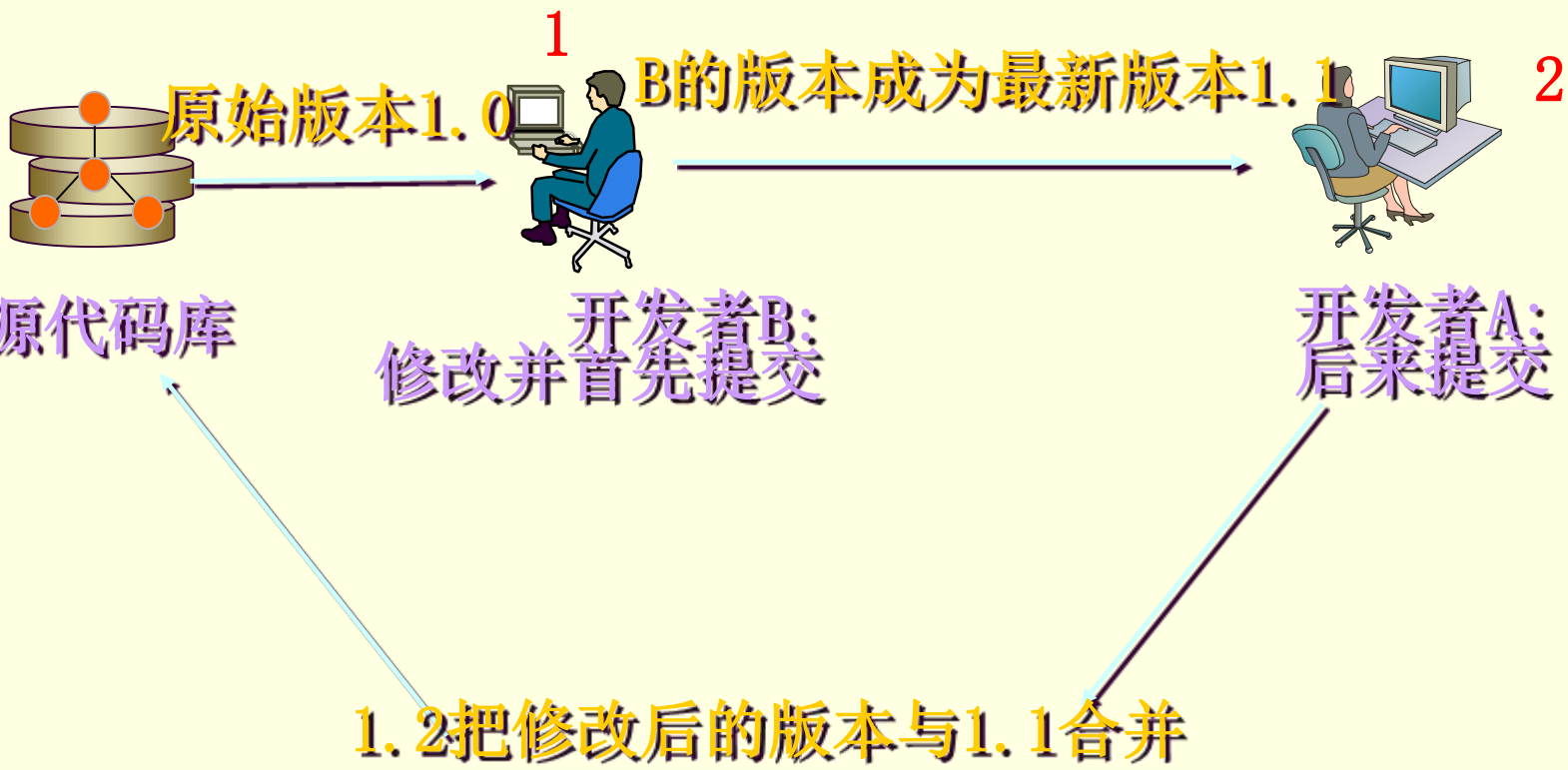
思想-并行开发中的冲突

yanbo@bit.edu.cn



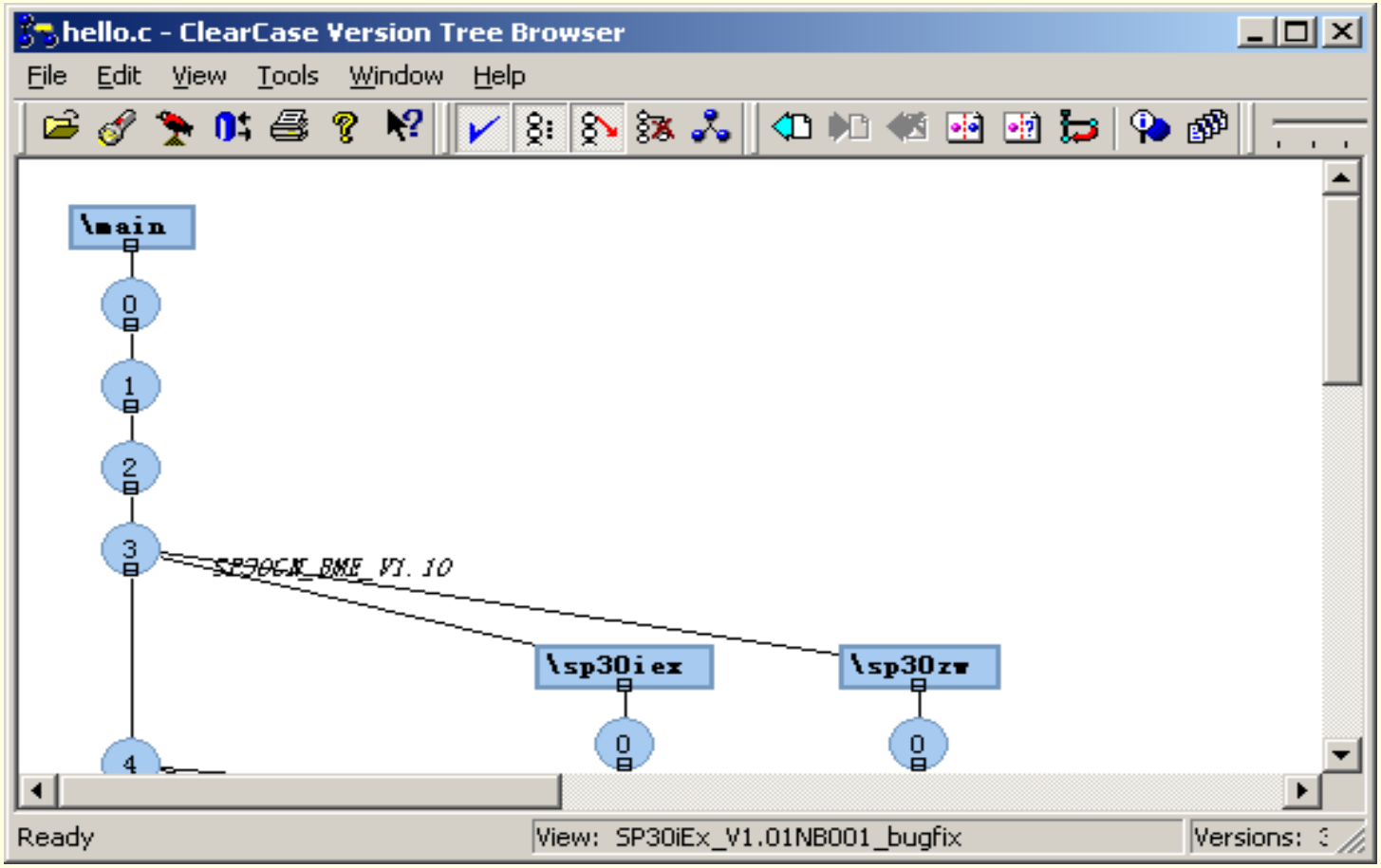
思想-解决冲突办法

yanbo@bit.edu.cn

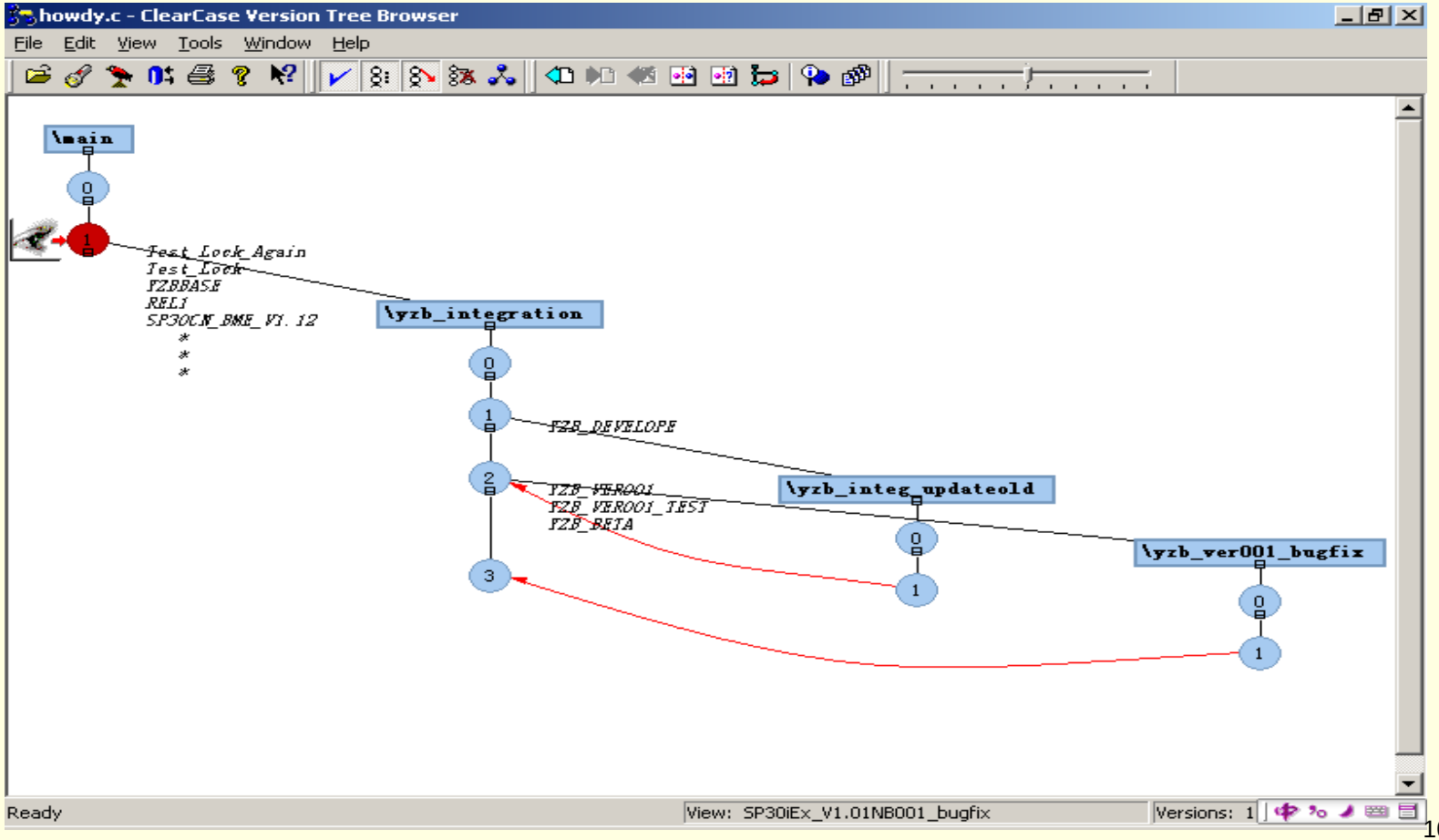


分支

yanbo@bit.edu.cn

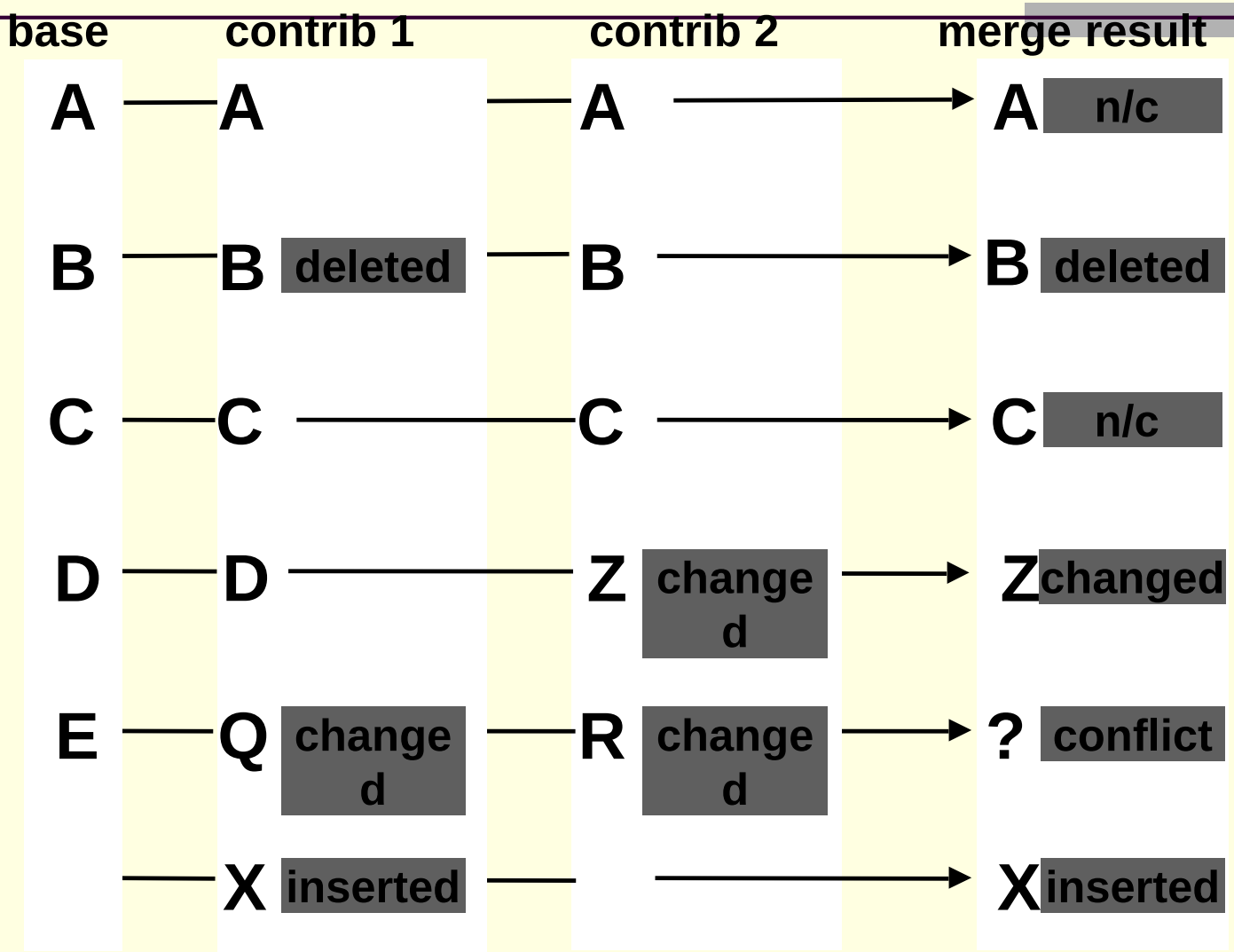


归并



归并

yanbo@bit.edu.cn



思想-工作空间管理

yanbo@bit.edu.cn

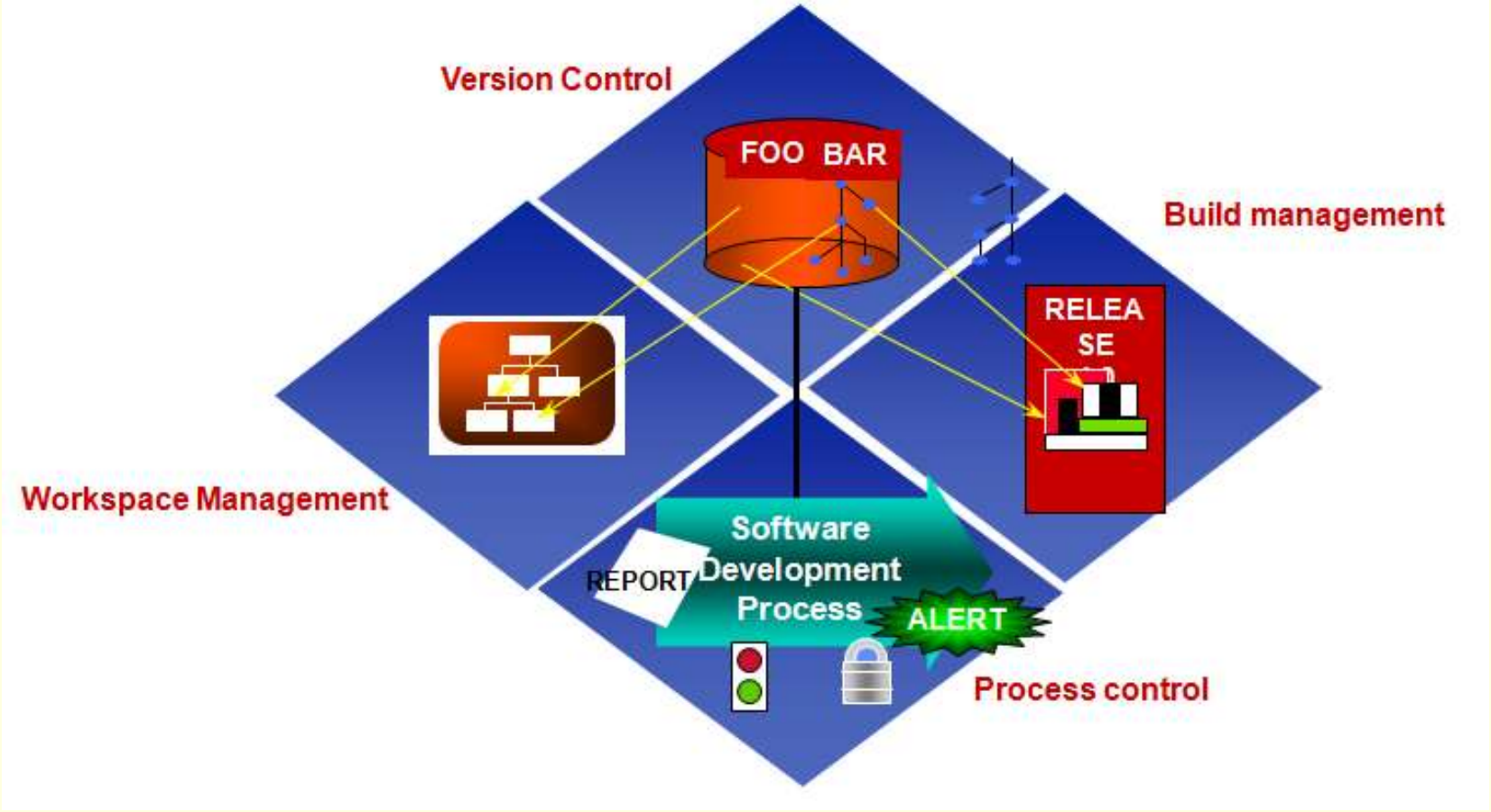
- 所谓“工作空间”，就是为了完成特定的开发任务（如开发新功能、进行软件测试、或修复BUG，等等），从版本库中选择一组正确的文件/目录的正确版本拷贝到开发人员的开发环境。
- 例如：为修复一个旧版本，如REL1中的BUG，开发人员首先需要在自己的开发环境中完全重现REL1 所对应的源文件和目录结构，也就是说，需要建立一个对应于REL1 的工作空间。

思想-工作空间管理

- 存在两类工作空间，一类是开发人员的私有空间，在私有空间中，开发人员可以相对独立地编写和测试自己的代码。
- 另一类工作空间是团队共享的集成空间，该空间用于集成所有开发人员的开发成果。
- 工作空间管理包括工作空间的创建、维护与更新、删除等。
- 工作空间应具备以下特点：稳定、一致、透明。

ClearCase 工作空间管理

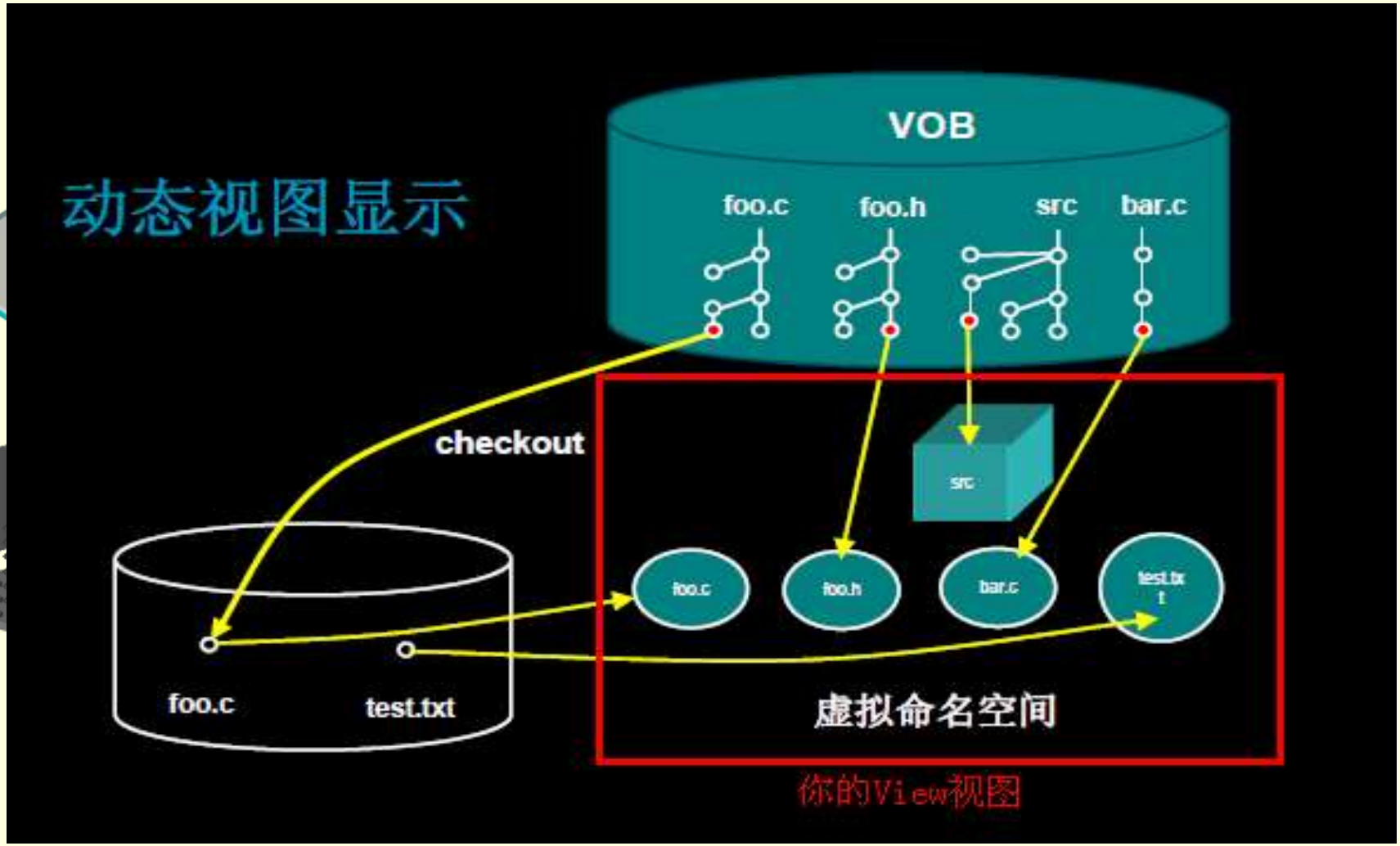
- 建立灵活的工作空间；



yanbo@bit.edu.cn

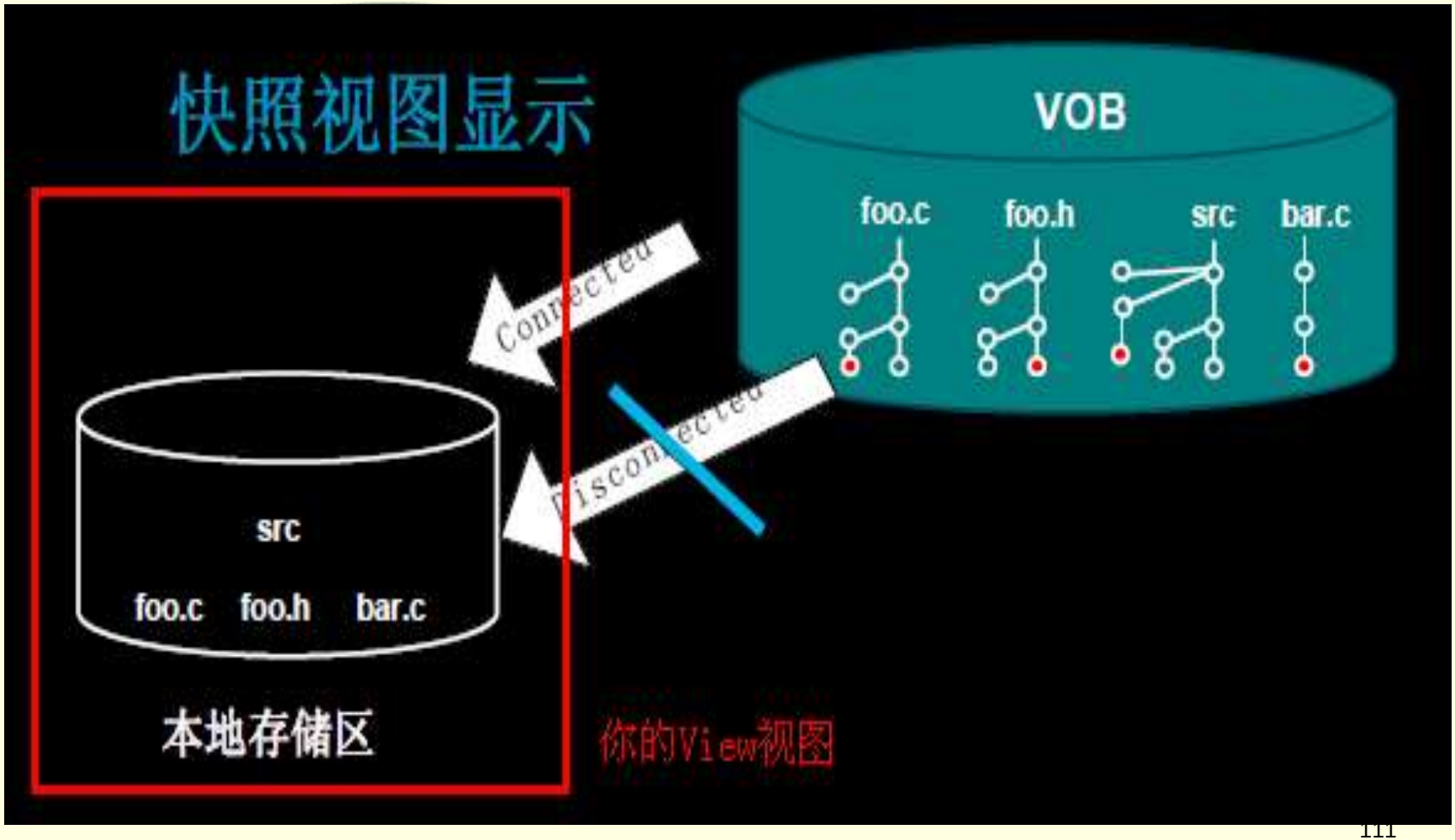
ClearCase 工作空间管理

yanbo@bit.edu.cn



ClearCase 工作空间管理

yanbo@bit.edu.cn



ClearCase 工作空间管理的益处

- 集成期间不影响开发工作继续进行
  - 在集成工作进行过程中能够保护个人的视图；
- 同步开发是可靠的
  - 为被检出的开发工作提供标识；
  - 自动检测多个同时被检出的文件并确保它们被适当的归并；

基于SCM的活动

项目经理角色

管理活动

- To do lists
- Allocate to team

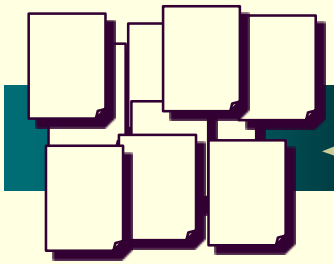
开发活动

Request	Priority	Owner
Add Web Interface	1	Terry
Bug 527	2	Sandy
Add local event handler	2	Kim

SCM 工具支持

管理工件

- Versioning: code, models, XML, HTML
- Parallel development



变更集

Add Web Interface

a.html	V5
c.xml	V3
b.jpg	V8

UCM -Unified Change Management

yanbo@bit.edu.cn

Out of the Box

- 基于变更管理的最佳实践
  - 无数的CM体验
  - 成千上万的用户
  - 易适用
  - 优化整个团队

抽象层次的提升

活动(Activity)
变更集
流(Stream)
活动的集合
组件（Components）
呈现软件结构
基线（Baselines）
组件的一个版本
项目（Project）
包含了组件和流

UCM：抽象层次的提升

简化变更管理，团队整体受益！

yanbo@bit.edu.cn

活动(Activity)

变更集

流(Stream)

控制隔离与集成的平衡

组件（Components）

呈现软件结构

基线（Baselines）

组件的一个版本

项目（Project）

包括了流和组件

项目经理

开发人员

集成人员

分析设计

测试人员

活动与工件

Activity（活动）

分配给开发人员的开发任务，包括实现产品新功能、修复产品缺陷、完善产品文档、设计及修改产品建模等。概念同“任务”。

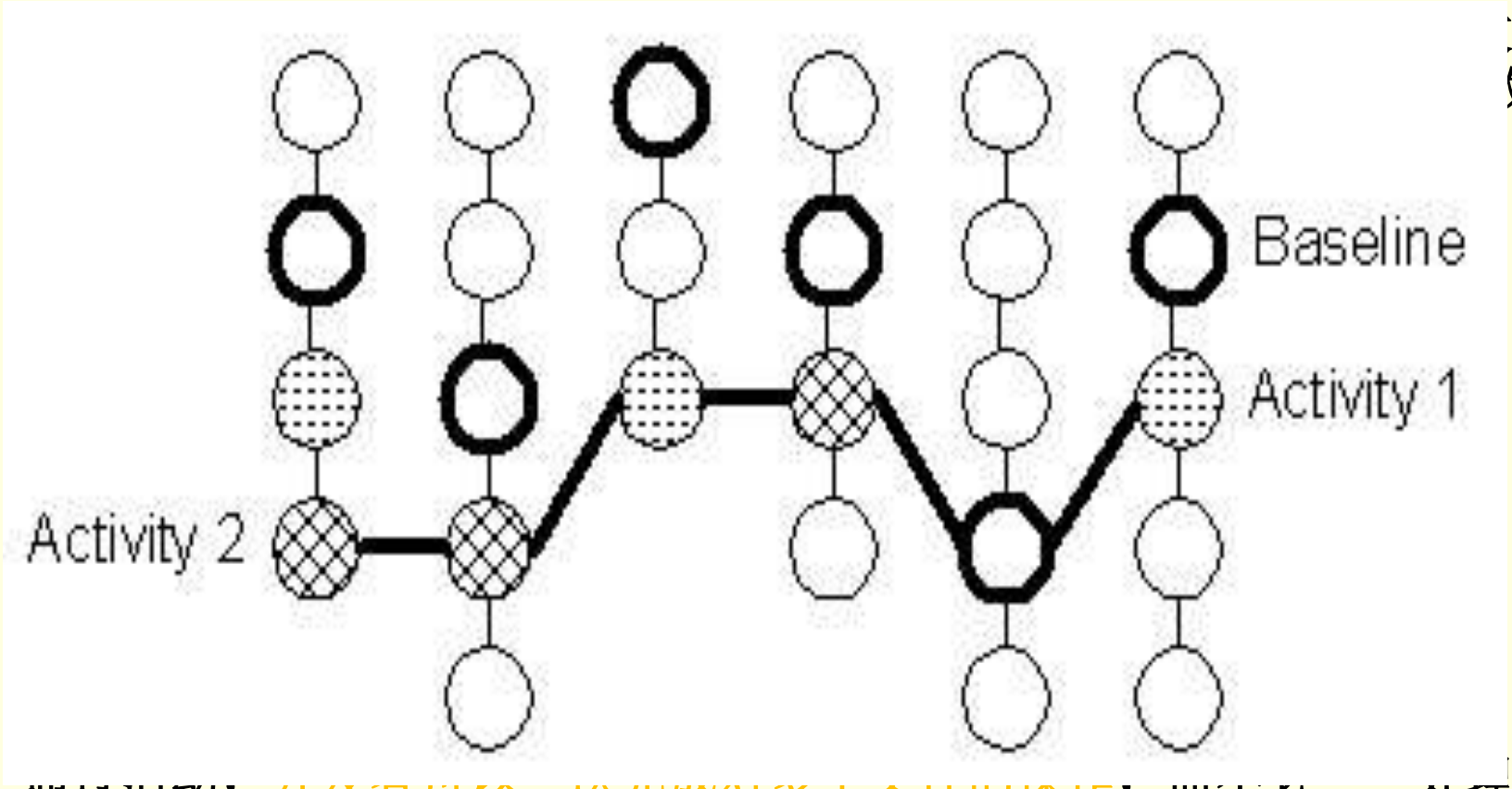
Artifact（工件）

在开发过程中生成的产品配置项。

UCM的活动和工件通过变更集建立联系。

活动

yanbo@bit.edu.cn



可以极大的提高工作效率。

活动

- 活动就是软件开发项目中的开发任务，一个活动包括一个文本标题和一个变更集，标题简要描述任务的目的范围，变更集标识了这个活动生成的代码版本。
- 通过活动，软件开发任务与开发出来的结果紧密联系起来。
- 活动除了与任务、文件版本关联外，还与UCM流和基线绑定。每个活动隶属特定的流，只能用来在该流上进行代码修改。每一个基线包含特定的配置项即文件的版本。

以活动为中心的组织和集成

Rational ClearQuest

管理活动

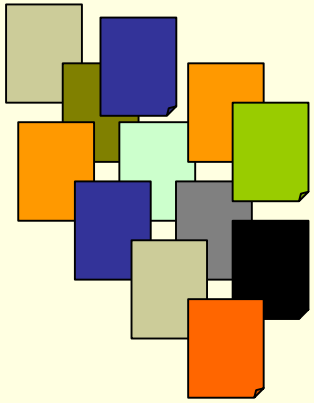
To Do Lists
Workflow

Rational ClearCase

管理工件

Versioning: code, models, XML, HTML

Parallel development



ClearQuest: Organized Activities

Request		Priority
		Owner
Special Promo	1	Terry
Bug 527	2	Sandy
Add GUI button	2	Kim

变更集

Special Promo

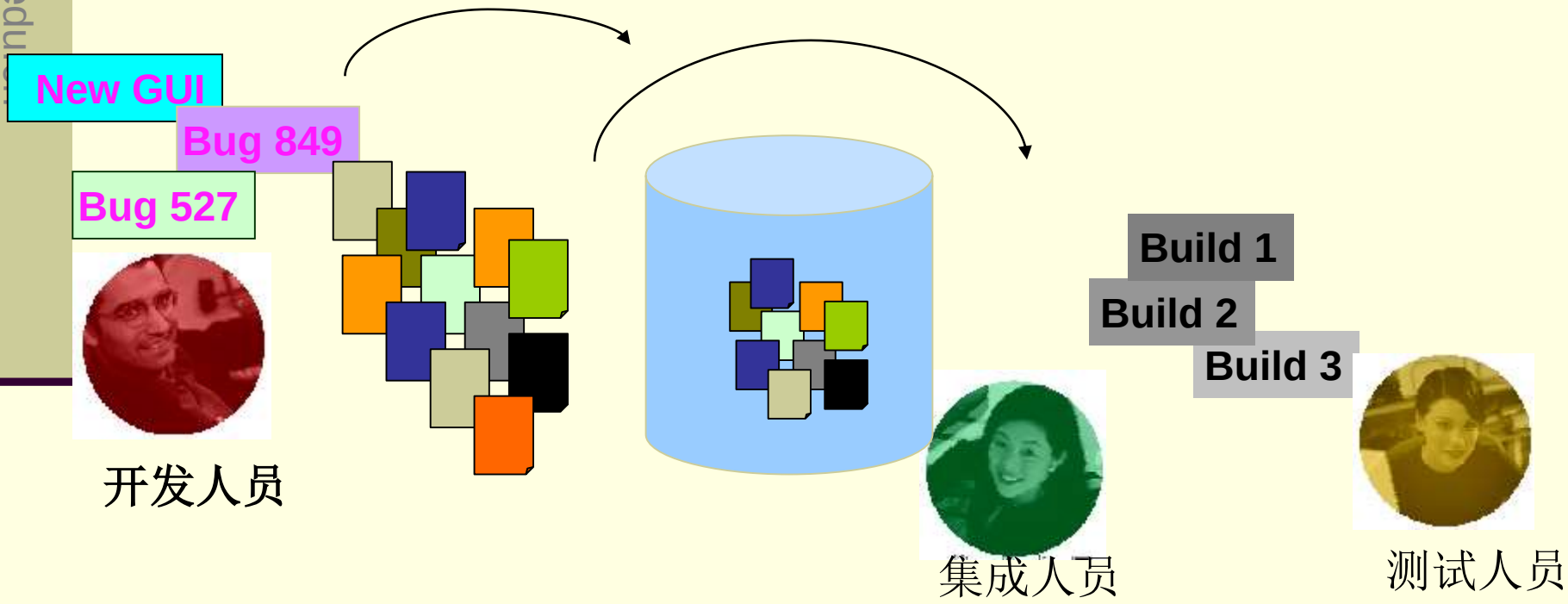
a.Html
V5
c.Xml
V3
b.Jpg
V8

统一活动与工件

引入活动带来的受益

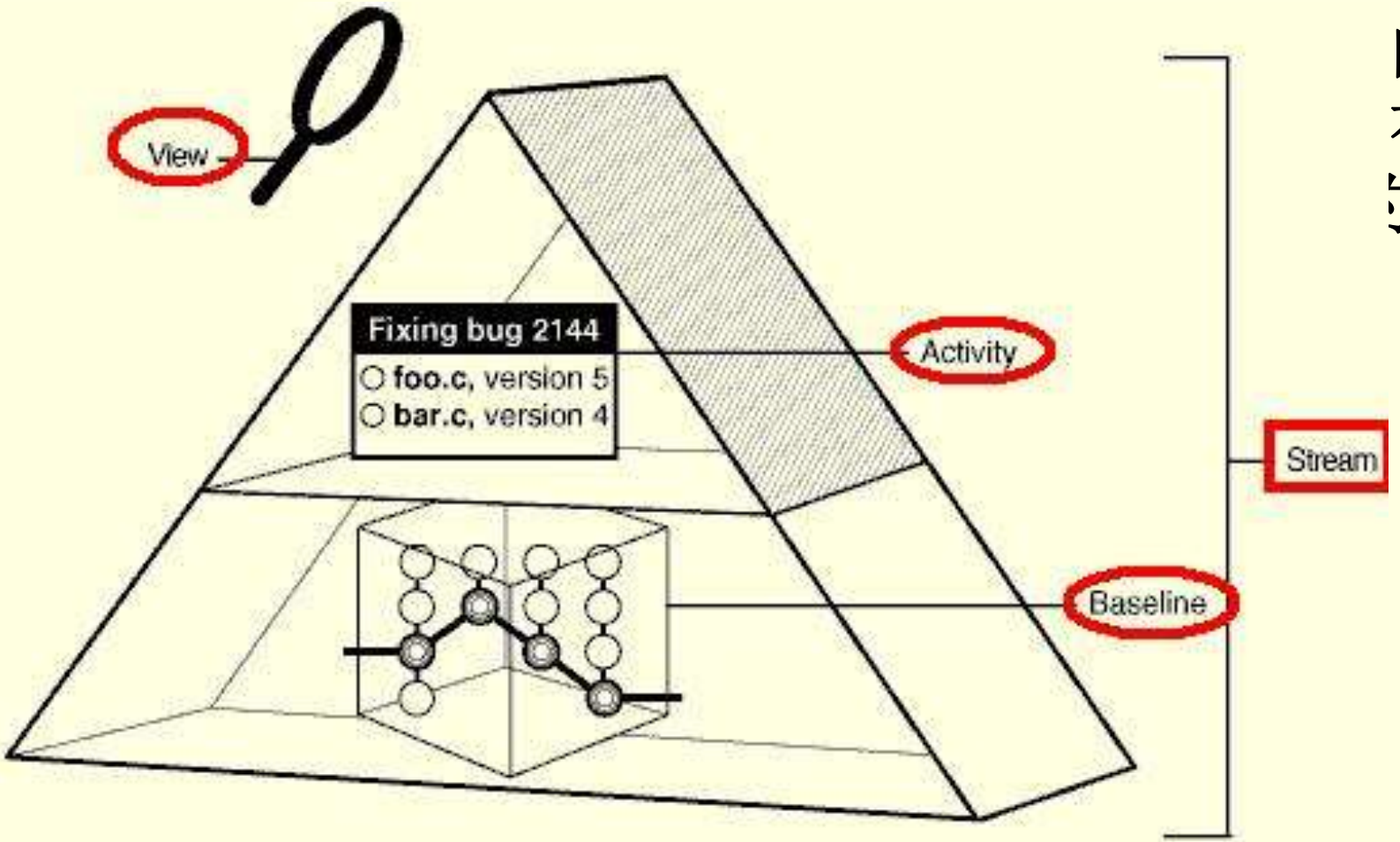
以“活动”递交 → 避免修改的丢失

基于活动的集成和测试 → 易于再现



流

的有



yanbo@bit.edu.cn

流

- 开发流：UCM 对象，决定了出现在开发视图中的元素的版本。包含开发人员的活动。开发流包含从集成流同步下来的版本以及之后开发人员建立的版本。
- 集成流：UCM 对象，记录了项目的基线，对外提供共享元素版本的访问能力。集成流和对应的集成视图构成最重要的共享工作区。
- 开发流、集成流为服务器端概念，可简单理解为服务器上的2个独立存储空间，特定阶段、特定版本集的文件、目录即存放在流中。
- 视图：为一个或多个用户提供工作区；包含VOB中元素的某个版本；有静态视图和动态视图两种形式。视图是通过一定的规则选择出来的各个元素特定版本的集合，用户通过视图存取、修改各个元素。

流

yanbo@bit.edu.cn

- 流的作用：
  - 流物理的存储了开发人员在视图中的活动；
  - 提供配置项的版本分支与合并机制；流封装了特定分支上的所有配置项的版本，开发人员通过在特定流上建立视图来获得这些版本，生成新版本，并可将版本在不同流之间移动，即执行分支与合并操作。

UCM: 参加项目与选择合适的工作区

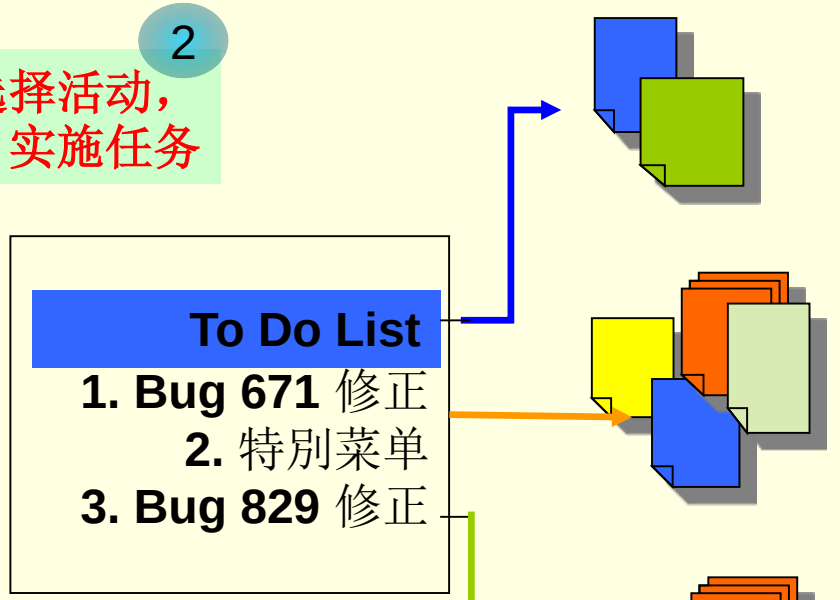
yanbo@bit.edu.cn

1

参加项目

2

选择活动，
实施任务

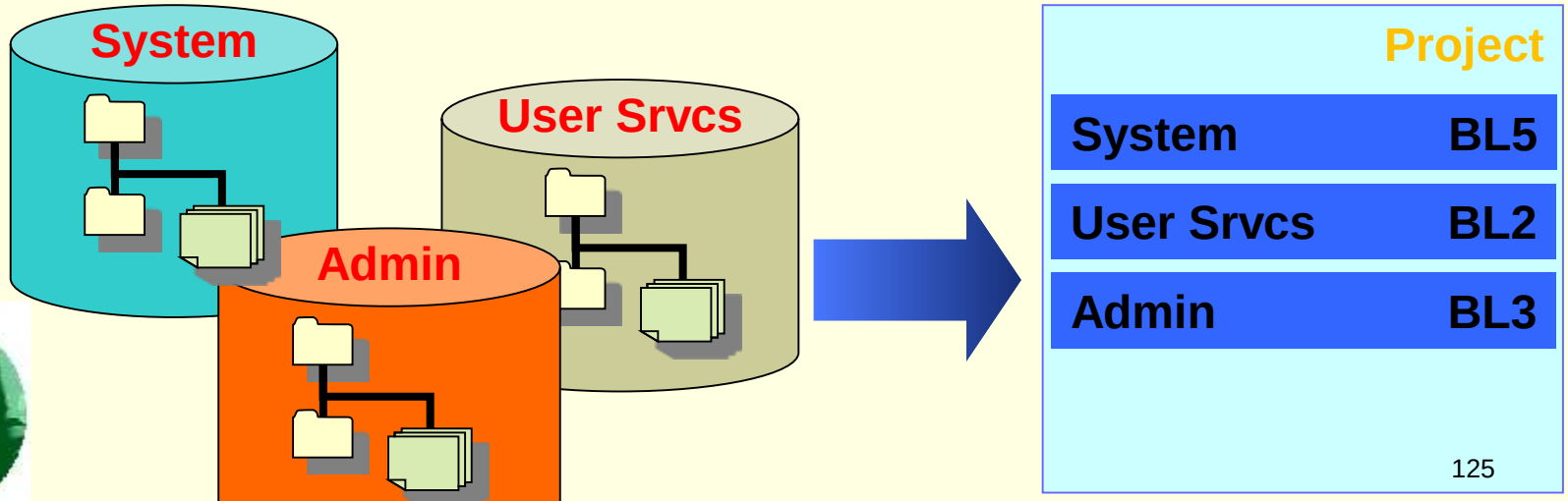


3

以活动递交变更
(递交变更集)

UCM：将工件组织成版本化的组件

- 组件：用于组织一个UCM项目中相关的目录和文件。同一个组件中的元素通常一起开发，集成和同步；一个项目至少包含一个组件，组件可被多个项目共享。
  - 把将要开发、集成、基线和发布的文件和目录聚集到一起
- “组件”的引入：
  - 有利于将逻辑设计与物理实现相对应；
  - 组件可以在不同项目之间重用；



集成人员

组件

yanbo@bit.edu.cn

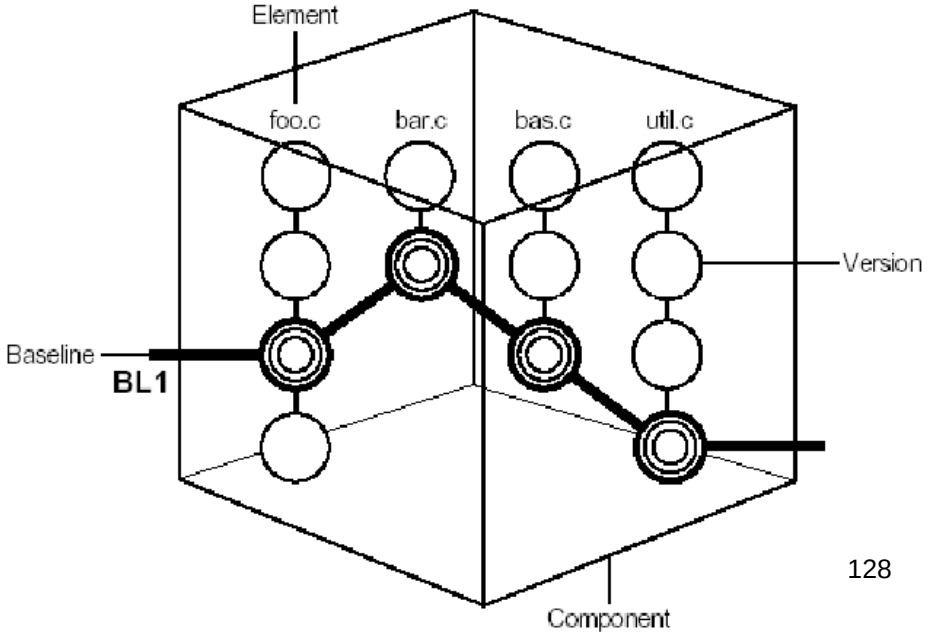
- UCM通过组件对象使软件项目中的代码和其他工件的组织结构能与软件系统的架构对应。
- 通过组件，对文档、代码的组织管理从目录这个比较低的层次上升到更高层次的模块化的软件结构系统或子系统。

基线

- UCM对象，代表了一个或多个组件的稳定配置。基线包含了活动和一个或多个组件中的每个可视元素的一个版本。开发人员可以基于某个基线创建开发流或同步某个已经存在的开发流。

基线

- 在软件开发过程中，可以将各个元素的不同的版本组合成一个基线。通过基线表示软件项目开发达到了一定的要求，也可以说是里程碑。项目管理员可以根据情况设置相应的基线，并随着项目的发展逐步设置新的基线。



UCM: 基线定级

yanbo@bit.edu.cn

基线的升级

项目经理

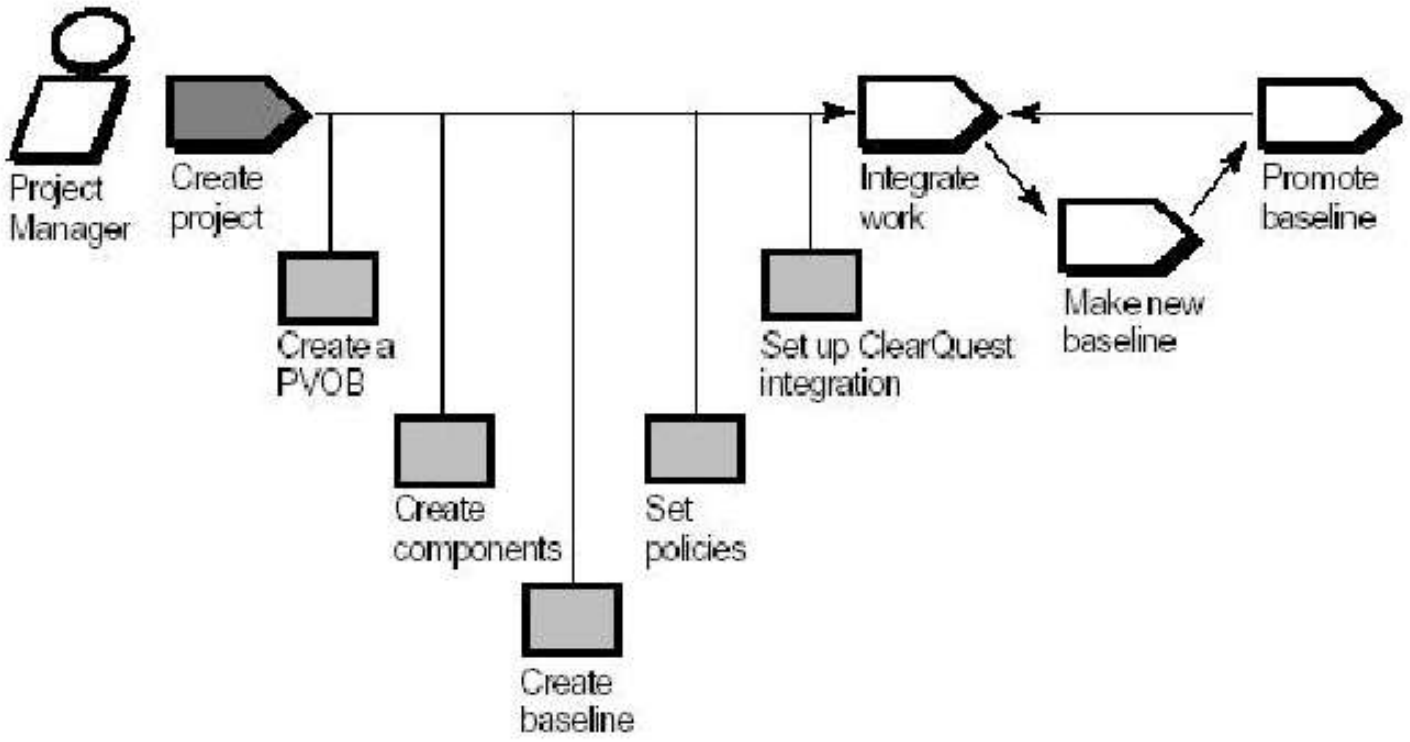
集成人员

项目管理

- UCM方式采用一种迭代开发过程，开发人员在同一个UCM项目中工作。
- 项目管理者负责创建项目，维护项目公共区域。
- 一个项目包括了公共区域和多个私有工作区域，私有工作区域允许开发人员在活动上各自独立地进行工作。

创建项目

yanbo@bit.edu.cn



项目工作过程

工作过程如下：

1.项目管理者创建项目并且为项目中的元素确定初始基线集合；

2.开发人员通过创建私有工作区域、获取项目基线内容，加入该项目；

3.开发人员创建活动并且一次在一个活动上工作，与活动相关的文件集合称为变更集；

4.当开发人员完成活动，并且在其私有工作区对其工作进行了测试后，通过执行交付将其工作与开发组共享。交付操作将开发者私有区域中的工作合并到项目共享区域。

项目工作过程

5.项目管理者在项目共享区域集成开发人员交付的工作。

6.项目管理者定期在项目共享域创建新的基线，用来集成开发人员的工作。

7.项目管理者执行快速验证测试，以保证新的基线可以正常工作。软件质量工程师将执行更多的扩展测试。

8.项目管理者定期在基线的质量和稳定性提高后调整基线的晋升级别（如：创建、测试、发布）以反映适当的里程碑。当新的基线经过了足够的测试，项目管理者可以将其指定为推荐基线。

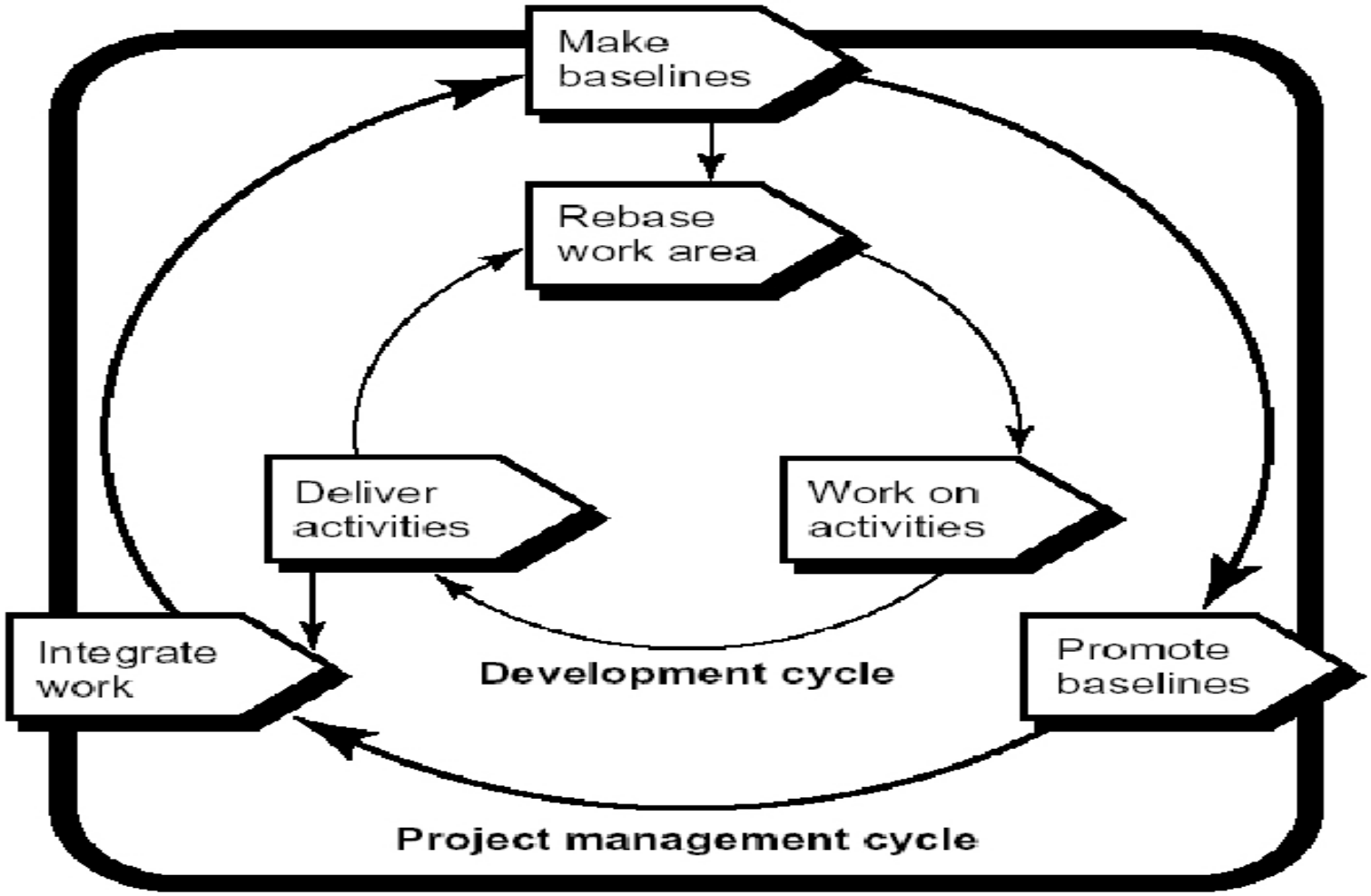
项目工作过程

9.开发者执行**rebase**操作来修改其私有工作区，使其包含新的推荐基线所确定的新版本集。

10.开发者继续如下的开发循环：基于活动进行开发工作、发布完成的活动、根据新基线修改其私有工作区域。

项目管理与开发循环

yanbo@bit.edu.cn



基于UCM模型的开发流程

1. 软件中的初始文件和目录被组织为版本化的工件，并保存在版本对象库（VOB）中。
2. 项目经理将项目中的任务分解为活动，并按一定的优先级别分配给开发人员。
3. 开发人员根据指派的活动在工作空间中对VOB中的工件进行修改，修改生成的每个文件或目录的版本都被自动记录在相应活动变更集中。
4. 开发人员在工作空间中完成代码修改，编译调试成功后就可以按活动将变更集提交并合并到项目集成流中，这一操作称为交付(Deliver)；交付是UCM实现开发隔离和合作的关键机制之一；

基于UCM模型的开发流程

5. 集成人员定期根据各个活动交付后而合并到项目集成流中的新代码来创建新的基线，并导入到集成工作空间进行构造，生成可安装包。经过基本测试后基线被提升为一定的状态以反映安装包的质量，同时安装包发布给测试人员进行深入测试；
6. 新基线提升发布后，开发人员基于此更新各自的工作空间，以获取最新稳定代码完成新的开发活动；

Q & A

yanbo@bit.edu.cn