MTAT.03.306
Requirements Engineering

RE Framework

MTAT.03.306

需求工程

可再生能源框架

Table of Contents

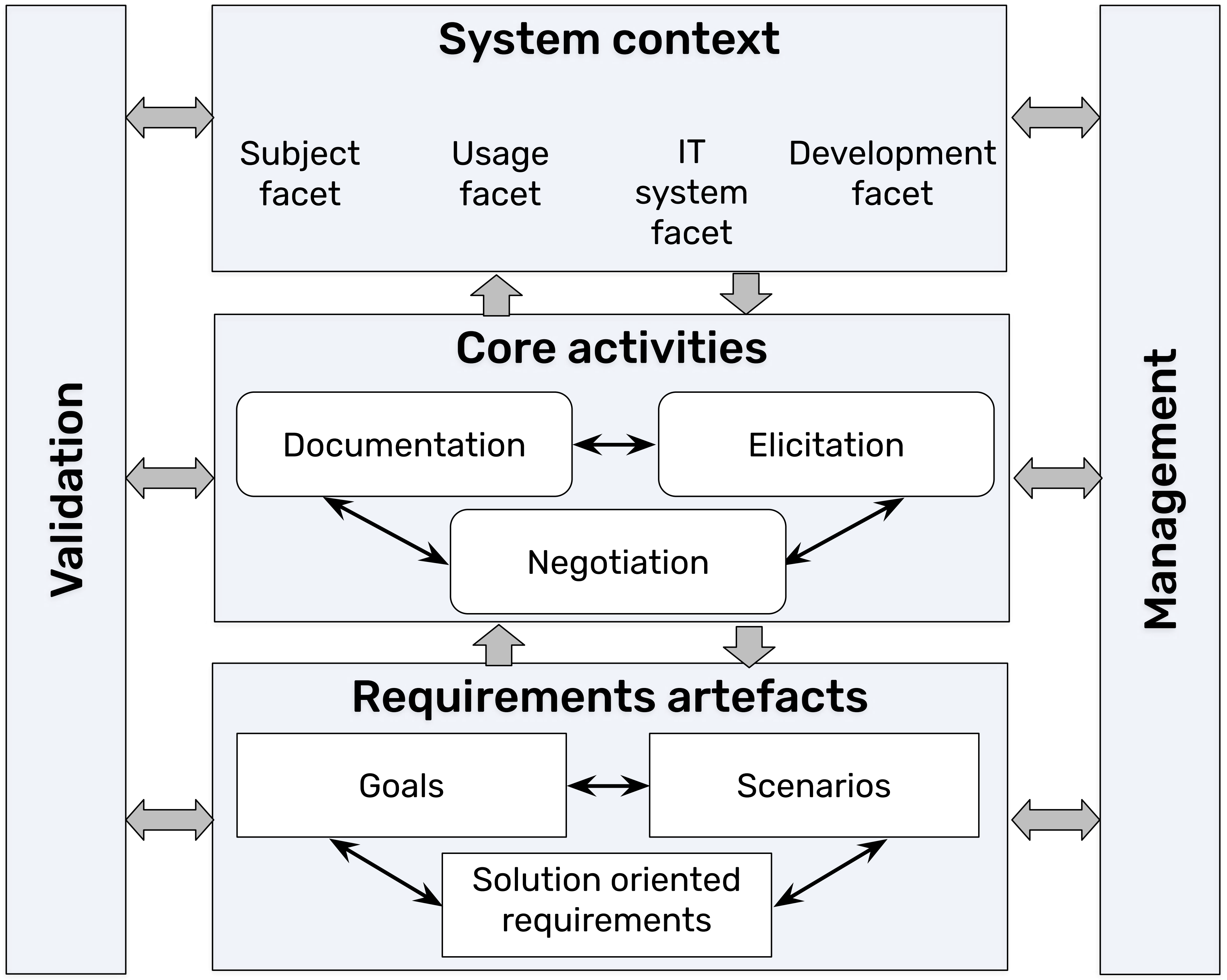
- RE framework
  - System context
  - Core activities
  - Requirements artefacts
  - Validation
  - Management

Pohl K. (2010) Requirements Engineering, Springer

目录

- RE 框架 – 系统上下文 – 核心活动 – 需求工件 – 验证 – 管理

Pohl K. (2010) 需求工程，施普林格


System context
Subject facet
Usage facet
IT system facet
Development facet
Core activities
Documentation
Elicitation
Negotiation
Requirements artefacts
Goals
Scenarios
Solution oriented requirements
Validation
Management

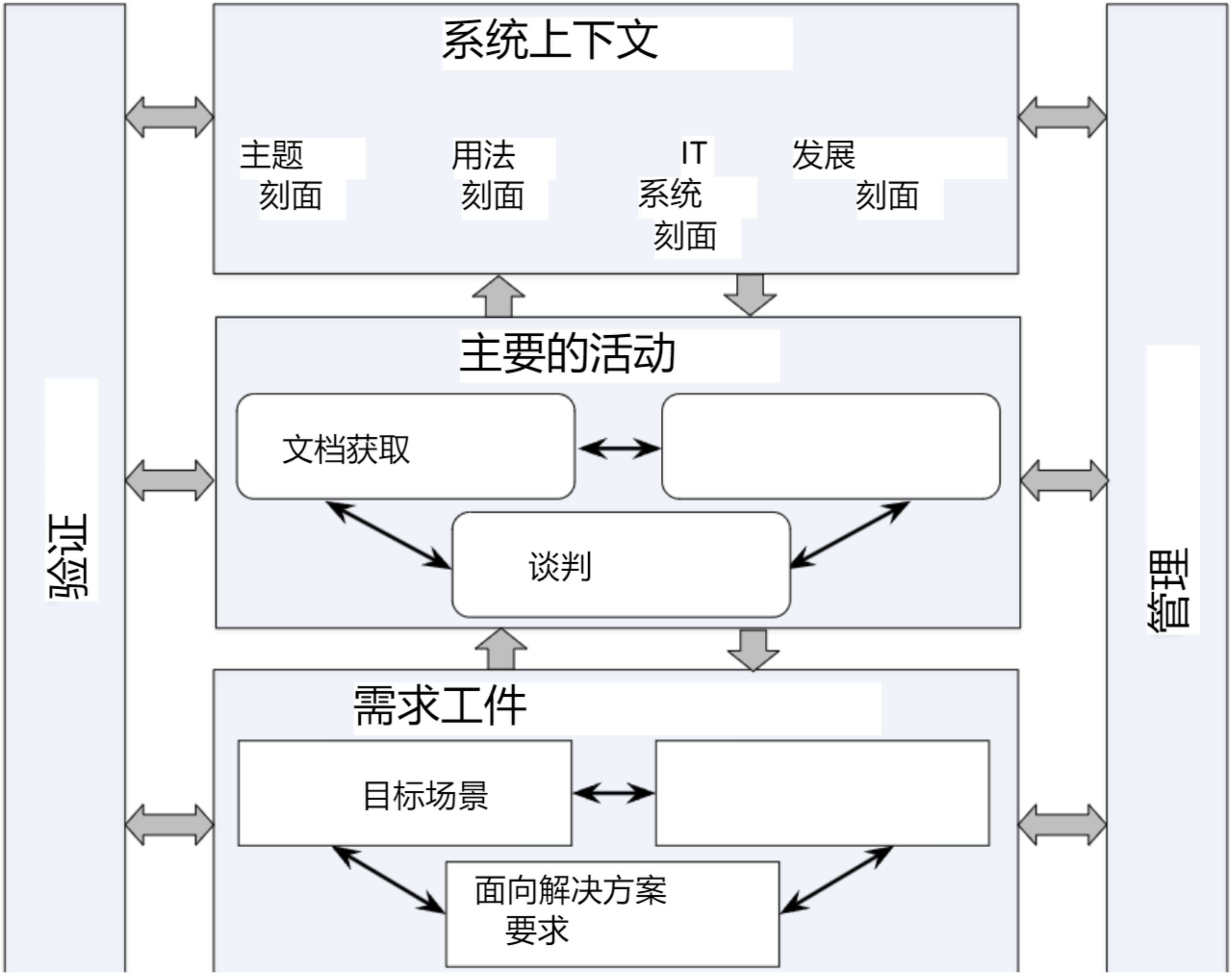

系统上下文
主题 刻面
用法 刻面
IT 系统 刻面
发展 刻面
主要的活动
文档获取
谈判
需求工件
目标场景
面向解决方案 要求
验证
管理

Table of Contents

- RE framework
  - **System context**
  - Core activities
  - Requirements artefacts
  - Validation
  - Management

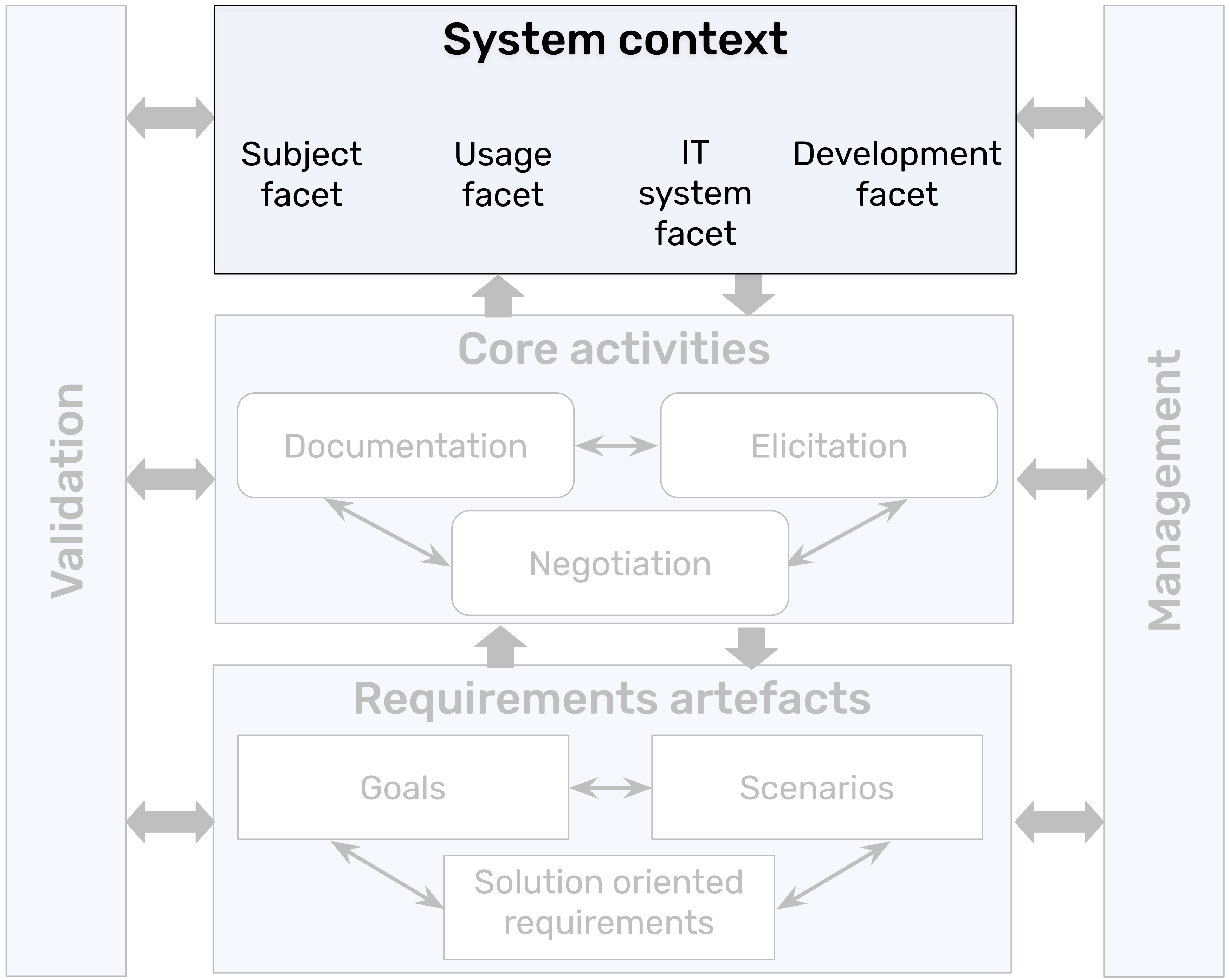
- Pohl K. (2010) Requirements Engineering, Springer

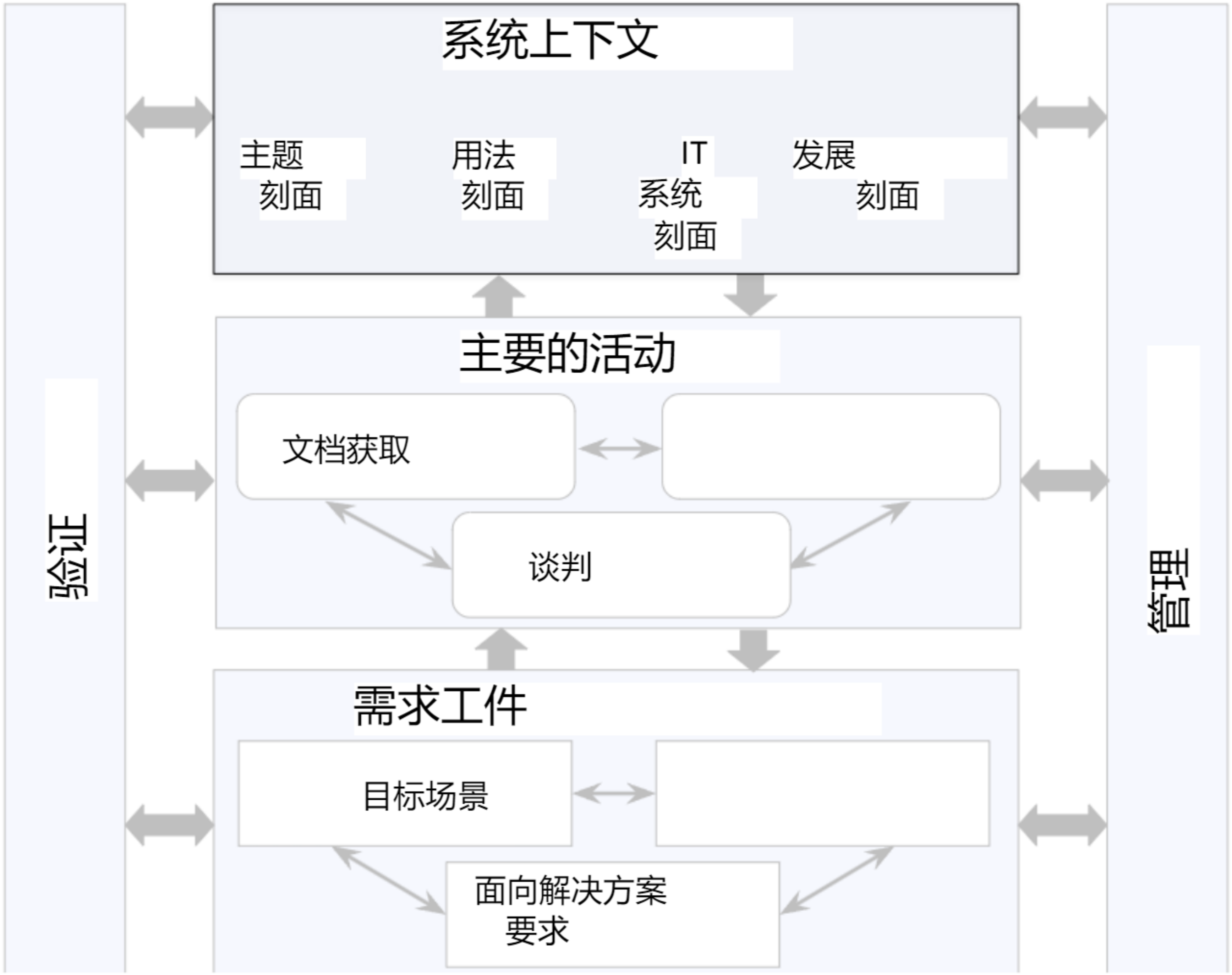
目录

- **RE** 框架 **–** 系统上下文

  - 主要的活动
  - 需求工件
  - 验证
  - 管理

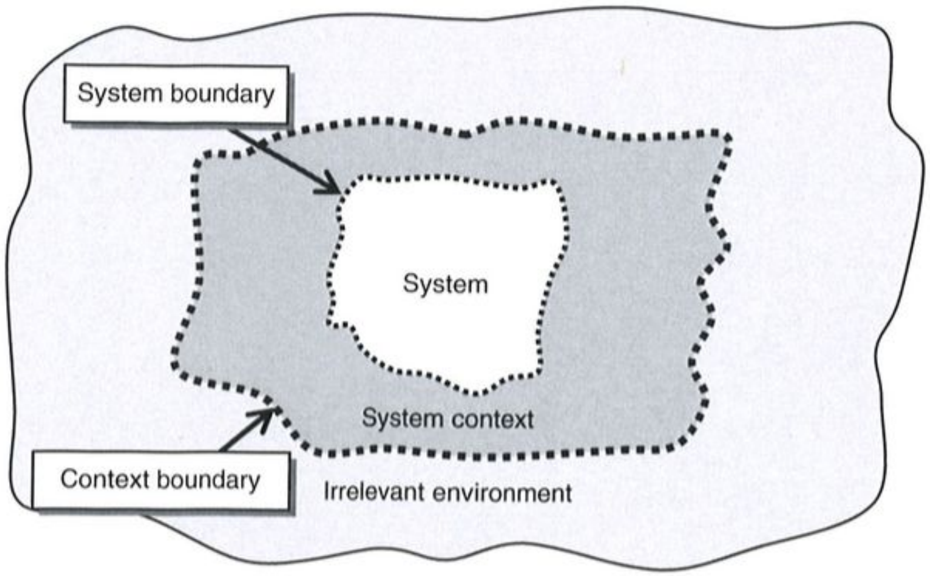
- Pohl K. (2010) 需求工程，施普林格


System context
Subject facet
Usage facet
IT system facet
Development facet
Core activities
Documentation
Elicitation
Negotiation
Requirements artefacts
Goals
Scenarios
Solution oriented requirements
Validation
Management


系统上下文
主题 刻面
用法 刻面
IT 系统 刻面
发展 刻面
主要的活动
文档获取
谈判
需求工件
目标场景
面向解决方案 要求
验证
管理

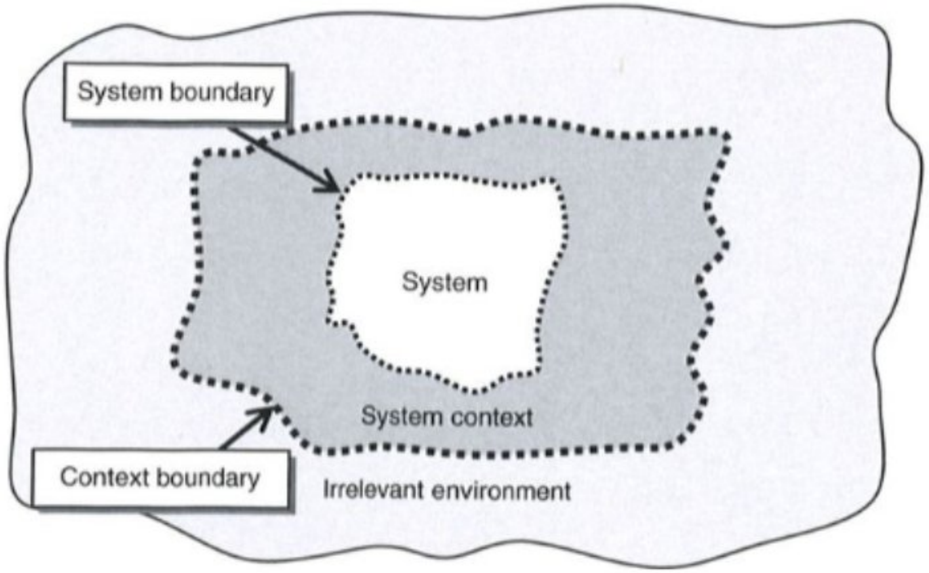
System Context

“the part of the system environment relevant for defining, understanding, and interpreting the system requirements of a system to be developed”



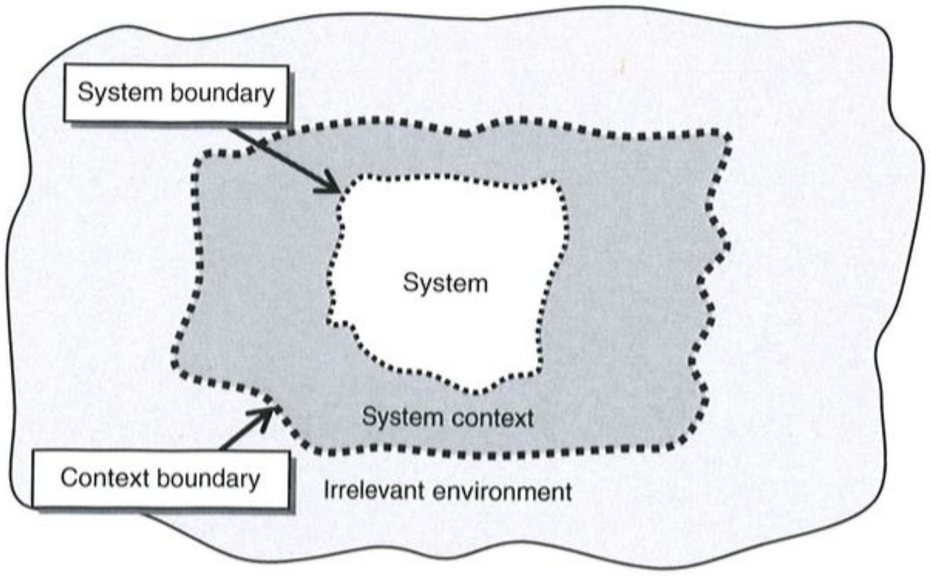
系统上下文

“ 系统环境中与定义、理解和解释待开发系统的系统需求相关的部分”



System Context

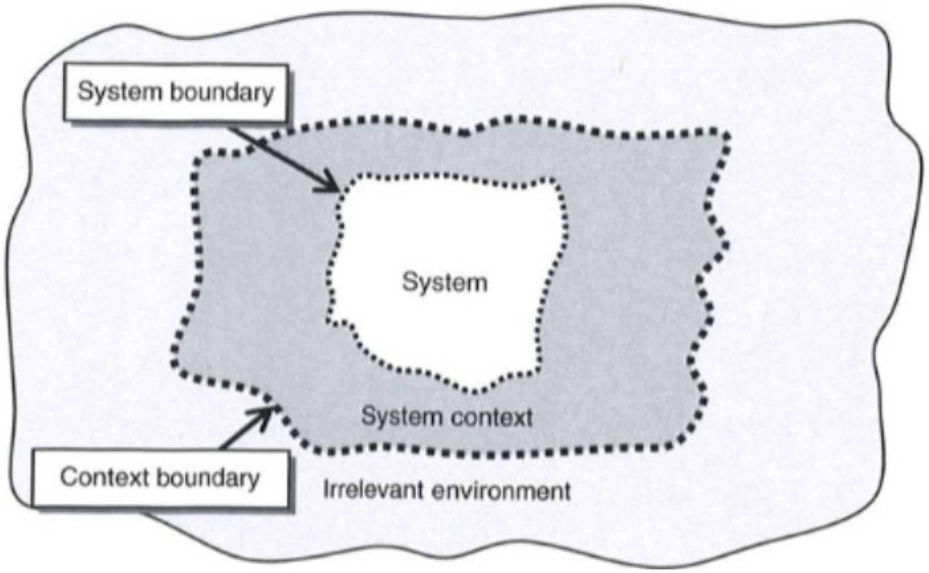
- **system boundary:**
  - Which aspects pertain to the system to be developed and which aspects belong in the system context?
- **context boundary:**
  - Which aspects pertain to the system context (i.e., have a relation to the system to be developed) and which aspects are part of the irrelevant environment?



系统上下文

- 系统边界：
  - ○哪些方面属于要开发的系统以及哪些方面属于系统上下文？

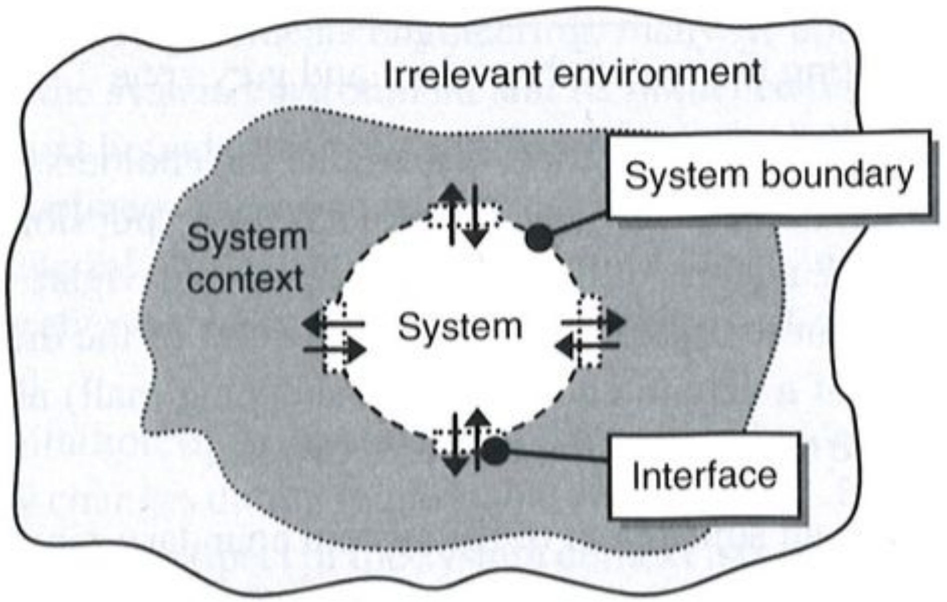
- 上下文边界：
  - ○ 哪些方面属于系统上下文（即与要开发的系统相关），哪些方面属于不相关环境的一部分？



System Context

Defining system boundaries through interfaces:

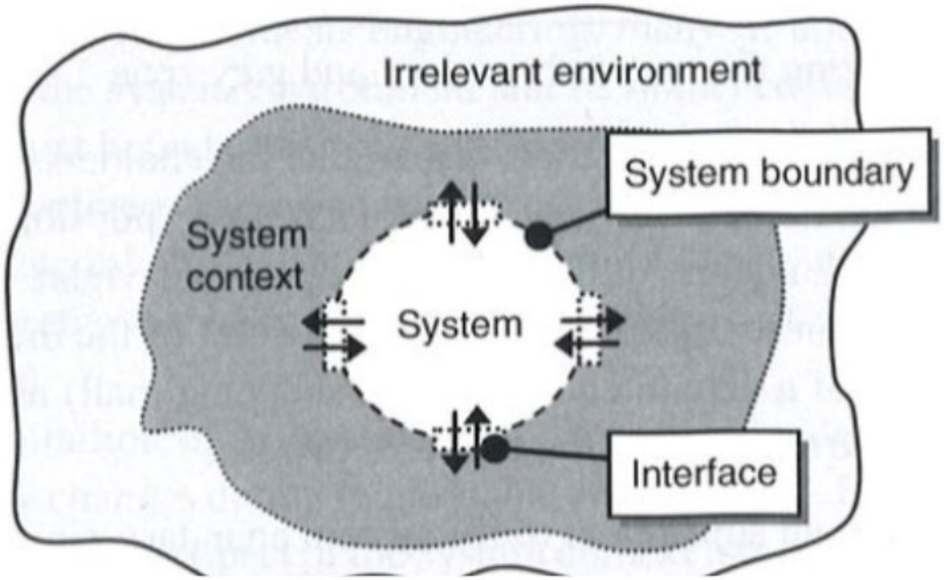
- human-machine interface
- hardware interface
- software interface



系统上下文

通过接口定义系统边界：

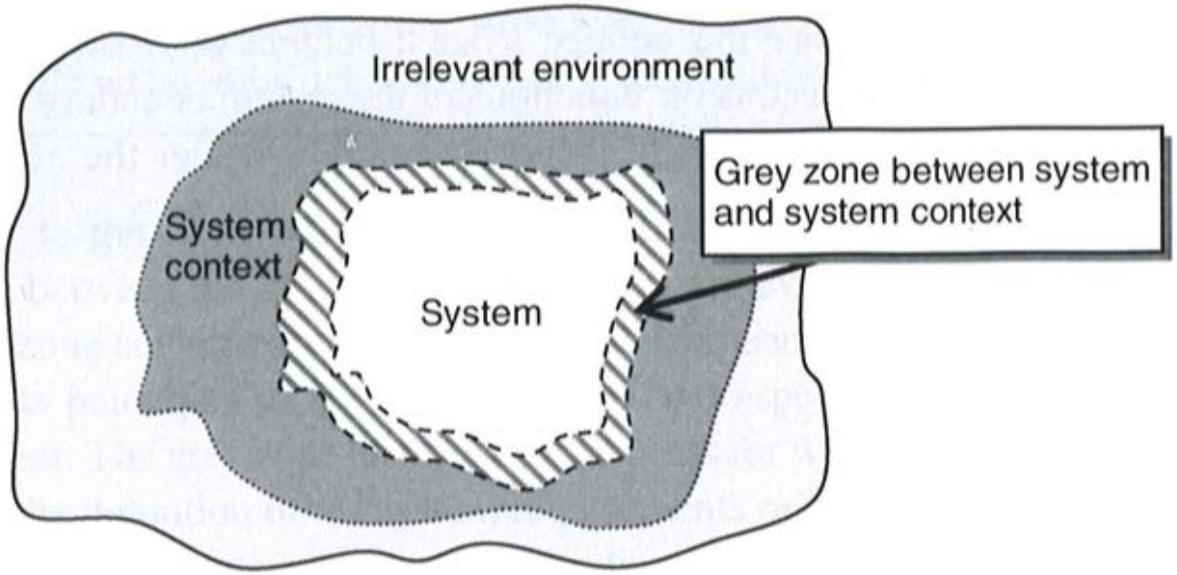
- 人机接口
- 硬件接口
- 软件界面



System Context

Gray zone between system and system context:

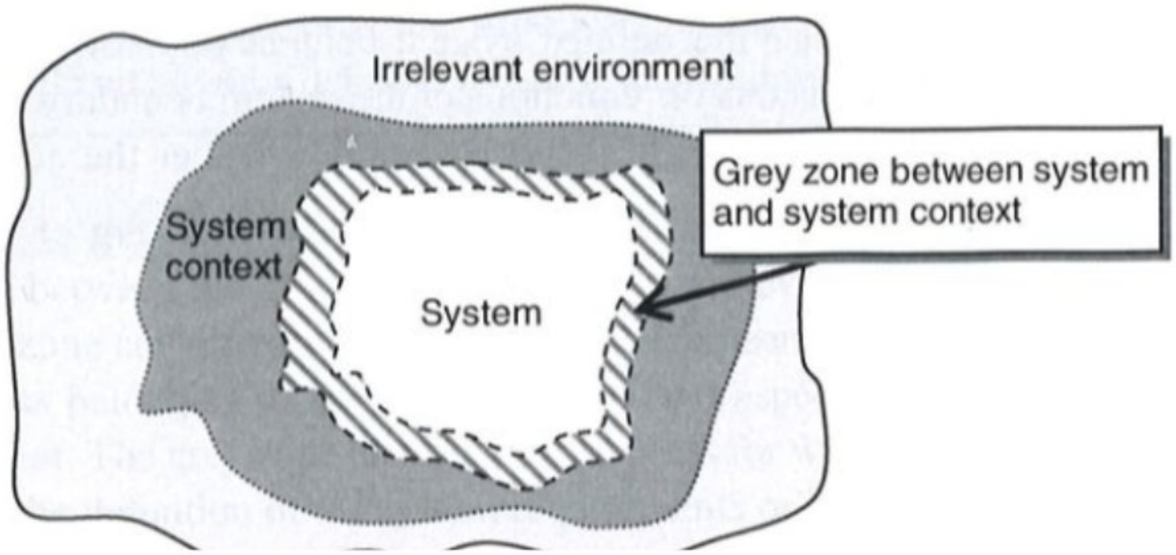
- describes vaguely defined system boundaries during RE process
- refers to the system scope change during RE



系统上下文

系统和系统上下文之间的灰色地带：

- 描述 RE 过程中模糊定义的系统边界
- 指RE期间系统范围的变化



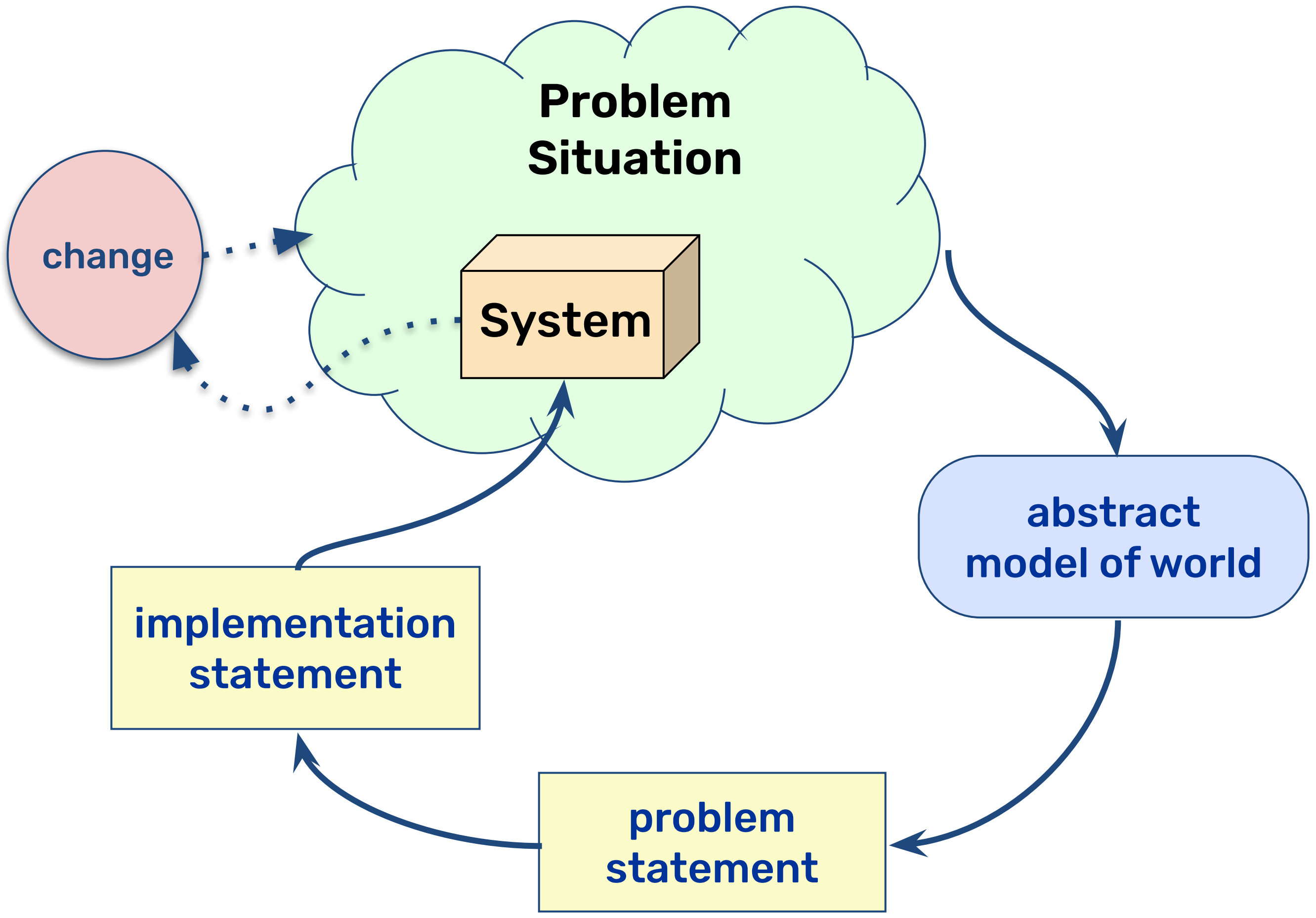
System Context

- **Subject facet**: objects and events that are relevant for the system,
    - because the system must store or process information about these objects
- **Usage facet**: aspects concerning the usage of the system by people and
    - other systems
- **IT system facet**: aspects concerning the operational or technical environment in which the system is deployed
- **Development facet**: aspects that influence the development of the system
    - imposed by law, or by client and relate to the development process

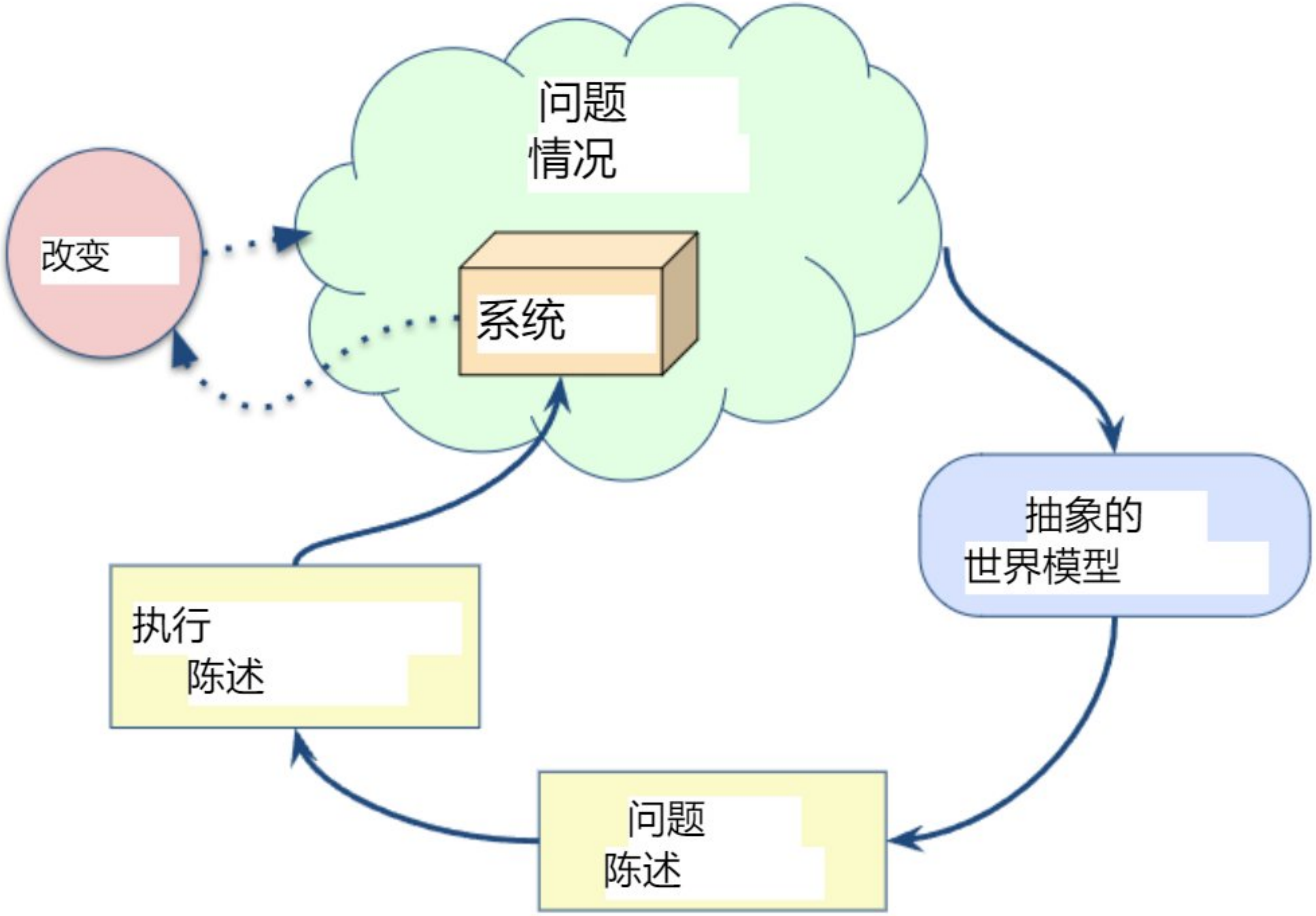
系统上下文

- 主题方面：与系统相关的对象和事件，
    - 因为系统必须存储或处理有关这些对象的信息
- 使用方面：有关人员和系统使用系统的方面
    - 其他系统
- IT 系统方面：涉及系统部署的操作或技术环境的方面
- 发展方面：影响系统发展的方面
    - 由法律或客户强制实施并与开发过程相关

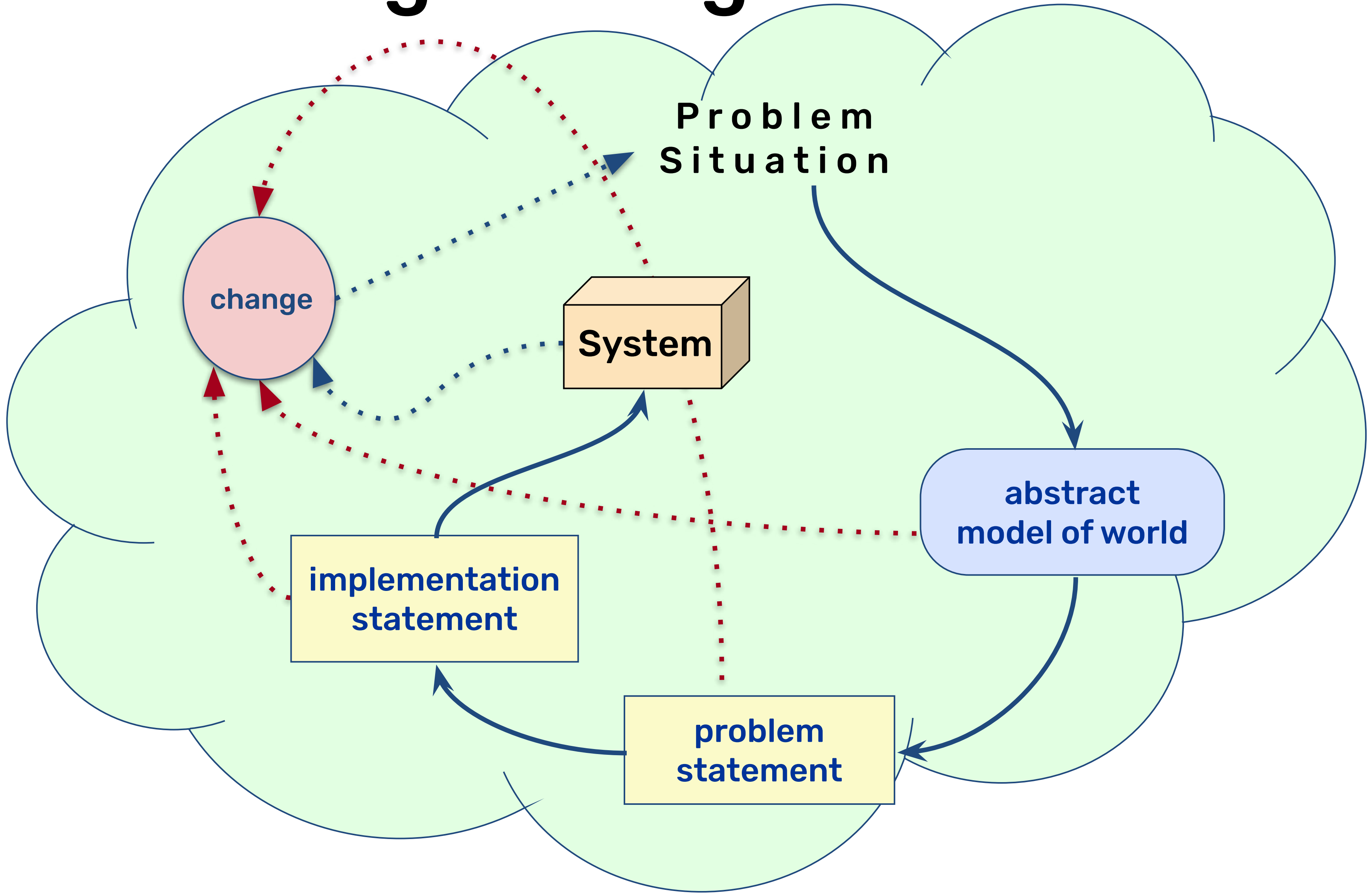
But design changes the world...



但设计改变世界……



But design changes the world…



但设计改变世界……

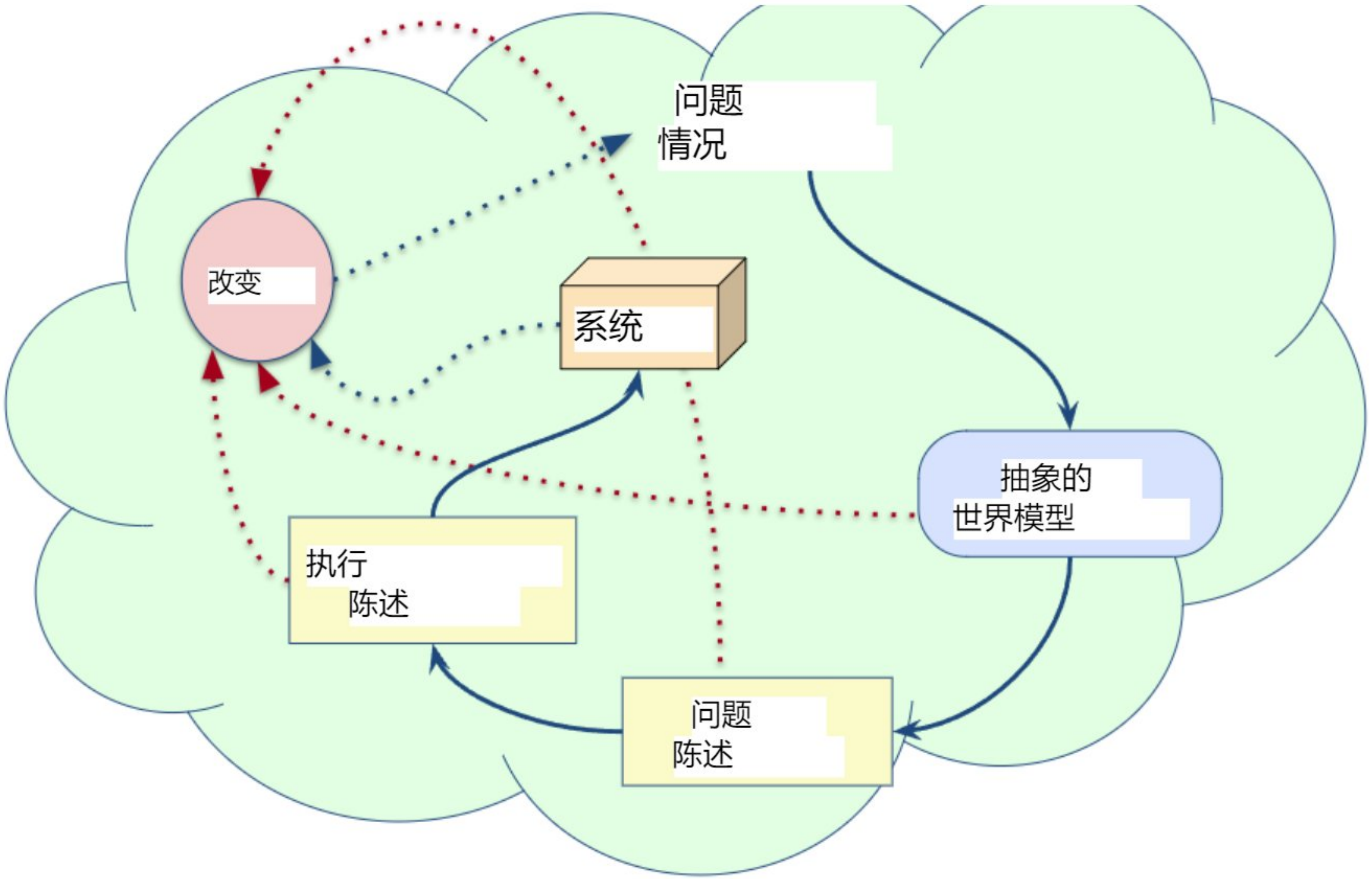


Table of Contents

- RE framework
  - System context
  - **Core activities**
  - Requirements artefacts
  - Validation
  - Management

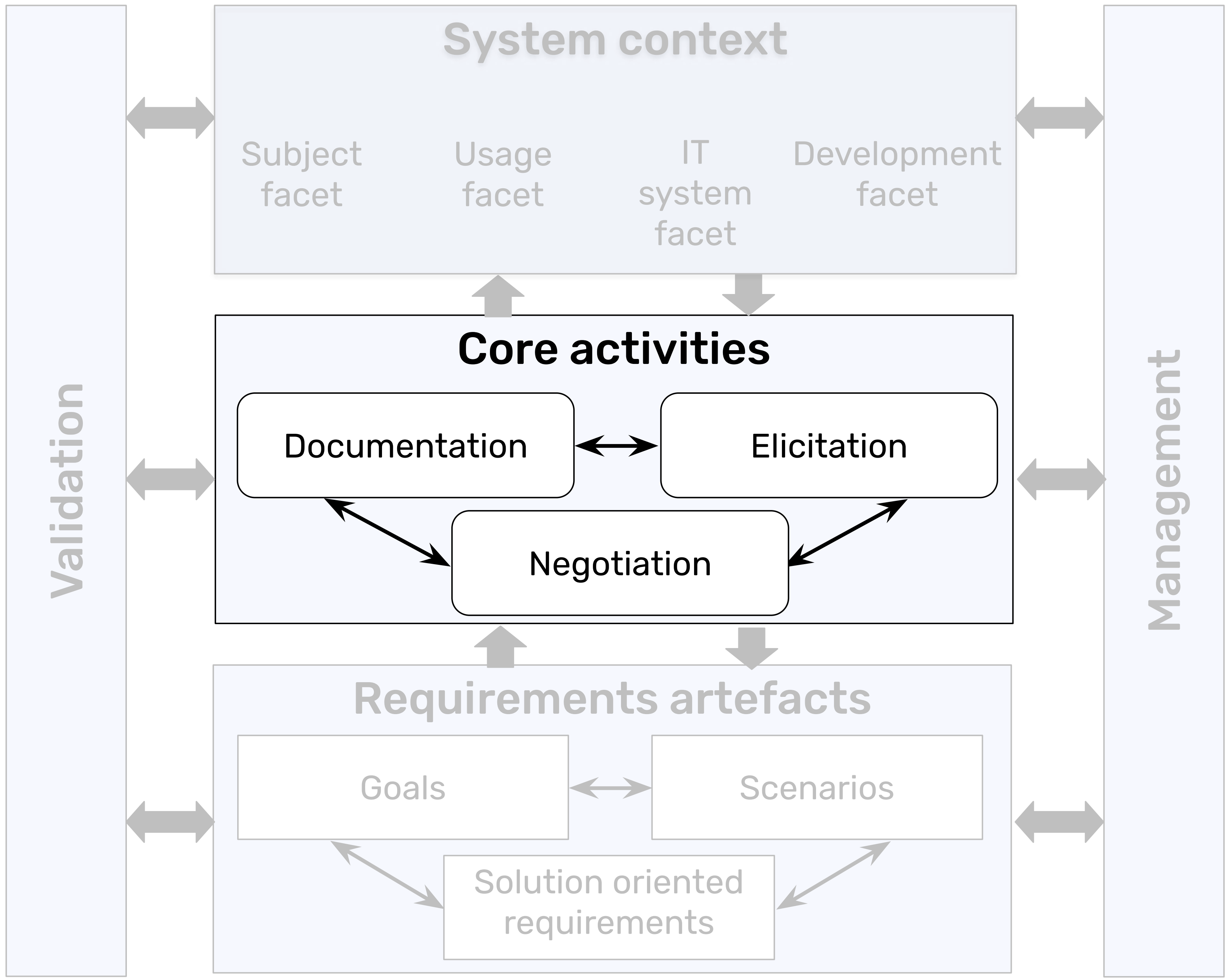
- Pohl K. (2010) Requirements Engineering, Springer

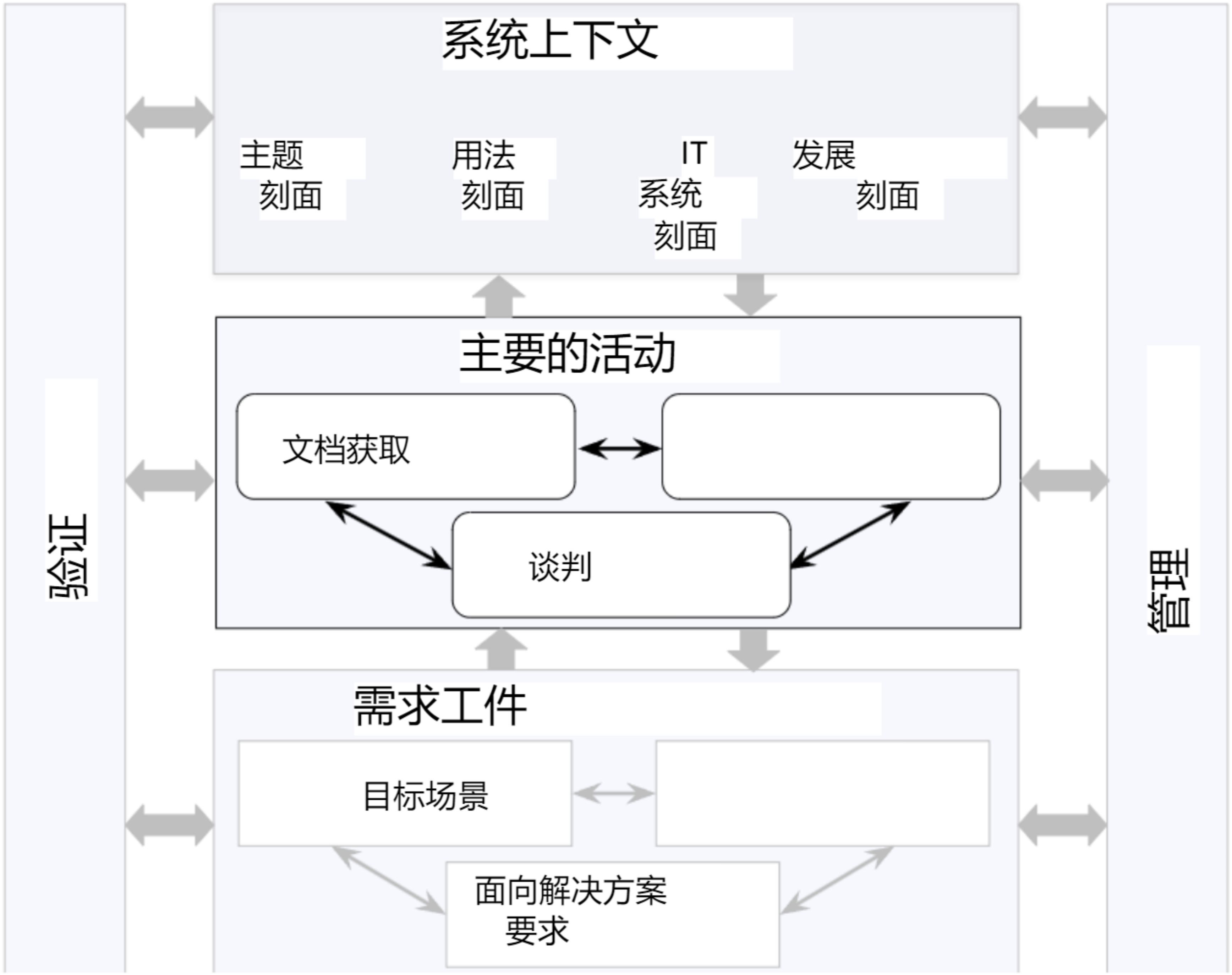
目录

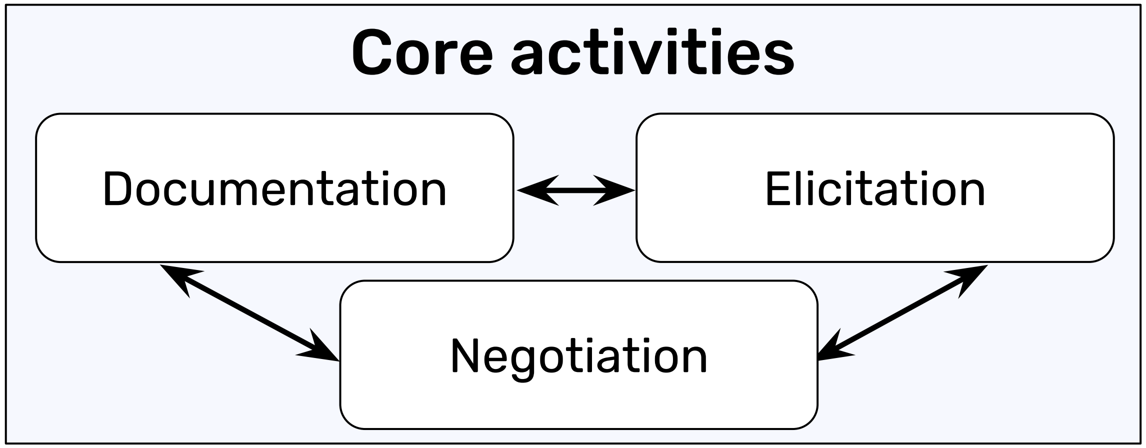
- RE 框架 – 系统上下文

  - 主要的活动
  - 需求工件
  - 验证
  - 管理

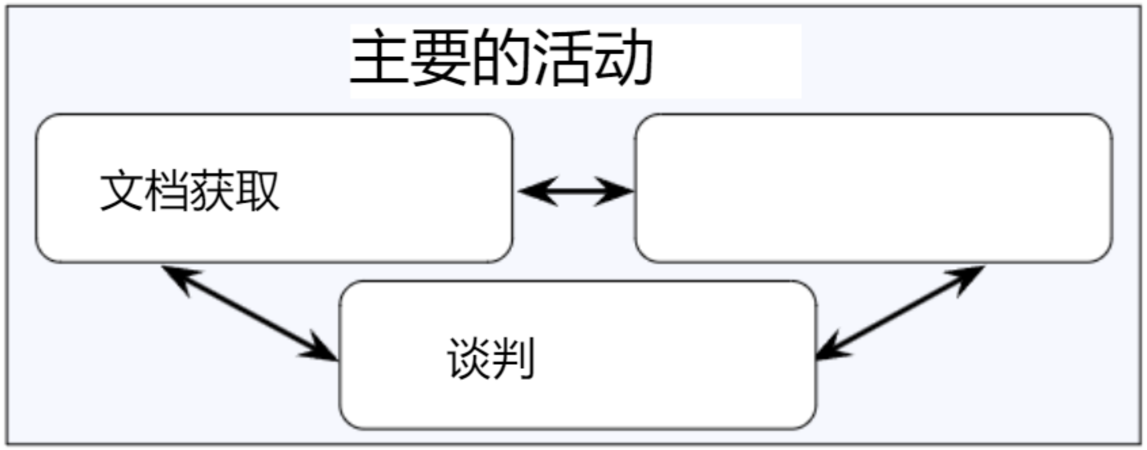
- Pohl K. (2010) 需求工程，施普林格


System context
Subject facet
Usage facet
IT system facet
Development facet
Core activities
Documentation
Elicitation
Negotiation
Requirements artefacts
Goals
Scenarios
Solution oriented requirements
Validation
Management

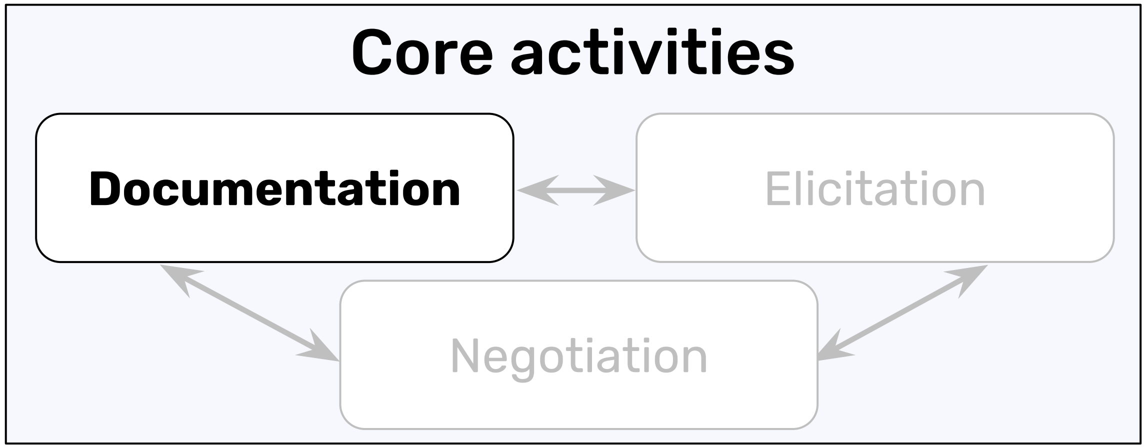

系统上下文
主题 刻面
用法 刻面
IT 系统 刻面
发展 刻面
主要的活动
文档获取
谈判
需求工件
目标场景
面向解决方案 要求
验证
管理



- performed iteratively / in parallel

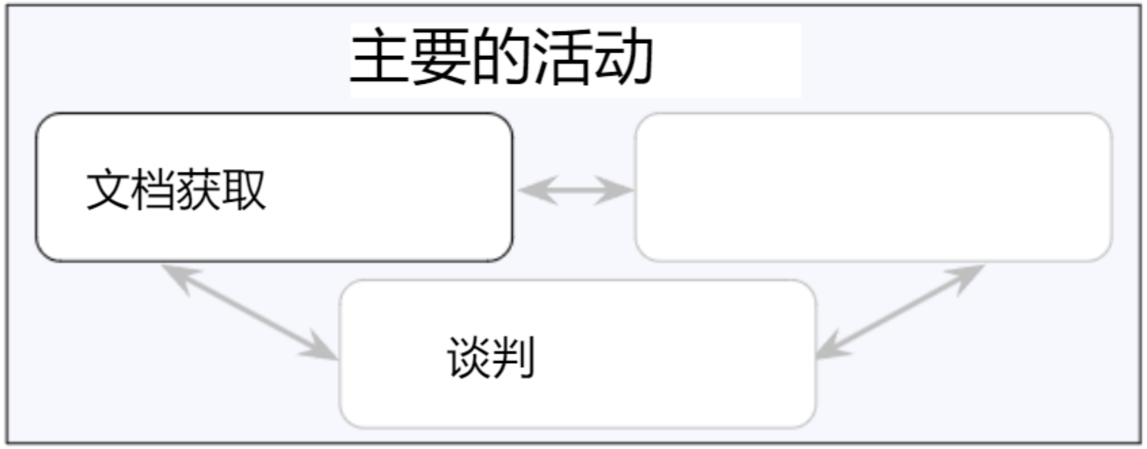


- 迭代/并行执行



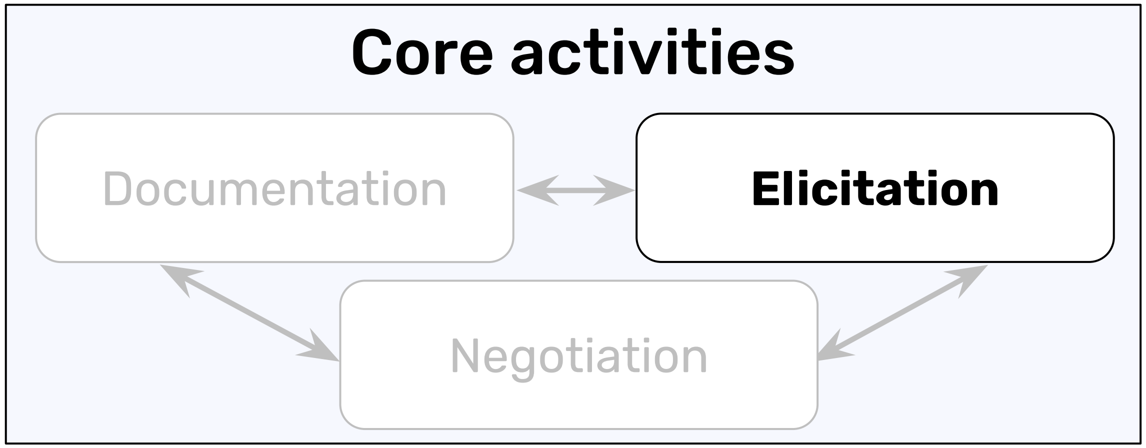
Document important information elicited or developed when performing a core the RE activity

- *i.e.*, documentation, elicitation, negotiation, validation and/or management



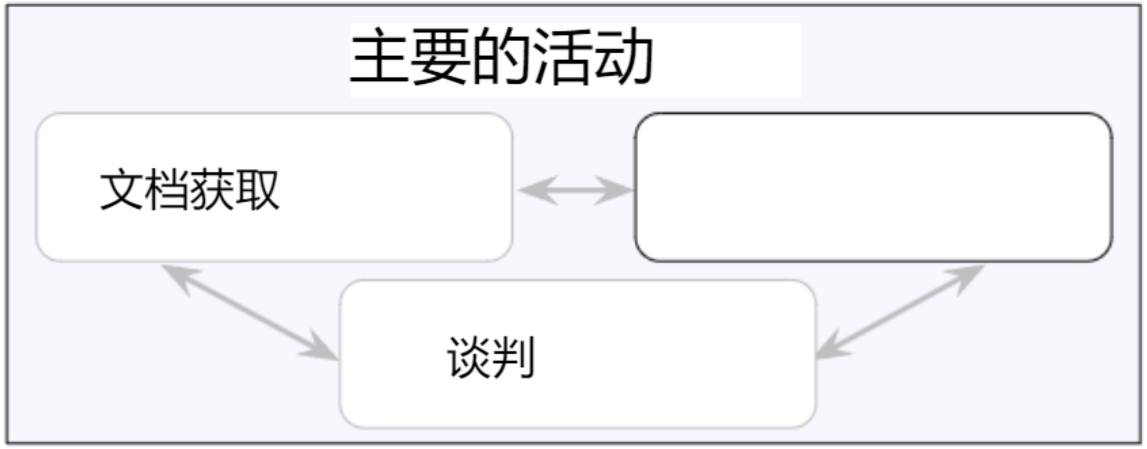
迭代执行

- 记录执行核心 RE 活动时得出或开发的重要信息
  - 即记录、启发、谈判、验证和/或管理



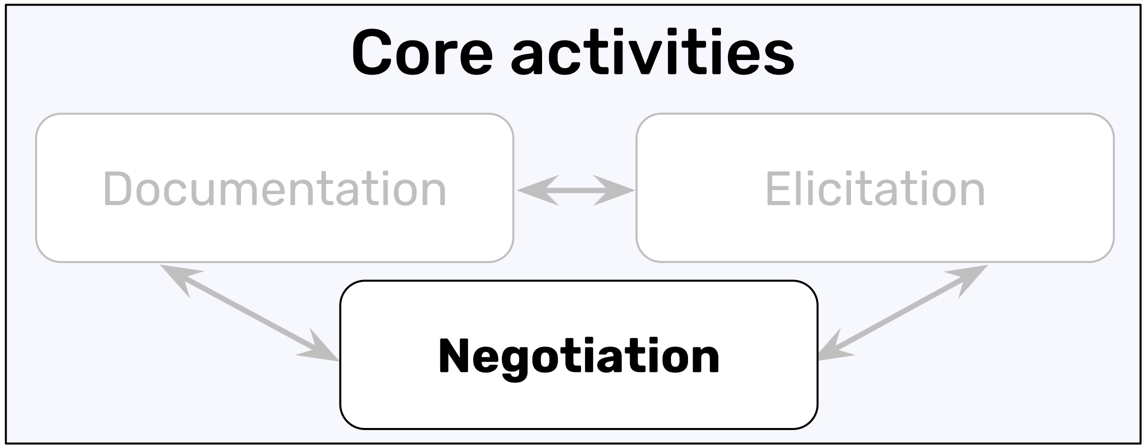
Achieve progress in the content dimension by <u>eliciting new requirements</u> as well as <u>detailed information</u> about existing requirements

- Elicit (extract) all requirements at the level of detail for the system to be developed



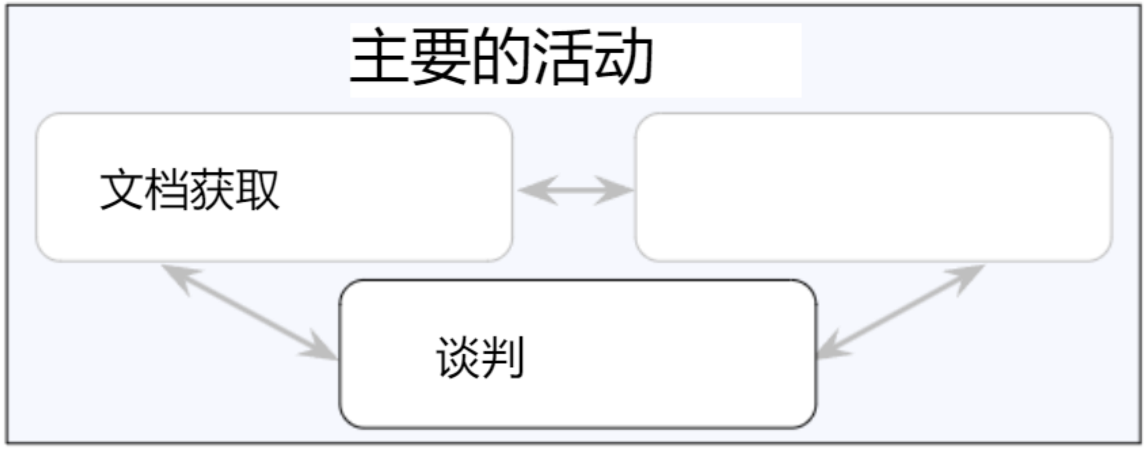
通过引出新的要求以及有关现有要求的详细信息，在内容维度上取得进展

- 引出（提取）要开发的系统的详细级别的所有需求



Achieve agreement among all stakeholders about the requirements

- has to deal with conflicts about requirements

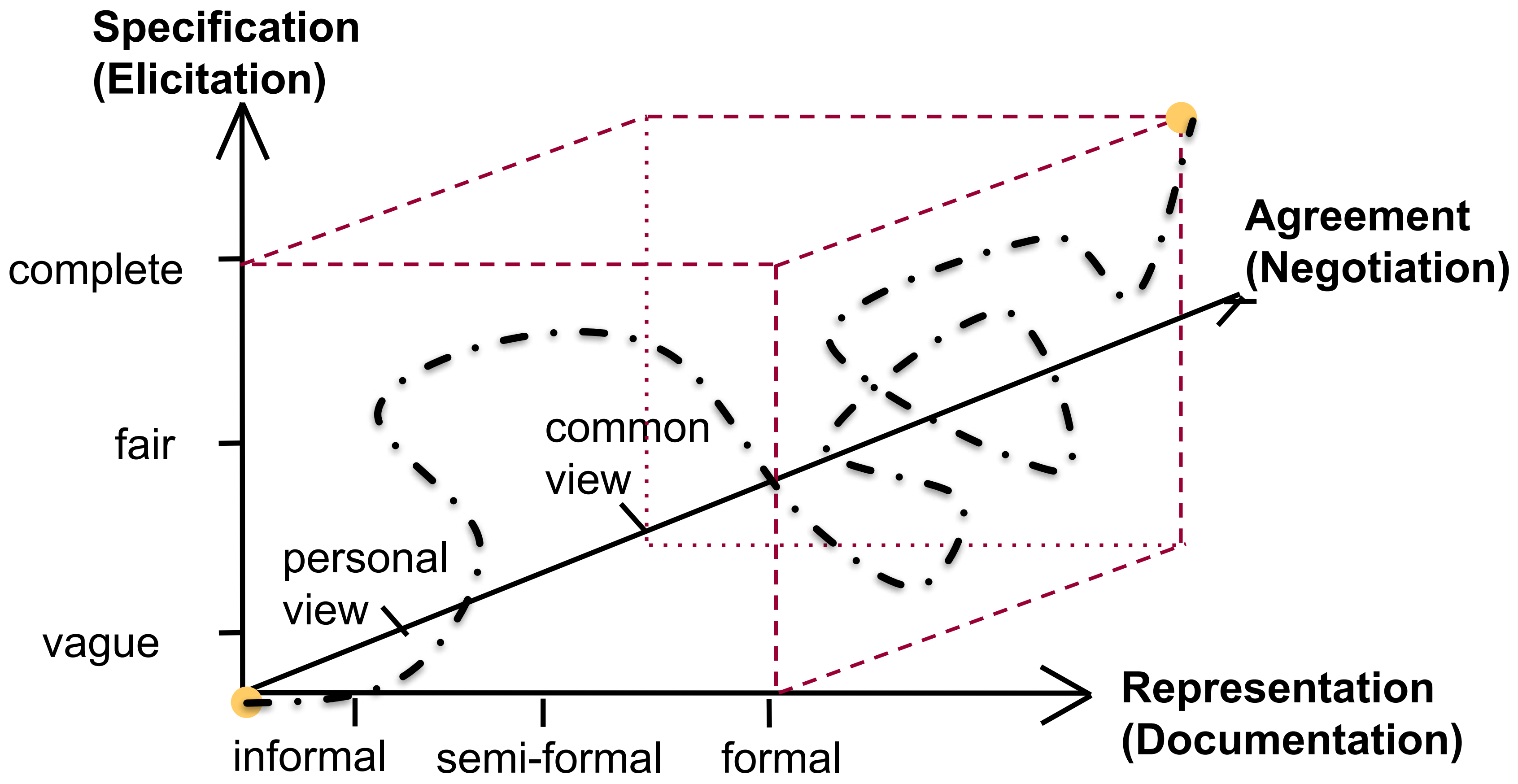


所有利益相关者就要求达成一致

- 必须处理需求冲突

Is there a “Requirements Lifecycle”

Source: Adapted from Pohl, CAISE 1993



是否有“ 需求生命周期”

资料来源：改编自 *Pohl*，*CAISE 1993*

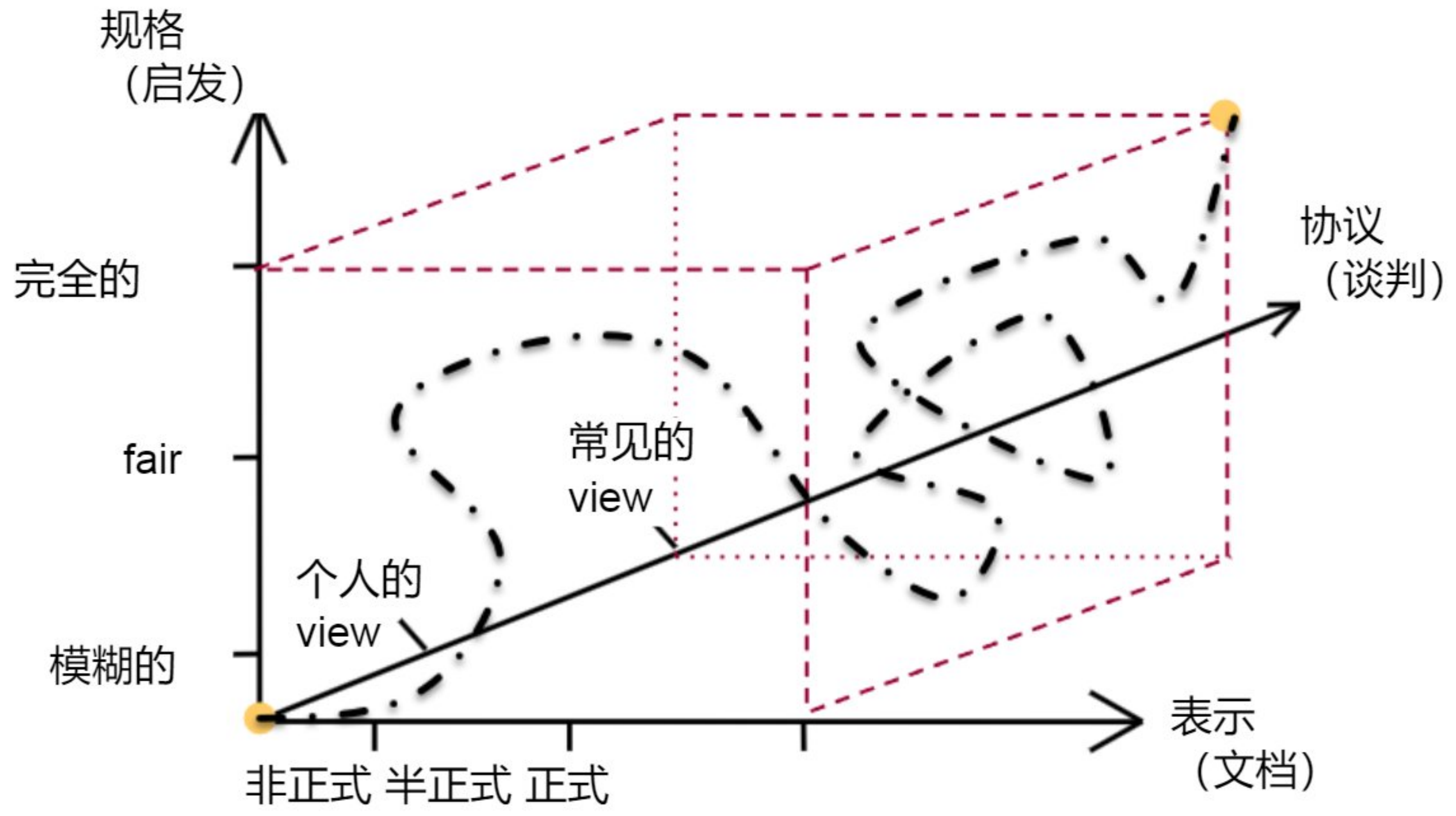


Table of Contents

- RE framework
  - System context
  - Core activities
  - **Requirements artefacts**
  - Validation
  - Management

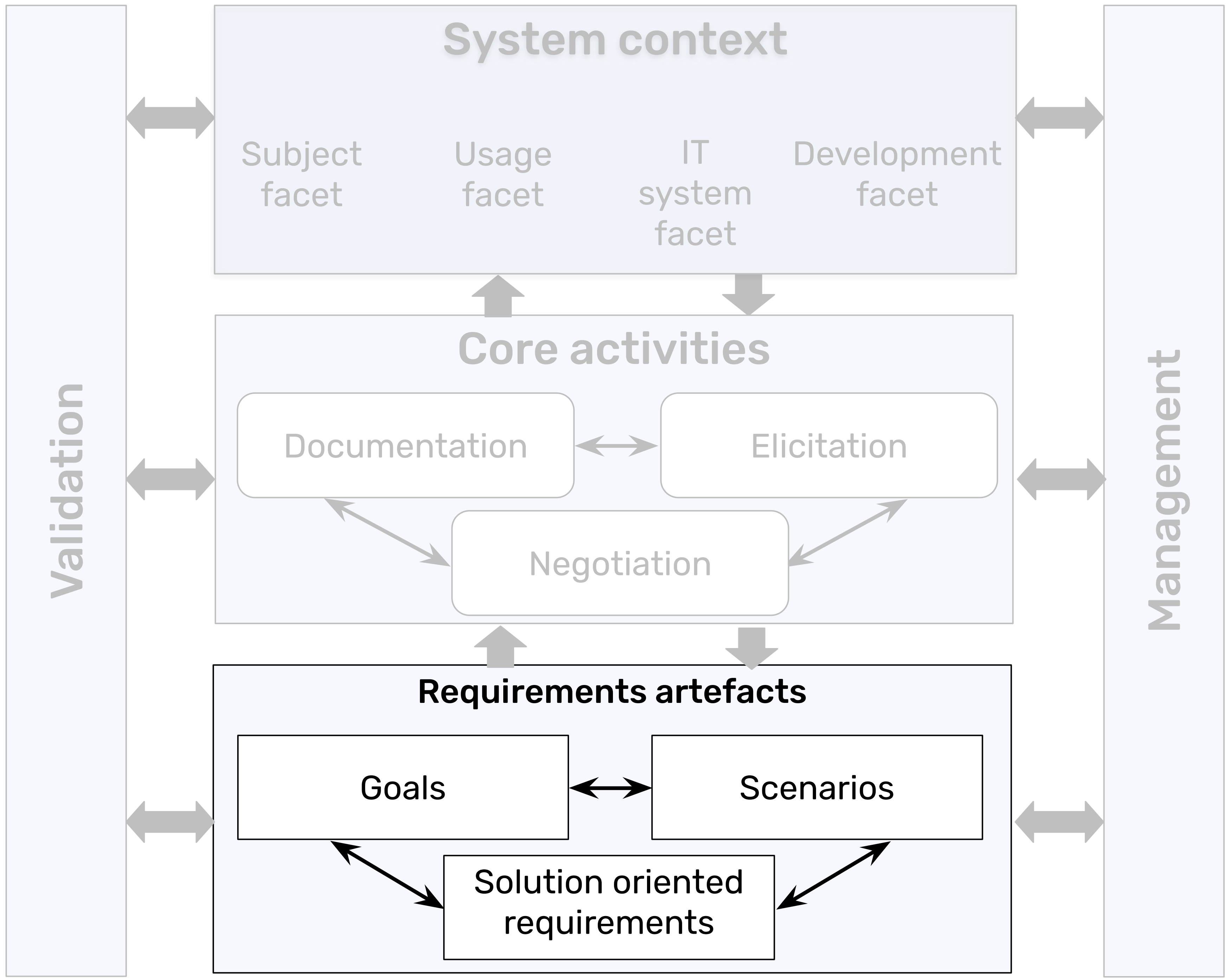
- Pohl K. (2010) Requirements Engineering, Springer

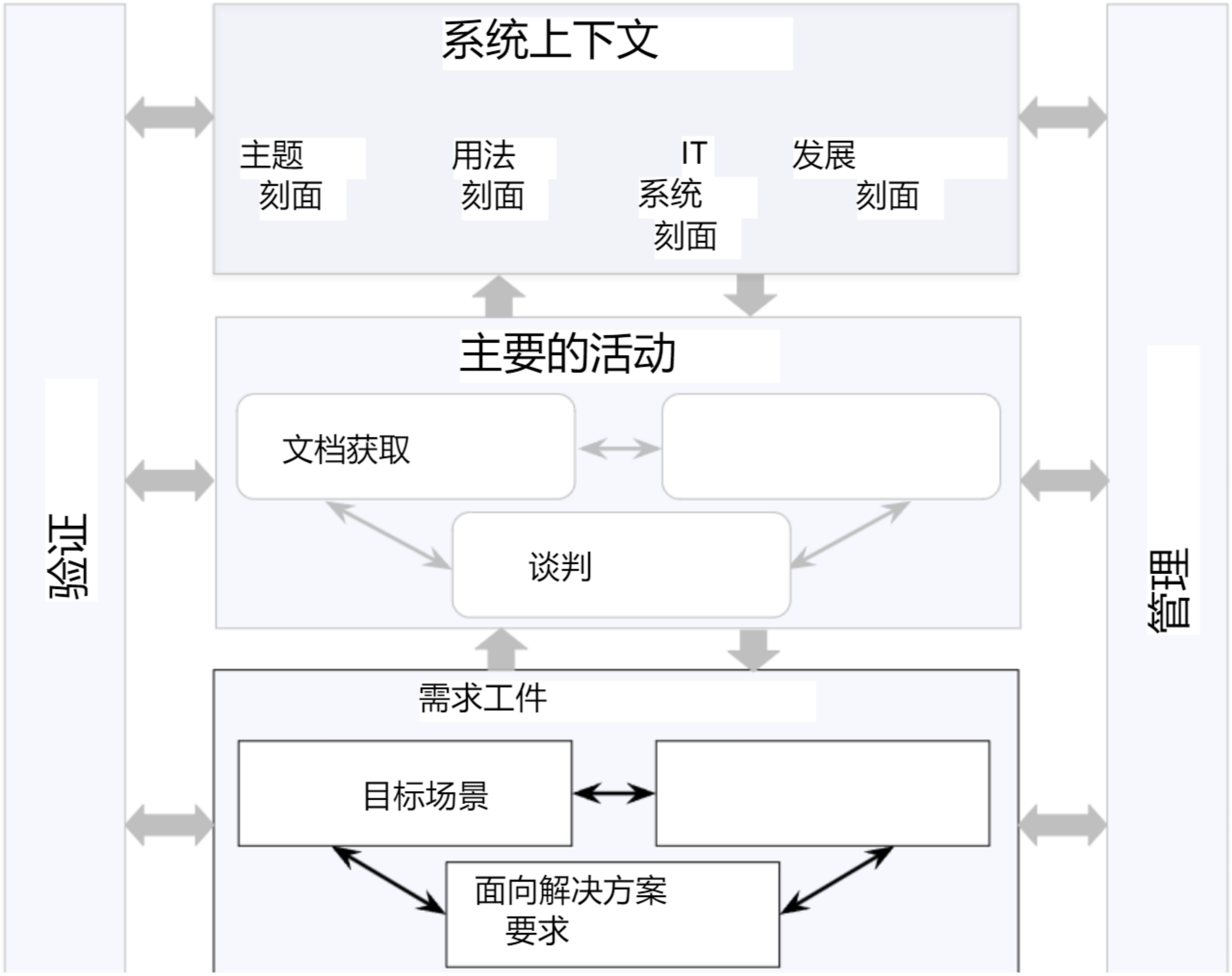
目录

- RE 框架 – 系统背景 – 核心活动

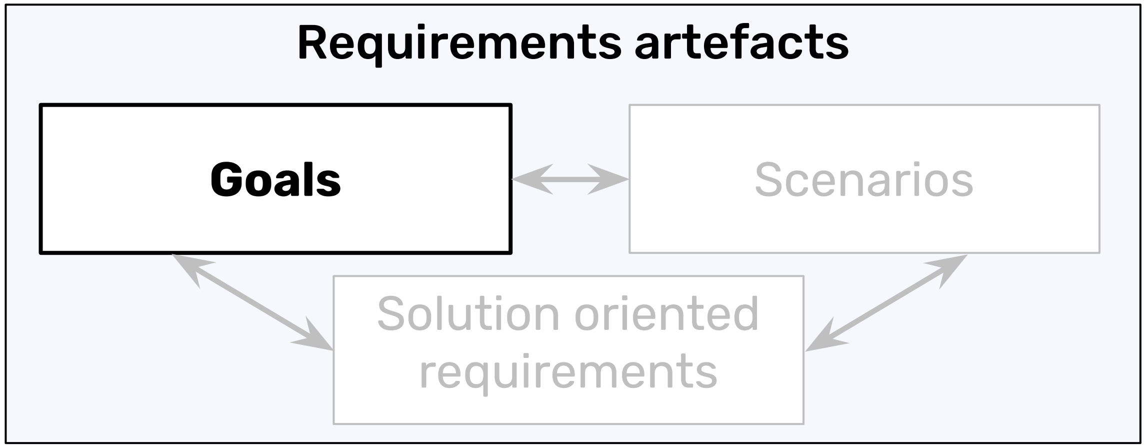
  - 需求工件
  - 验证
  - 管理

- Pohl K. (2010) 需求工程，施普林格

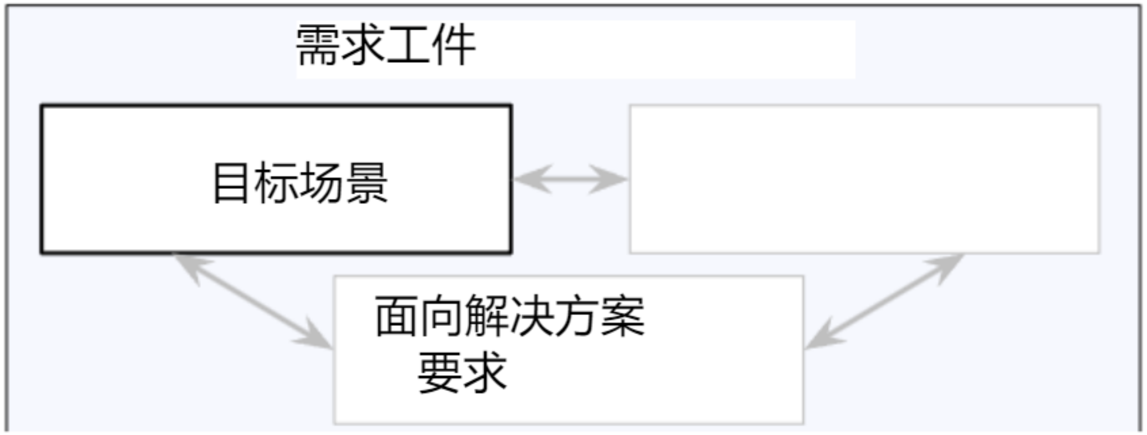

System context
Subject facet
Usage facet
IT system facet
Development facet
Core activities
Documentation
Elicitation
Negotiation
Requirements artefacts
Goals
Scenarios
Solution oriented requirements
Validation
Management


系统上下文
主题 刻面
用法 刻面
IT 系统 刻面
发展 刻面
主要的活动
文档获取
谈判
需求工件
目标场景
面向解决方案 要求
验证
管理

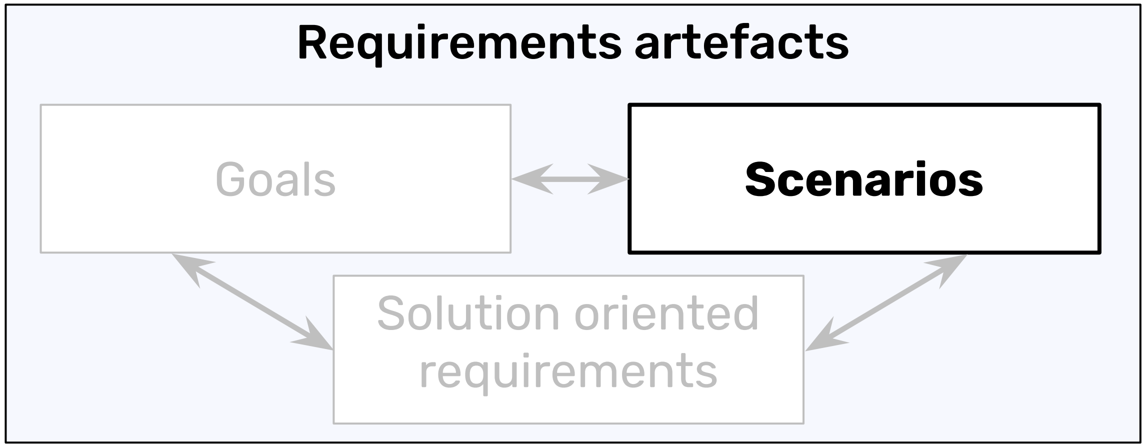
Intention with regard to objectives, properties, or use of the system



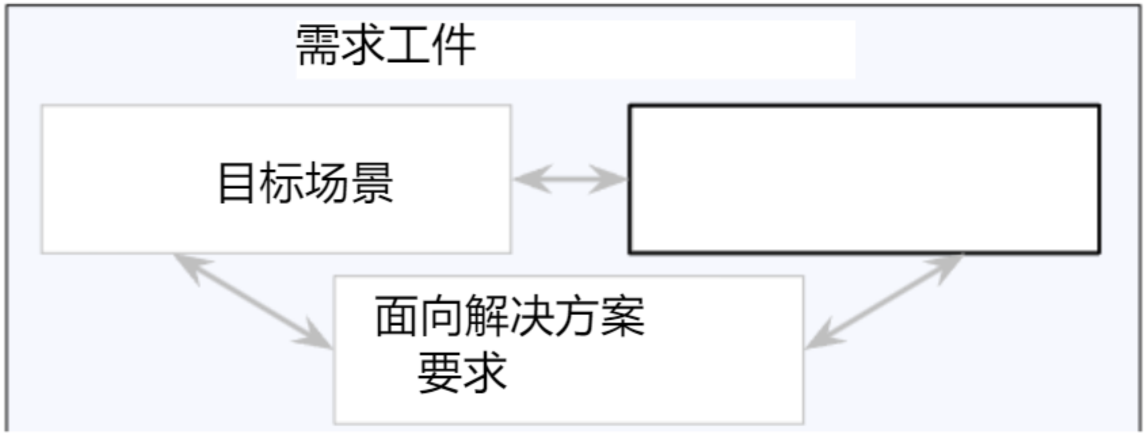
关于系统的目标、属性或使用的意图



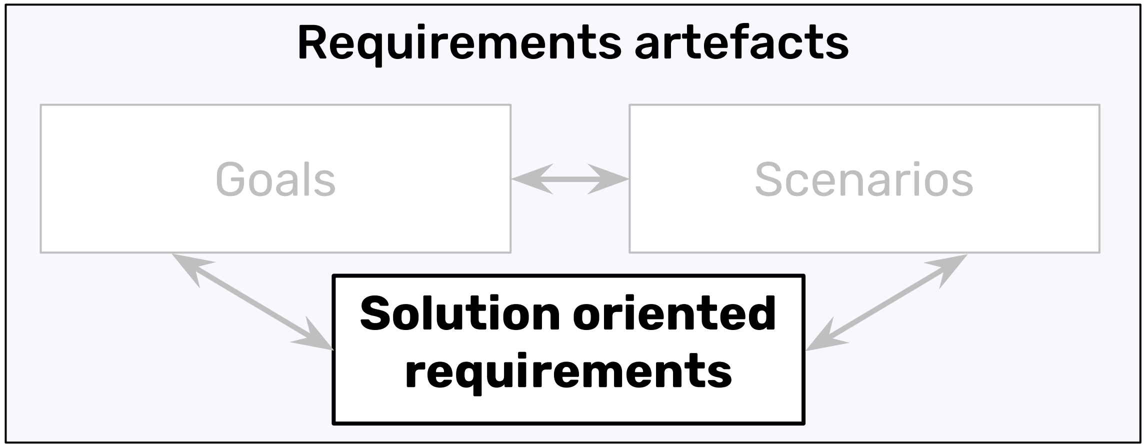
Document sequences of interactions in which the system satisfies some goals or fails to satisfy them



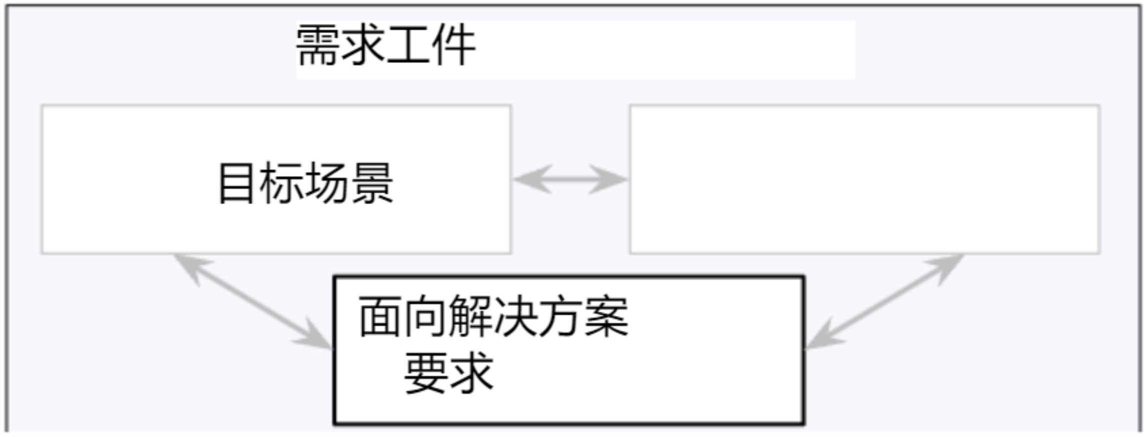
或系统的使用 记录系统满足或未能满足某些目标的交互序列



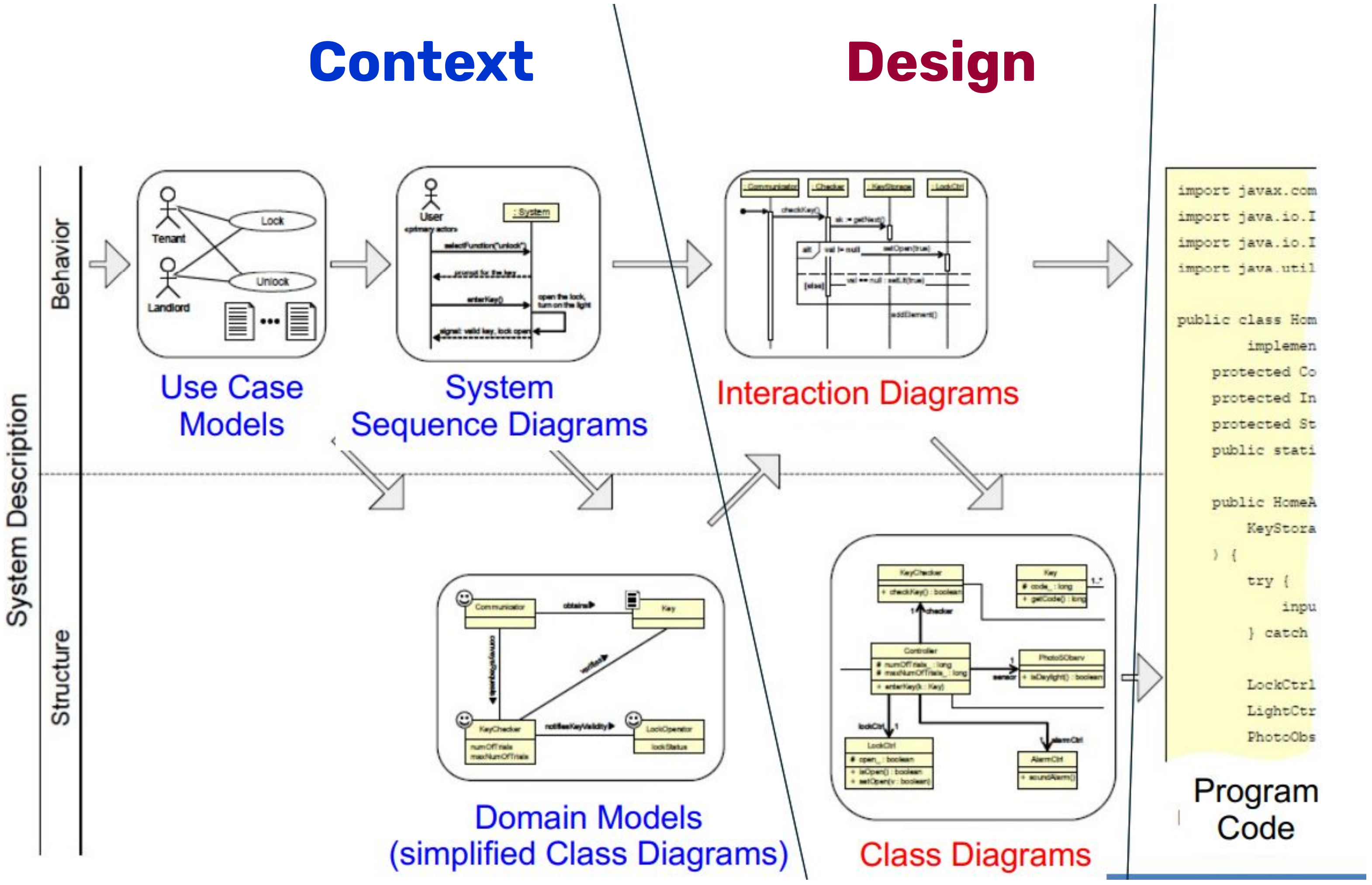
Specify requirements at the required level of detail, the desired properties and features of the system to be developed



指定所需详细程度的需求、要开发的系统的所需属性和功能



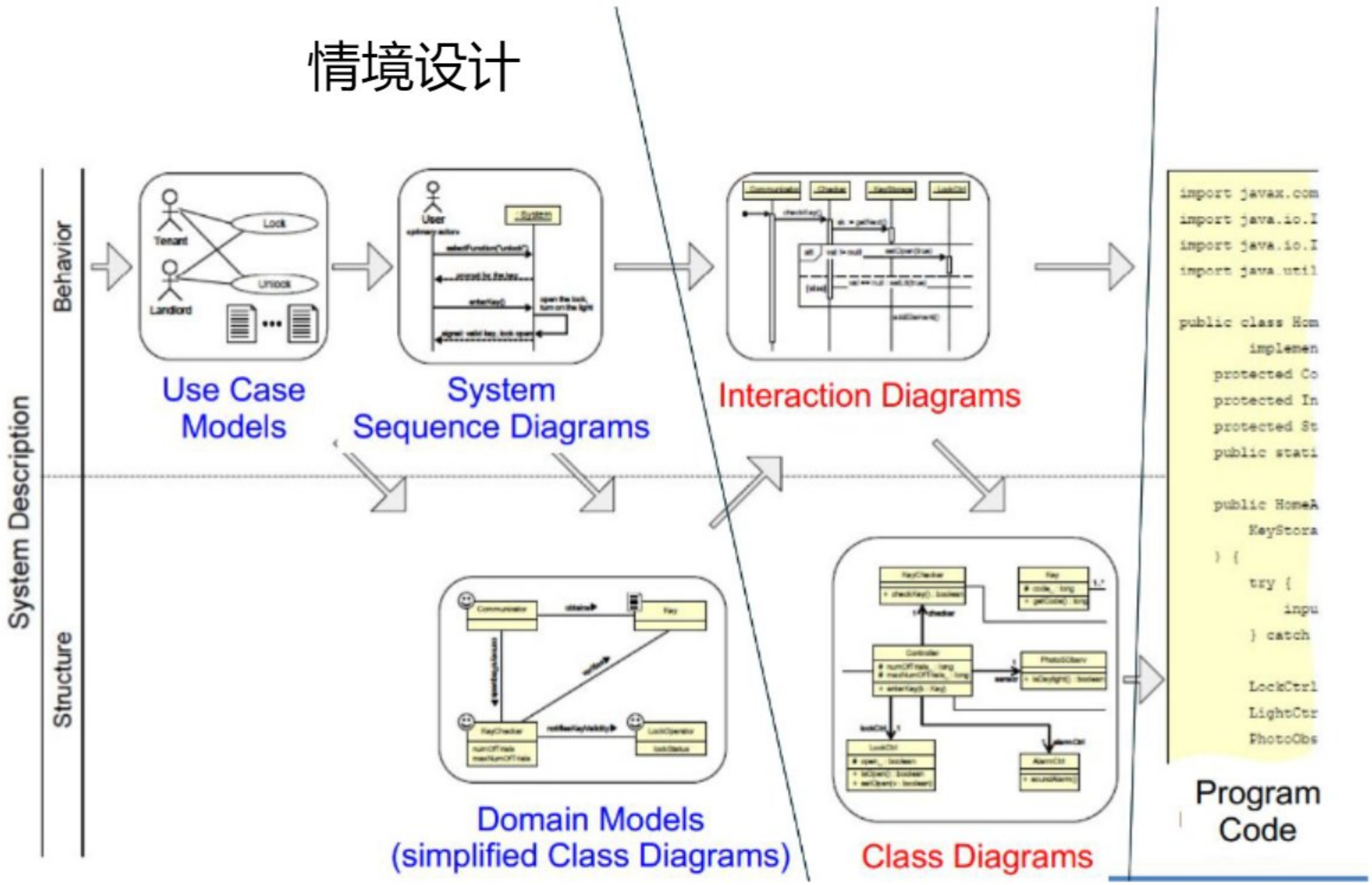
Requirements artefacts



```
import javax.com
import java.io.I
import java.io.I
import java.util

public class Hom
        implemen
    protected Co
    protected In
    protected St
    public stati

    public HomeA
        KeyStora
    ) {
        try {
            inpu
        } catch

        LockCtrl
        LightCtr
        PhotoObs
```

需求工件

情境设计



```
import javax.com
import java.io.I
import java.io.I
import java.util

public class Hom
        implemen
    protected Co
    protected In
    protected St
    public stati

    public HomeA
        KeyStora
    ) {
        try {
            inpu
        } catch

        LockCtrl
        LightCtr
        PhotoObs
```

Table of Contents

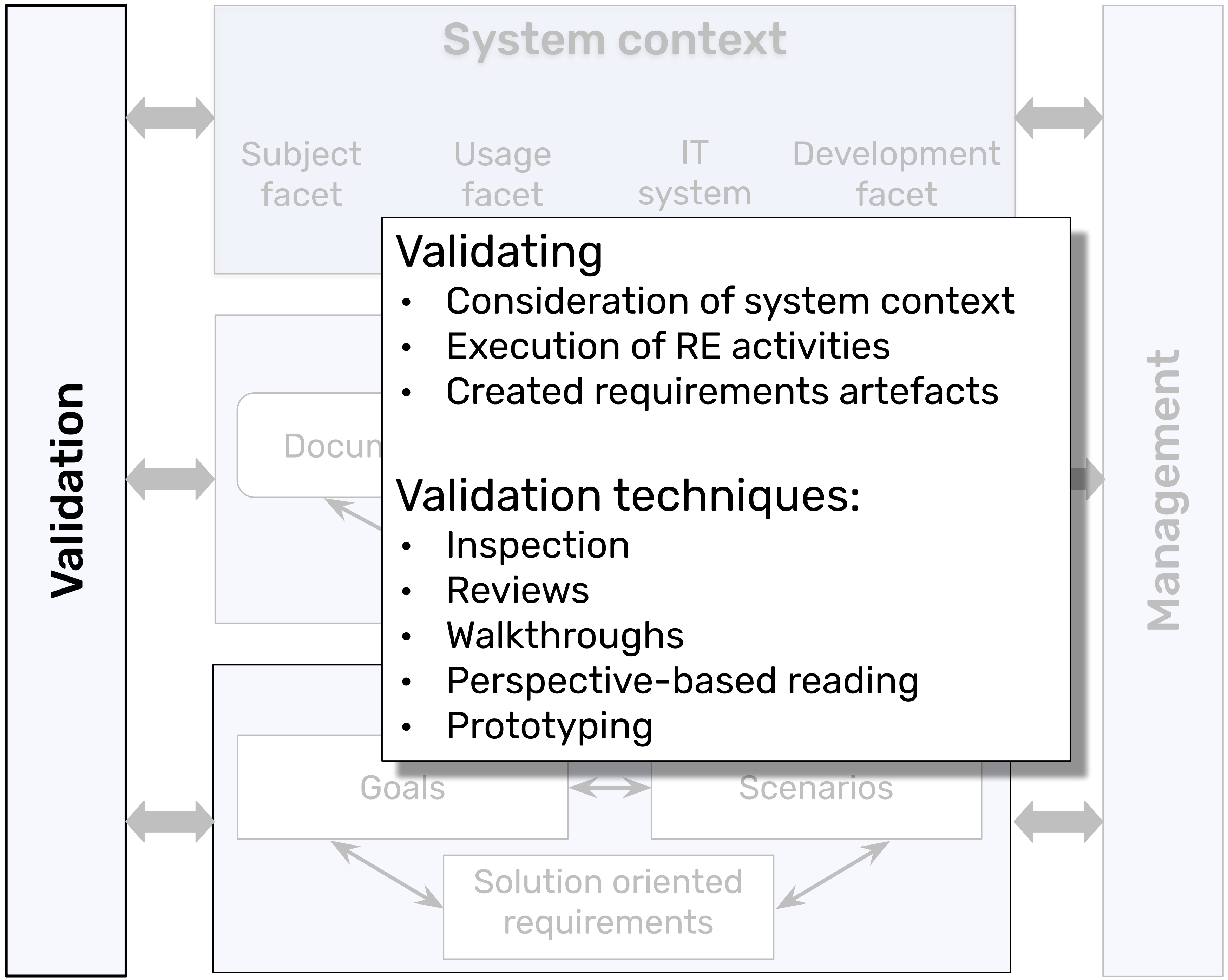
- **RE framework**
  - System context
  - Core activities
  - Requirements artefacts
  - **Validation**
  - Management

- Pohl K. (2010) Requirements Engineering, Springer

目录

- 可再生能源框架
  - 系统上下文
  - 主要的活动
  - 需求工件
  - 验证
  - 管理

- Pohl K. (2010) 需求工程，施普林格

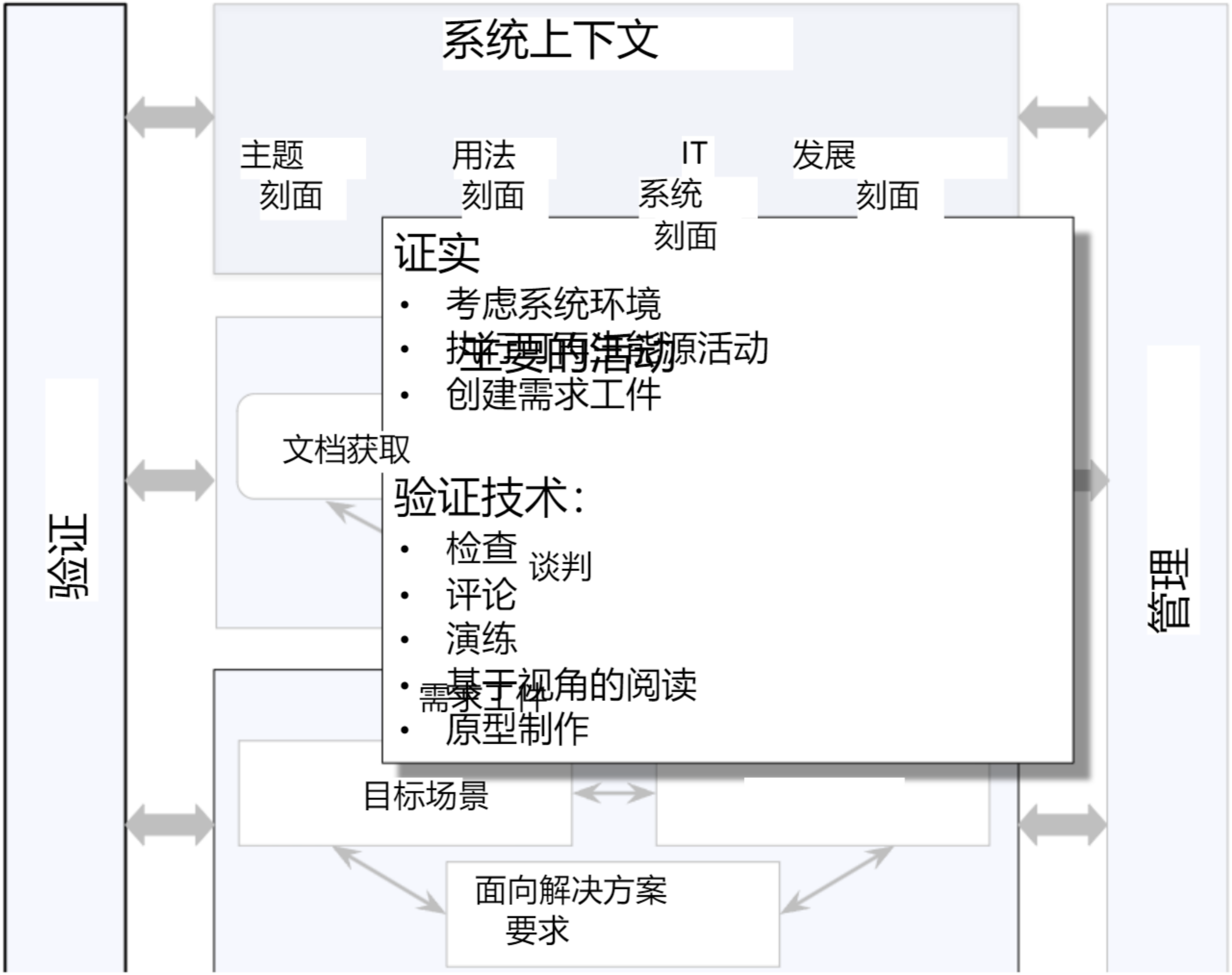


Validating

- Consideration of system context
- Execution of RE activities
- Created requirements artefacts

Validation techniques:

- Inspection
- Reviews
- Walkthroughs
- Perspective-based reading
- Prototyping



证实

- 考虑系统环境
- 执行面向潜能源活动
- 创建需求工件

验证技术：

- 检查
- 评论
- 演练
- 基于视角的阅读
- 原型制作

Table of Contents

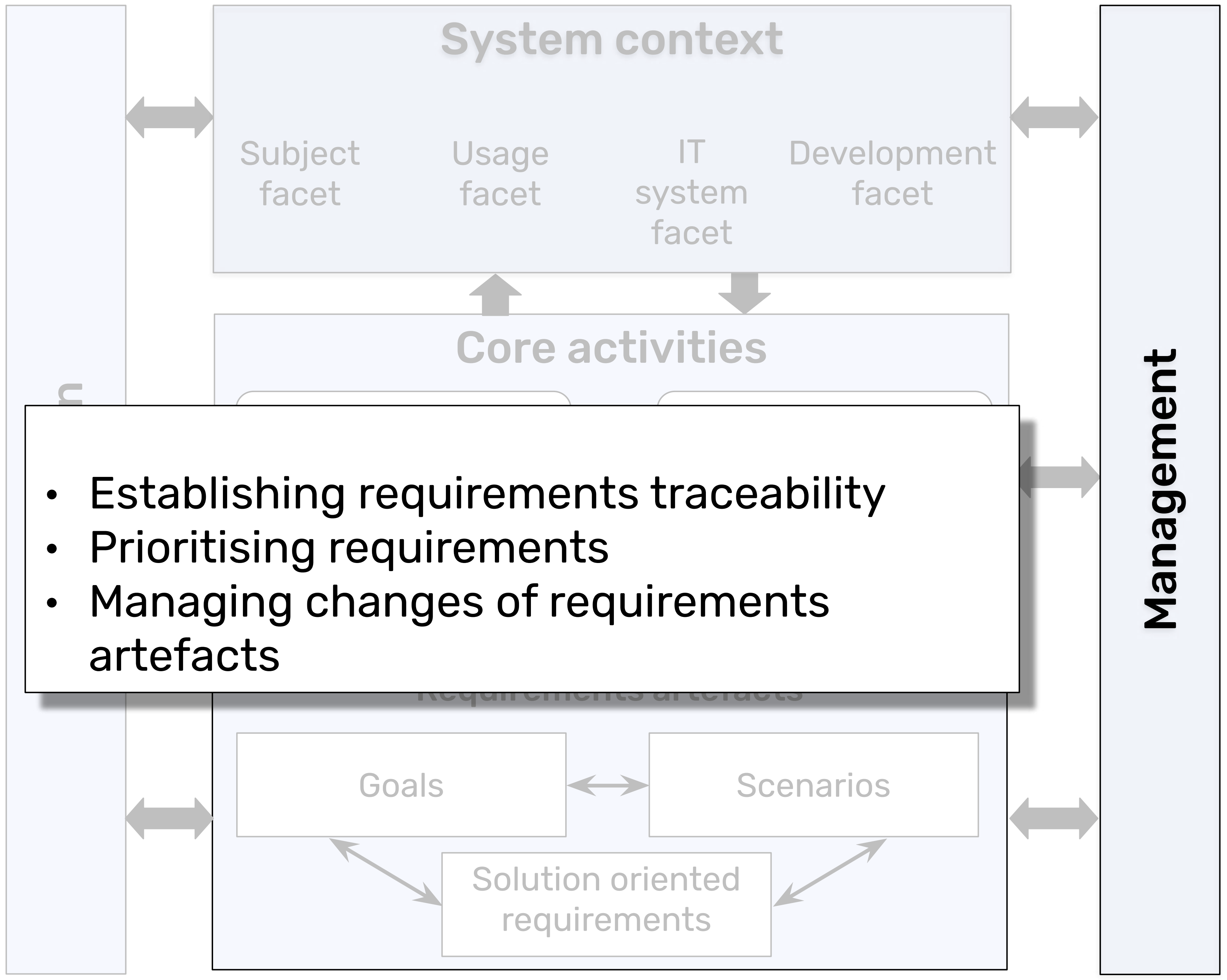
- **RE framework**
  - System context
  - Core activities
  - Requirements artefacts
  - Validation
  - **Management**

- Pohl K. (2010) Requirements Engineering, Springer

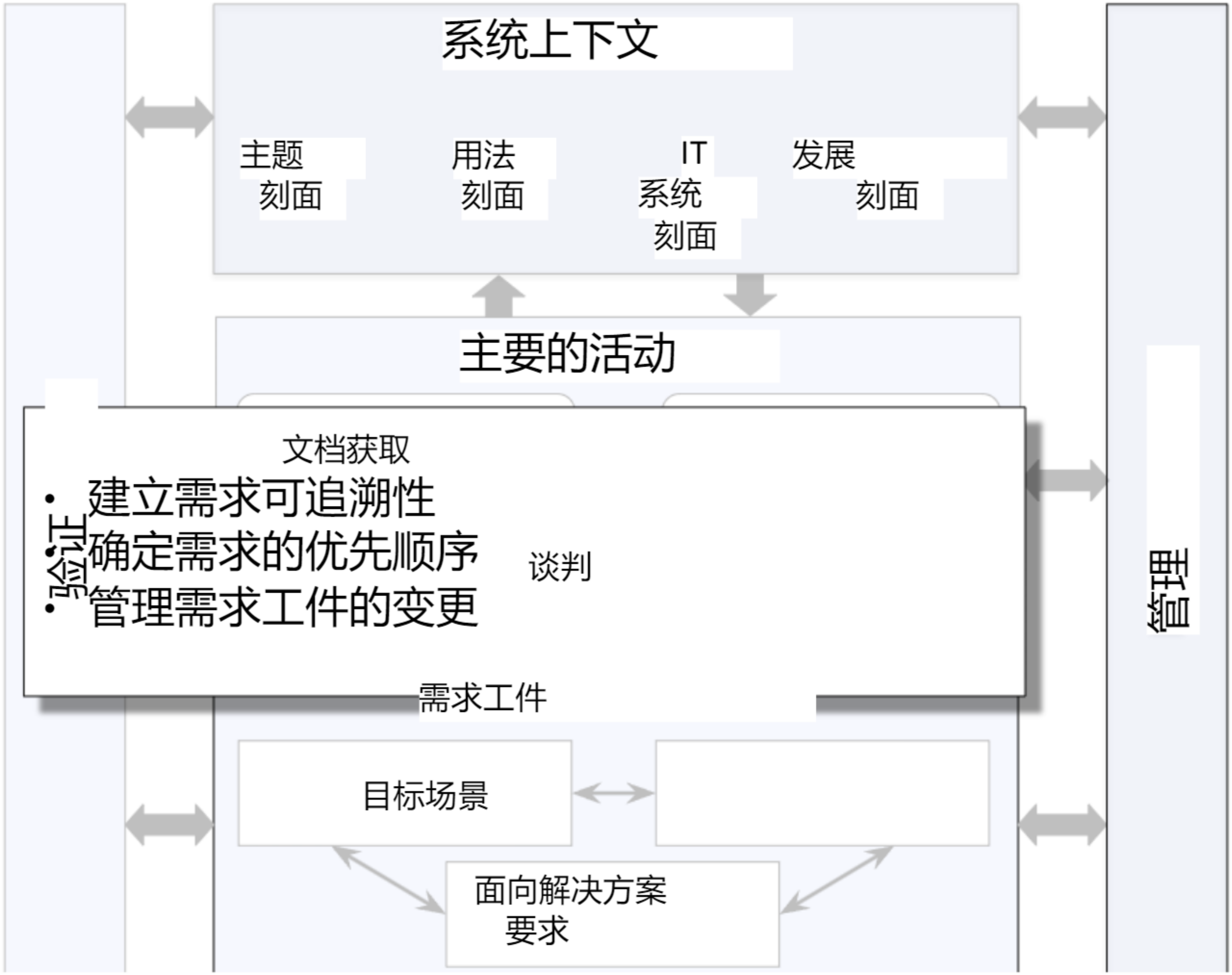
目录

- 可再生能源框架
  - 系统上下文
  - 主要的活动
  - 需求工件
  - 验证
  - 管理

- Pohl K. (2010) 需求工程，施普林格

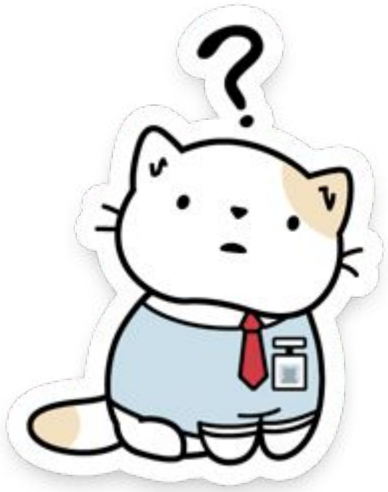


- Establishing requirements traceability
- Prioritising requirements
- Managing changes of requirements artefacts



- 建立需求可追溯性
- 确定需求的优先顺序
- 管理需求工件的变更

Any questions



任何问题

How does RE framework fit to the software development lifecycles (SDLCs)?

RE 框架如何适应软件开发生命周期 **(SDLC)**?

Lifecycle of Engineering Project

- Lifecycle models
  - Useful for comparing projects in general terms
  - Not enough detail for project planning
- Examples:
  - Sequential models: Waterfall, V model
  - Rapid Prototyping
  - Phased Models: Incremental, Evolutionary
  - Iterative Models: Spiral
  - Agile Models: eXtreme Programming, Scrum, Kanban

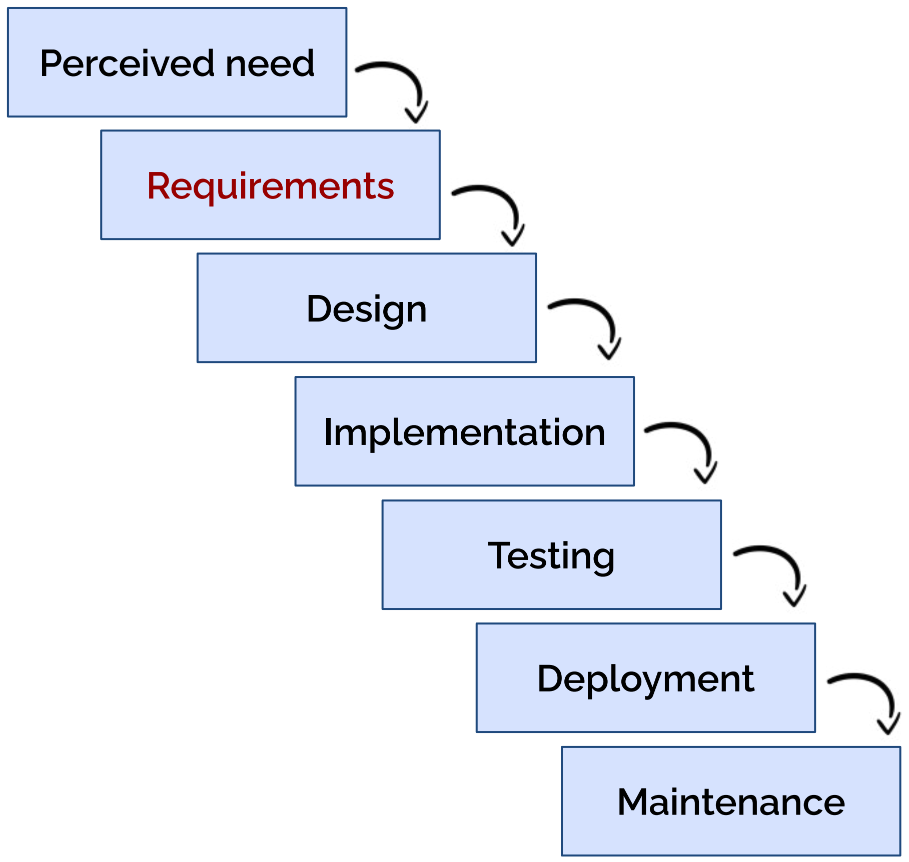
的生命周期

工程

• 生命周期模型 – 对于一般性比较项目很有用 – 对于项目规划来说细节不够

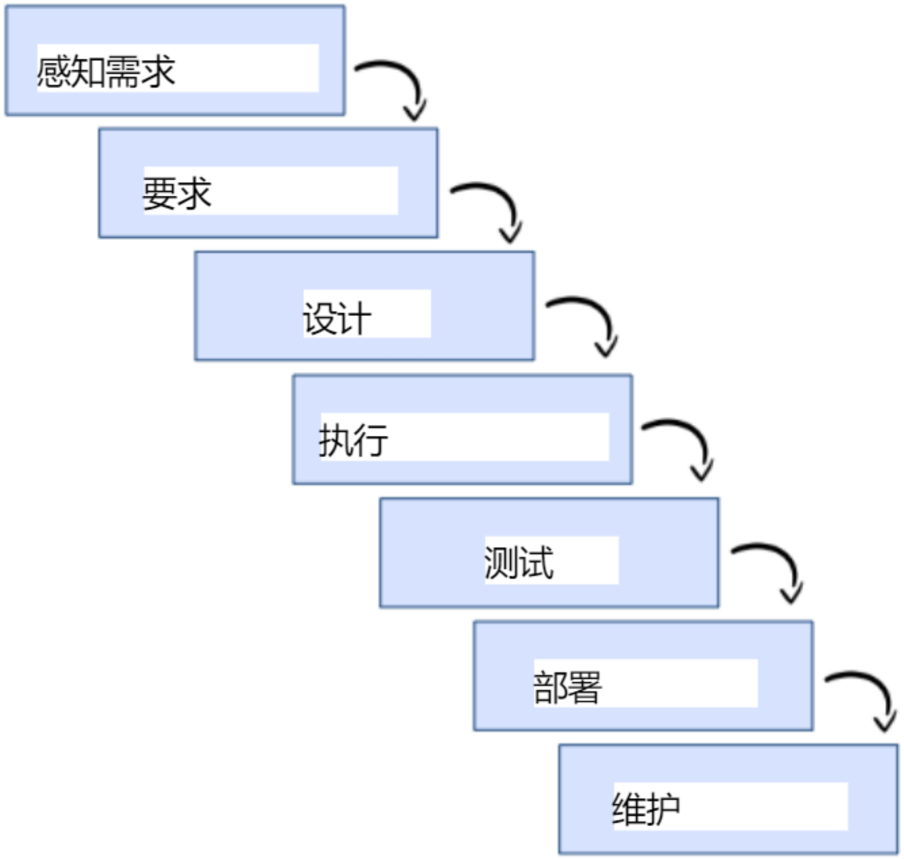
• 示例： – 顺序模型：瀑布式、V 模型 – 快速原型 – 分阶段模型：增量式、演化式 – 迭代模型：螺旋式 – 敏捷模型：极限编程、Scrum、看板

Waterfall Model



- **View of development:**
  - A sequential process of stepwise refinement
  - Largely a high level management view
  - All requirements are refined up front
  - All the steps are done once in the strict order

瀑布模型



- 发展观： – 一个连续的过程

  逐步细化
  - 很大程度上是高水平管理观点
  - 所有要求均预先完善
  - 所有步骤均按严格顺序完成一次

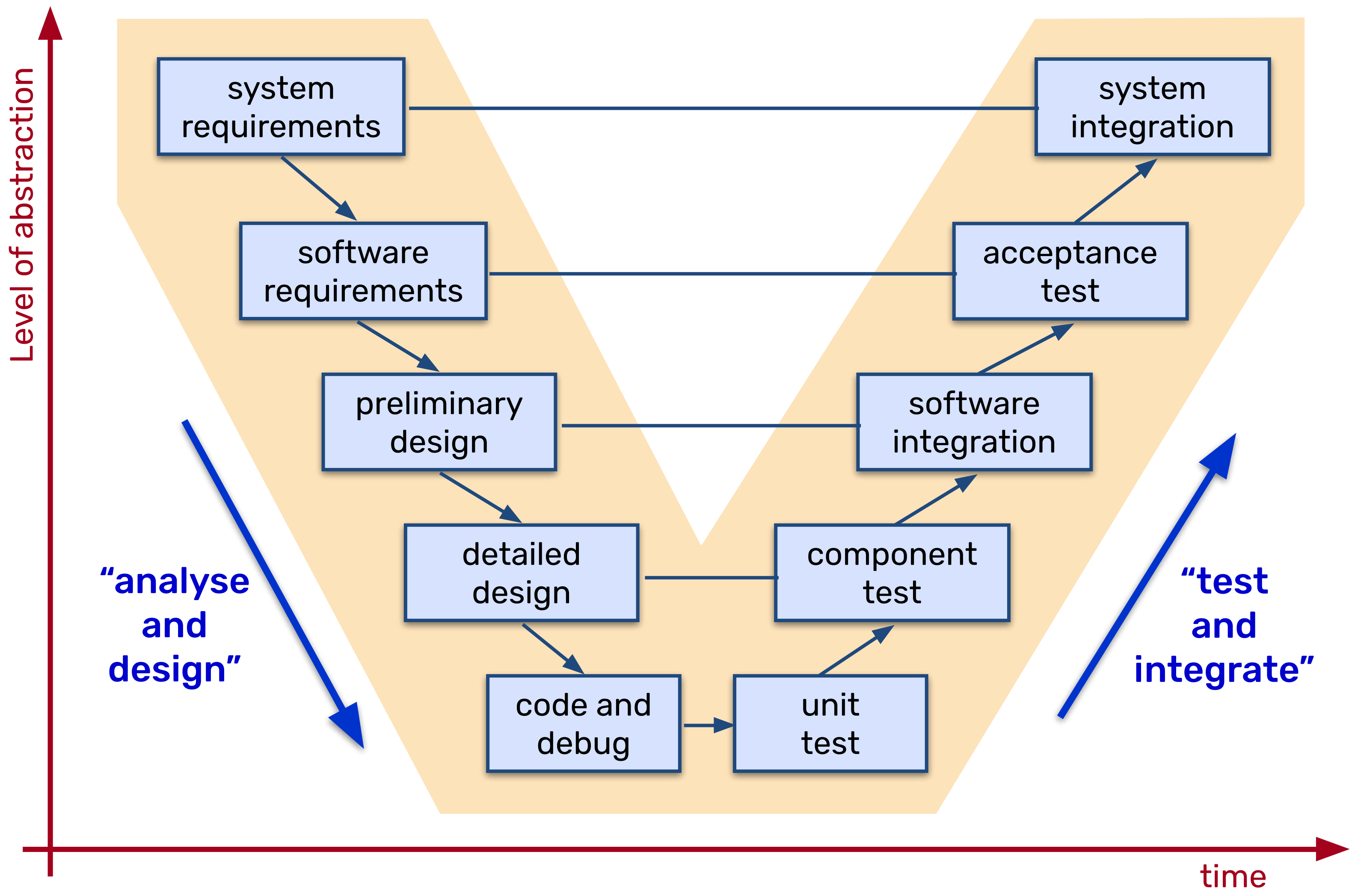
Waterfall Model

- **Pros:**
  - Easy to understand
  - Easy to monitor timewise
  - The outcome is crystal clear
- **Cons**:
  - Static view of requirements - ignores volatility
  - Lack of user involvement once specification is written
  - Unrealistic separation of specification from design
  - Doesn’t accommodate prototyping, reuse, etc.
  - Long delivery time

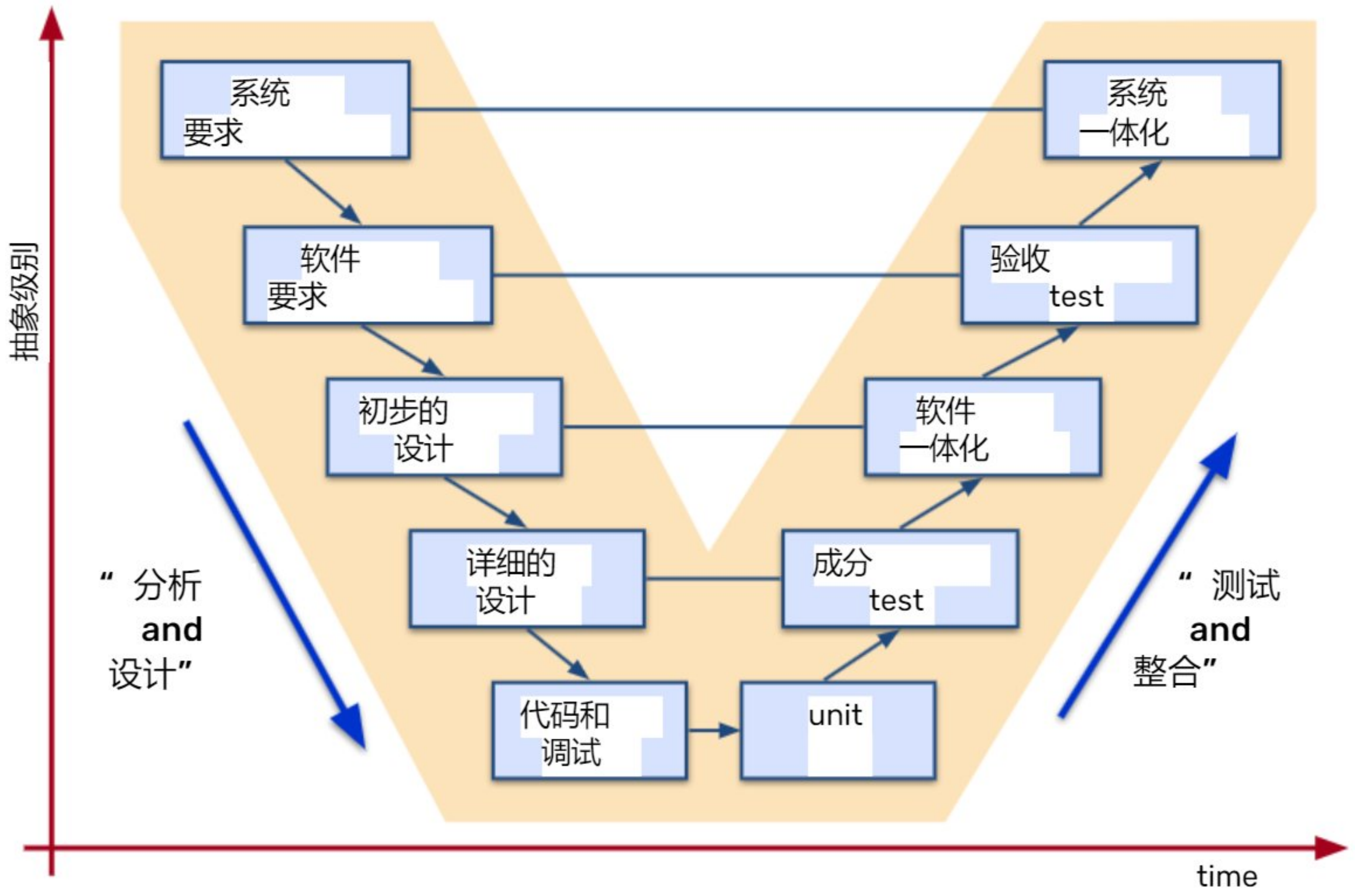
瀑布模型

- 优点：
  - 容易明白
  - 易于及时监控 – 结果一目了然
- 缺点：
  - 需求的静态视图 – 忽略波动性 – 编写规范后缺乏用户参与 – 规范与设计不切实际的分离 – 不适应原型设计、重用等 – 交付时间长

V-Model


Level of abstraction
system requirements
software requirements
preliminary design
detailed design
code and debug
unit test
component test
software integration
acceptance test
system integration
"analyse and design"
"test and integrate"
time

V型


系统
要求
软件
要求
初步的
设计
详细的
设计
代码和
调试
unit
成分
test
软件
一体化
验收
test
系统
一体化
抽象级别
“ 分析
and
设计”
“ 测试
and
整合”
time

V-Model

Pros:

- Simple and easy to use
- Each phase has specific deliverables
- Works well for small projects where requirements are easily understood

Cons:

- Little flexibility and adjusting scope is difficult and expensive
- No clear path for problems found during testing phases
- Doesn't accommodate prototyping, reuse, etc.
- Long delivery time

V型

优点：

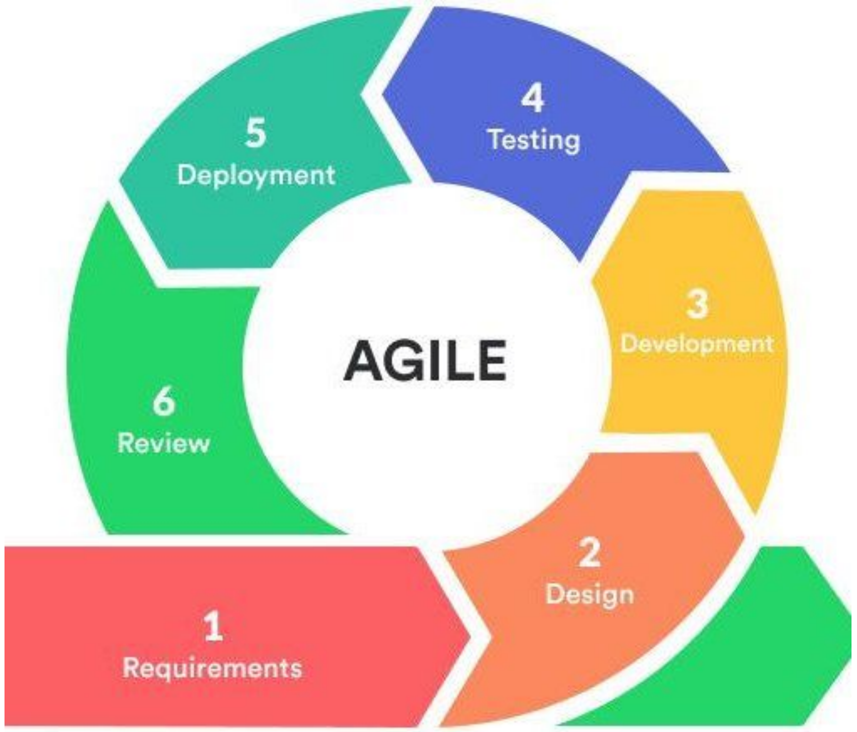
– 简单易用 – 每个阶段都有特定的可交付成果 – 非常适合需求易于理解的小型项目

缺点：

– 灵活性小，调整范围困难且昂贵 – 测试阶段发现的问题没有明确的路径 – 不适应原型设计、重用等 – 交付时间长

Agile Models

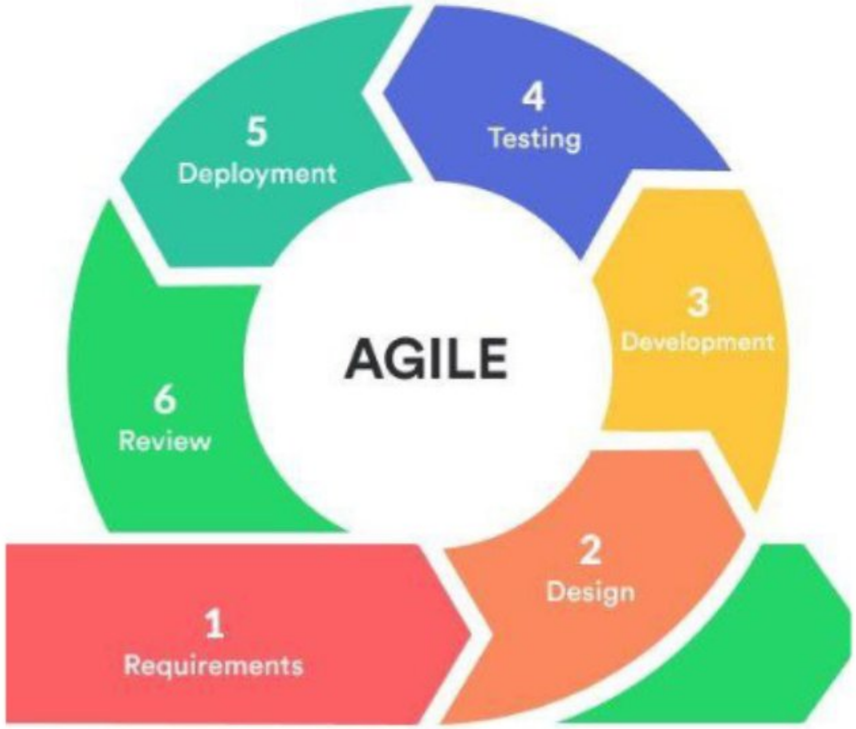
Source: Adapted from Nawrocki et al, RE'02



- Develops software iteratively
- Delivers multiple 'software increments'
- Reduce communication barriers
  - Programmer interacts with customer
- Reduce document-heavy approach
  - Documentation is expensive and of limited use
- Have faith in the people
  - Don't need fancy process models to tell them what to do!
- Is driven by customer descriptions of what is required (scenarios)
  - Rather than focusing on the contract
- Follow the manifesto of 12 principles

敏捷模型

资料来源：改编自 *Nawrocki* 等人，*RE' 02*



- 迭代开发软件
- 提供多个“软件增量”
- 减少沟通障碍
  - 程序员交互
    顾客
- 减少大量文档的方法 ○ 文档成本高昂且缺乏

  有限使用
- 对人民有信心 ○ 不需要花哨的流程模型

  告诉他们该怎么做！
- 由客户对需求（场景）的描述驱动 ○ 而不是关注

  合同
- 遵循 12 条原则宣言

Agile Models

Pros:

- Recognizes that plans are short-lived
- Adapts as changes occur
- Improved quality by finding and fixing defects quickly and identifying expectation mismatches early

Cons:

- Code can be hard to maintain
  - Focuses on working with software and lacks documentation efficiency
- Relies on oral communication
  - Mis-interpretation possible
- Assumes single customer representative
  - Multiple viewpoints not possible
- Only short term planning
  - No longer term vision

https://images.squarespace-cdn.com/content/v1/567bb0614bf118911ff0bedb/1579721881351-CHLC89T4TS6WN0207LSF/ke17ZwdGBToddI8pDm48kEHI8Rr6fdiYXgoBPndr-I8UqsxRUqqbr1mOJYKfIPR7LoDQ9mXPOjoJoqy81S2I8N_N4V1vUb5AoIIIbLZhVYxCRW4BPu10St3TBAUQYVKcHb4Mgx5VOko7O0Fa25dY_2NkKFadhjWtiVoqQ0gwguBwW4eD-34ex314_74FYepK/Agile-Methodology.jpg

敏捷模型

优点：

– 认识到计划是短暂的 – 随着变化的发生而进行调整 – 通过快速发现和修复缺陷以及及早识别期望不匹配来提高质量

缺点：

– 代码可能难以维护

• 专注于使用软件，缺乏文档效率 – 依赖口头交流

• 可能产生误解——假设单一客户代表

• 不可能采用多种观点——只能进行短期规划

  • 不再是长期愿景

Agile Models

Agile methodology includes

- Agile Scrum
- Extreme Programming (XP)
- Lean Software Development
- Kanban
- Feature-driven development (FDD)
- Rapid application development (RAD)
- Scumban
- Dynamic systems development method (DSDM)

https://images.squarespace-cdn.com/content/v1/567bb0614bf118911ff0bedb/1579721881351-CHLC89T4TS6WN0207LSF/ke17ZwdGBToddI8pDm48kEHI8Rr6fdiYXgoBPndr-I8UqsxRUqqbr1mOJYKfIPR7LoDQ9mXPOjoJoqy81S2I8N_N4V1vUb5AoIIIbLZhVYxCRW4BPu10St3TBAUQYVKcHb4Mgx5VOko7OOFa25dY_2NkKFadhjWtiVoqQ0gwguBwW4eD-34ex314_74FYepK/Agile-Methodology.jpg

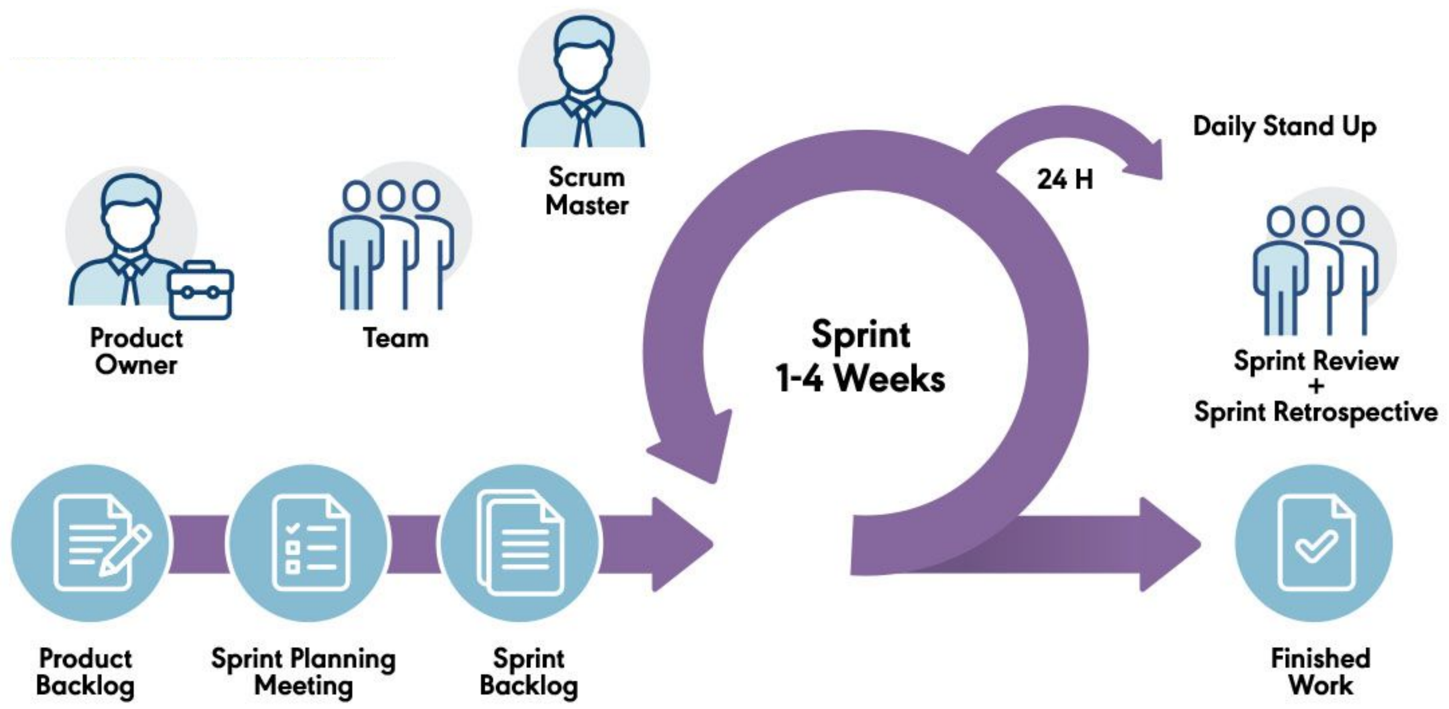
敏捷模型

敏捷方法论包括

- 敏捷 Scrum
- 极限编程（XP)
- 精益软件开发
- 看板
- 功能驱动开发（FDD）
- 快速应用程序开发 (RAD)
- 斯坎班
- 动态系统开发方法（DSDM ）

[illegible]

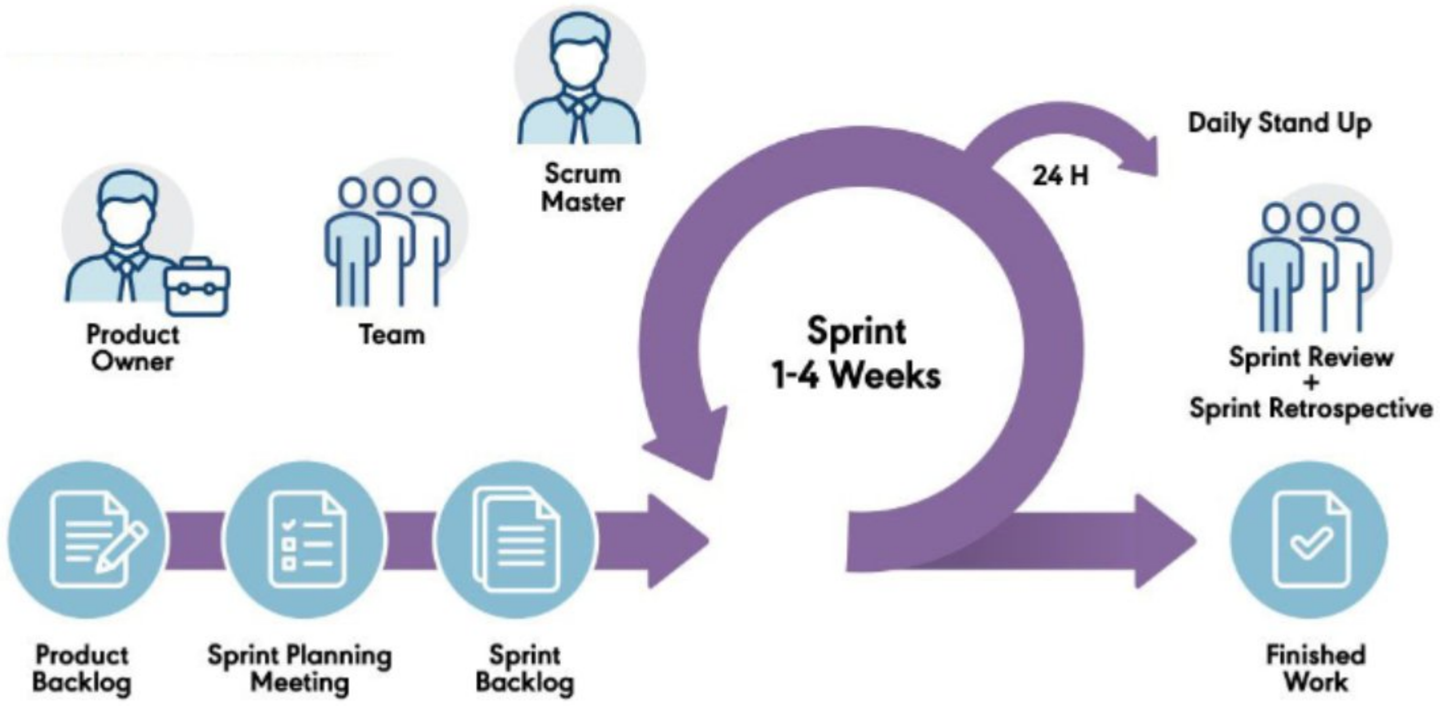
Scrum



- Short sprints with small deliverables
- Specific roles assignment
  - product owner, scrum master, and development/scrum team

https://www.pm-partners.com.au/wp-content/uploads/2021/06/blog-scrum-process.jpg

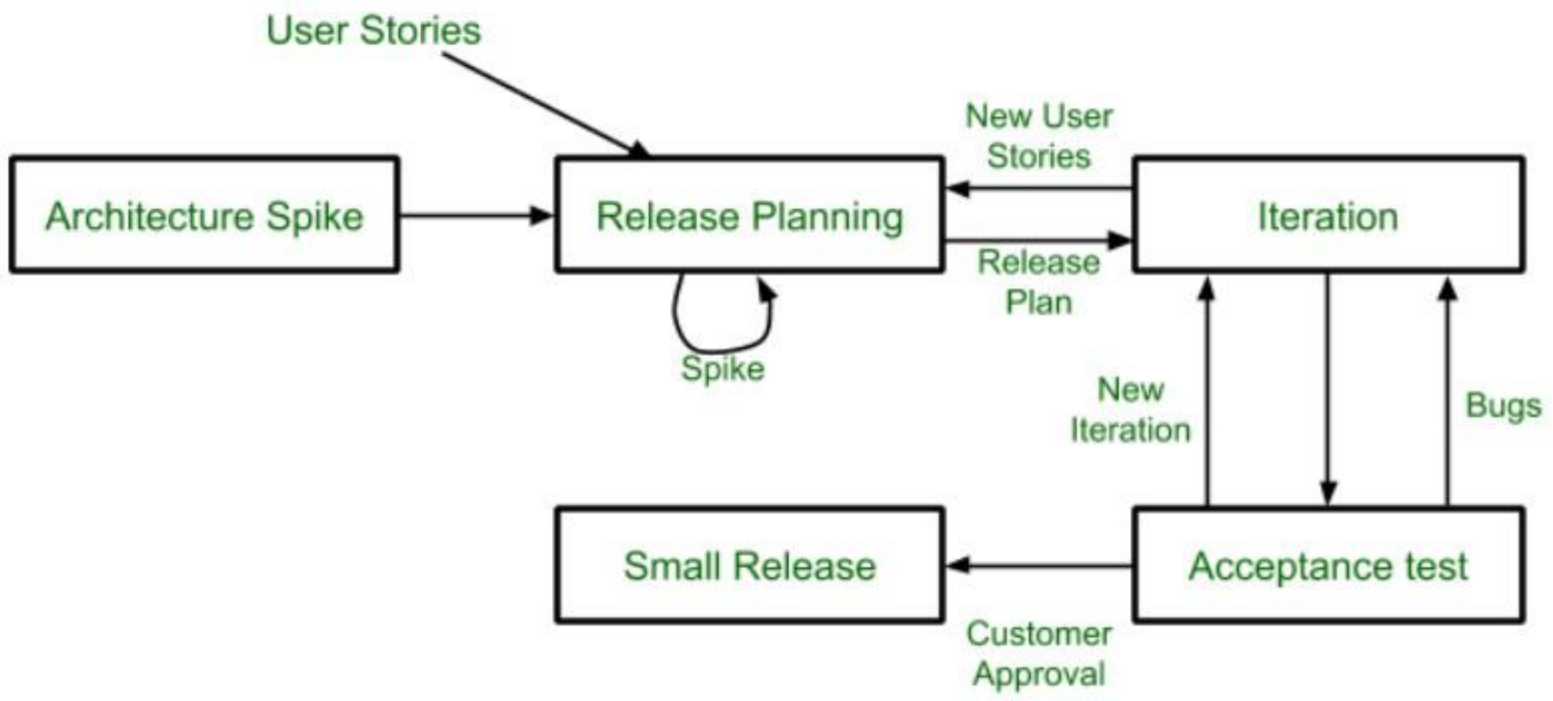
Scrum



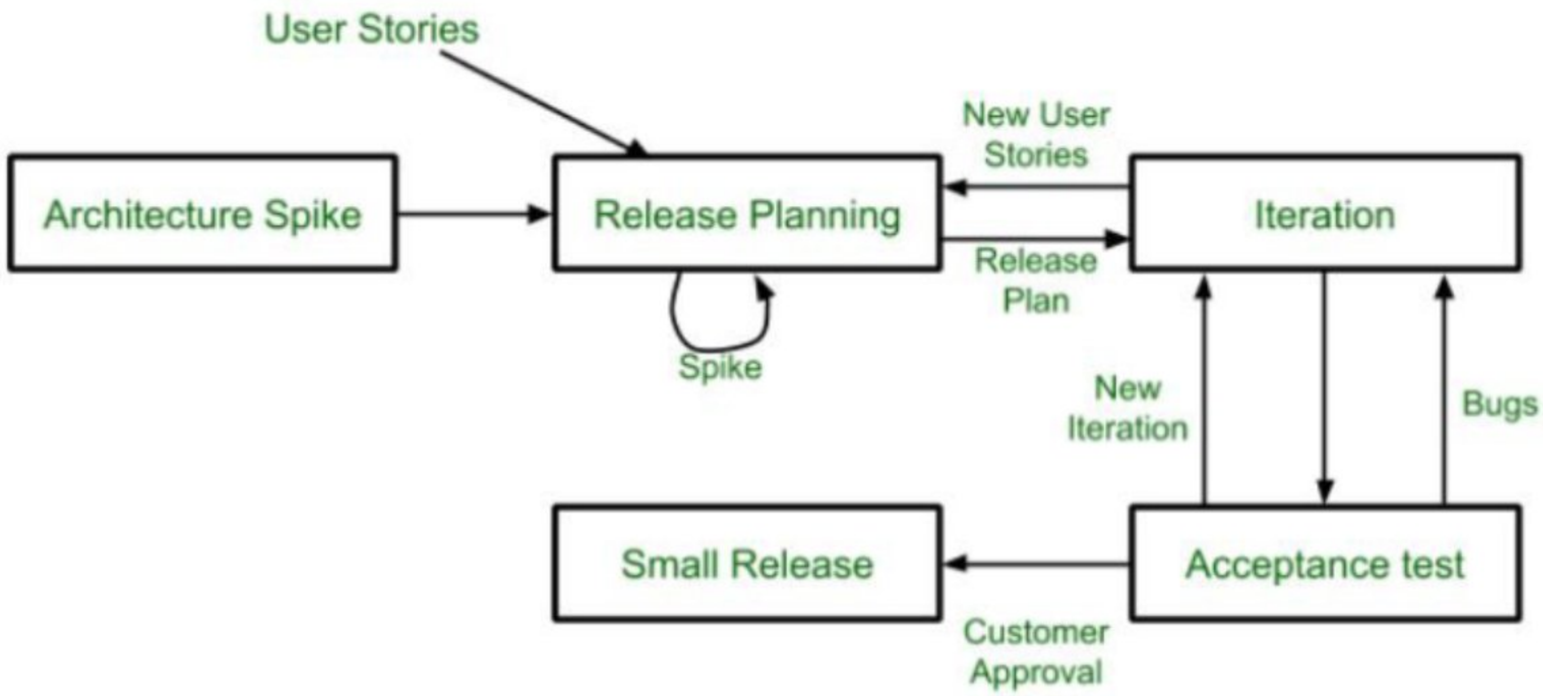
- 短冲刺，交付成果少
- 具体角色分配 ○ 产品所有者、Scrum Master 和开发/Scrum 团队

https://www.pm-partners.com.au/wp-content/uploads/2021/06/blog-scrum-process.jpg

Extreme Programming



极限编程


User Stories
Architecture Spike
Release Planning
New User Stories
Iteration
Release Plan
Spike
New Iteration
Bugs
Small Release
Acceptance test
Customer Approval

Extreme Programming

- Instead of a requirements spec, use:
  - User story cards
  - On-site customer representative
- Pair Programming
- Small releases
  - E.g. every two or three weeks
- Planning game
  - Select and estimate user story cards at the beginning of each release
- Emphasise strong engineering practices
- Pair programming
- Test-driven development (TDD)
- The program code is the design doc

极限编程

- 使用： ○ 用户故事卡 ○ 现场客户代表，而不是需求规范

- 结对编程
- 小版本发布 ○ 例如每两到三周

- 规划游戏 ○ 在每个游戏开始时选择并评估用户故事卡

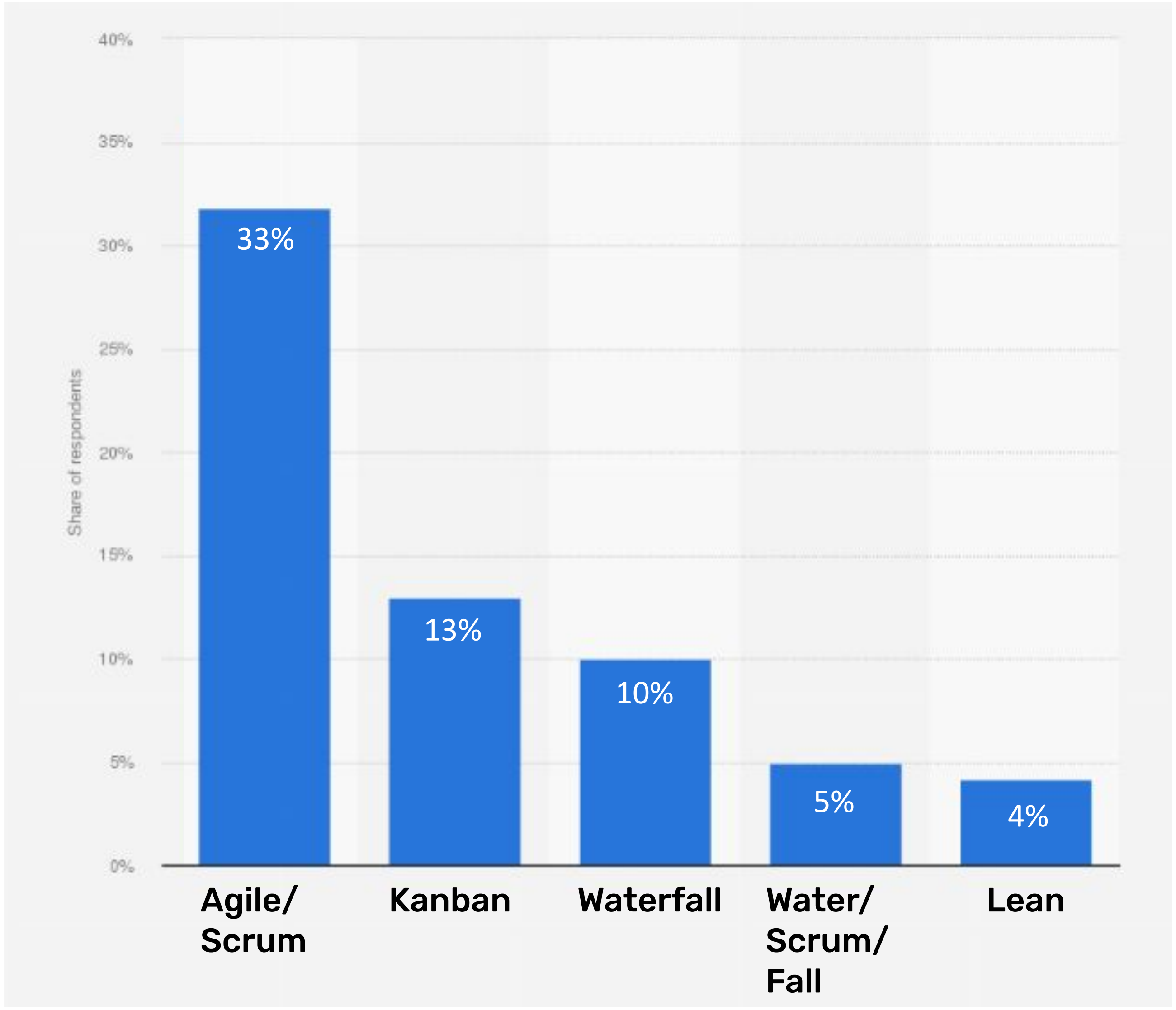
  发布
- 强调强有力的工程实践
- 结对编程
- 测试驱动开发（TDD）
- 程序代码就是设计文档

Where is RE in these lifecycles?

RE 在哪里?

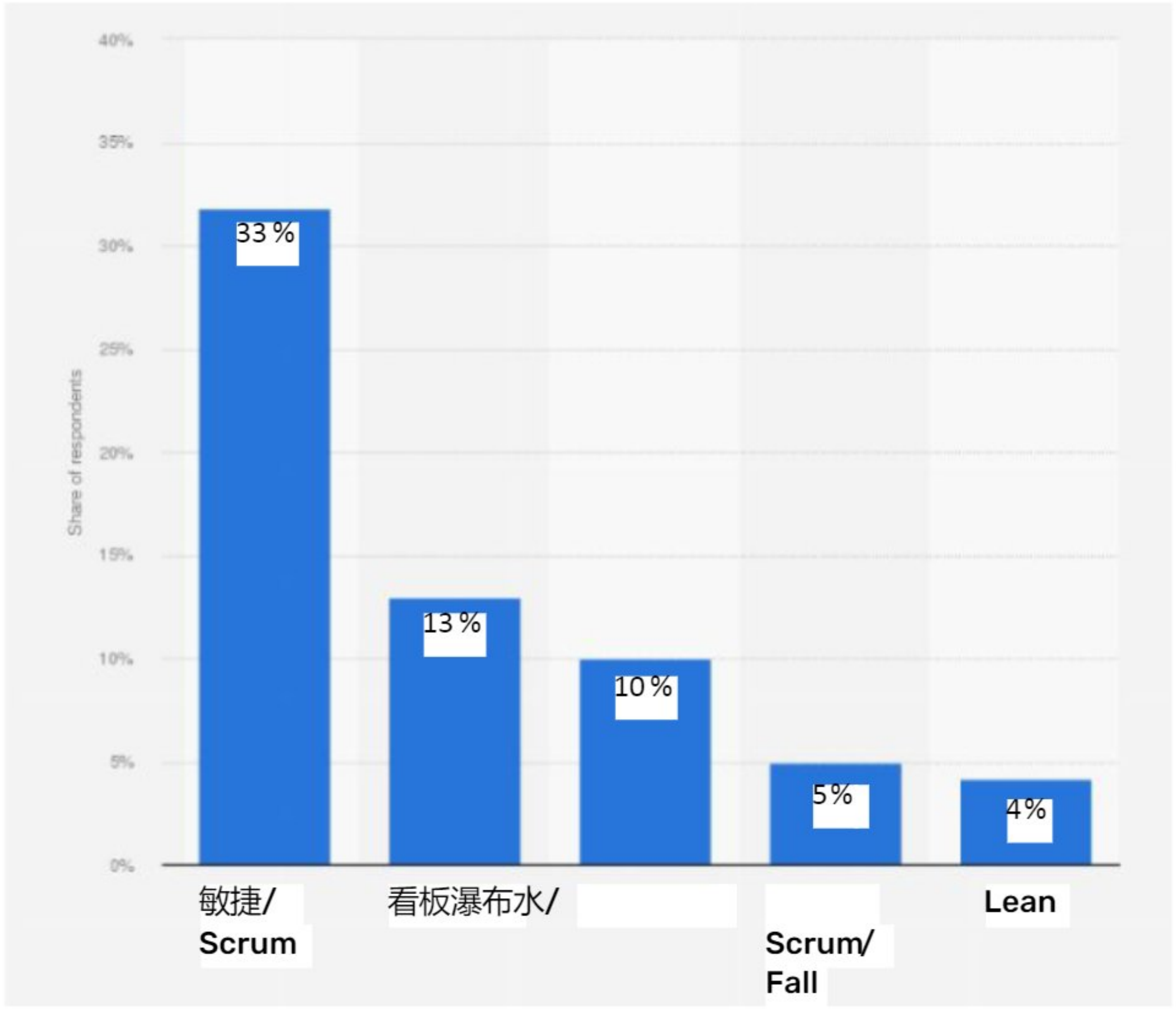
生命周期?

Usage of SDLCs



https://www.statista.com/statistics/1233917/software-development-methodologies-practiced/

SDLC 的使用



https://www.statista.com/statistics/1233917/sofftware-development-methodologies-practiced/

Why learn RE if it is not much relevant in Agile SDLCs ?

- Not all the systems developed by Agile/Scrum
- Documenting changes in the system is also a kind of requirements management ➜ new features defined context for the future features
- **RE is not only about documentation**

如果 **RE** 与敏捷 **SDLC** 关系不大，为什么要学习 **RE**?

· 并非所有系统都是由敏捷/Scrum 开发的 · 记录系统中的变更也是一种需求管理 ➜ 新功能为未来功能定义了上下文 · RE 不仅仅是文档

Scrum example

- Product Owner is a requirements engineering expert
  - elicits requirements (a.k.a. user stories)
  - manages the scope (system boundary)
  - documents (for effective team communication)
  - manages 'specification' (a.k.a. product backlog) by communicating them, prioritising, defining acceptance criteria
- Product Backlog is a dynamic set of requirements
  - Backlog refinement = requirements validation
- Scrum team members
  - use RE modelling techniques to communicate the RE implementation/understanding

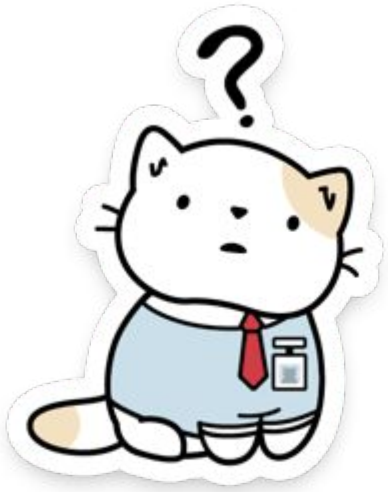
Scrum 示例

- 产品负责人是需求工程专家 ○ 引出需求（又名用户故事）

  ○ 管理范围（系统边界） ○ 文件（用于有效的团队沟通） ○ 通过沟通、确定优先级、定义验收标准来管理“规范”（又名产品待办事项）

- 产品待办事项列表是一组动态的需求
  - ○ 待办事项细化 = 需求验证
- Scrum 团队成员
  - ○ 使用 RE 建模技术来传达 RE 实施/理解

Any questions



任何问题