IBM Software Group

监控项目过程，控制项目风险－开发过程监控解决方案

Rational. software

IBM 软件部　雷勇

ON DEMAND BUSINESS

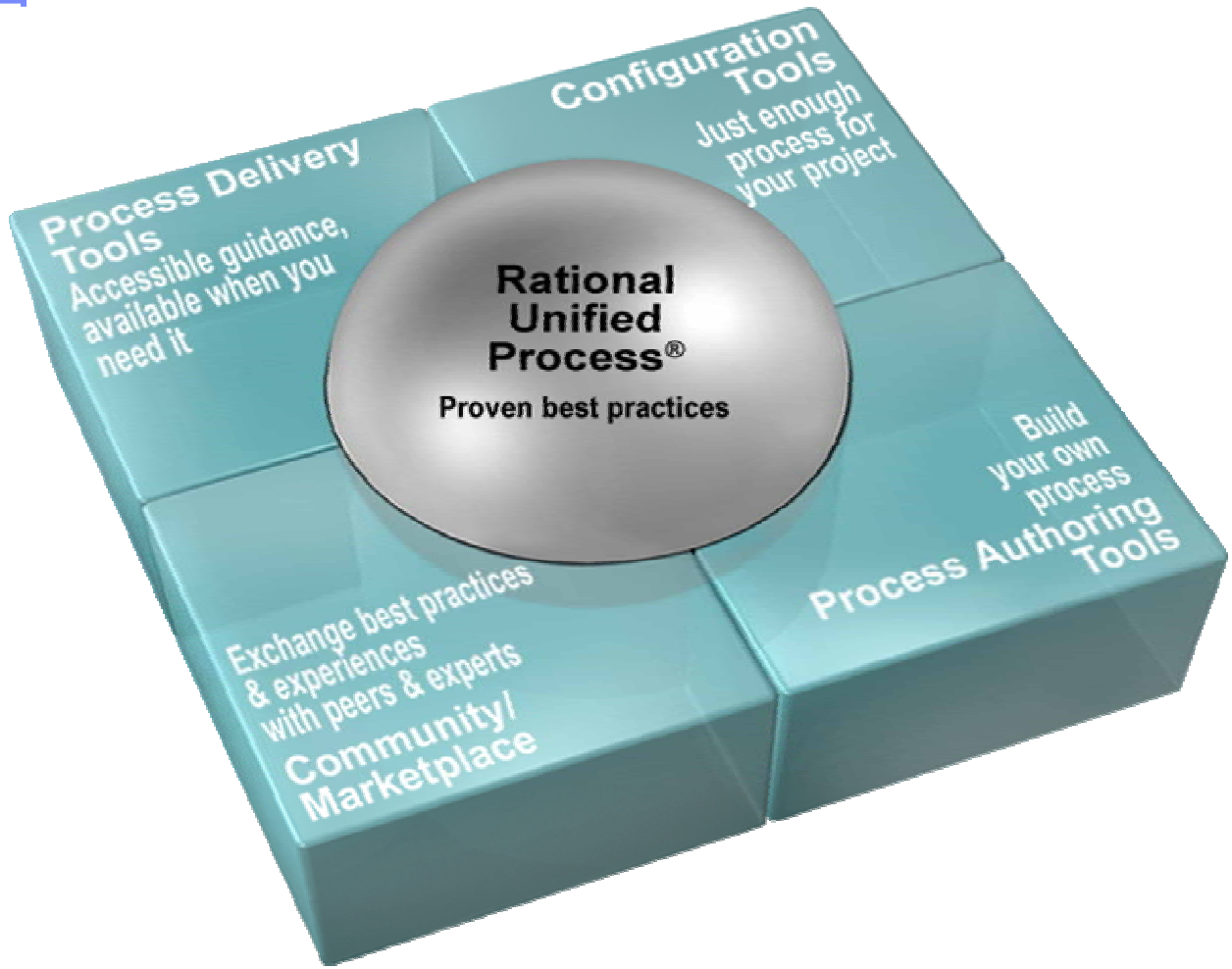
议程

- 统一软件开发过程（Rational Unified Process）
- 管理迭代化的软件开发项目
  - 瀑布模型VS迭代化开发
  - 软件项目的开发阶段
  - 制定项目开发计划
  - 迭代化开发的指导原则

什么是软件开发过程

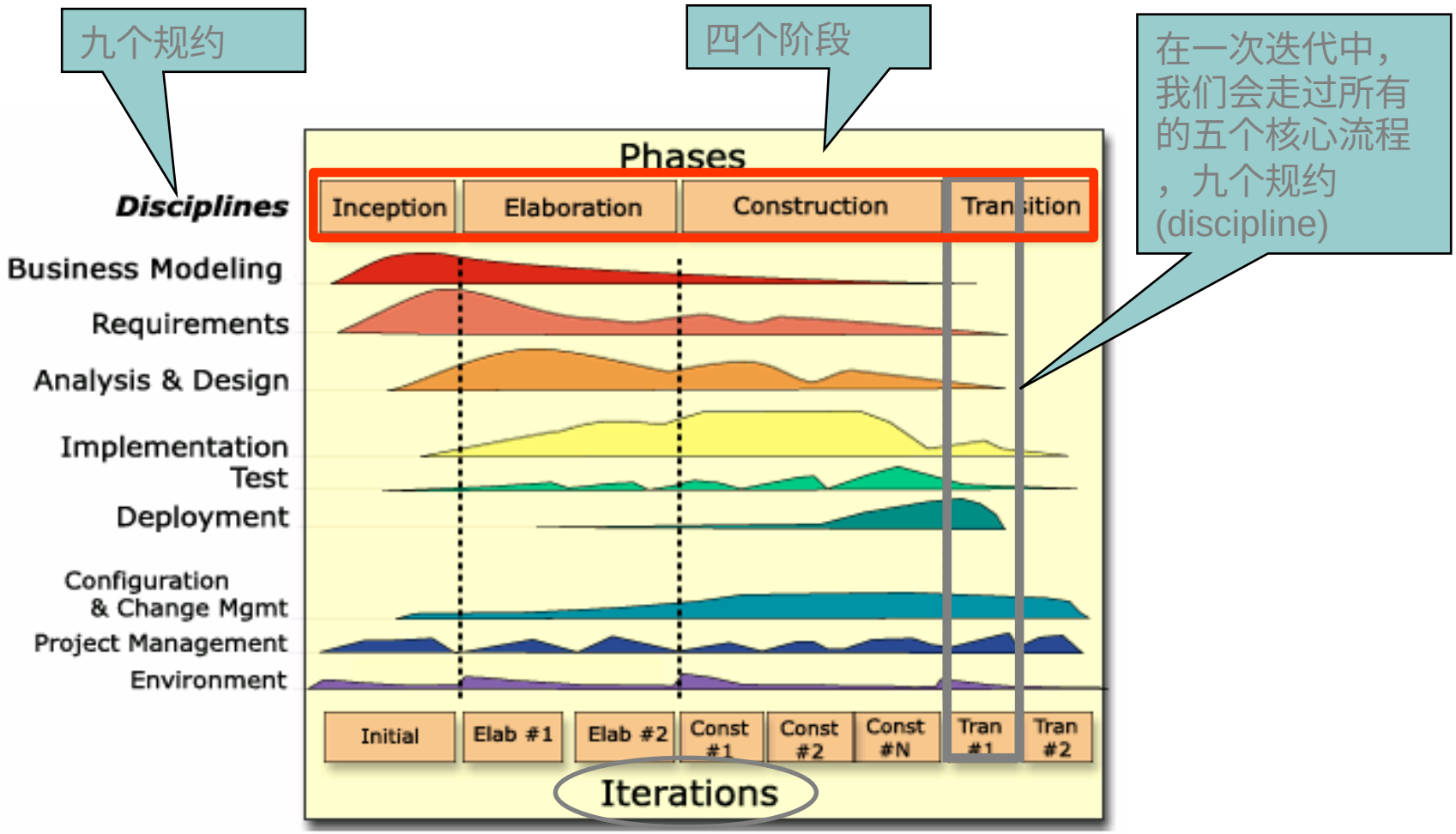
一个过程定义了为达到某个确定的目的，需要什么人在什么时候以何种方式做何种工作。对于软件工程而言，其目标是构造一个新的软件产品或者完善一个旧的软件产品。

RUP平台



Proven. Practical. Flexible.

RUP: 风险驱动, 基于 Use-Case 技术, 以架构为中心, 迭代化, 可配置的软件开发流程



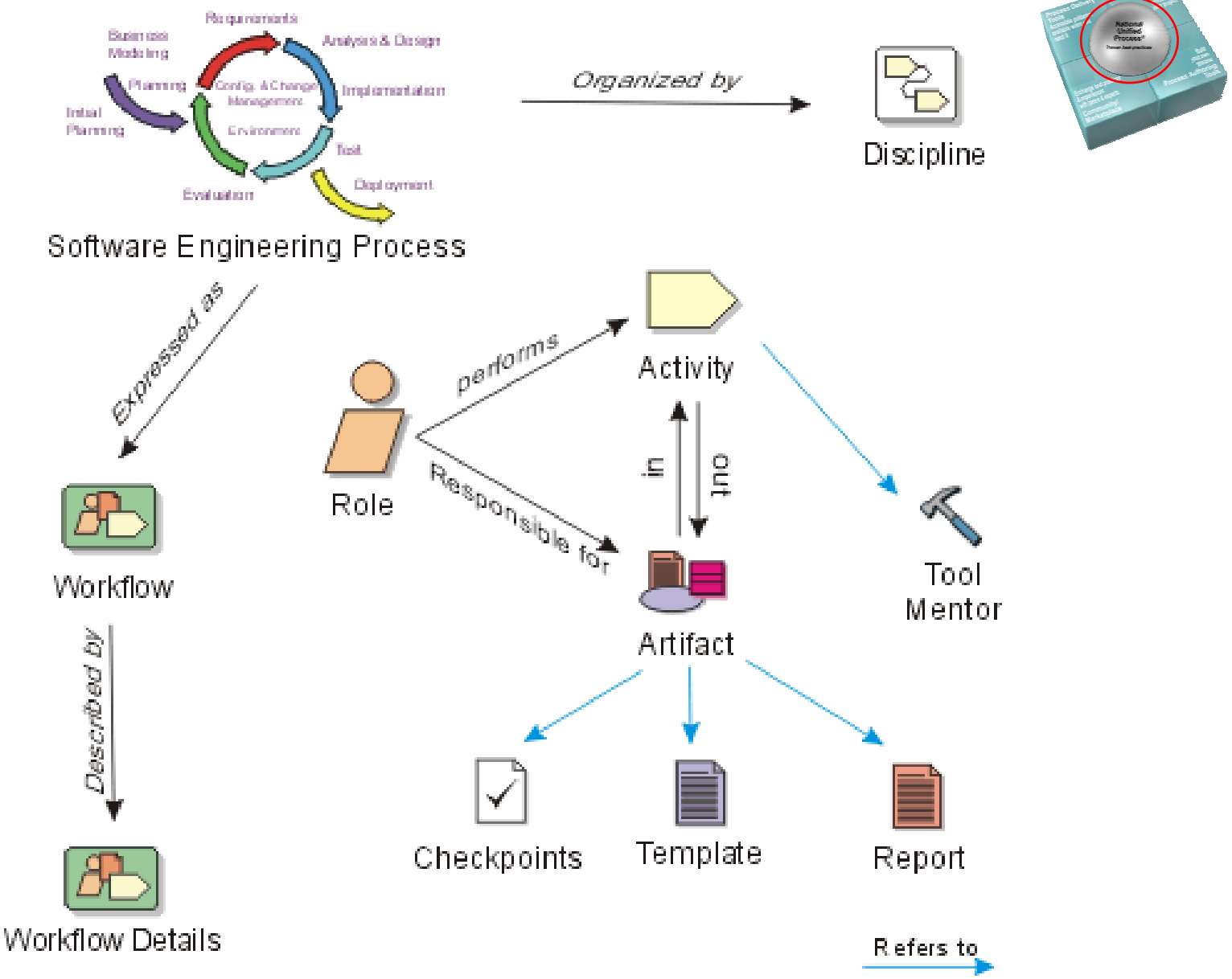
核心概念

- 用例驱动(Use-case driven)表明开发过程是沿着一个流进行，一系列从用例获得的工作流前进的。用例被确定，用例被设计、实现，最后用例成为测试人员构造测试用例的基础。

- 迭代是按预先计划所进行的一系列开发活动，通过迭代会产生一个软件发布结果（内部或外部的）,并且根据预先制定的标准来对该结果进行评估

- 构架是系统在其所处环境中的最高层次的概念。软件系统的构架是通过接口交互的重要构件（在特定时间点）的组织或结构，这些构件又由一些更小的构件和接口组成。构架可以递规解析为通过接口交互的部件、连接部件的关系以及组装部件的一些限制条件。

关键概念


Requirements
Business Modeling
Analysis & Design
Planning
Config. & Change Management
Implementation
Initial Planning
Environment
Test
Evaluation
Deployment
Software Engineering Process
Organized by
Discipline
Expressed as
Workflow
Described by
Workflow Details
Role
performs
Responsible for
Activity
in
out
Artifact
Tool Mentor
Checkpoints
Template
Report
Refers to

关键概念

- **工作流程**：所有角色、活动和工件的简单列举不能组成一个流程；我们需要采用一种有意义的顺序描述产生有价值结果的活动，并显示角色之间的交互作用。一个*工作流程*就是一系列活动，这些活动产生的结果是可见的价值。

- **工作流程明细**：这些图显示所涉及的角色、输入和输出工件、以及执行的活动。

- **角色**：定义了在软件工程组织的环境中，个人或协同工作的多人小组的行为和职责。角色代表项目中个人承担的任务，并定义其如何完成工作。

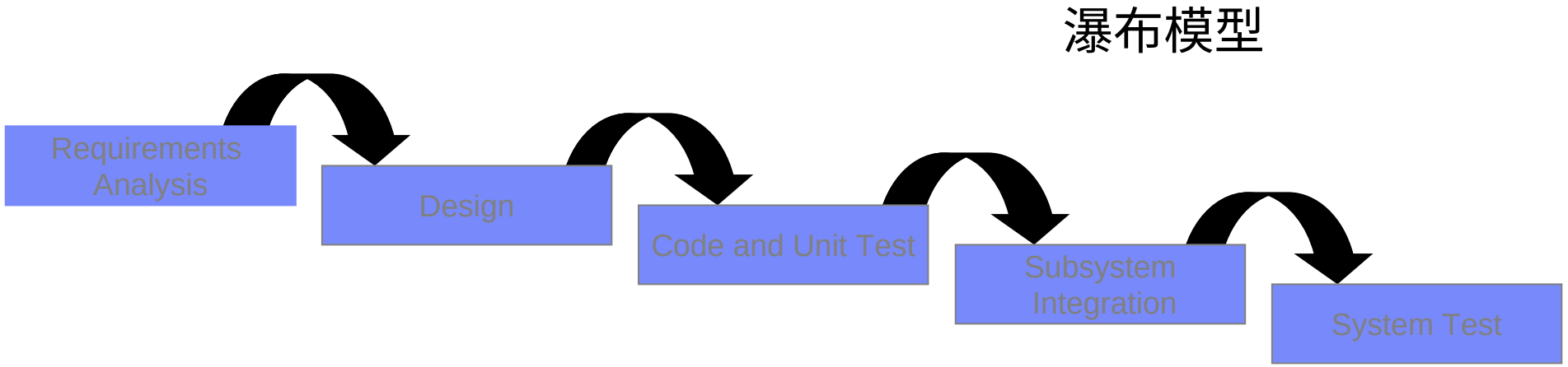
- 角色从事活动，而活动了定义角色执行的工作。**活动**是参与项目的角色为提供符合要求的结果而进行的工作。

- 活动说明主要侧重于工作的结果，而**工件指南**侧重于工作的过程。**检查点**提供快速参考，帮助您评估工件的质量。

议程

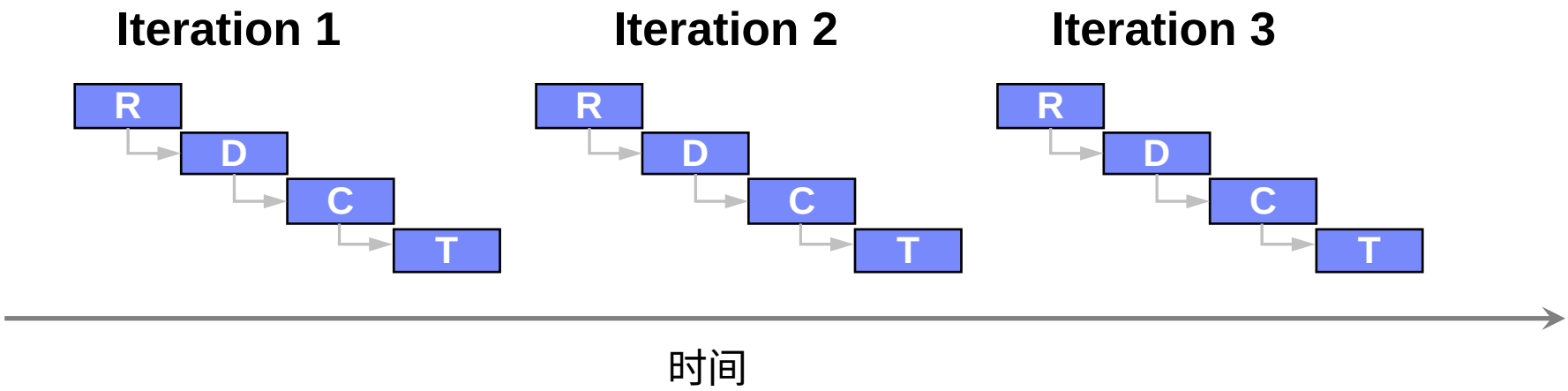
- 统一软件开发过程（Rational Unified Process）
- 管理迭代化的软件开发项目
  - 瀑布模型VS迭代化开发
  - 软件项目的开发阶段
  - 制定项目开发计划
  - 迭代化开发的指导原则

传统的瀑布模型



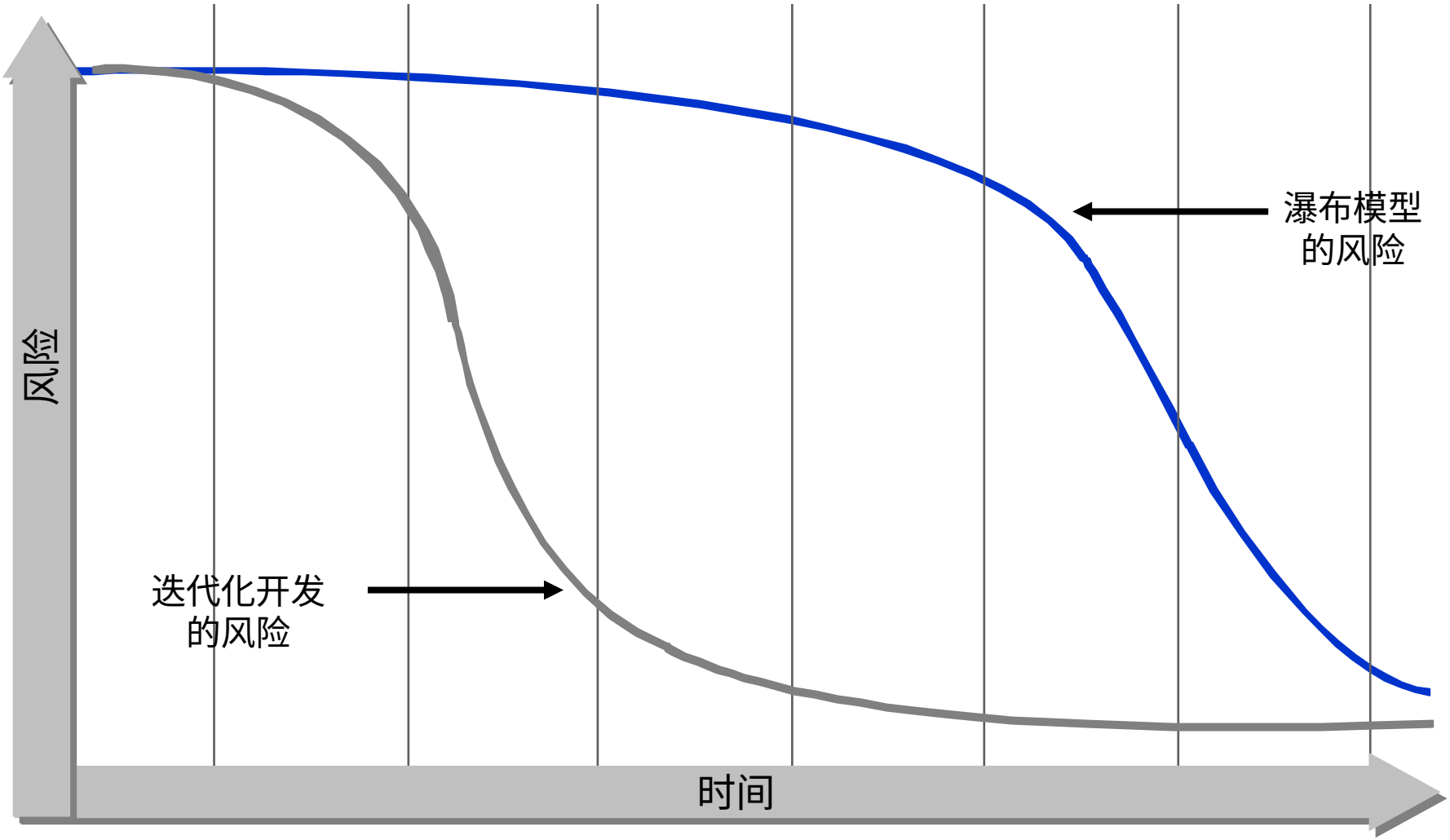
- 重要项目风险要到项目后期才能消除
- 文档驱动的开发流程，对于项目的预见性较差
- 集成测试和系统测试在项目的最后阶段才做
- 无法做到提前部署
- 经常导致额外的迭代

迭代化的软件项目生命周期模型



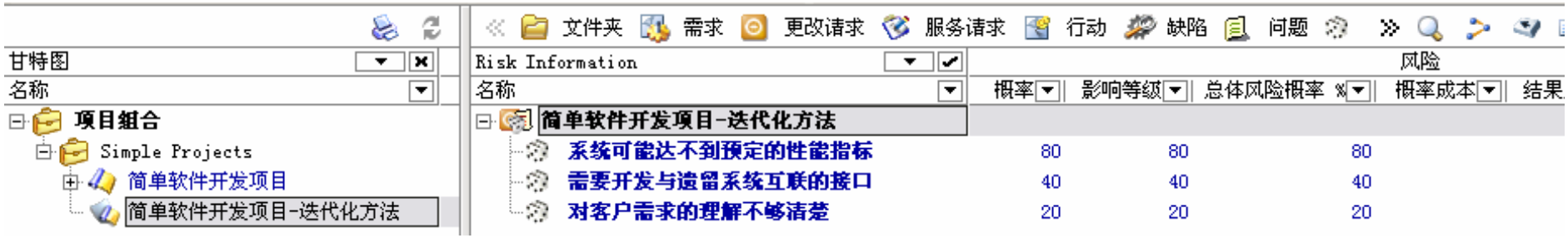
- 把复杂的问题分解成一系列相对简单的问题
- 早期的迭代解决风险最高的问题
- 每次迭代都增加系统的功能并产生一个可运行的结果
- 每次迭代都包括有测试工作

迭代化开发 vs. 瀑布模型

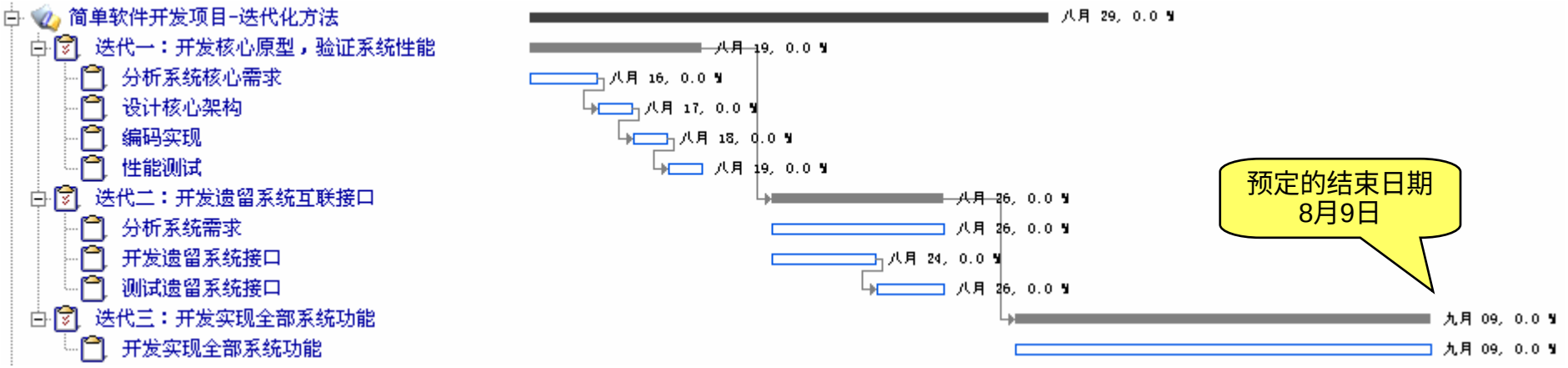


迭代计划的依据：风险驱动

- 优先解决那些高风险的问题

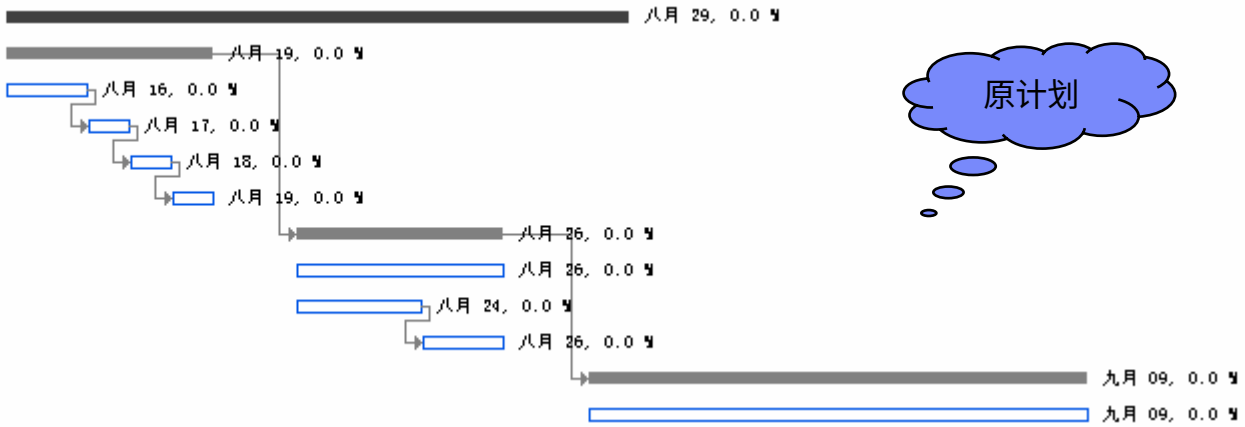


- 根据上面的风险列表制定的迭代开发计划

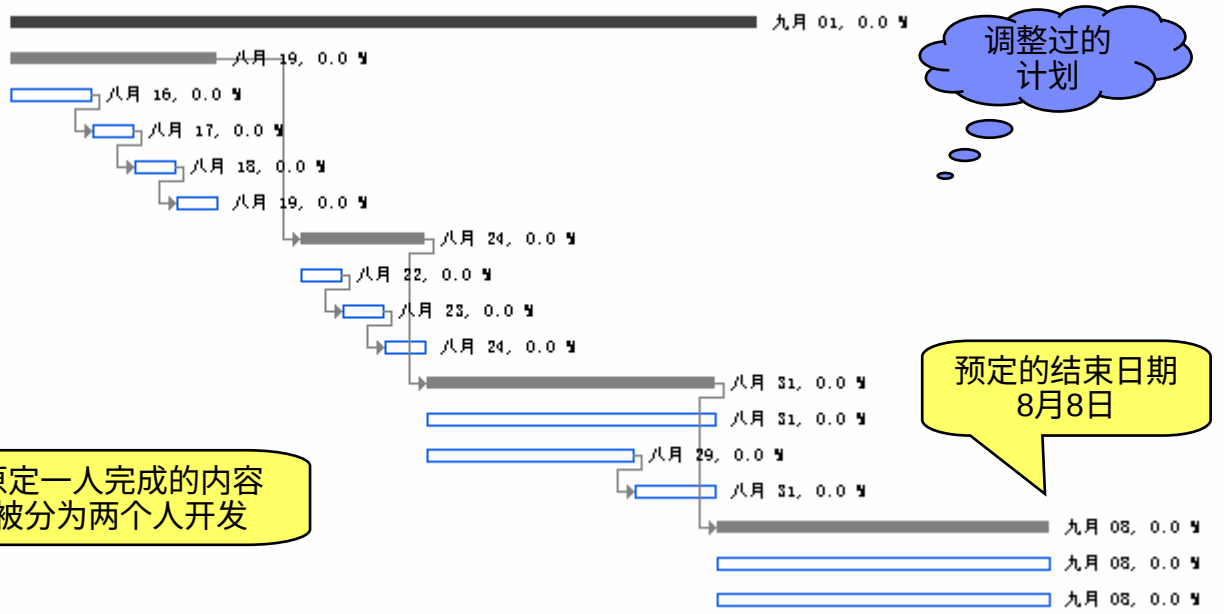


迭代一失败之后，调整过的计划

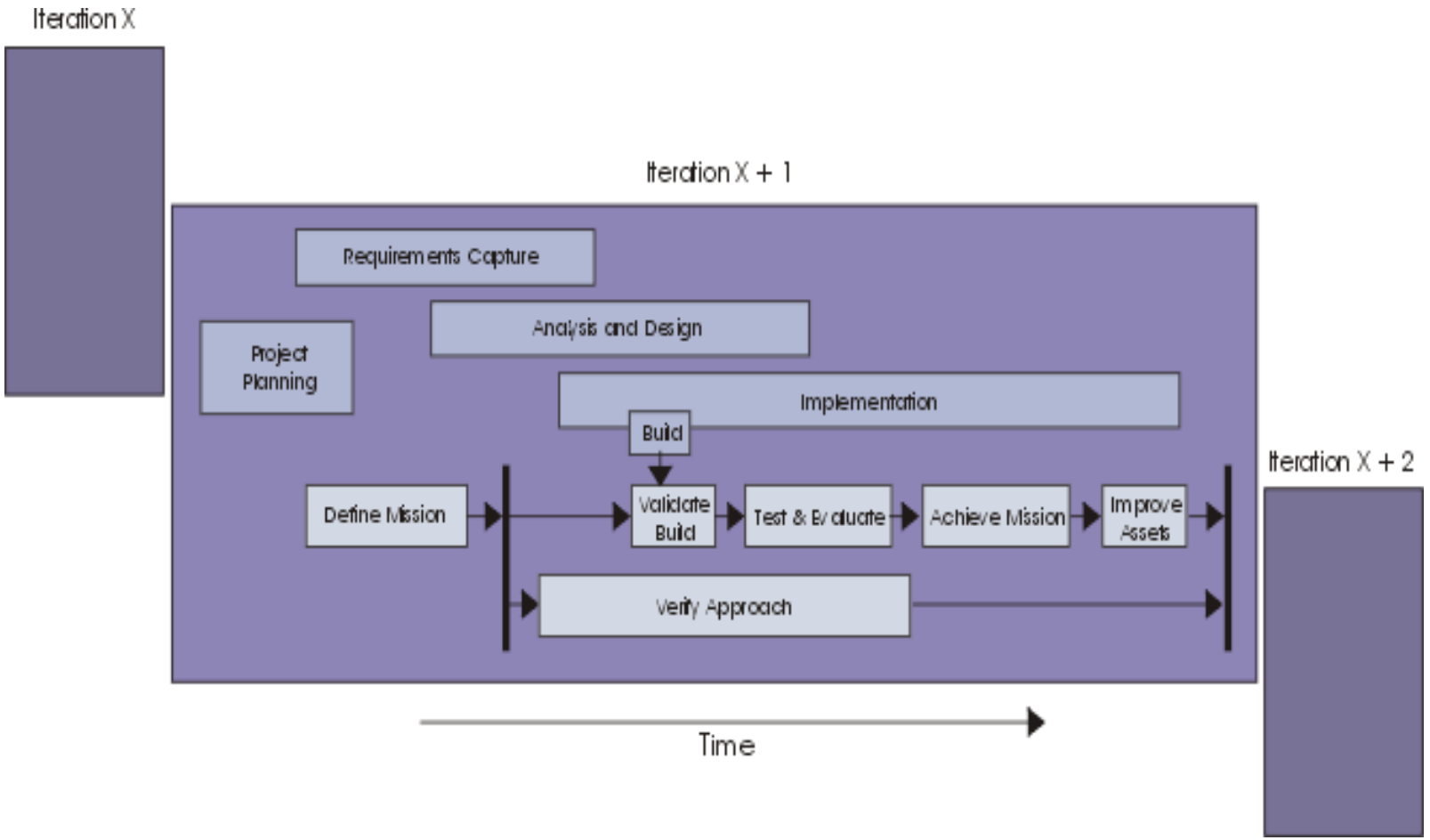
- 简单软件开发项目-迭代化方法
  - 迭代一：开发核心原型，验证系统性能
    - 分析系统核心需求
    - 设计核心架构
    - 编码实现
    - 性能测试
  - 迭代二：开发遗留系统互联接口
    - 分析系统需求
    - 开发遗留系统接口
    - 测试遗留系统接口
  - 迭代三：开发实现全部系统功能
    - 开发实现全部系统功能



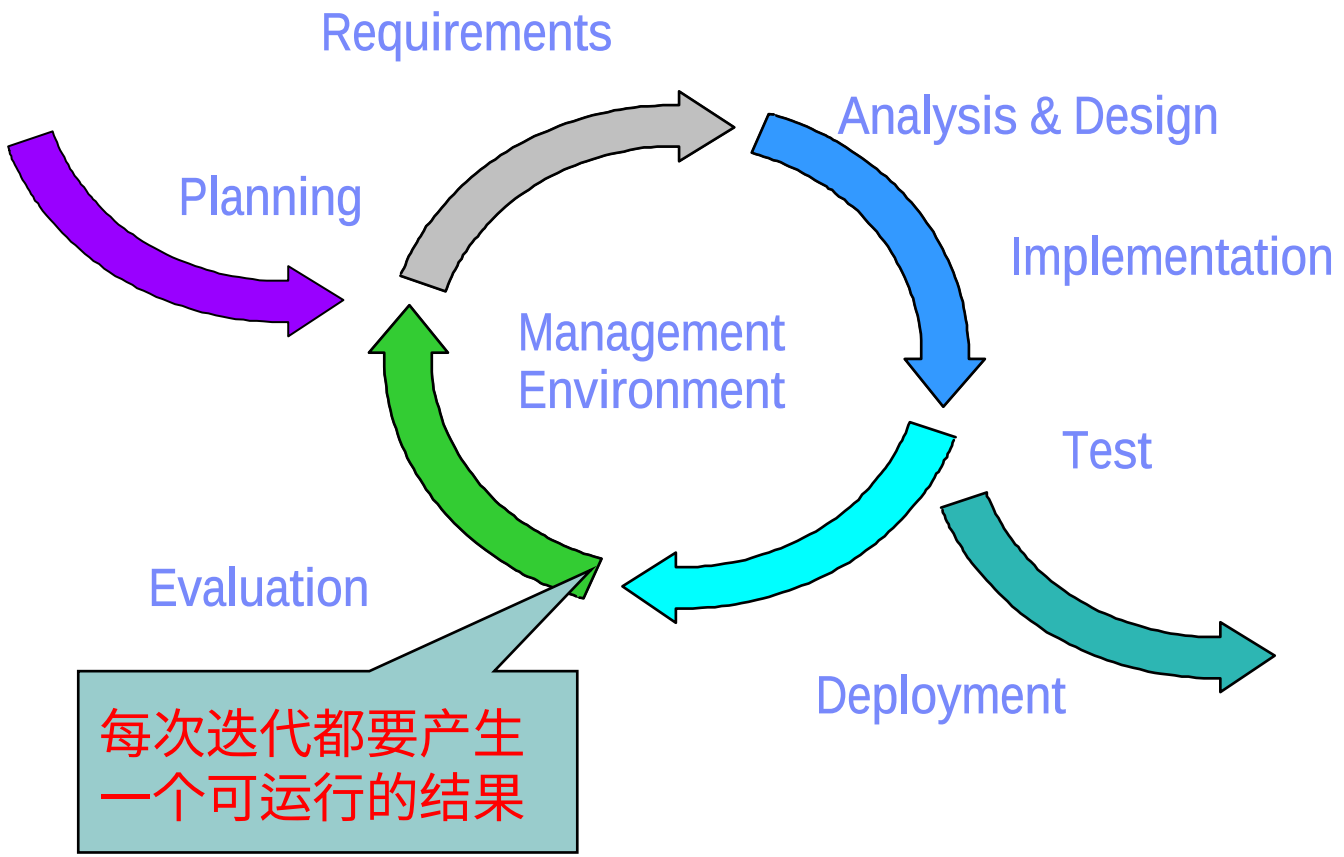
- 简单软件开发项目-迭代化方法
  - 迭代一：开发核心原型，验证系统性能
    - 分析系统核心需求
    - 设计核心架构
    - 编码实现
    - 系统性能测试
  - 迭代二：重新开发核心原型，验证系统性能
    - 修改核心架构设计
    - 修改编码实现
    - 系统性能测试
  - 迭代三：开发遗留系统互联接口
    - 分析系统需求
    - 开发遗留系统接口
    - 测试遗留系统接口
  - 迭代四：开发实现全部系统功能
    - 开发实现剩余系统功能1
    - 开发实现剩余系统功能2



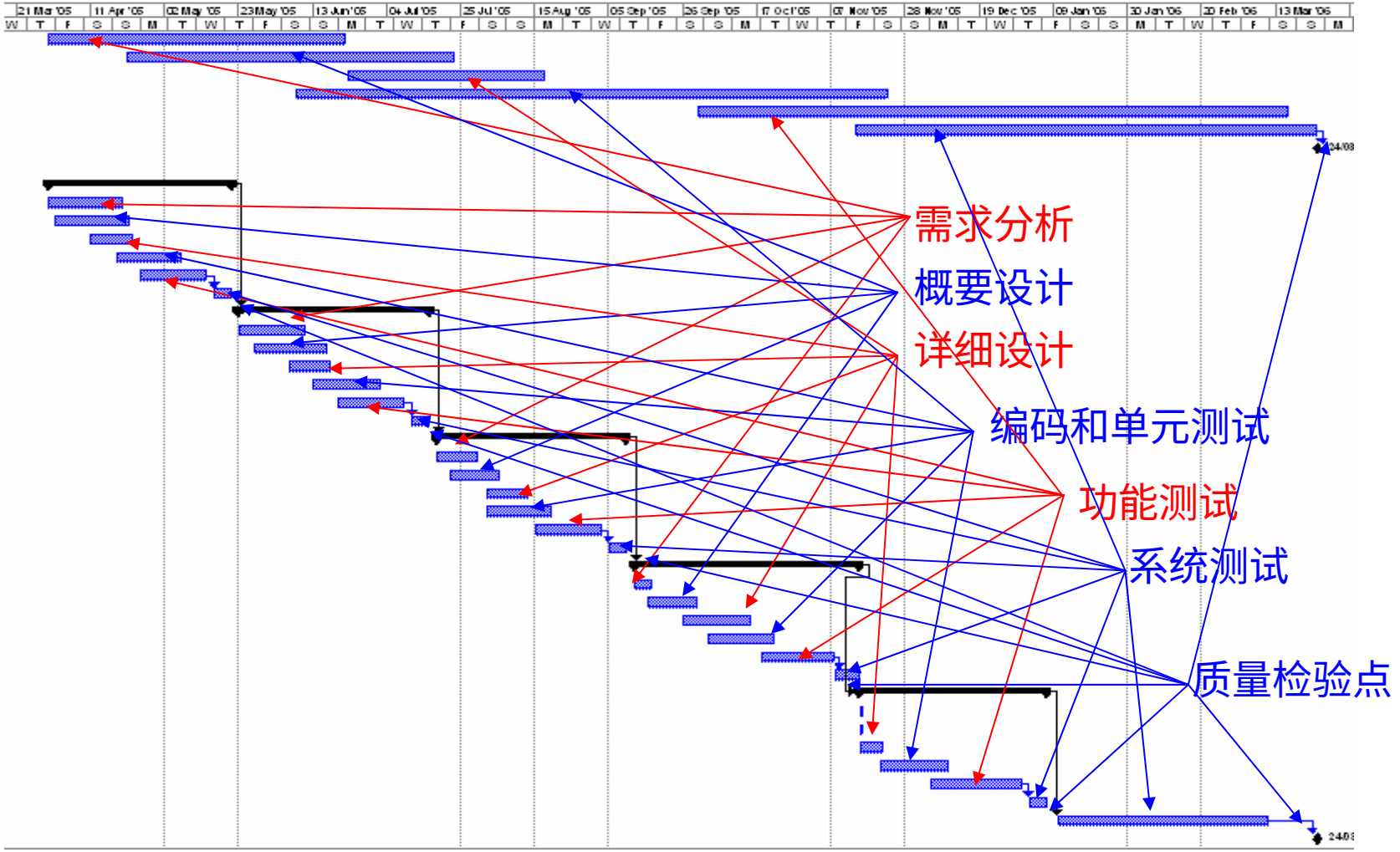
一次迭代


Iteration X
Iteration X + 1
Requirements Capture
Project Planning
Analysis and Design
Implementation
Build
Define Mission
Validate Build
Test & Evaluate
Achieve Mission
Improve Assets
Verify Approach
Iteration X + 2
Time

迭代化开发产生可运行的结果



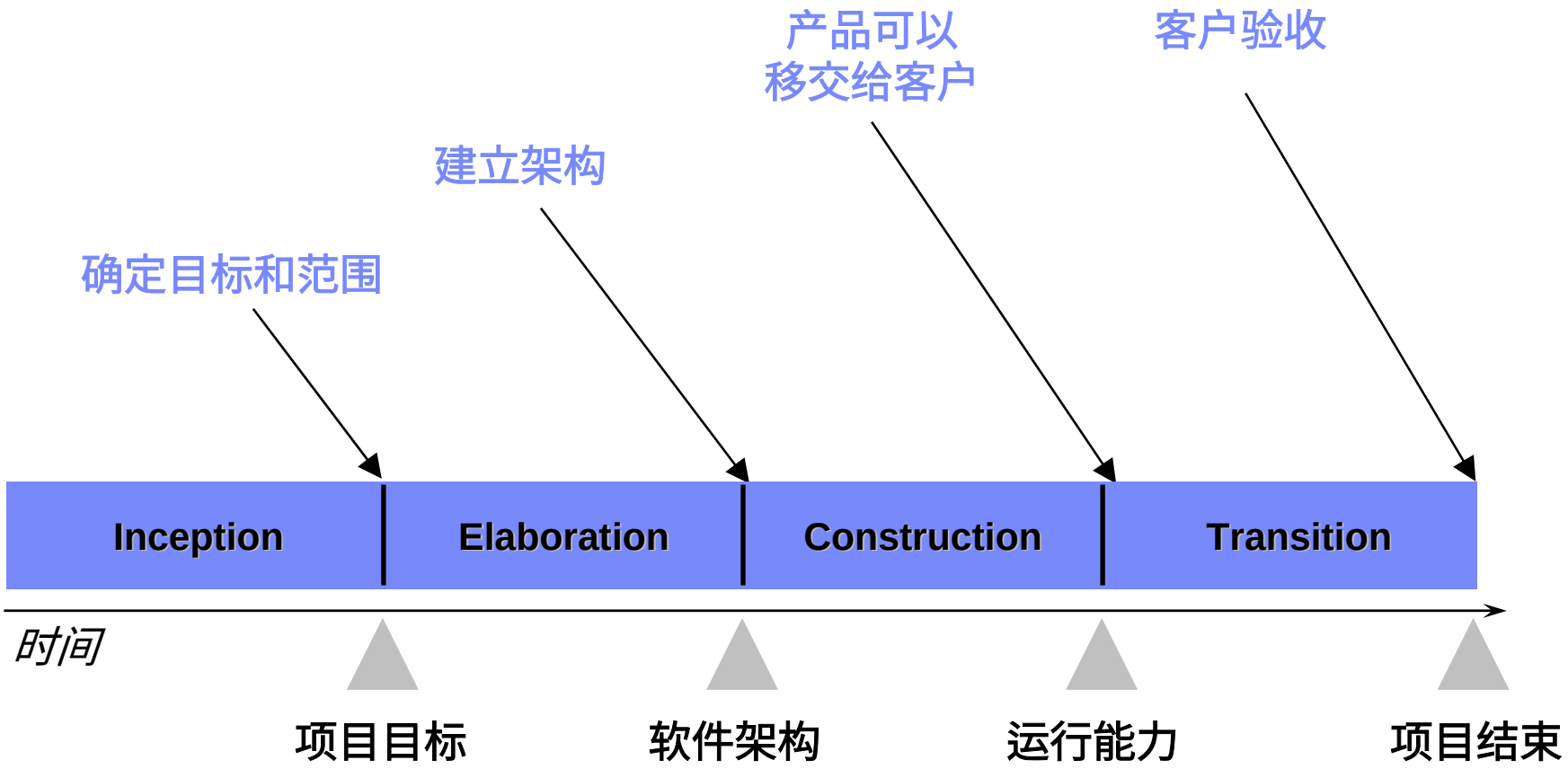
瀑布模型到迭代模型的映射



议程

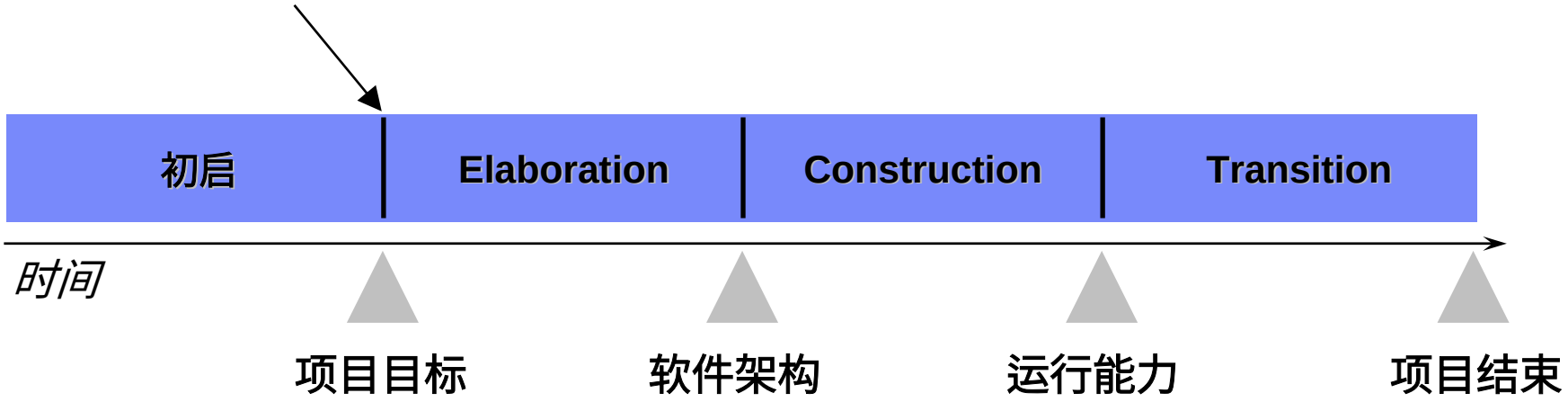
- 统一软件开发过程（Rational Unified Process）
- 管理迭代化的软件开发项目
  - 瀑布模型VS迭代化开发
  - 软件项目的开发阶段
  - 制定项目开发计划
  - 迭代化开发的指导原则
-

软件项目的四个阶段


产品可以
移交给客户
客户验收
建立架构
确定目标和范围
Inception
Elaboration
Construction
Transition
时间
项目目标
软件架构
运行能力
项目结束

初启阶段(Inception)

- 确定项目开发的目标和范围
- 定义主要的需求：用例以及主要的用例场景
- 根据一些主要的用例场景来构建一个基本架构
- 估算开发周期和成本
- 估计潜在的风险



手机开发项目 – 初启阶段

初启迭代1

手机功能列表：

. 通话

. 短信

. 地址簿

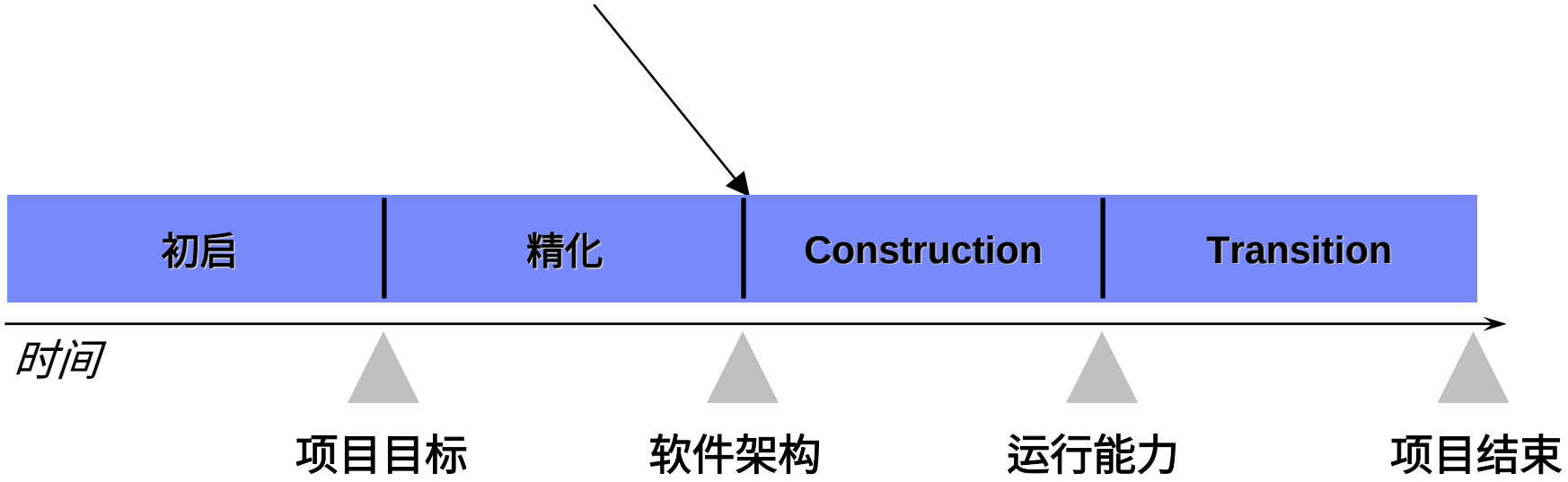
. 游戏

…

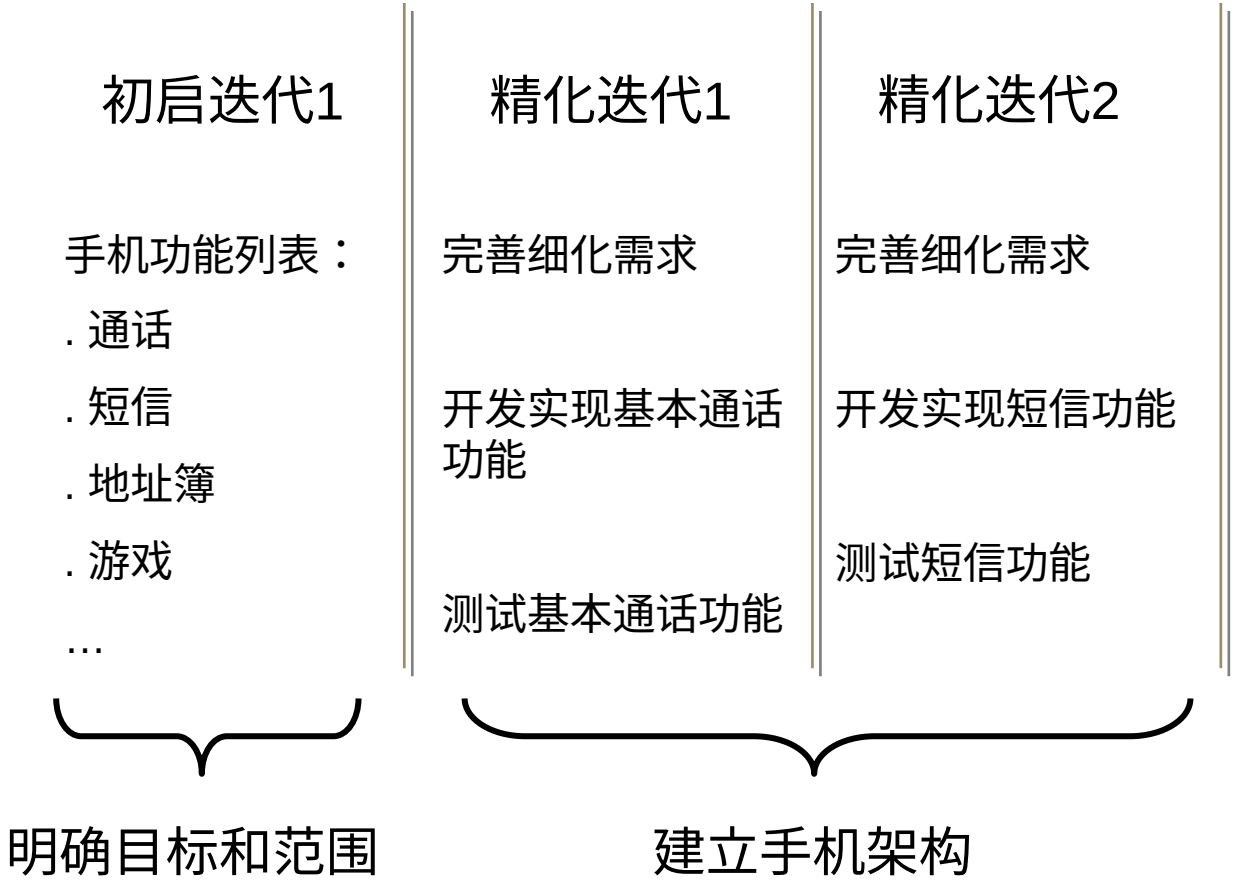
明确目标和范围

细化阶段(Elaboration)

- 尽快稳定并验证软件架构
- 定义所有的需求：所有的用例及用例场景
- 确定项目开发前景(vision)
- 为构建阶段(Construction)制定一个详细的开发计划

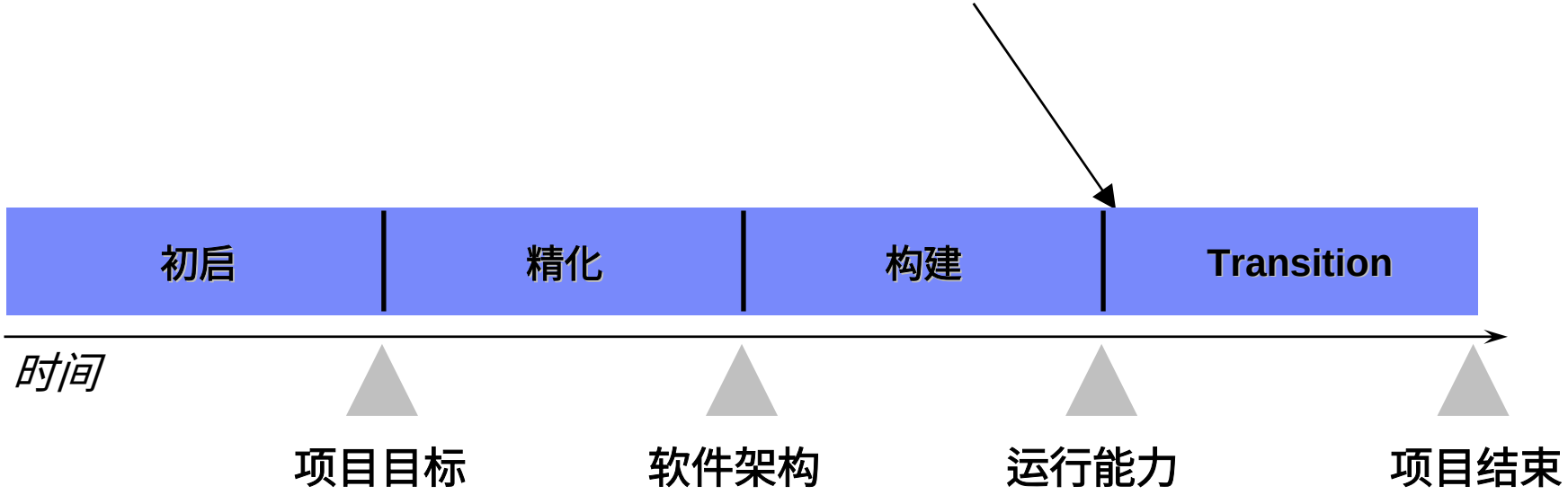


手机开发开发项目 – 细化阶段

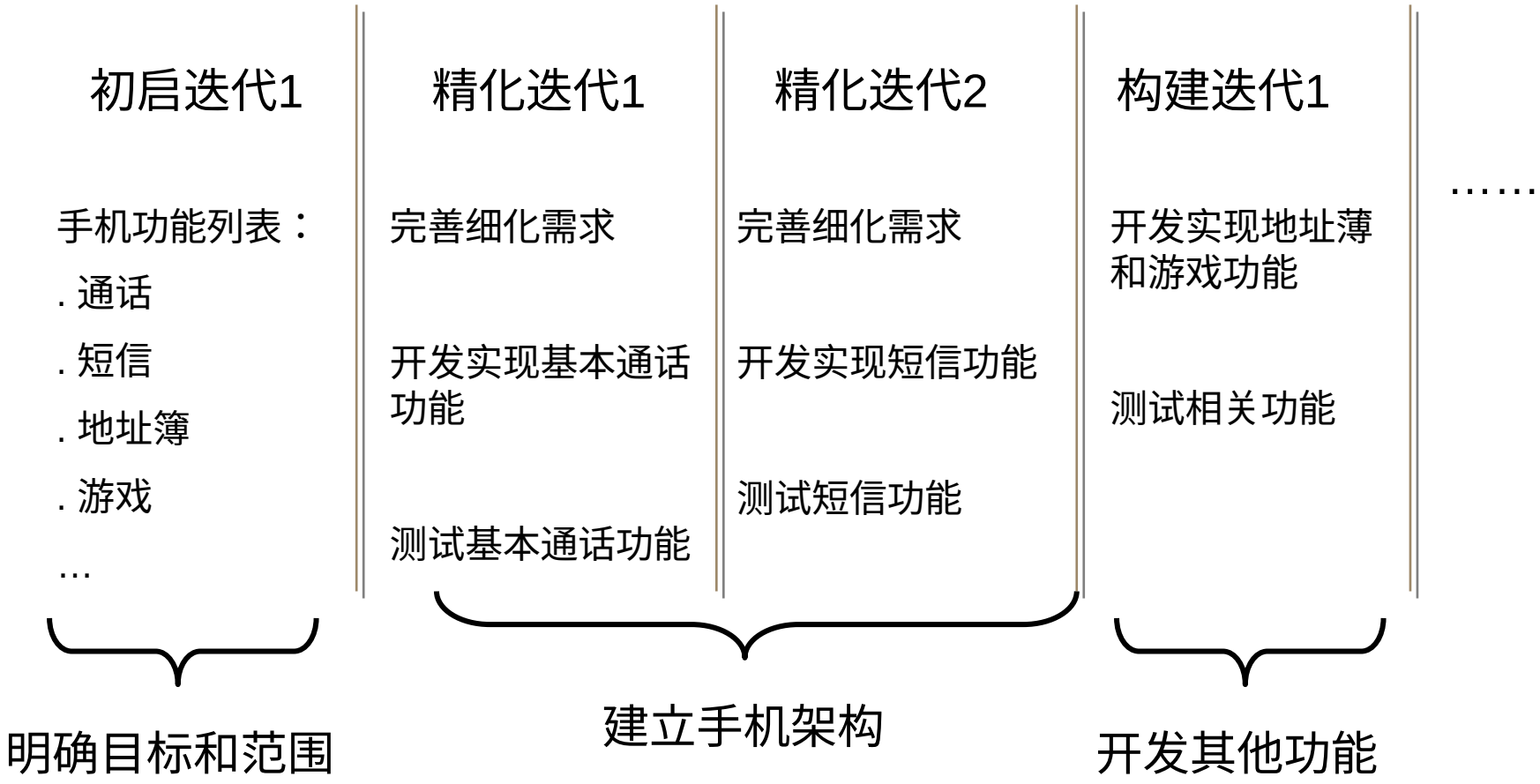


构建阶段(Construction)

- 尽快完成软件产品的开发
- 尽可能降低开发成本，优化开发资源和避免不必要的返工
- 在保证开发进度的同时达到足够的软件质量
- 获得一些有用的版本 (alpha, beta等)

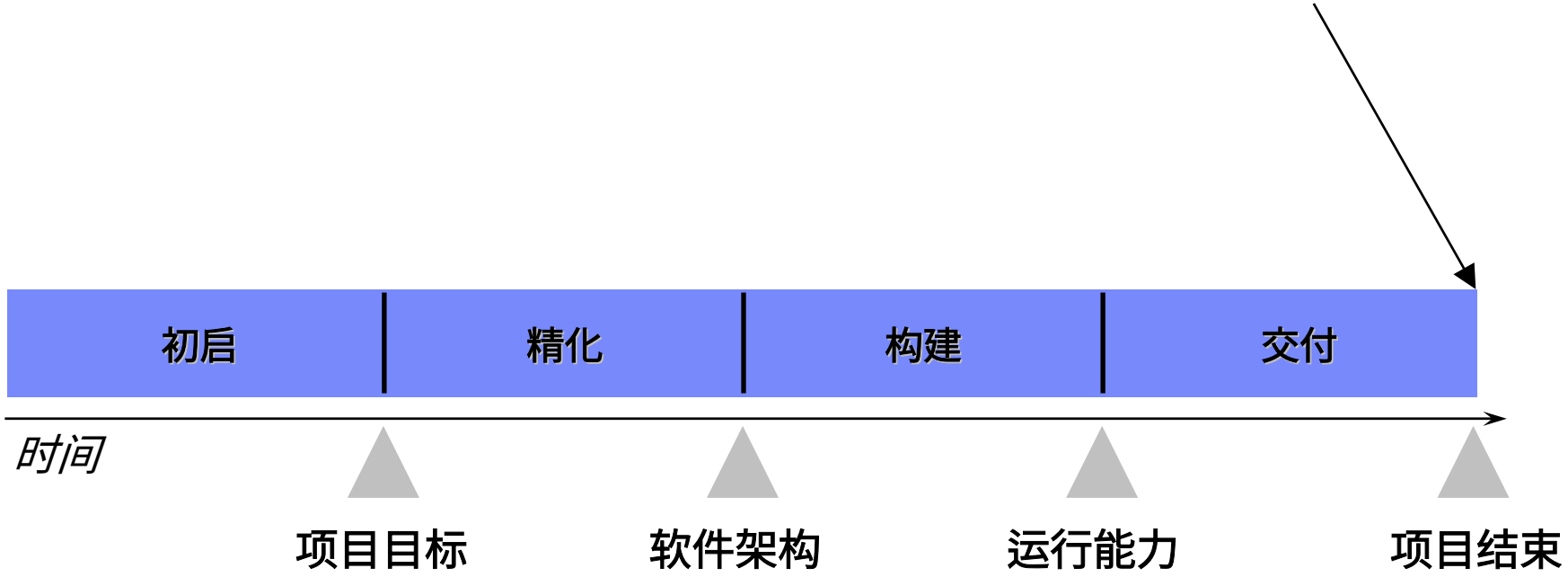


迭代化开发案例 - 手机开发

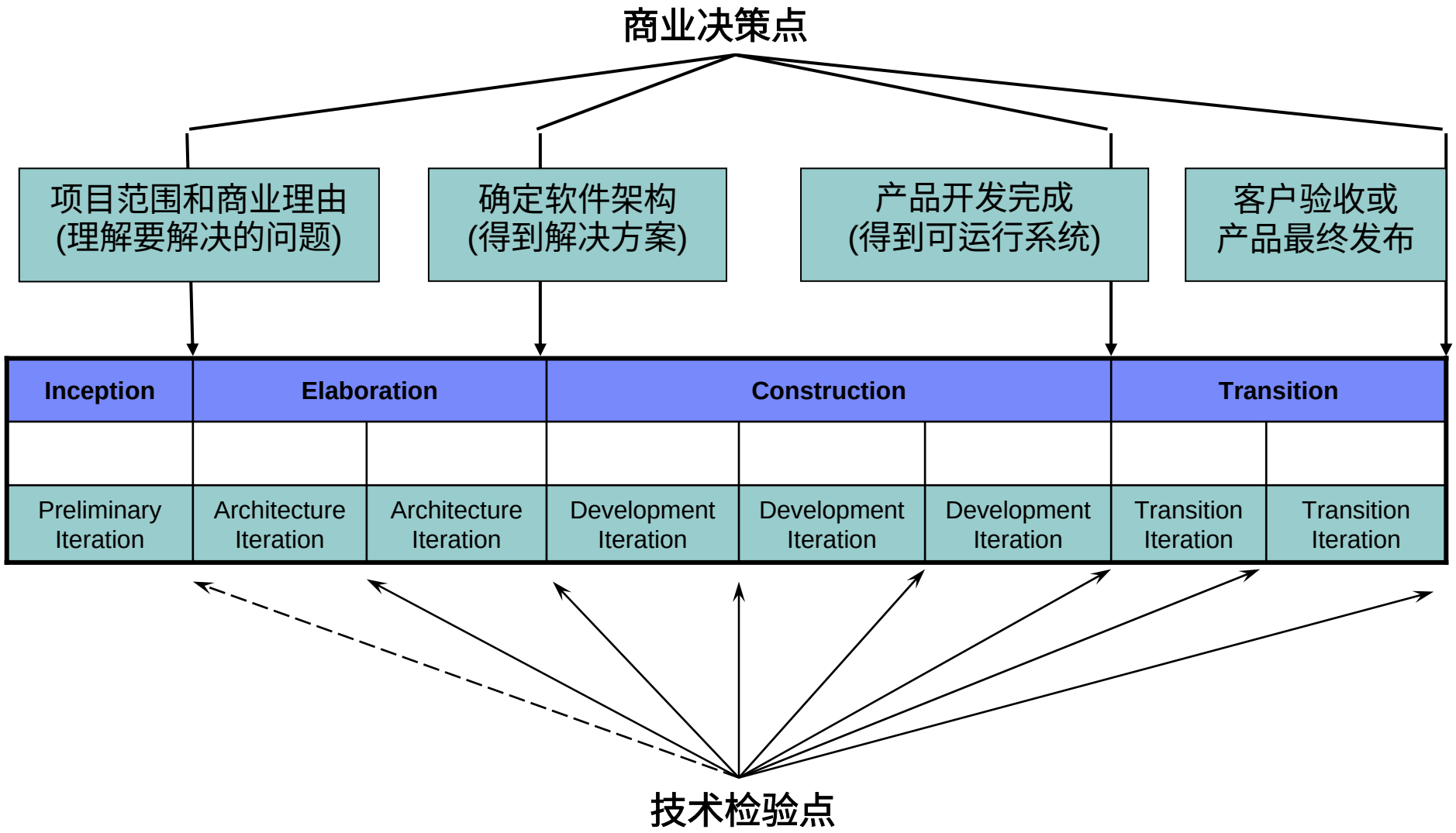


交付阶段(Transition)

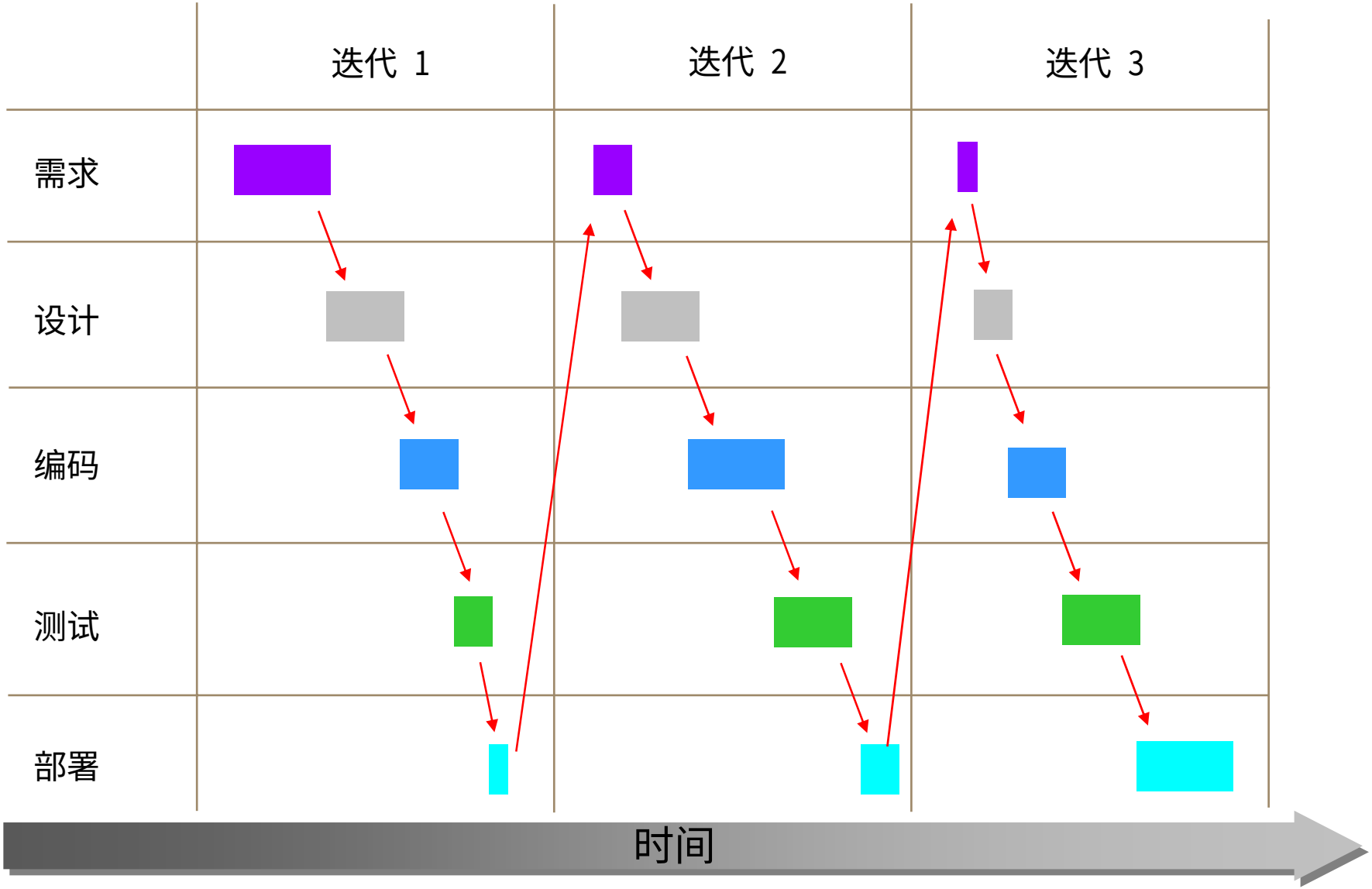
- 获得涉众的认同：
  产品部署已经完成并且满足预定的质量标准
- 尽快达到最终稳定的产品基线



项目的关注点随时间而发生变化



项目的关注点随时间而发生变化

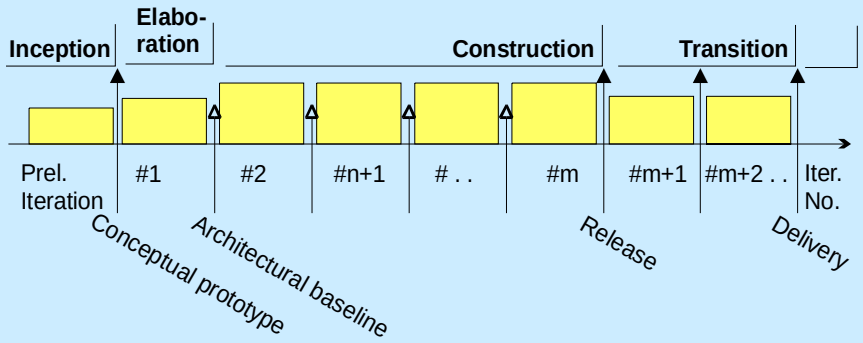


议程

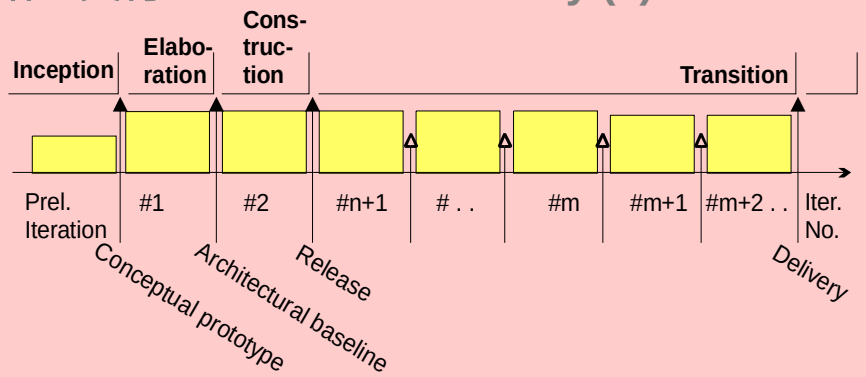
- 统一软件开发过程（Rational Unified Process）
- 管理迭代化的软件开发项目
  - 瀑布模型VS迭代化开发
  - 软件项目的开发阶段
  - 制定项目开发计划
  - 迭代化开发的指导原则
-

迭代化开发的策略

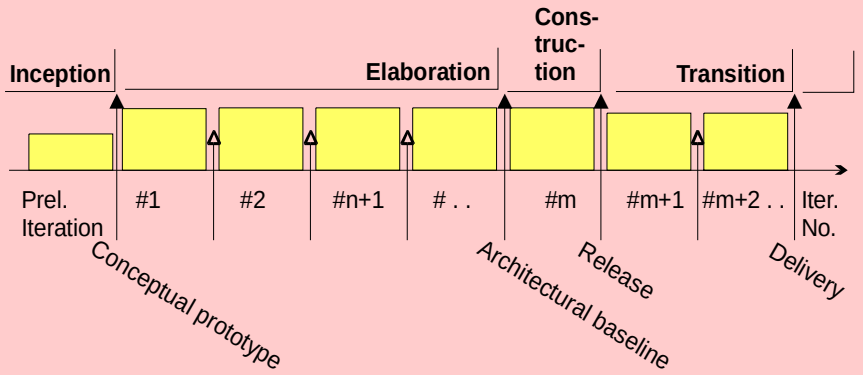
增量式 Incremental (1)



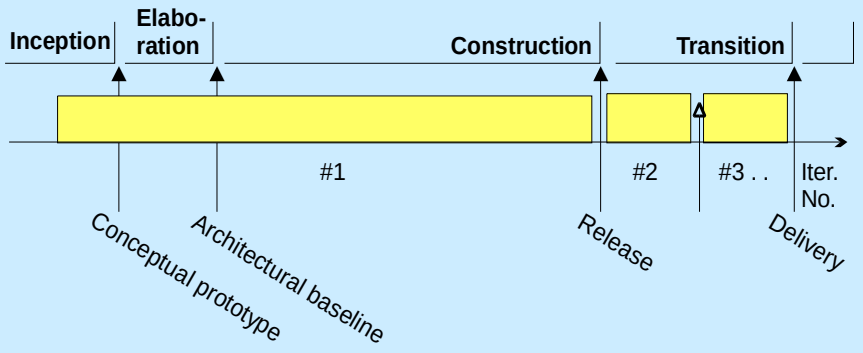
增量交付 Incremental delivery (3)



演进式 Evolutionary (2)



“Grand design” (4)



迭代的周期

- 迭代开始时需要制定迭代计划和确定本次迭代需要实现的需求，迭代结束时必须产生一个可运行的发布版本
- 根据具体项目的规模和复杂度，典型的迭代为4到6个星期
- 影响迭代周期的主要有以下这些因素：
  - 软件公司的规模、团队的稳定性和软件开发能力成熟度
  - 对迭代化流程的熟悉程度
  - 项目的大小
  - 项目中所采用技术的复杂度
  - 软件开发的自动化程度：管理代码、发布信息、功能/性能测试等

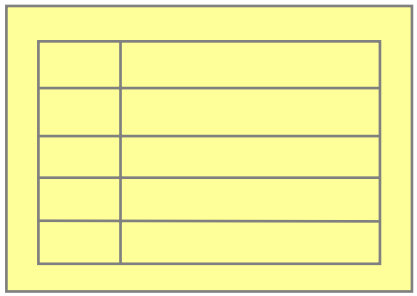
迭代数量

- 经验值：6 ± 3

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

制定项目开发计划

项目开发计划

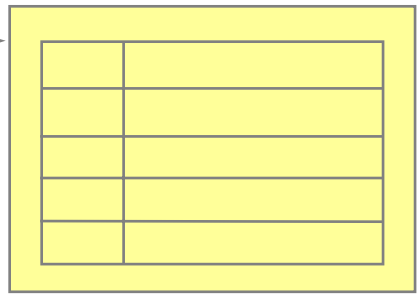


主要开发阶段和里程碑

什么时候完成项目

项目要做什么

阶段计划



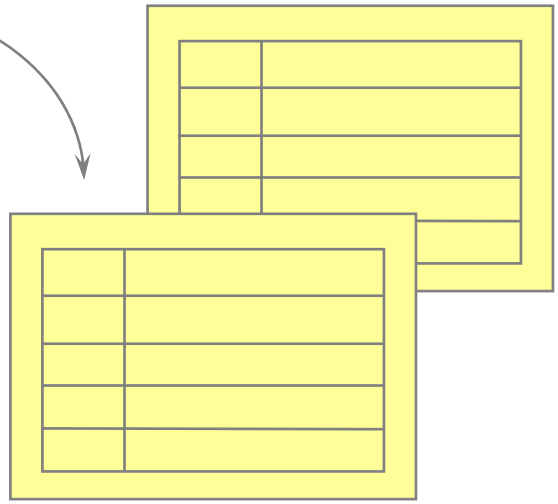
每个阶段分别有那些迭代

迭代的数量

迭代的目标

迭代的周期

迭代计划



下一个迭代计划

开发路标

框架性计划

细化的计划

议程

- 统一软件开发过程（Rational Unified Process）
- 管理迭代化的软件开发项目
  - 瀑布模型VS迭代化开发
  - 软件项目的开发阶段
  - 制定项目开发计划
  - 迭代化开发的指导原则

迭代化开发的指导原则

- 尽早降低项目风险
- 尽早确定软件架构

什么是风险?

风险是人们目前拥有或即将产生的一种顾虑，担心某种因素会严重影响到项目的成功。 —RUP

常见的风险类型:

- 技术/架构风险
  - 未经证实的技术，从未涉足的应用领域
- 资源风险
  - 缺乏足够的开发人员、技术、资金
- 商业风险
  - 兑争对手、ROI、供应商
- 进度风险
  - 无法按时完成项目

项目经理需要维护一张风险列表来对风险进行跟踪并评估风险的优先等级

风险管理策略

- 避免风险
- 风险转移
- 接受风险（必须有一个应急方案）
  - 维护风险列表
  - 立即采取必要的行动来降低风险影响
  - 制定一个应急方案

在初启阶段就应该建立风险列表，在每一迭代中都要对风险列表进行更新

迭代化开发是由风险驱动的

- 早期的迭代应优先解决那些影响最大的风险
- 应持续地对风险进行评估，风险会随着时间而发生变化
- 项目经理负责维护风险列表，该风险列表是制定迭代计划的重要依据

软件架构是项目管理的基础

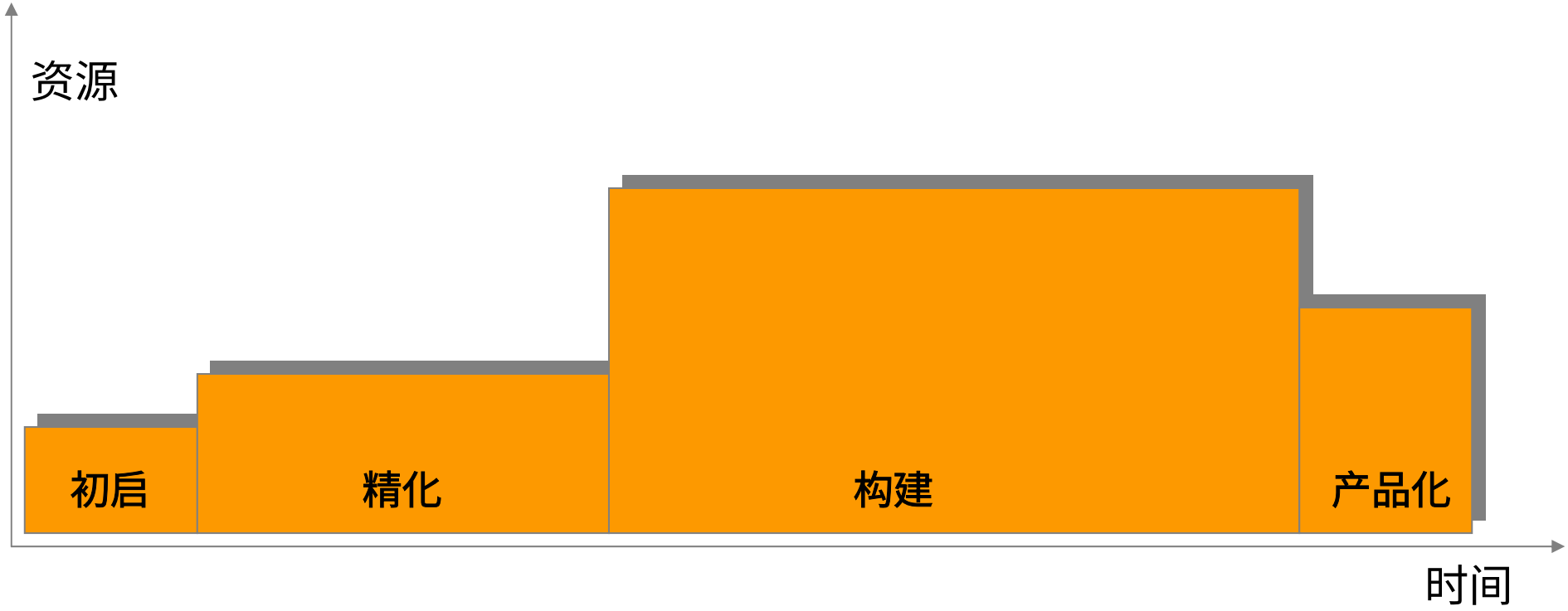
- 稳定的架构是项目开发中一个重要的里程碑
- 架构中的问题往往是项目失败的一个重要原因
- 架构是保证项目计划可预见性的一个重要因素
- 架构决定了一系列关键决策和风险控制点

尽早确定系统架构

- 架构
  - 确定了项目处于哪一个阶段
  - 确定了迭代的内容
  - 决定了开发资源的分配

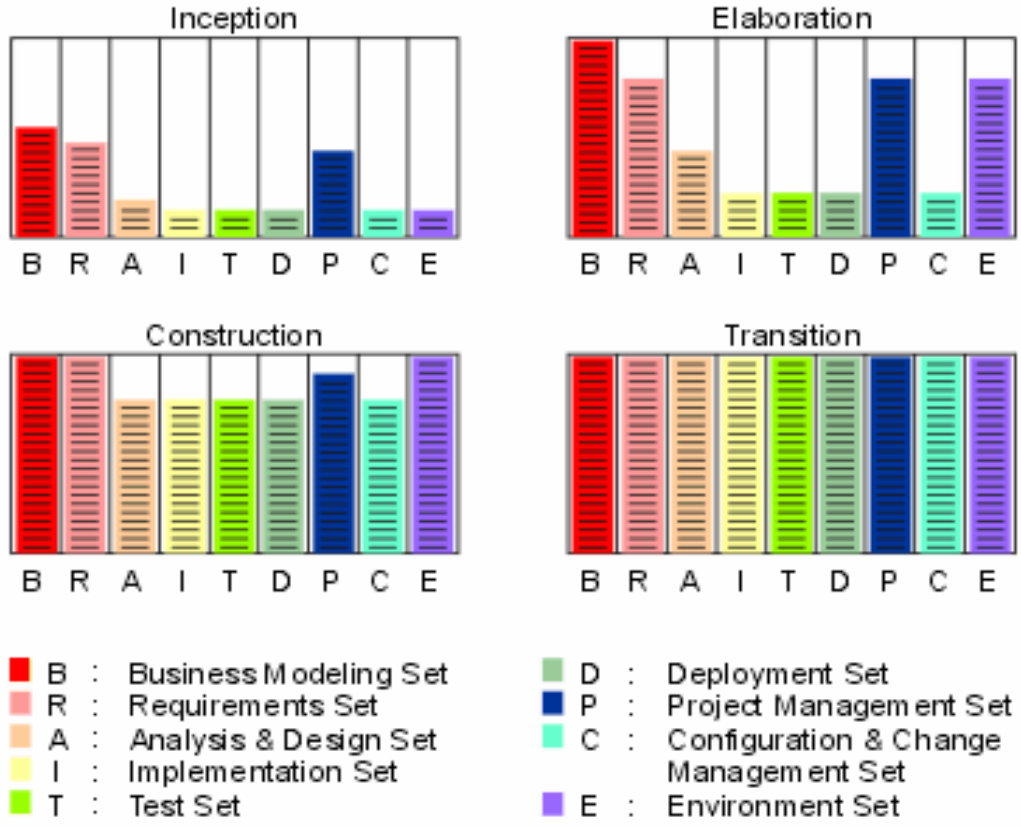
- 在精化阶段确定架构，在后续的迭代中不断完善

迭代化开发中资源和时间的典型比例



	初启	精化	构建	产品化
工作量	5%	20%	65%	10%
时间	10%	30%	50%	10%

项目生命周期中工件的变化



Information set evolution over the development phases.

With the iterative approach, artifact sets mature over time.


Questions


Thank
You