4.2 传统软件开发过程的特点及存在的问题

4.2.1 为什么首先、单独讨论软件开发过程?

- 采用什么样的软件开发过程，对于开发的成功率是至关重要的;
- 软件开发方法并不能绝对控制软件开发过程，更不能替代软件开发过程。许多不同的开发方法，采用的是同样或类似的软件开发过程;
- 在软件开发过程的不同阶段，需要选择最合适的开发方法，因为还没有任何一种软件开发方法是完备的。

4.2 传统软件开发过程的特点及存在的问题

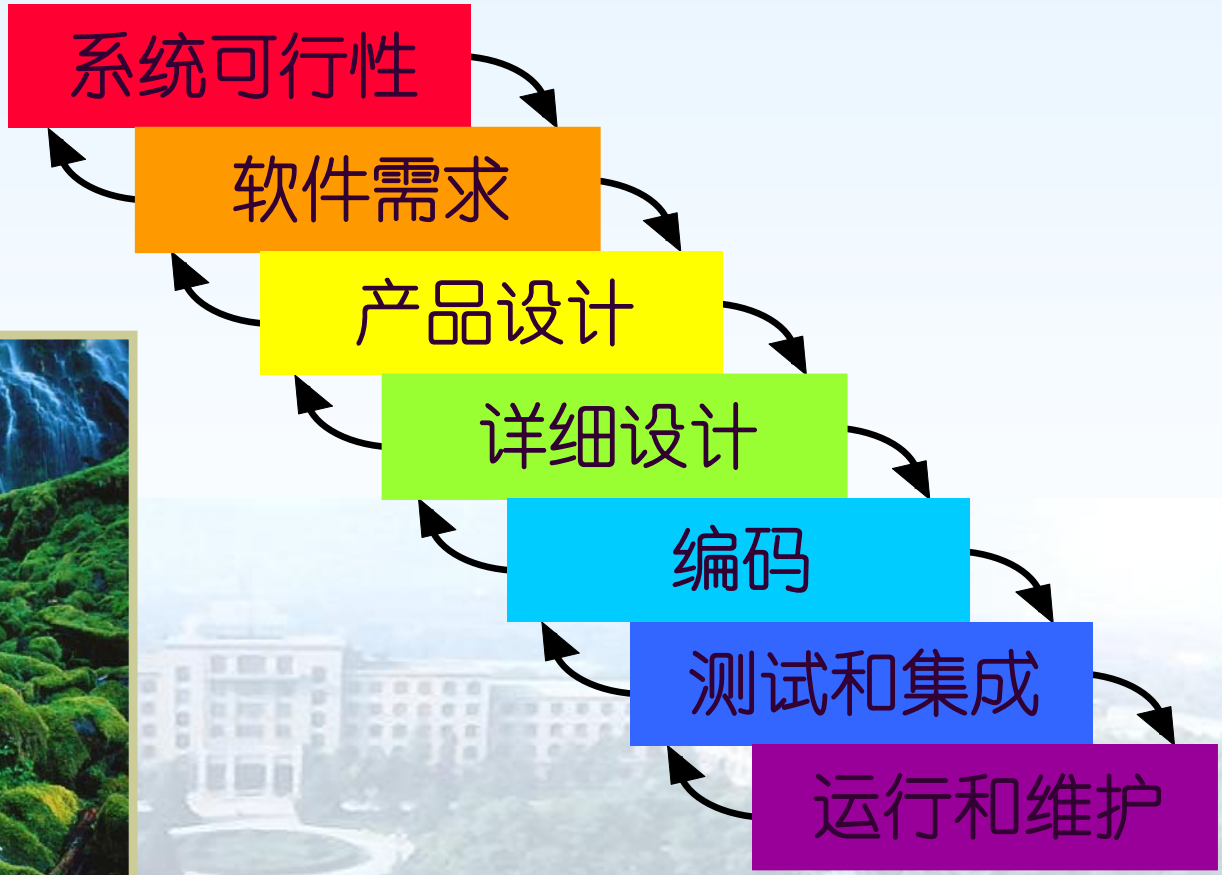
4.2.2 传统的软件开发过程模型

- 瀑布模型
- 原型模型
- 螺旋线模型
- 渐进式模型
- 增量式模型

4.2 传统软件开发过程的特点及存在的问题

4.2.2 传统的软件开发过程模型

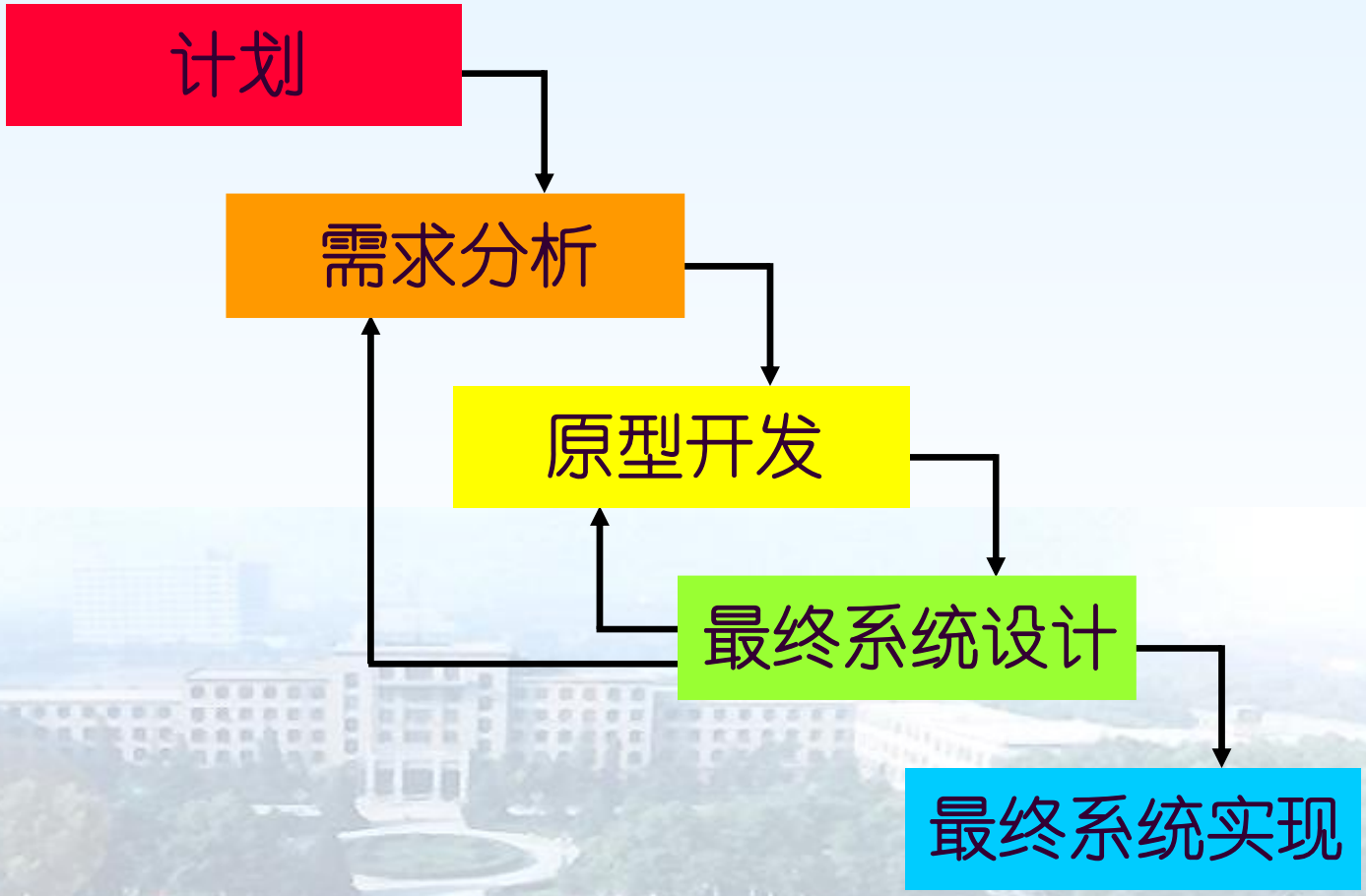
- 瀑布模型
- 原型模型
- 螺旋线模型
- 渐进式模型



4.2 传统软件开发过程的特点及存在的问题

4.2.2 传统的软件开发过程模型

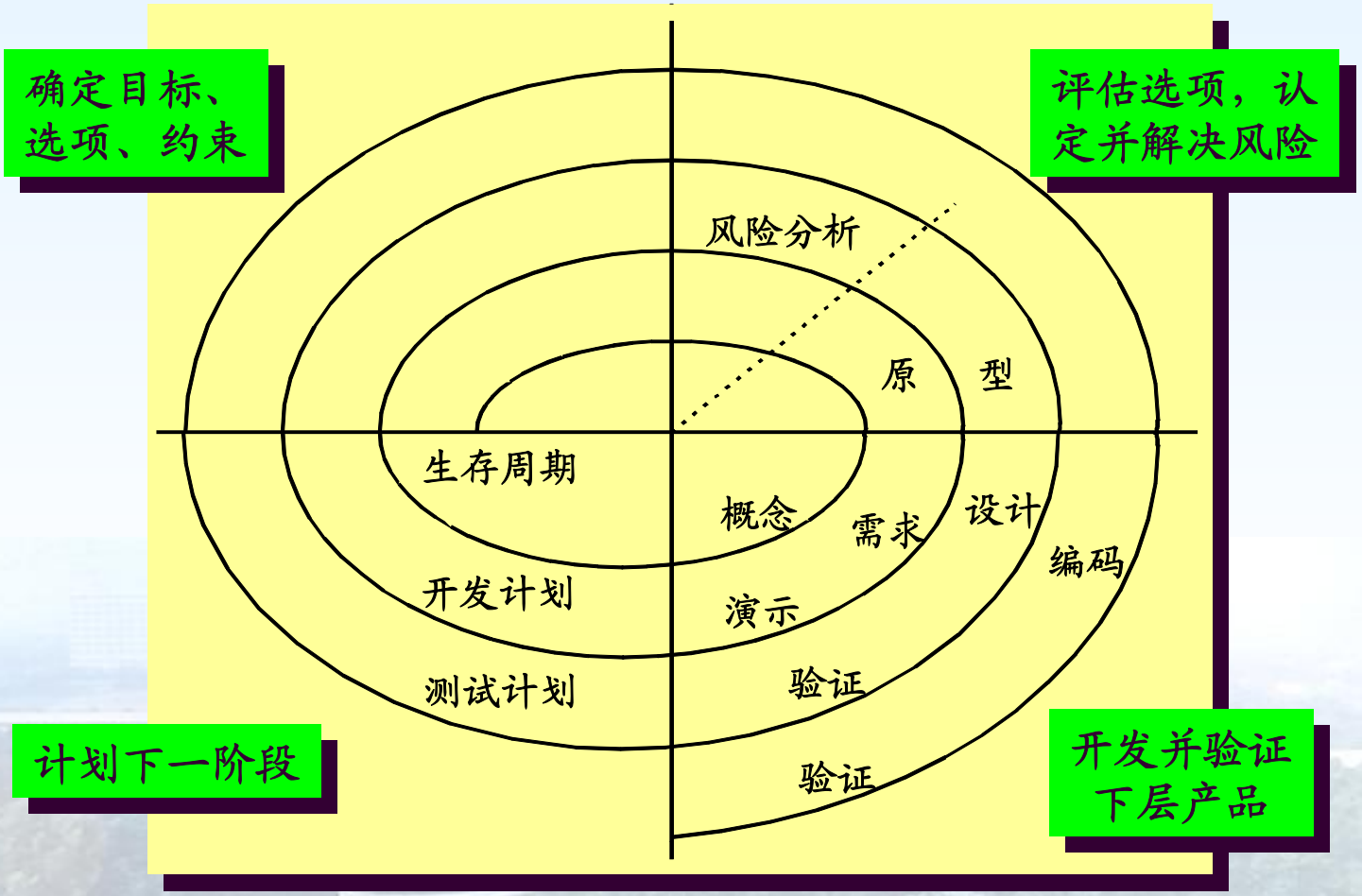
- 瀑布模型
- 原型模型
- 螺旋线模型
- 渐进式模型
- 增量式模型



4.2 传统软件开发过程的特点及存在的问题

4.2.2 传统的软件开发过程模型

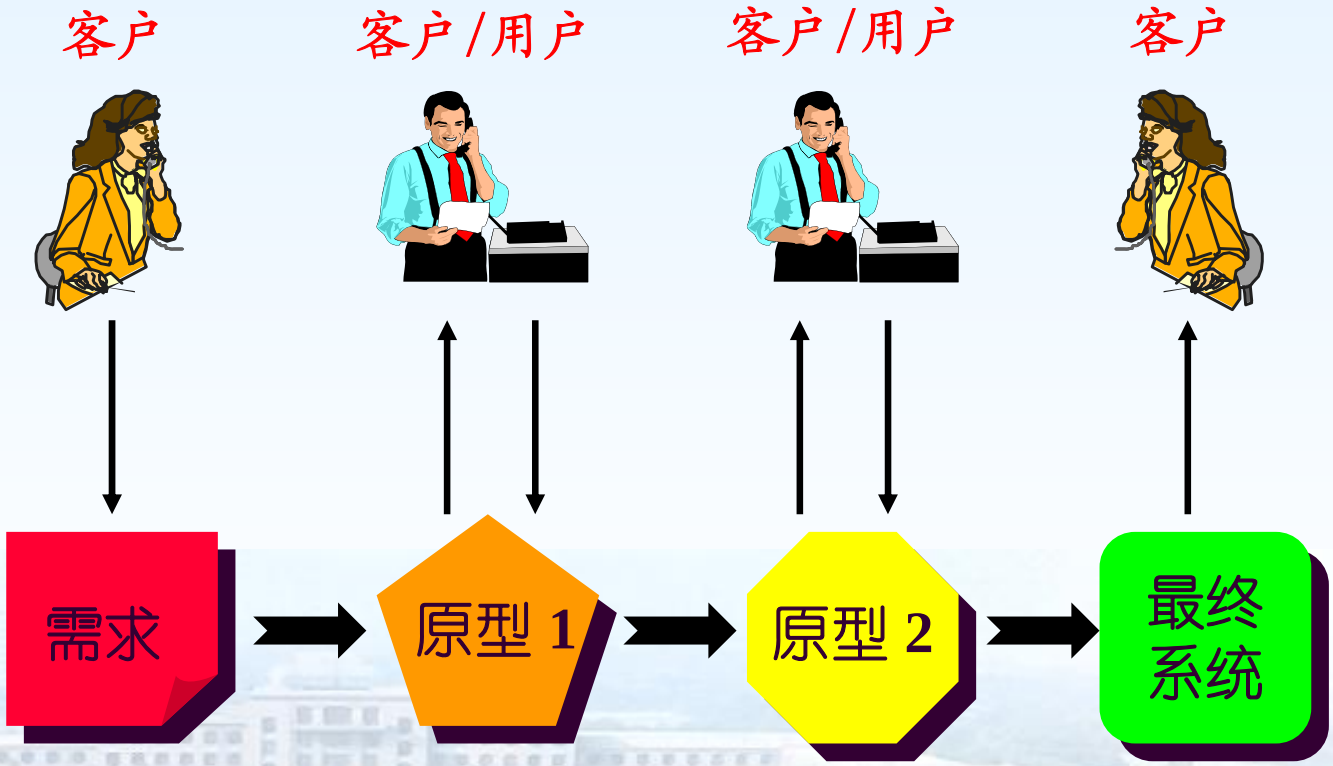
- 瀑布模型
- 原型模型
- 螺旋线模型
- 渐进式模型
- 增量式模型



4.2 传统软件开发过程的特点及存在的问题

4.2.2 传统的软件开发过程模型

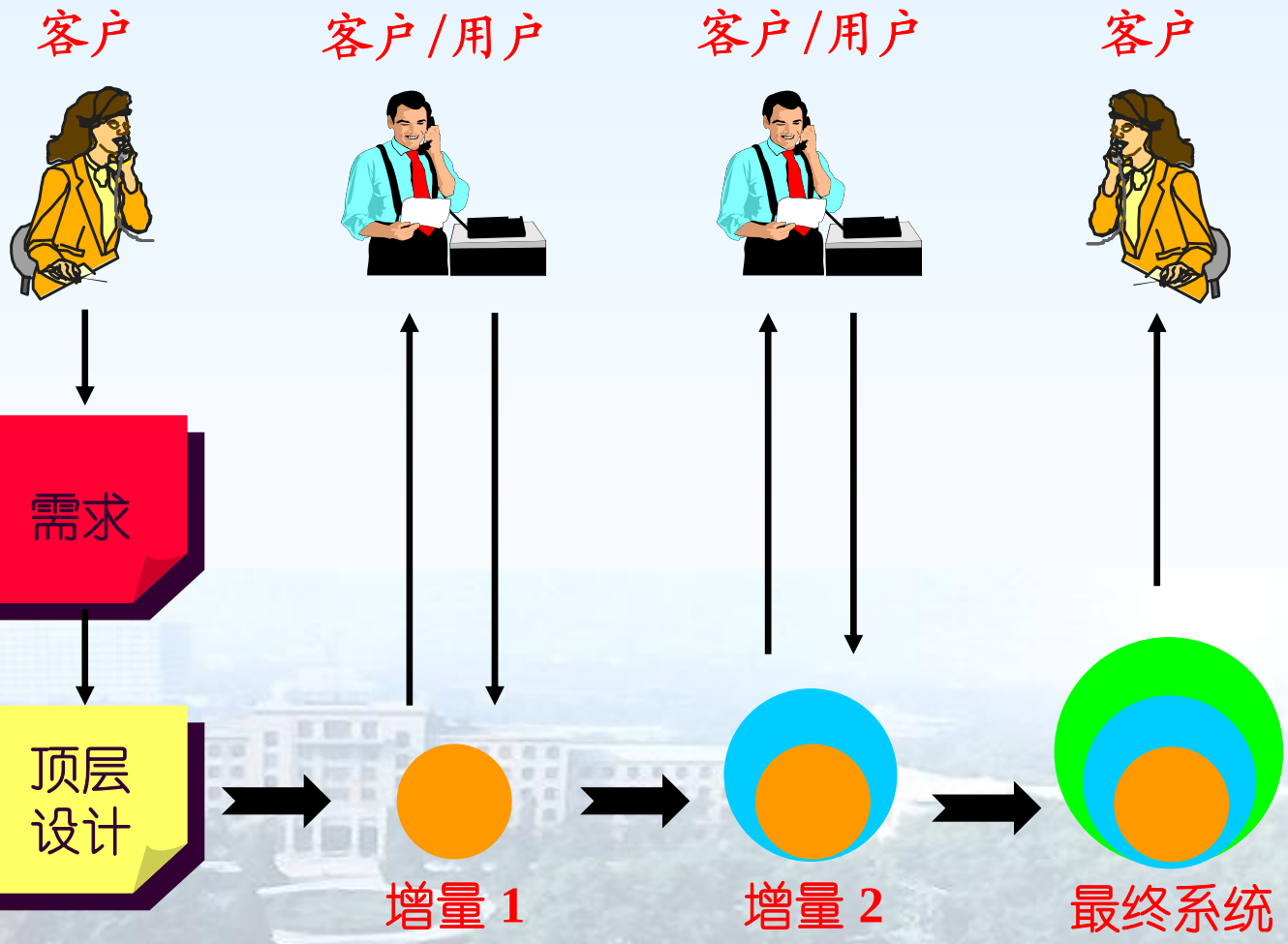
- 瀑布模型
- 原型模型
- 螺旋线模型
- 渐进式模型
- 增量式模型



4.2 传统软件开发过程的特点及存在的问题

4.2.2 传统的软件开发过程模型

- 瀑布模型
- 原型模型
- 螺旋线模型
- 渐进式模型
- 增量式模型



4.2 传统软件开发过程的特点及存在的问题

4.2.3 传统的软件开发过程存在的问题

- 采用的是被动适应需求变化和开发问题的方式（瀑布、原型、渐进）；
- 逐步演化的计划也是逐步形成的，对开发过程缺少有效的控制（螺旋线）；
- 要求与客户/用户的每次交流都给出“冻结”的阶段结果，对于复杂系统是不现实的（增量）；
- 对于重做一部分已得出的结果，因缺少对策而本质上是恐惧的；
- 更多考虑的是管理的方便性，对实施缺少有效的支持。

4.3 传统软件开发方法的特点及存在的问题

4.3.1 方法论的概念

- 方法论:
  - 在系统分析与设计中所采用的方式、策略、方向、活动。
- 典型的方法论:
  - 功能分解方法
  - 结构化方法
  - 信息建模方法（E-R方法）
  - 面向对象方法

4.3 传统软件开发方法的特点及存在的问题

4.3.2 功能分解方法

- 功能分解方法将一个系统看成是由若干个功能构成的一个集合，每个功能可以划分成若干个加工（子功能），一个加工又进一步分解成若干个加工步骤（子加工）乃至过程。这种方法可以用下面的式子来表示:

  功能分解＝功能＋子功能＋功能接口

- 分解的结果用一个结构图来表示，体现了特定的功能模块，也体现了模块之间的层次的关系，最终的程序规格说明与过程化的语言所提供的基本控制结构（即顺序结构、选择结构和迭代结构）有着很好的对应性。

4.3 传统软件开发方法的特点及存在的问题

4.3.2 功能分解方法

例:

- 商业计划
- 财政
- 产品计划
- 材料
- 生产计划
- 生产
- 研究
- 零售
- 销售
  - 股票控制
  - 订货服务
    - 接受订货
      - 确认顾客付款能力
      - 检查产品库存
      - 出错处理过程
      - 积压订货处理过程
      - 大宗顾客订货
    - 交付订货
    - 开具帐单
    - 发货
  - 包装
  - 发运
- 结帐
- 人事

4.3 传统软件开发方法的特点及存在的问题

4.3.2 功能分解方法

- 功能分解方法的弱点是:
  - 需求不一定能被直接、客观地映射到功能上，何况还存在非功能需求;
  - 作为基点的功能，其需求通常是不稳定的;
  - 功能划分只能体现加工的层次，未必能体现数据的层次;
  - 一般只适用于规模小、简单的系统。

4.3 传统软件开发方法的特点及存在的问题

4.3.3 结构化方法

- 结构化方法用数据流图（DFD）分析系统的数据流，着眼于数据加工流程，有固定的模式转换成设计。这种方法可以用下面的式子来表示:

  结构化方法＝数据（控制）流＋数据（控制）变换＋
  数据（控制）存贮＋终结点＋小型规格说明＋数据词典

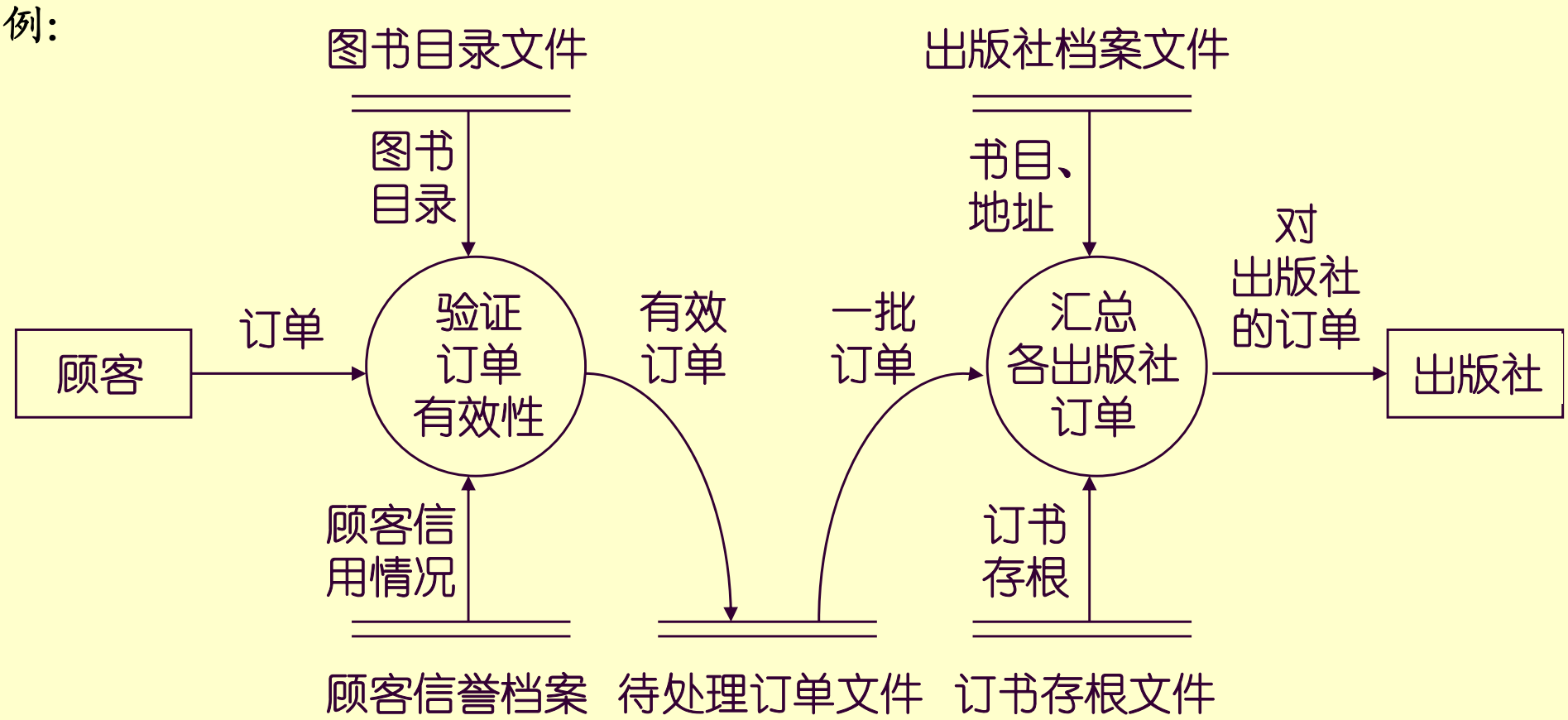
  其中终结点就是通常所谓的数据源和数据池。

- 这种方法能够帮助分析人员搞清系统的处理对象应当经历的加工过程，可以检查数据和加工是否发生了遗漏和不一致。

4.3 传统软件开发方法的特点及存在的问题

4.3.3 结构化方法

例：



4.3 传统软件开发方法的特点及存在的问题

4.3.3 结构化方法

- 结构化方法的弱点是:
  - 有些需求不是以数据流为主干的;
  - DFD表现不了数据的结构和联系，需附加数据字典;
  - 数据流的分解是指数级的，可能发生数据字典“爆炸”。

4.3 传统软件开发方法的特点及存在的问题

4.3.4 信息建模方法

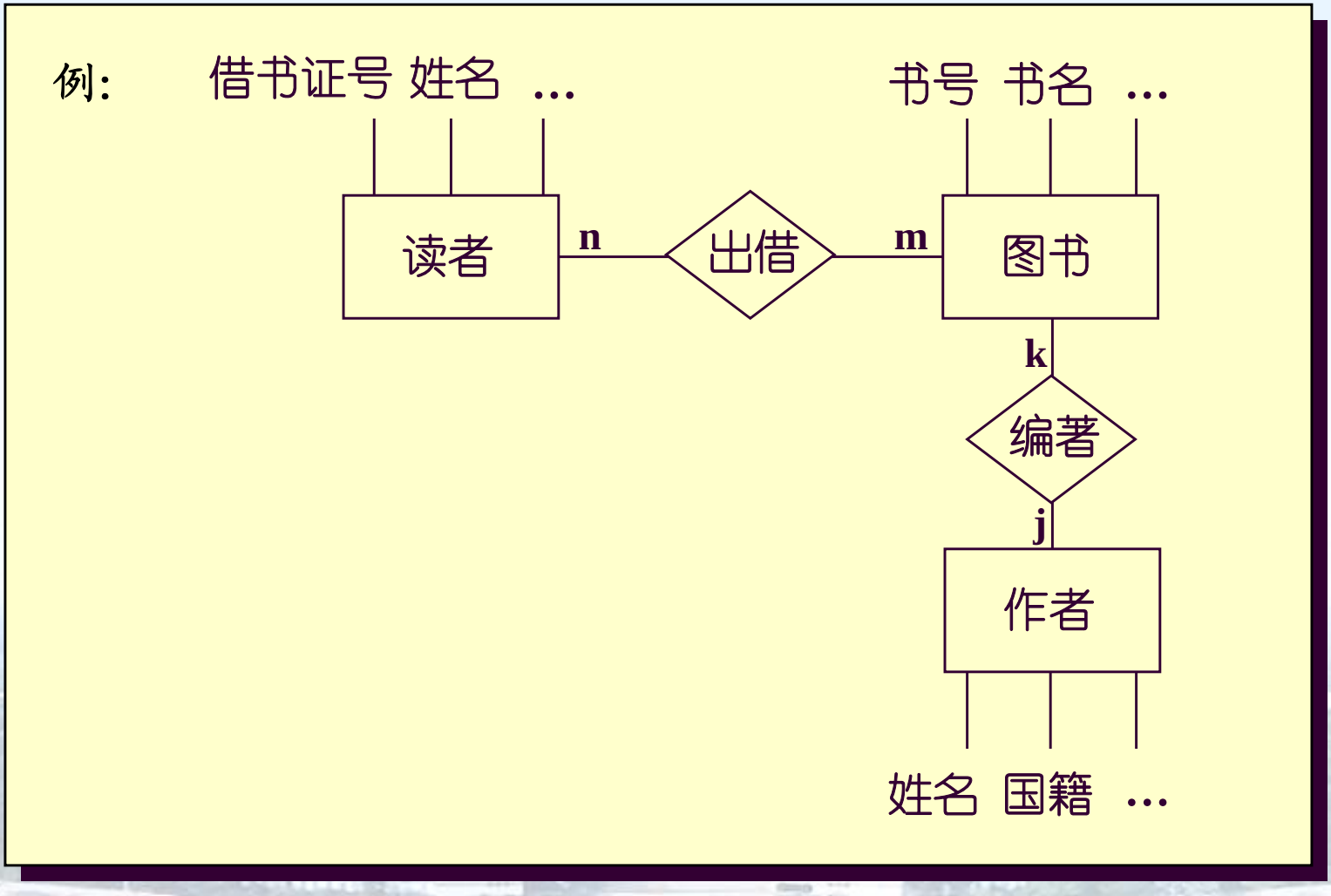
- 信息建模方法用扩展的 E-R 图分析系统中的实体及其间的联系，从数据的角度对问题空间建模，对于获得对问题空间的认识很有帮助。这种方法可以用下面的式子来表示:

  信息建模＝对象＋属性＋联系＋父类型／子类型＋关联对象

  这里，对象指的是现实世界中某种事物的表示，以及这种事物的一些实例。
- 这种方法强调数据的结构特征以及数据之间的联系，适用于数据密集、结构/联系复杂、数据加工操作简单的系统。

4.3 传统软件开发方法的特点及存在的问题

4.3.4 信息建模方法



4.3 传统软件开发方法的特点及存在的问题

4.3.4 信息建模方法

- 信息建模方法的弱点是:
  - 表示不了对操作（加工）的需求;
  - 体现不了数据之间的相似性关联;
  - 在表示模型上与设计有较大的区别，需要转换。

4.3 传统软件开发方法的特点及存在的问题

4.3.5 传统方法的思维特征

- 功能分解方法采用程序员的思维方式，没有计算机和软件知识的用户很难理解。
- 结构化方法采用一般工程技术人员的思维方式，一部分用户能够理解。
- 信息建模方法采用关系数据库设计人员的思维方式，一般用户也可以理解。

4.3 传统软件开发方法的特点及存在的问题

4.3.6 传统方法存在的共同问题

- 分别建立功能模型和信息（数据）模型，从而难以在这两种不同表示的模型之间，有效地验证和检查一致性和正确性;
- 从分析模型到设计模型的转换是一件困难的工作;
- 对行为的建模缺少支持。

4.4 Rational统一开发过程（RUP）

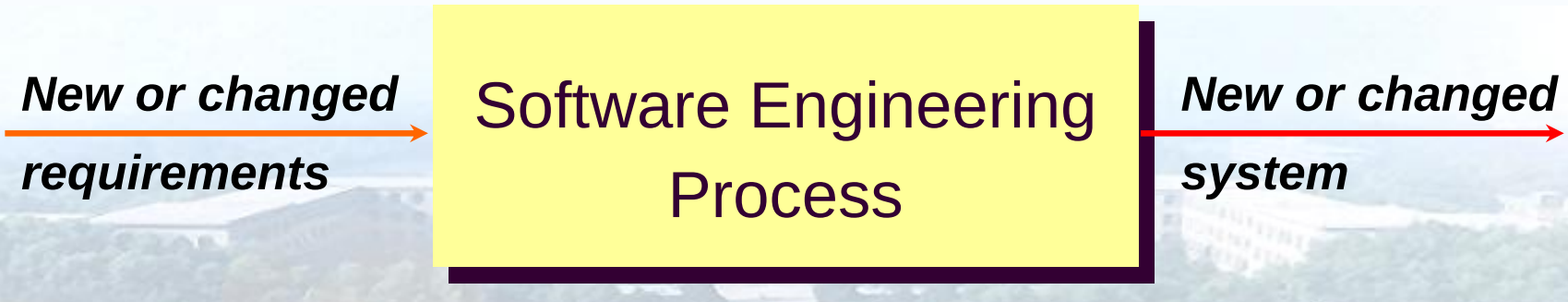
4.4.1 什么是RUP?

- 在开发UML（统一建模语言）的过程中，一直在考虑以UML的概念为基础的开发过程。
  - 作为一种用于描述、说明、可视化地构造软件开发各个阶段的中间制品（并产生相应的文档）的语言， UML只是一种手段而不是结果。
  - 软件开发的结果应当是坚固的、可伸缩的、可升级的软件系统，为了获得这一结果，需要语言（如UML）和软件过程（如RUP）两个方面的支持。

4.4 Rational统一开发过程（RUP）

4.4.1 什么是RUP?

- 什么是软件工程过程？
  - A process defines Who is doing What, When and How to reach a certain goal.
  - In software engineering the goal is to build a software product or to enhance an existing one.



4.4 Rational统一开发过程（RUP）

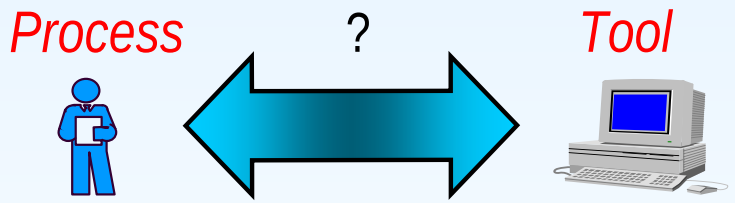
4.4.1 什么是RUP?

- The problem … ...
  - Most software projects use no well-defined process. Instead team members (re-)invent process as they go.
  - If process is used, different functional teams use normally inconsistent processes and modeling languages.

4.4 Rational统一开发过程（RUP）

4.4.1 什么是RUP?

- The problem … ...
  - Processes are not linked properly to tools, or are not properly automated.



4.4 Rational统一开发过程（RUP）

4.4.1 什么是RUP?

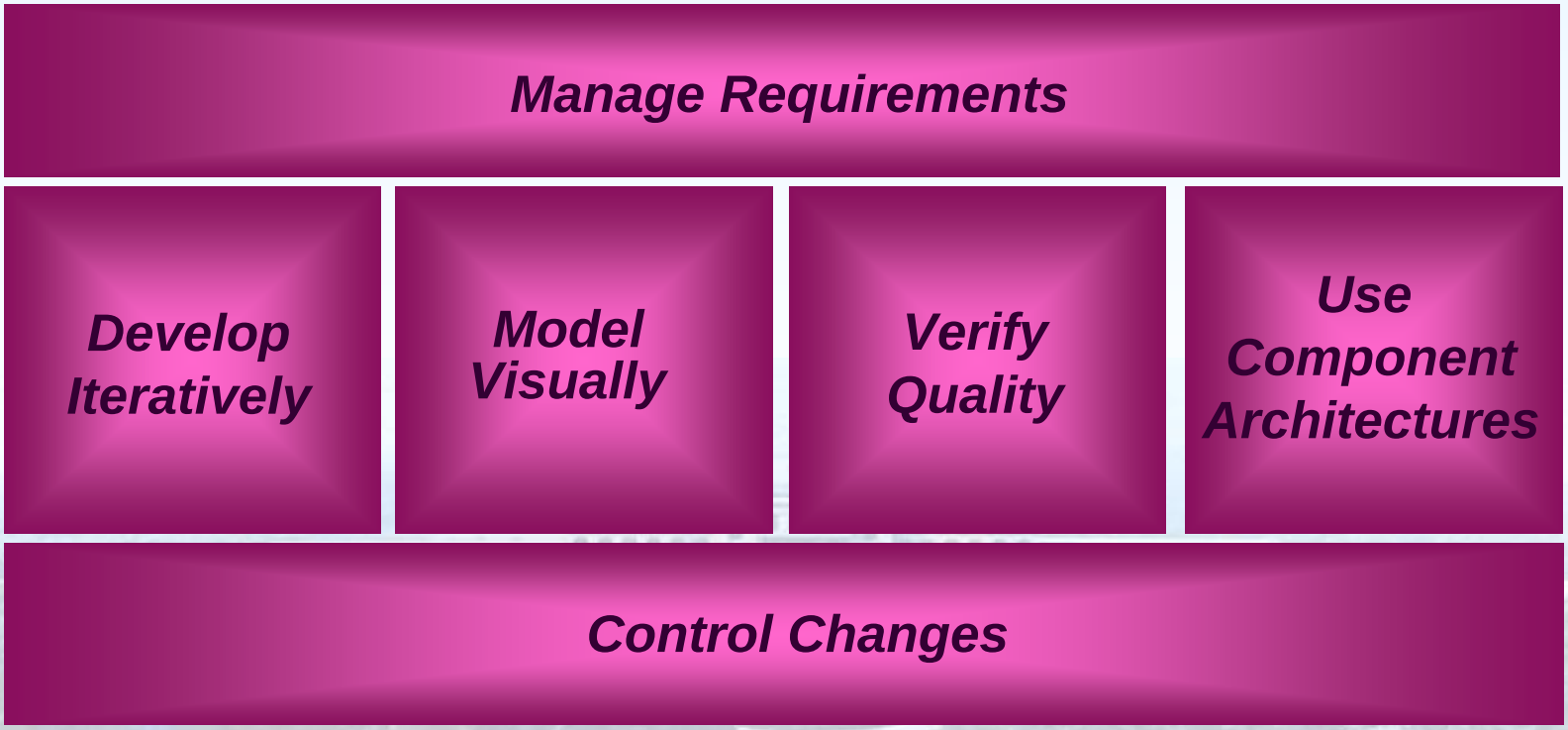
- RUP (Rational Unified Process) 是由 Rational 公司的一种软件开发过程产品，是目前影响较大的、面向对象的软件开发过程。RUP提出了一整套以UML为基础的开发准则，用以指导软件开发人员以UML为基础进行软件开发。
- RUP从已形成的各种面向对象分析与设计方法中吸取精华，为软件开发人员以UML为基础进行软件开发提供了一个普遍的软件过程框架，可以适应不同的软件系统、应用领域、组织类型和项目规模。
- OMG正在以RUP为基础进行标准化工作，目标是UP。

4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

Describes the effective implementation of key “Best Practices”.

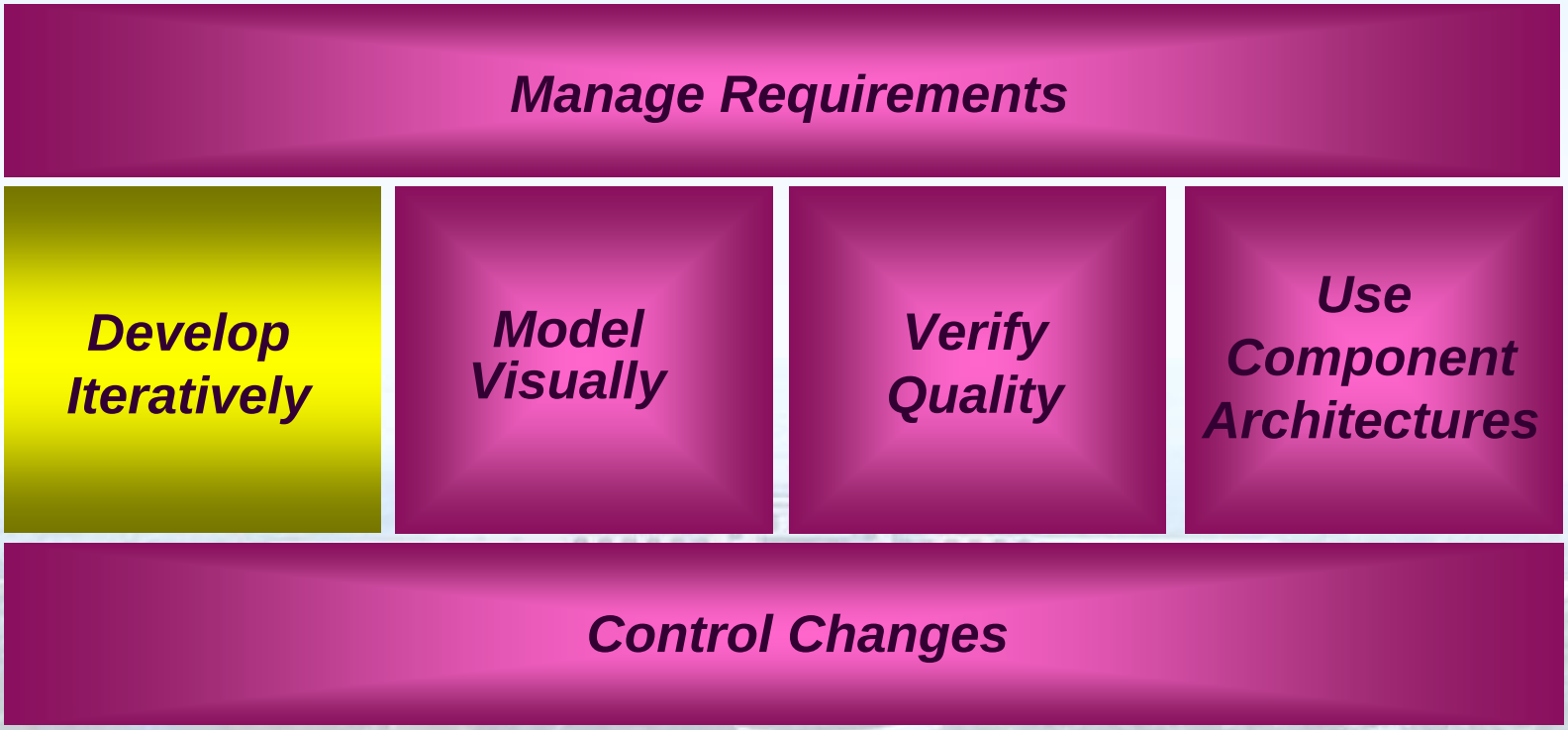


4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

1. Develop Software Iteratively:



4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

1. Develop Software Iteratively:
   - An initial design will likely be flawed（*有缺陷*） with respect to its key requirements.
   - Late - phase discovery of design defects results in costly over-runs and/or project cancellation.

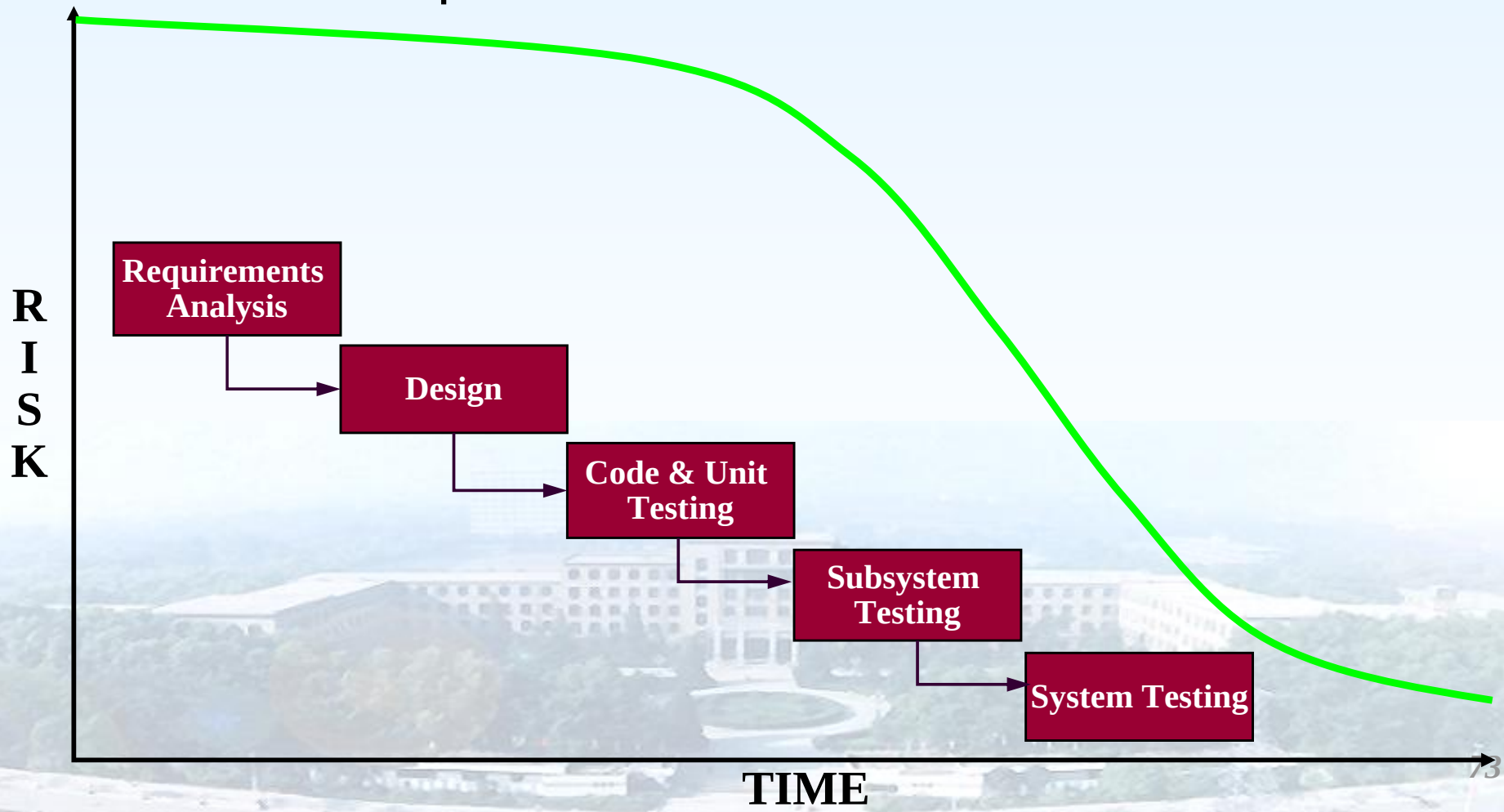
The time and money spent implementing a faulty design are not recoverable !

4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

Waterfall Development: Risk vs. Time:

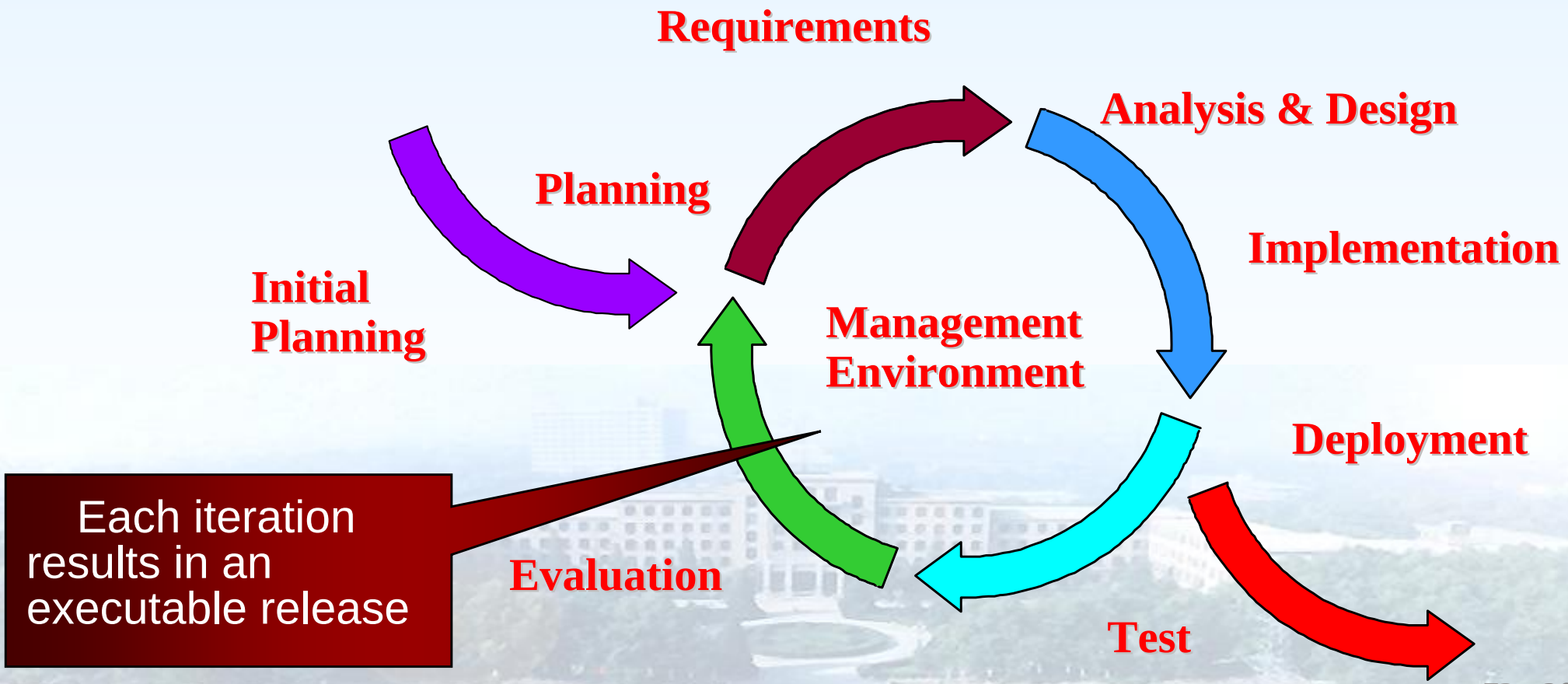


4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

Iterative Development（迭代式开发）：

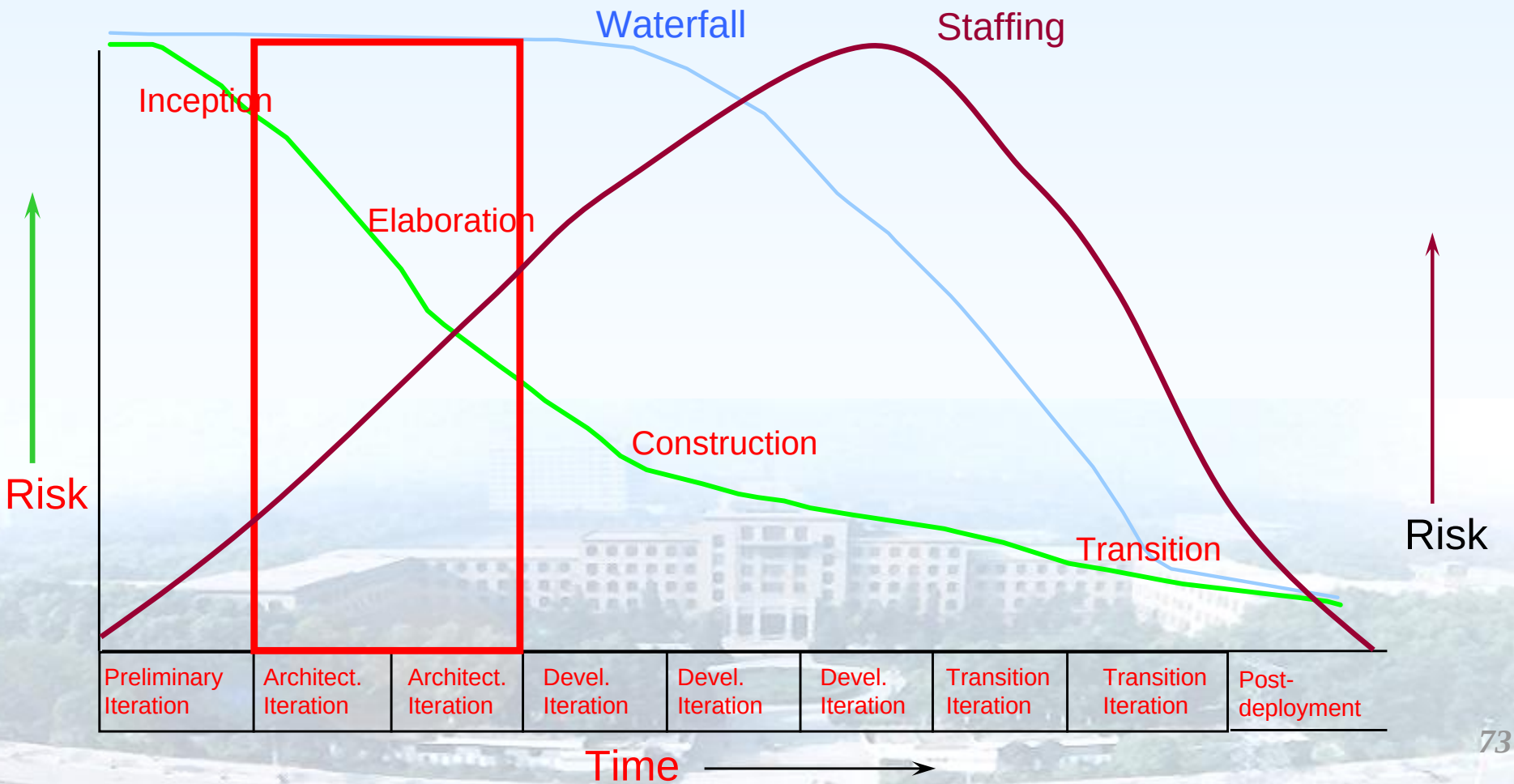


4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

Risk Profile of an Iterative Development:



4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

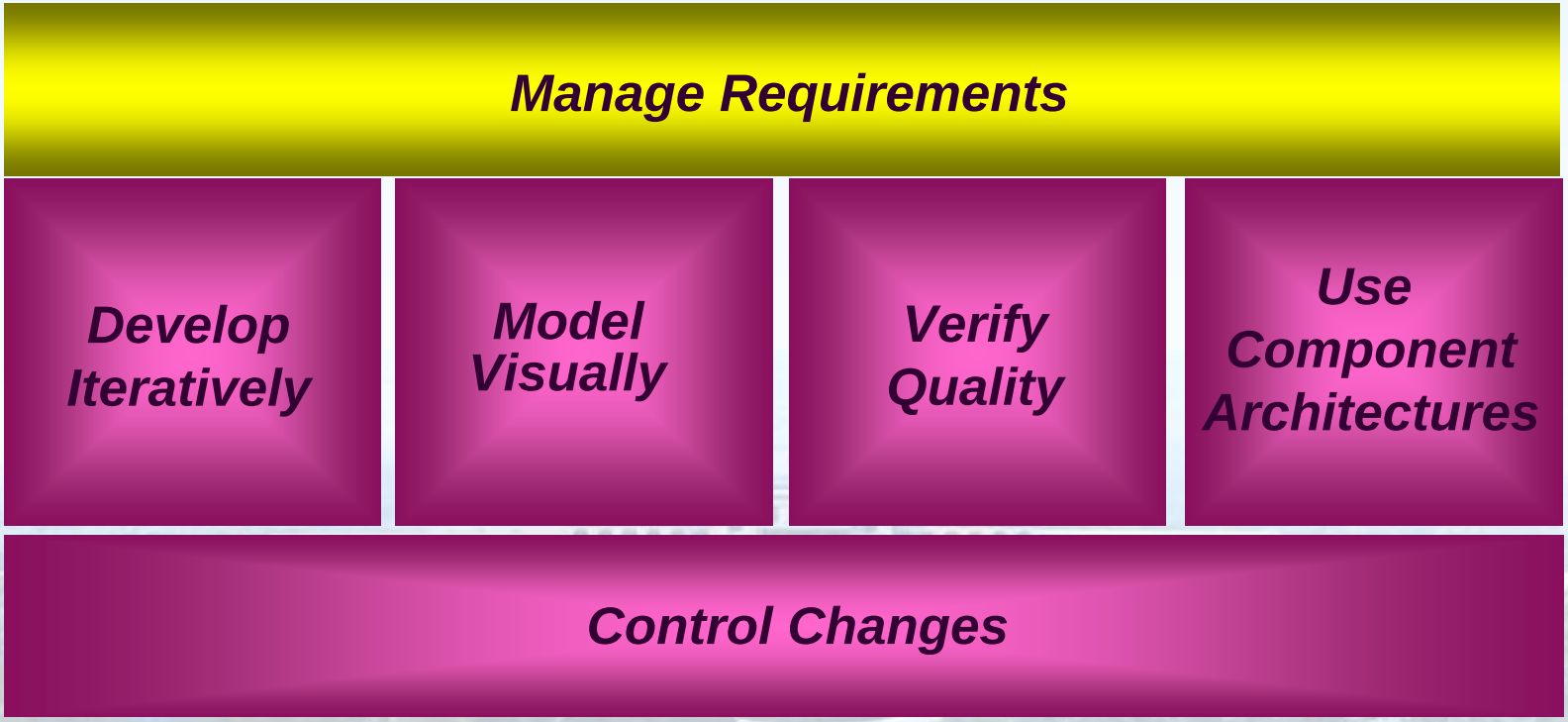
- Iterative Development Characteristics:
  - A Critical risks are resolved before making large investments. – 在进行大规模的投资之前就解决了关键的风险问题。
  - Initial iterations enable early user feedback. – 使得早期的用户反馈在初始迭代中就能出现。
  - Testing and integration are continuous. – 连续进行测试和集成。
  - Objective milestones provide short-term focus. – 各个目标里程碑提供了短期的焦点（阶段性的中心）。
  - Progress is measured by assessing implementations. – 对过程的测量是通过对实现的评定（而不仅仅是文档）来进行的。
  - Partial implementations can be deployed. – 可以对局部的实现进行部署。

4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

2. Manage Your Requirements:



4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

2. Manage Your Requirements:

- Elicit（得出）, organize, and document required functionality and constraints.
- Track and document tradeoffs（折衷） and decisions.
- Business requirements are easily captured and communicated through use cases.
- Use cases（用例） are important planning instruments.

4.4 Rational统一开发过程（RUP）

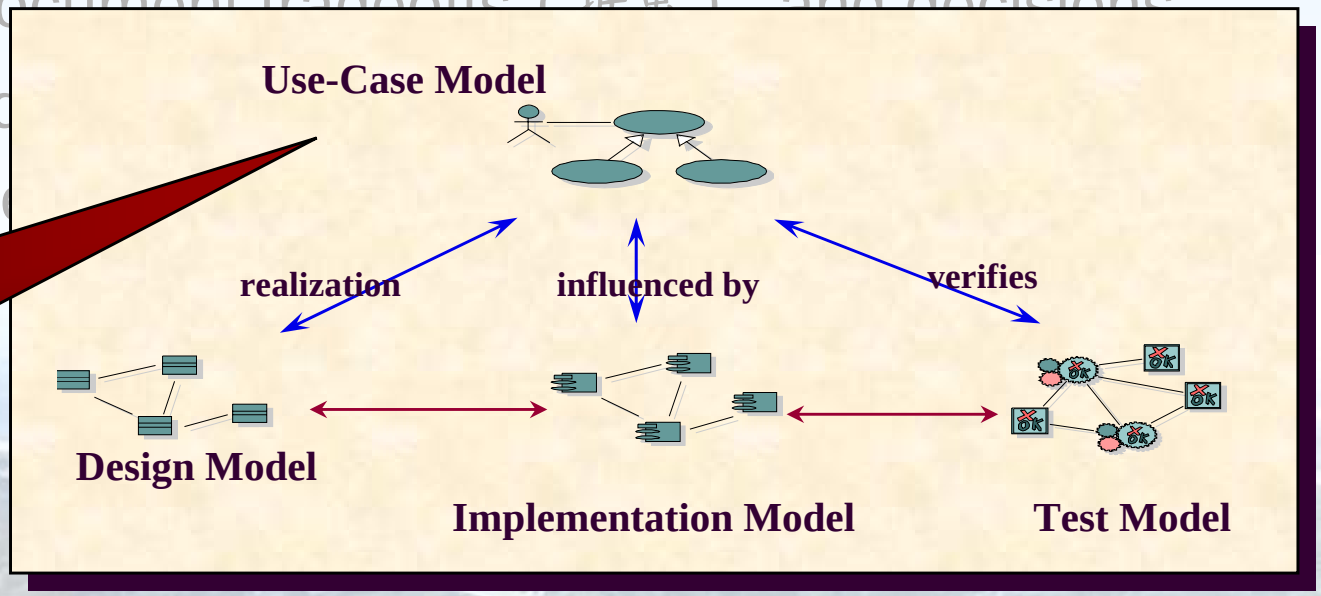
4.4.2 RUP的思路：Implementing Best Practices

– Booch

2. Manage Your Requirements:

- Elicit（得出）, organize, and document required functionality and constraints.
- Track and document tradeoffs（折衷）, and decisions
- Business rec
communicat

Use Cases drives the work from analysis through test

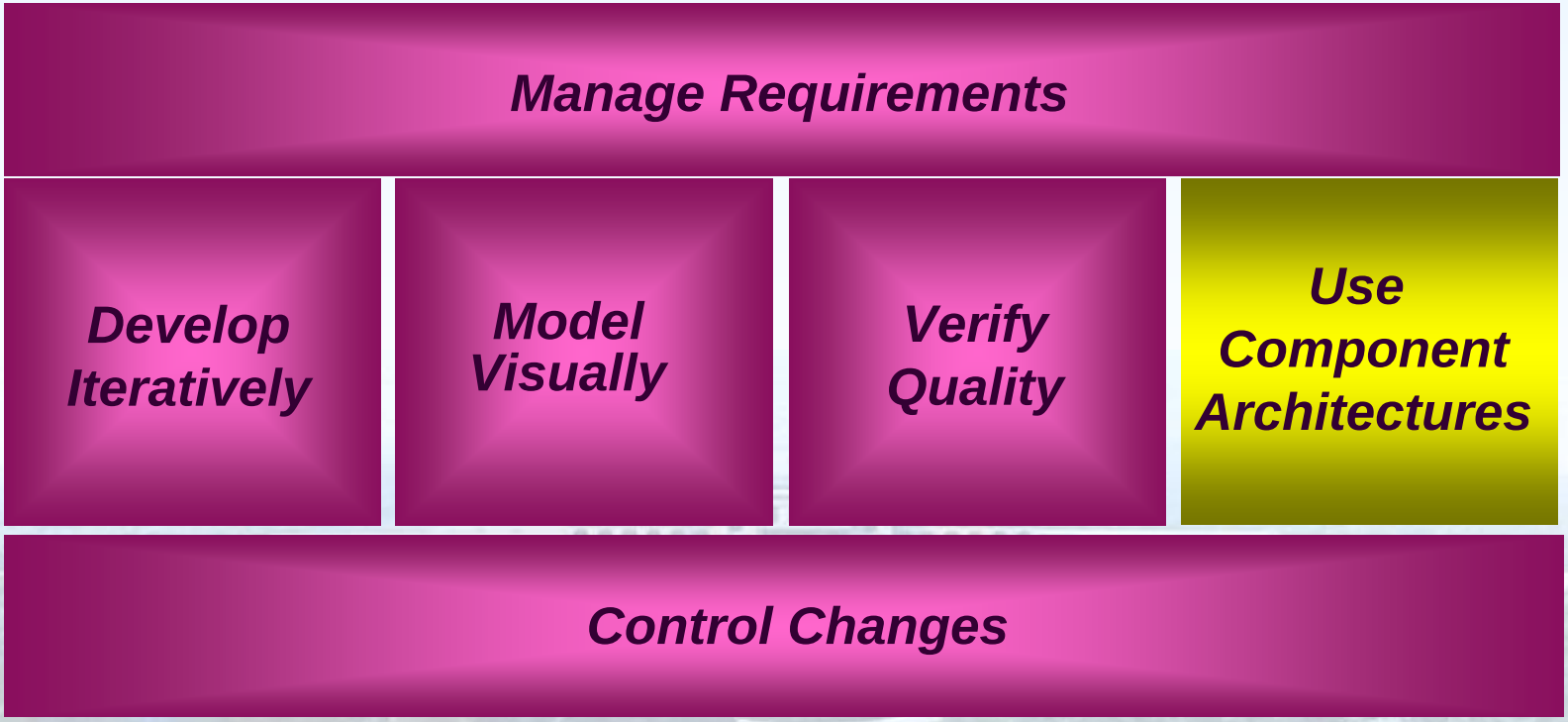


4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

3. Employ Component-based Architecture:



4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

3. Employ Component-based Architecture:

- Design, implement and test your architecture up-front! – 对体系结构进行自下而上的设计、实现和测试。
- A systematic approach to define a “good” architecture – 用一种系统化的做法来定义好的体系结构。
  - resilient to change by using well-defined interfaces – 采用定义明确的接口来使得变更有弹性。
  - by using and reverse engineering components – 采用现成的和通过逆向工程得到的构件。
  - derived from top rank use cases – 由高级别的用例来驱动。
  - intuitively understandable – 易于直观上的理解。

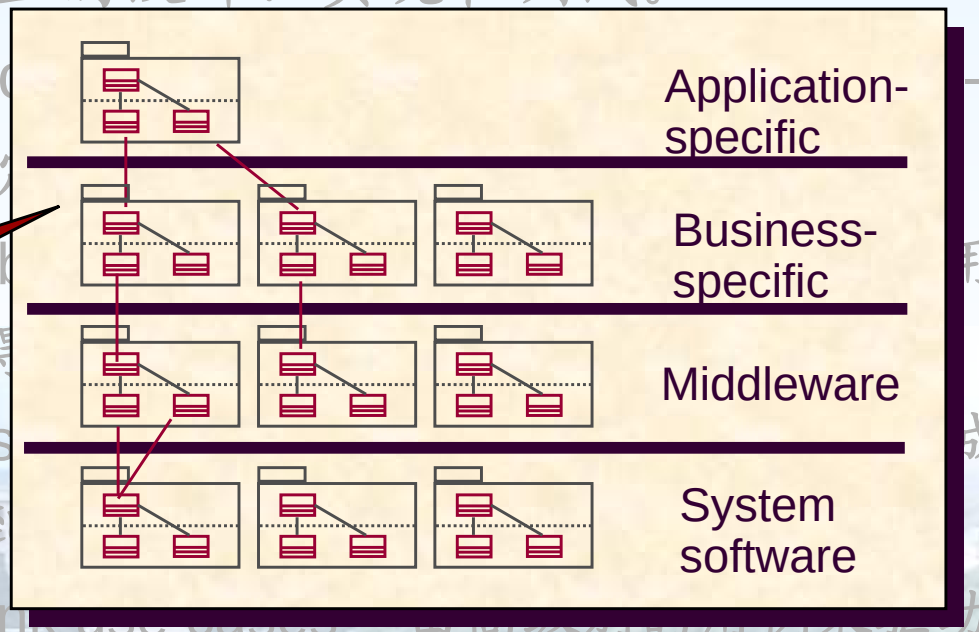
4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

3. Employ Component-based Architecture:

- Design, implement and test your architecture up-front! – 对体系结构进行自下而上的设计、实现和测试。
- A systematic approac – 用一种系统化的做法来
  - resilient to cha 用定使得
  - ers 成的得到
  - derived from top rank use cases 动。
  - intuitively understandable – 易于直观上的理解。



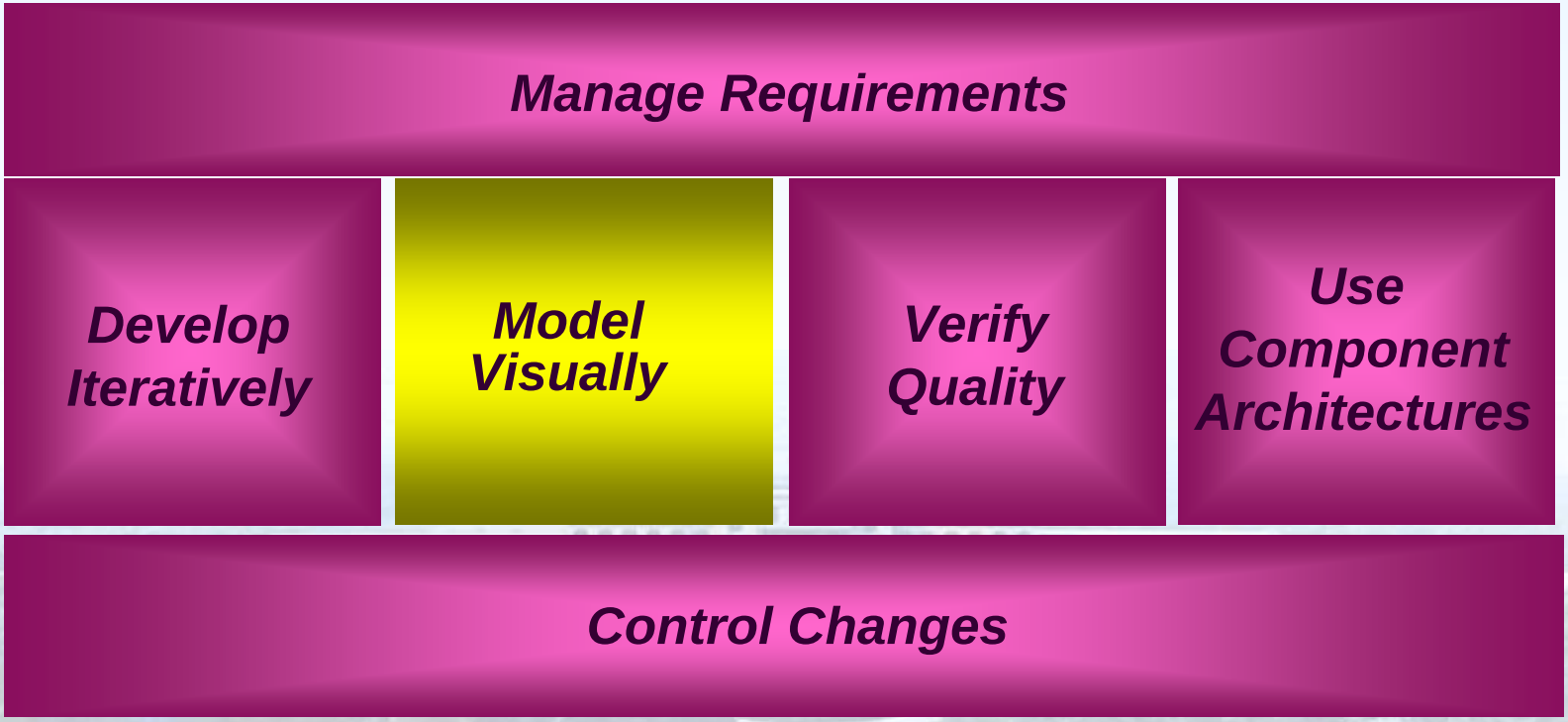
Component-based Architecture with layers

4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

4. Model Software Visually:



4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

4. Model Software Visually:

- Capture the structure and behavior of architectures and components.
- Show how the elements of the system fit together.
- Maintain consistency between a design and its implementation.
- Promote unambiguous communication.

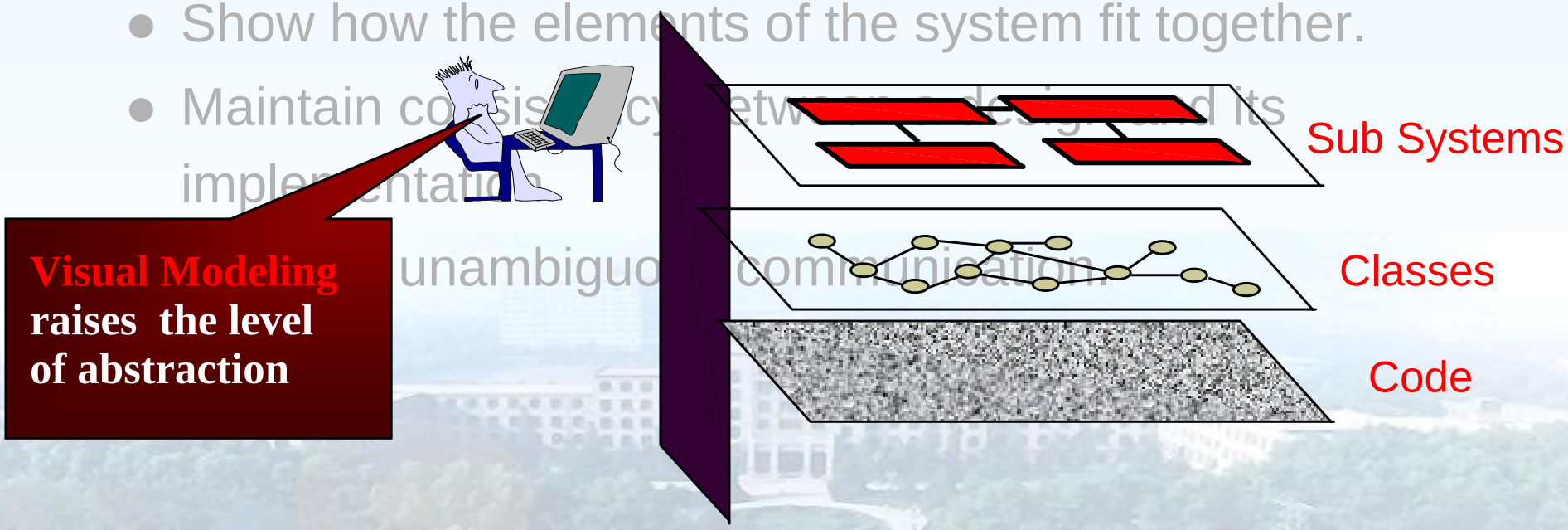
4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

4. Model Software Visually:

- Capture the structure and behavior of architectures and components.
- Show how the elements of the system fit together.
- Maintain consistency between a design and its implementation.
- Facilitate unambiguous communication.

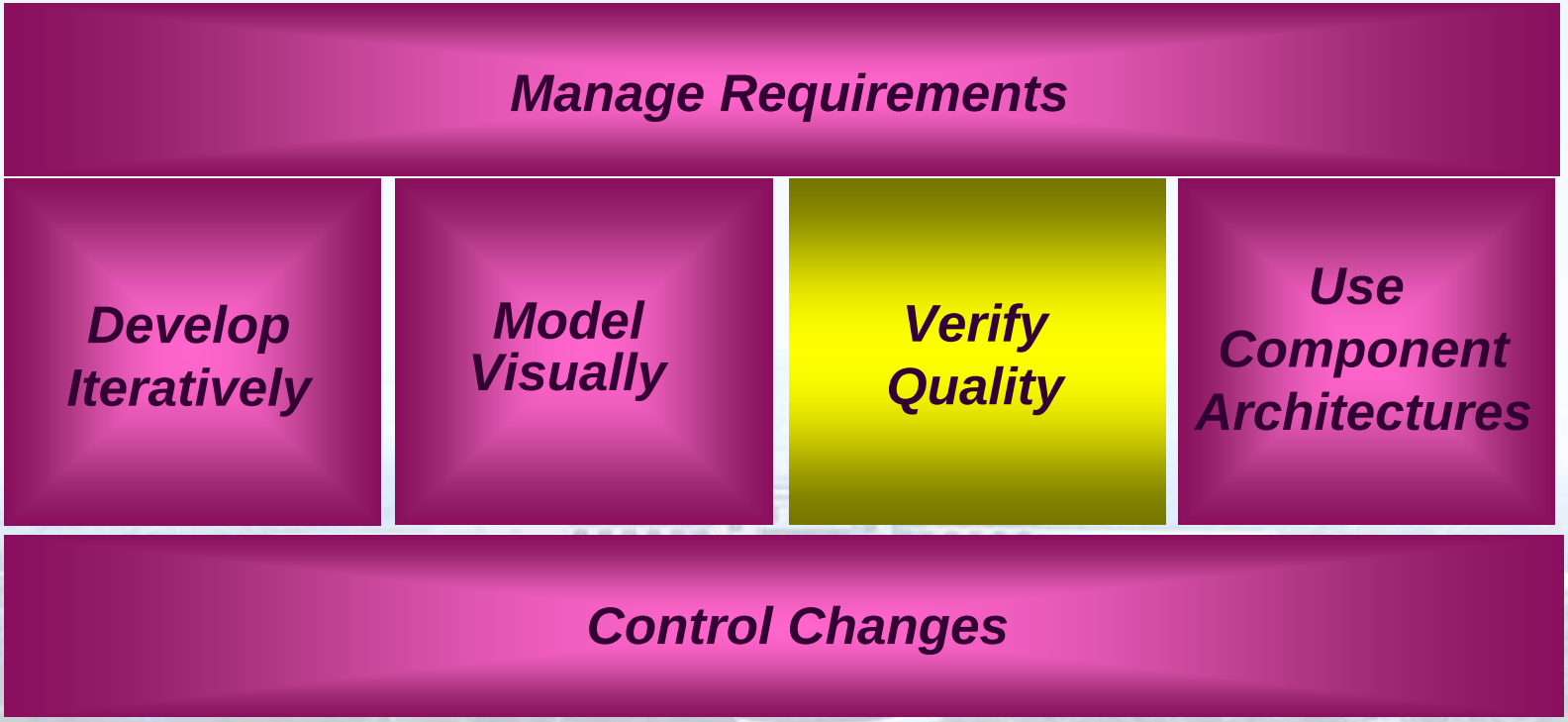


4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

5. Verify Software Quality:



4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

5. Verify Software Quality:

- Create tests for each key scenario to ensure that all requirements are properly implemented.
- Unacceptable application performance hurts as much as unacceptable reliability.
- Verify software reliability - memory leaks（内存泄漏）, bottle necks（性能瓶颈）.
- Test every iteration - automate test!

4.4 Rational统一开发过程（RUP）

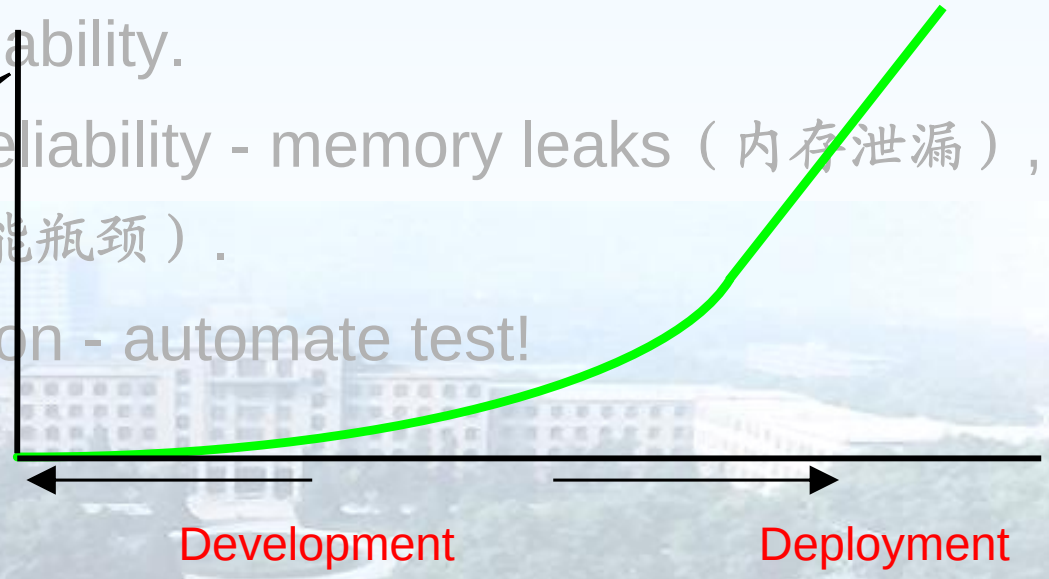
4.4.2 RUP的思路：Implementing Best Practices

– Booch

5. Verify Software Quality:

- Create tests for each key scenario to ensure that all requirements are properly implemented.
- Unacceptable application performance hurts as much as unacceptable reliability.
- Verify software reliability - memory leaks（内存泄漏）, bottlenecks（性能瓶颈）.
- Integration - automate test!

Software problems are 100 to 1000 times more costly to find and repair after deployment

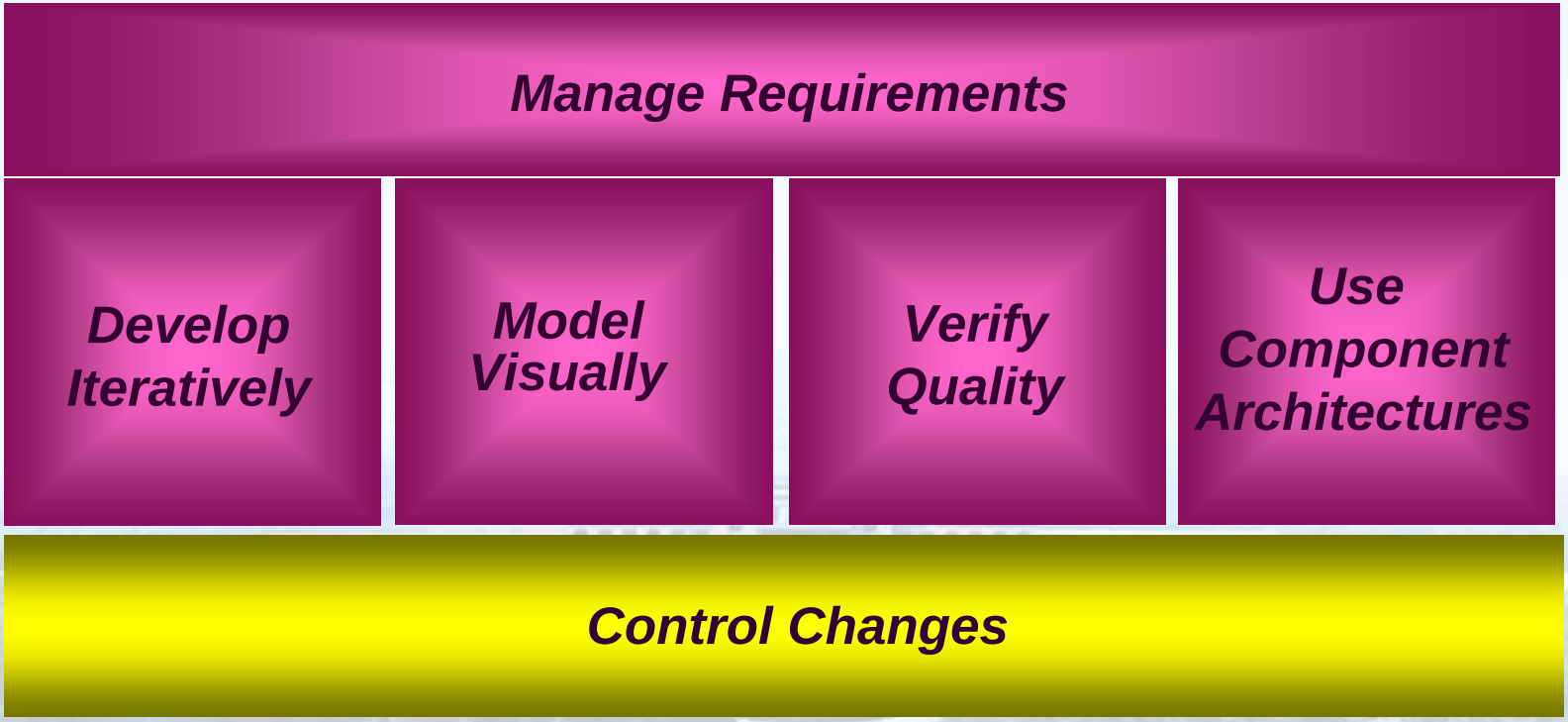


4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

6. Control Changes to Software:



4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

6. Control Changes to Software:

- Control, track and monitor changes to enable iterative development.
- Establish secure workspaces for each developer
  - Provide isolation from changes made in other workspaces.
  - Control all software artifacts - models, code, docs, etc.
- Automate integration and build management.

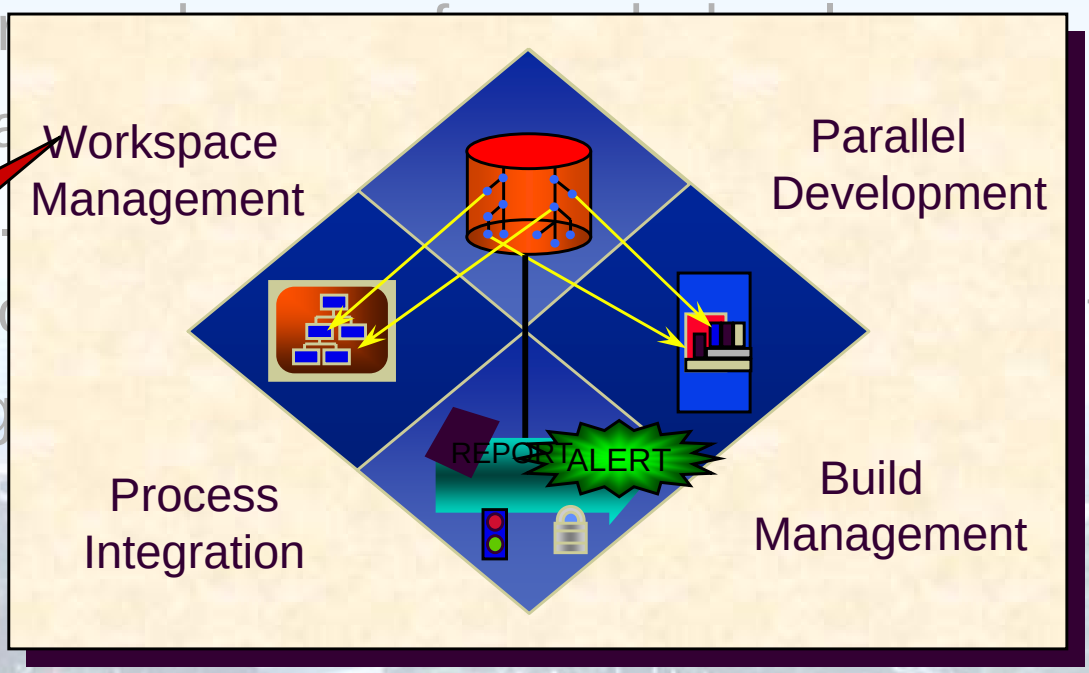
4.4 Rational统一开发过程（RUP）

4.4.2 RUP的思路：Implementing Best Practices

– Booch

6. Control Changes to Software:
   - Control, track and monitor changes to enable iterative development.
   - Establish secur
     - Provide isola
       workspaces.
       tc.

CM is more than just check-in and check-out



4.4 Rational统一开发过程（RUP）

4.4.3 RUP的基本特征

- 受控的迭代式增量开发
- 用例（Use Cases）驱动
- 以软件体系结构为中心

4.4 Rational统一开发过程（RUP）

4.4.3 RUP的基本特征

- 受控的迭代式增量开发:
  - 将软件开发分为一系列小的迭代过程，在每个迭代过程中逐步增加信息、进行细化;
  - 根据具体情况决定迭代的次数、每次迭代延续的时间以及迭代工作流;
  - 每次迭代都选择目前对风险影响最大的用例进行，以分解和降低风险。

4.4 Rational统一开发过程（RUP）

4.4.3 RUP的基本特征

- 用例驱动:
  - 采用用例来捕获对目标系统的功能需求;
  - 采用用例来驱动软件的整个开发过程，保证需求的可跟踪性，确保系统所有功能均被实现;
  - 将用户关心的软件系统的业务功能模型和开发人员关心的目标软件系统的功能实体模型结合起来，提供一种贯穿整个软件生存周期的开发方式，使得软件开发的各个阶段的工作自然、一致地协调起来。

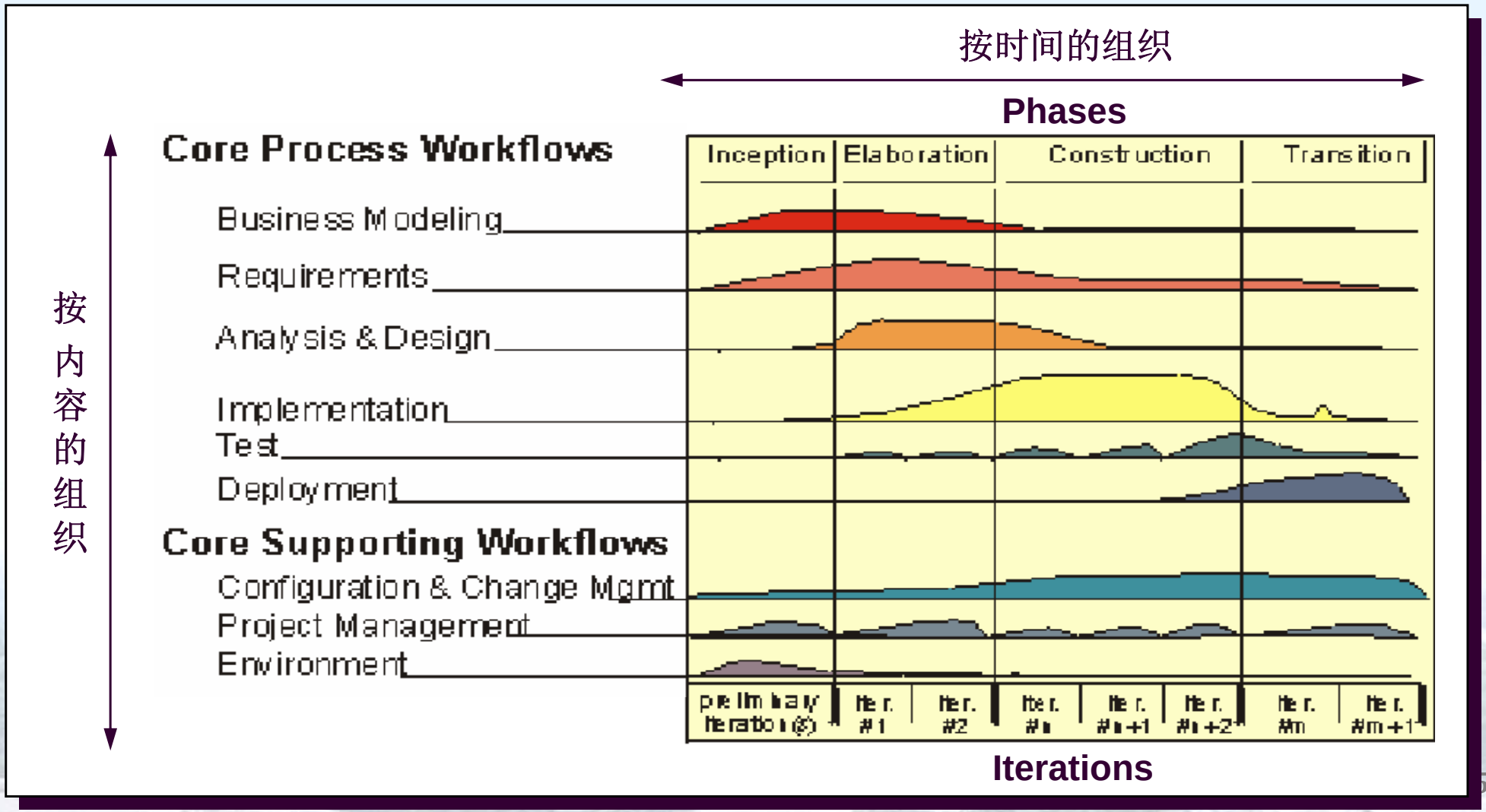
4.4 Rational统一开发过程（RUP）

4.4.3 RUP的基本特征

- 以软件体系结构为中心:
  - 强调在开发过程的早期，识别出与软件体系结构紧密相关的用例，并通过对这些用例的分析、设计、实现和测试，形成体系结构框架;
  - 在后续阶段中对已形成的体系结构框架进行不断细化，最终实现整个系统;
  - 在开发过程的早期形成良好的软件体系结构，有利于对系统的理解、支持重用和有效的组织软件开发。

4.4 Rational统一开发过程（RUP）

4.4.4 RUP 中受控的迭代式增量开发的二维模型



4.4 Rational统一开发过程（RUP）

4.4.4 RUP 中受控的迭代式增量开发的二维模型

- 时间（Time）
  - 周期（Cycles）：一个RUP可以分为若干个周期
    - 阶段（Phases）：起始、演化、构造、提交
      - 迭代（Iterations）：每个阶段进行若干次
- 核心工作流（Core Workflow）
  - 工作流（Workflow）：对应于特定的迭代的连续活动
    - 活动（Activities）：需求定义、分析、设计、实现和测试
      - 中间制品（Artifacts）：活动的结果

4.4 Rational统一开发过程（RUP）

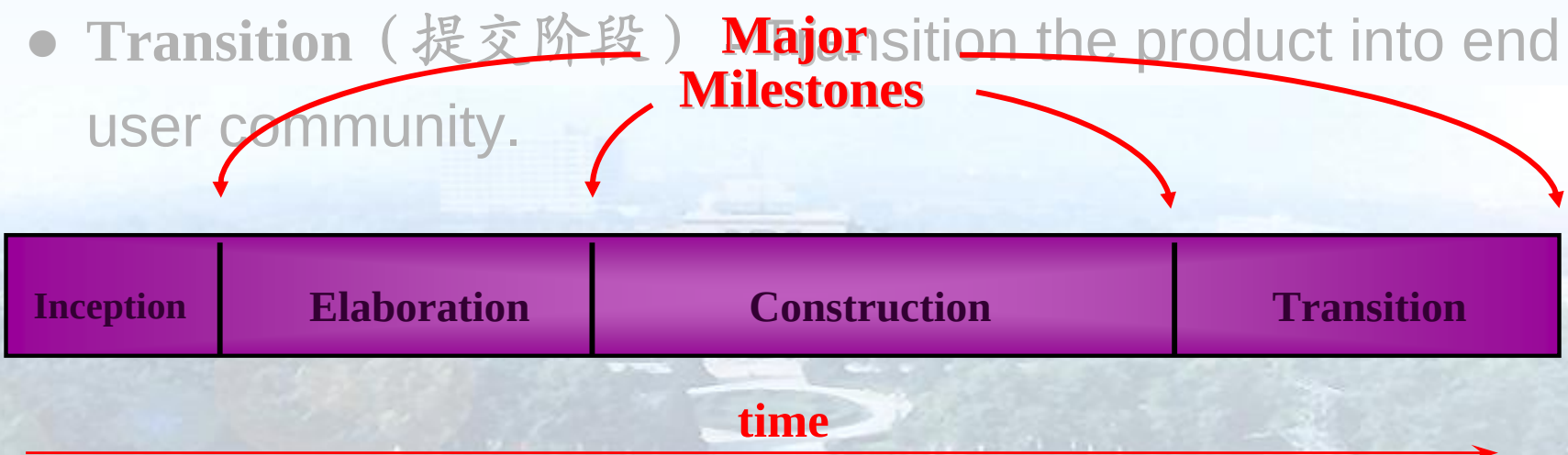
4.4.4 RUP 中受控的迭代式增量开发的二维模型

- **RUP 将一个周期的开发过程分为四个阶段:**
  - **Inception**（起始阶段）- Define the scope of project.
  - **Elaboration**（演化阶段） - Plan project, specify features, baseline architecture.
  - **Construction**（构建阶段） - Build the product.
  - **Transition**（提交阶段） - Transition the product into end user community.

4.4 Rational统一开发过程（RUP）

4.4.4 RUP 中受控的迭代式增量开发的二维模型

- RUP 将一个周期的开发过程分为四个阶段:
  - Inception（起始阶段）- Define the scope of project.
  - Elaboration（演化阶段） - Plan project, specify features, baseline architecture.
  - Construction（构建阶段） - Build the product.
  - Transition（提交阶段） - Transition the product into end user community.



4.4 Rational统一开发过程（RUP）

4.4.4 RUP 中受控的迭代式增量开发的二维模型

- 起始阶段:
  - 意图:
    - To establish the business case（*商业计划*） for a new system or for a major update of an existing system
    - To specify the project scope
  - 结果:
    - A general vision of the project's requirements, i.e., the core requirements
      - Initial use-case model and domain model (10-20% complete)
    - An initial business case, including:
      - Success criteria
      - An initial risk assessment
      - An estimate of resources required

4.4 Rational统一开发过程（RUP）

4.4.4 RUP 中受控的迭代式增量开发的二维模型

- 演化阶段:
  - 意图:
    - To analyze the problem domain
    - To establish a sound architectural foundation
    - To address（标定） the highest risk elements of the project
    - To develop a comprehensive plan showing how the project will be completed
  - 结果:
    - Use-case and domain model 80% complete
    - An executable architecture and accompanying documentation
    - A revised business case, including revised risk assessment
    - A development plan for the overall project

4.4 Rational统一开发过程（RUP）

4.4.4 RUP 中受控的迭代式增量开发的二维模型

- 构建阶段:
  - 意图:
    - To incrementally develop a complete software product which is ready to transition into the user community
  - 产品:
    - A complete use-case and design model
    - Executable releases of increasing functionality
    - User documentation
    - Deployment documentation
    - Evaluation criteria for each iteration
    - Release descriptions, including quality assurance results
    - Updated development plan

4.4 Rational统一开发过程（RUP）

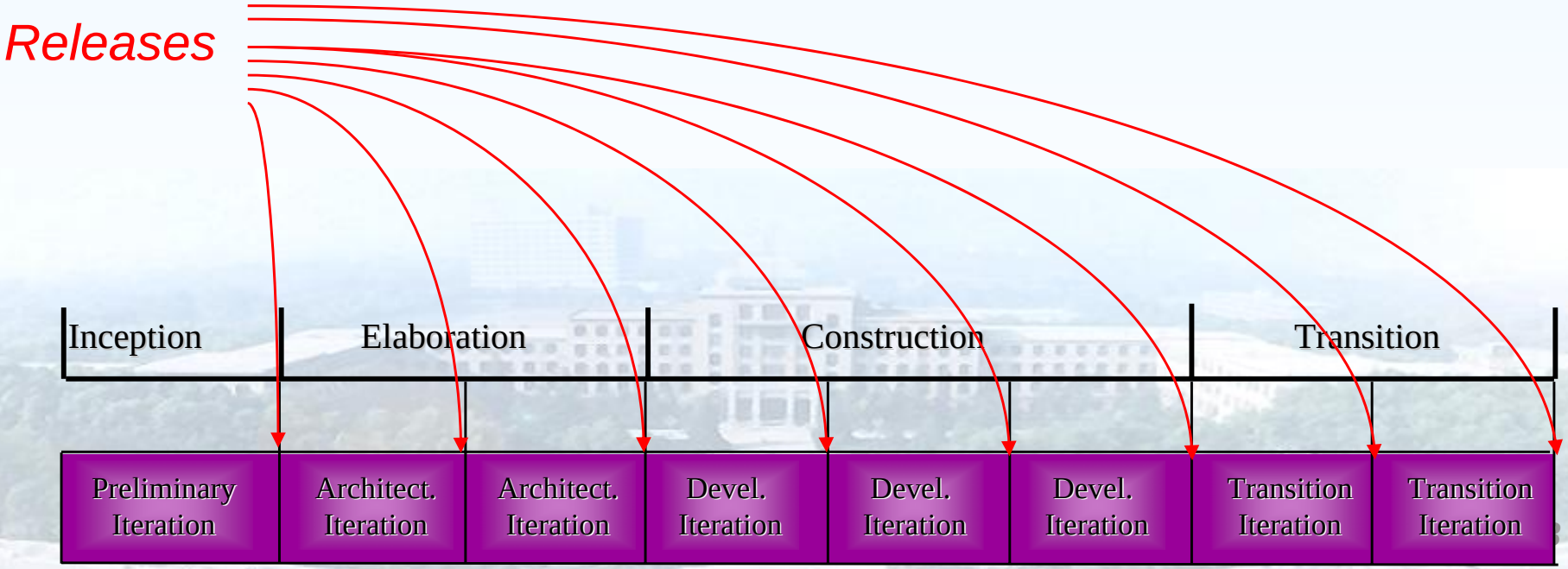
4.4.4 RUP 中受控的迭代式增量开发的二维模型

- 提交阶段:
  - 意图:
    - To transition the software product into the user community
  - 产品:
    - Executable releases
    - Updated system models
    - Evaluation criteria for each iteration
    - Release descriptions, including quality assurance results
    - Updated user manuals
    - Updated deployment documentation
    - “Post-mortem” analysis of project performance

4.4 Rational统一开发过程（RUP）

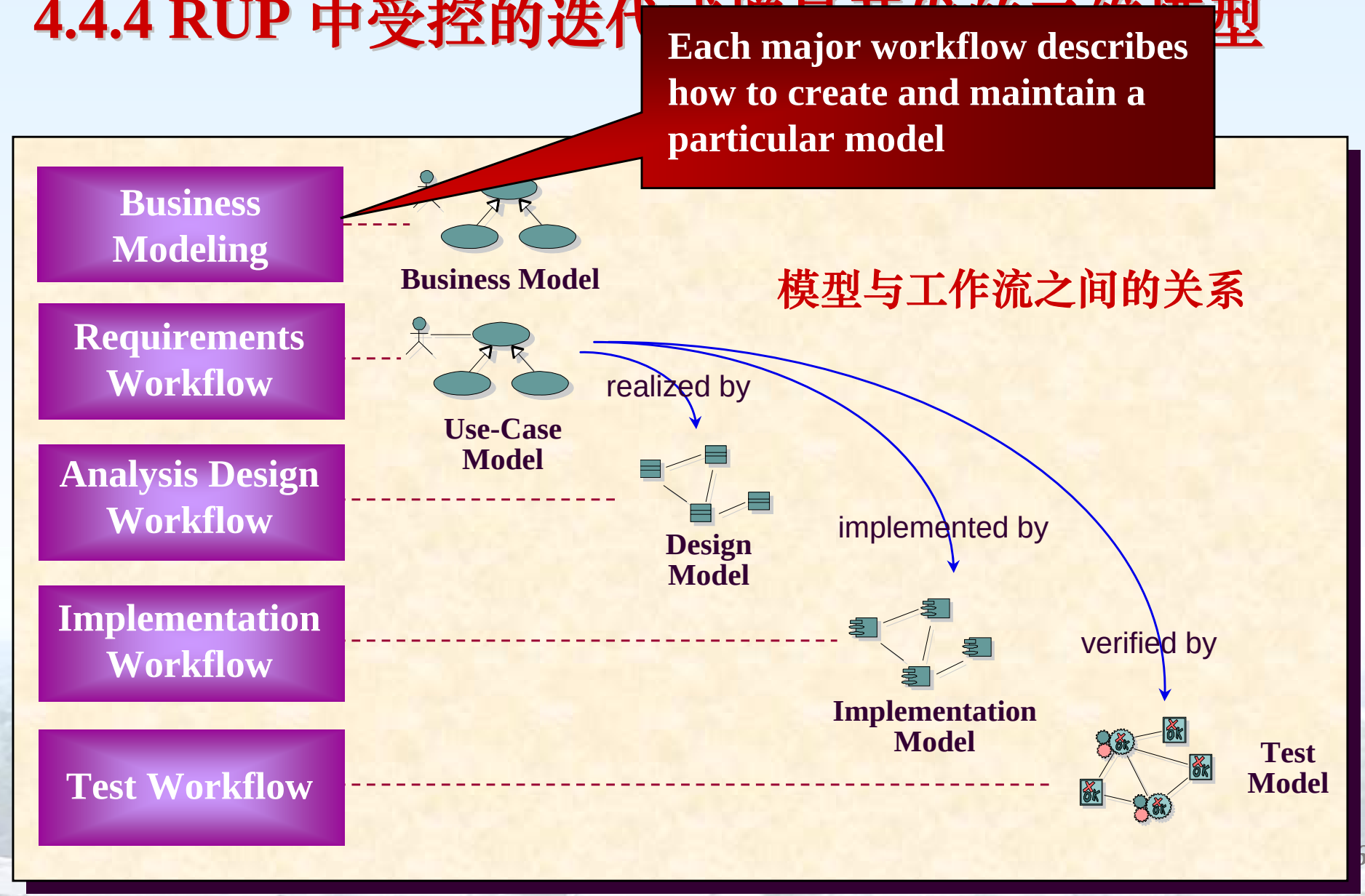
4.4.4 RUP 中受控的迭代式增量开发的二维模型

- 迭代与阶段之间的关系：
  - An iteration is a distinct sequence of activities with an established plan and evaluation criteria, resulting in an executable release (internal or external).



4.4 Rational统一开发过程（RUP）

4.4.4 RUP 中受控的迭代式增量开发的工作模型



4.4 Rational统一开发过程（RUP）

4.4.5 RUP产品

- **RUP 是一个基于 Web 的产品:**
  - Interactive knowledge base accessible from tools.
  - Powerful graphical navigation, search engine, index …
  - Guidelines, templates, tool mentors at your finger tips.

4.4 Rational统一开发过程（RUP）

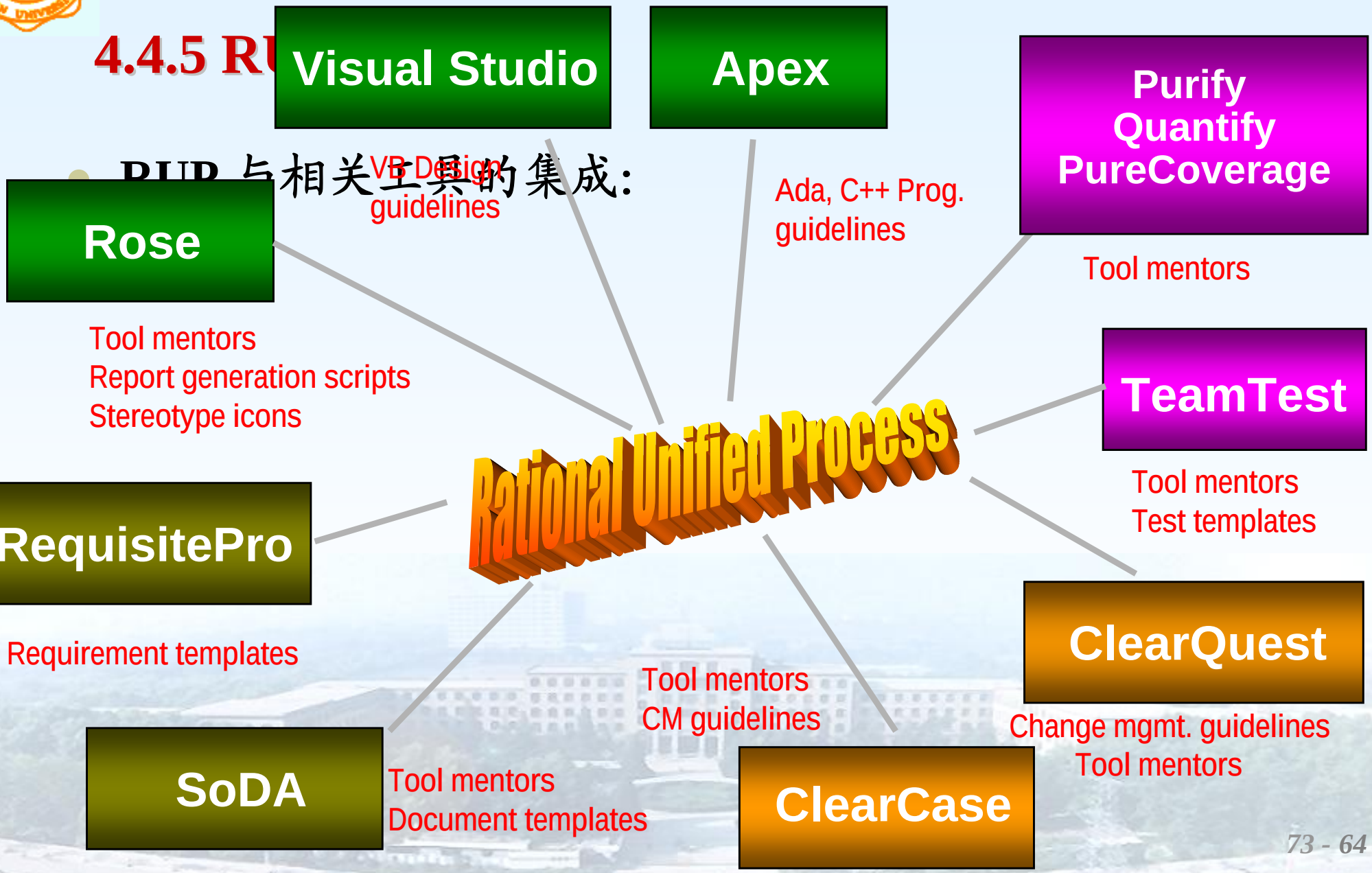
4.4.5 RUP产品

- **RUP 中的知识库的内容包括:**
  - Extensive guidelines for all team members.
  - Tool mentors (most Rational tools).
  - Templates
    - Rational Rose (examples and template for how to structure your Rose models)
    - Word (30+)
    - SoDA (10+)
    - MS Project
  - Development kit - guidelines, tools, templates for customizing the process.
  - Access to Resource Center (white papers, updates, hints, and add-on products).

4.4 Rational统一开发过程（RUP）

4.4.5 RU

- RUP与相关工具的集成:



4.4 Rational统一开发过程（RUP）

4.4.6 RUP带来的观念变化

- 更强的计划性：
  - 迭代开发意味着要有更强的预见性和计划性，阶段的划分、阶段内的迭代都需要仔细规划。这要求项目管理者承担更大的责任，而所换来的则是开发任务的具体化和可预见性。
- 坦然面对迭代过程中一部分中间制品的推倒重来：
  - 不要恐惧这样的现实，由于迭代过程的细化和相应工具的支持，其影响是可以控制的。
- 坦然面对中间制品的“不美观”：
  - 在一些迭代中产生的中间制品，虽然外观上不能令用户和投资者满意，但其作用和价值是完美的。这时，项目管理者要充当一个“意见缓冲区”，对外树立可信赖和讲信用的形象。

4.4 Rational统一开发过程（RUP）

4.4.6 RUP带来的观念变化

- 尽早进行困难的工作:
  - 强调与实现的关系密切，而将困难工作放在开发后期进行是十分有害的，会使开发后期突然遇到“集成地狱”而使开发过程失去控制。
- 把软件放在首位:
  - 过分强调规格说明（问题空间的描述）的作用是不恰当的，因为用户所购买的是软件（解空间）而不是规格说明。在开发过程中，需求和规格说明都是允许变化的。
- 加强开发过程监控与量化管理:
  - 项目管理者要注重对各种变化与扰动的测量与监控，应先示范，后发布文档，以使得开发人员信服。

4.4 Rational统一开发过程（RUP）

4.4.6 RUP带来的观念变化

- 既需要好的项目管理者，也需要好的体系结构设计师：
  - 一个成功的软件项目同时需要这两种人，而且项目管理者本人就应当懂得体系结构设计。
- 确定迭代的数量、持续时间和内容：
  - RUP给出了一定的指南，但主要靠项目管理者根据实际情况确定，因而责任更加重大。

4.4 Rational统一开发过程（RUP）

4.4.7 我们使用 RUP 的体会

- 确实是一种可操作的软件开发过程;
- 典型的分而治之（*Divide and Conquer*）思想;
- 理解 **RUP** 比掌握 **UML** 容易，且总体效果上更明显;
- 由于将开发过程划分成若干较小的子过程/迭代，易于预测进度、降低风险;
- 设计每个迭代时，规定了迭代任务、建模目标、开发工具和程序设计语言，使得项目管理更为具体;
- 在开发早期就得出了关键部件的基本实现结果，对于建立和增强用户和开发组成员的信心都十分有利;

4.4 Rational统一开发过程（RUP）

4.4.7 我们使用 RUP 的体会

- 对项目管理者的素质和责任要求有十分明显的提高;
- 最难的是确定迭代数量、顺序和内容，RUP 本身没有现成的答案，需要积累和提炼，需要经验;
- 就迭代开发过程本身，估计并不一定非要采用面向对象开发方法和 UML（可能效果会有差别）。

4.4 Rational统一开发过程（RUP）

4.4.8 从软件工程发展过程看 RUP

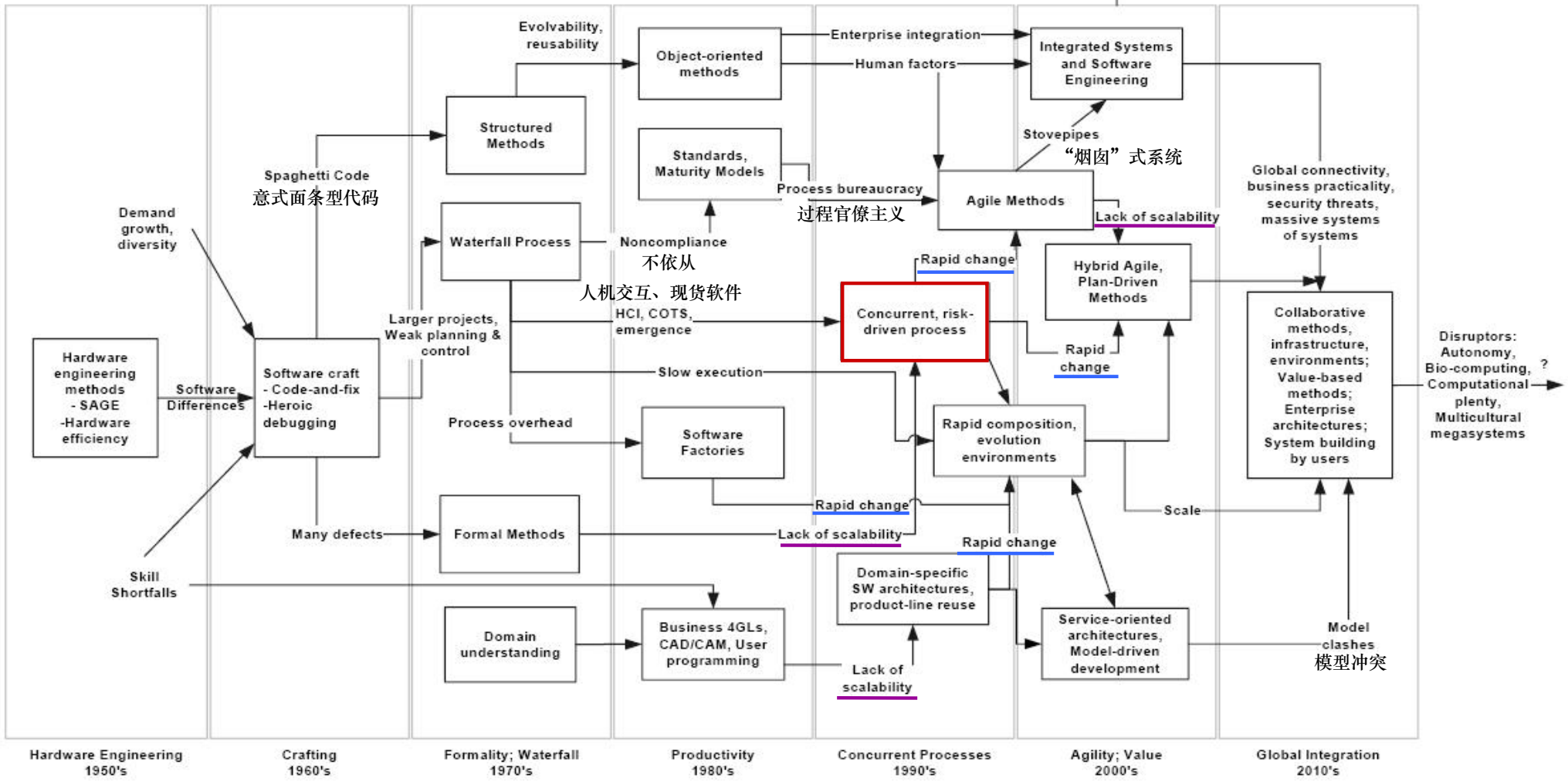


Figure 6. A Full Range of Software Engineering Trends （*Barry Boehm* 论文）

要点与引伸

- 软件开发过程与软件开发方法之间并没有一一对应的关系。
- **RUP** 带来的观念变化影响了软件工程的许多基本概念。
- 对软件开发过程的管理是为了更好地支持和促进软件开发，而不是制约软件开发。
- 软件开发成功与否的标志，不只是开发出实现了用户需求的产品，而且还包含了时间、成本、对维护与扩充的支持等重要因素，因此需要开发过程的有效支持。

下一次课的内容

- 面向对象系统分析与设计－软件和软件开发过程的多视图特征
  - 为什么单一视图解决不了问题？“4+1”视图；各种视图的 UML 表示手段
- 面向对象系统分析与设计－UML的基本概念、结构和作用
  - UML 的发展过程；UML的目标；UML 的基本概念与结构
- 面向对象系统分析与设计－UML基本表示
  - 包（Package）、依赖（Dependency）、注释（Note）；用例图（Use-Case Diagram）
- 面向对象系统分析与设计－UML静态建模表示—类图
  - 类（Class）；模板（Template）；概括（Generalization）；聚集（Aggregation）；关联（Association）

下一次课的内容（续）

- 面向对象系统分析与设计－UML动态建模表示
  - 状态图（State-Transition / Statechart Diagram）；
  - 活动图（Activity Diagram）；
  - 序列图（Sequence Diagram）；
  - 协同图（Collaboration Diagram）
- 面向对象系统分析与设计－UML系统实现表示
  - 组件图（Component Diagram）；
  - 部署图（Deployment Diagram）
- 面向对象系统分析与设计－代码生成与文档生成
  - 从模型生成代码;
  - 在Rose中设定代码生成的特性;
  - 文档自动生成的原理