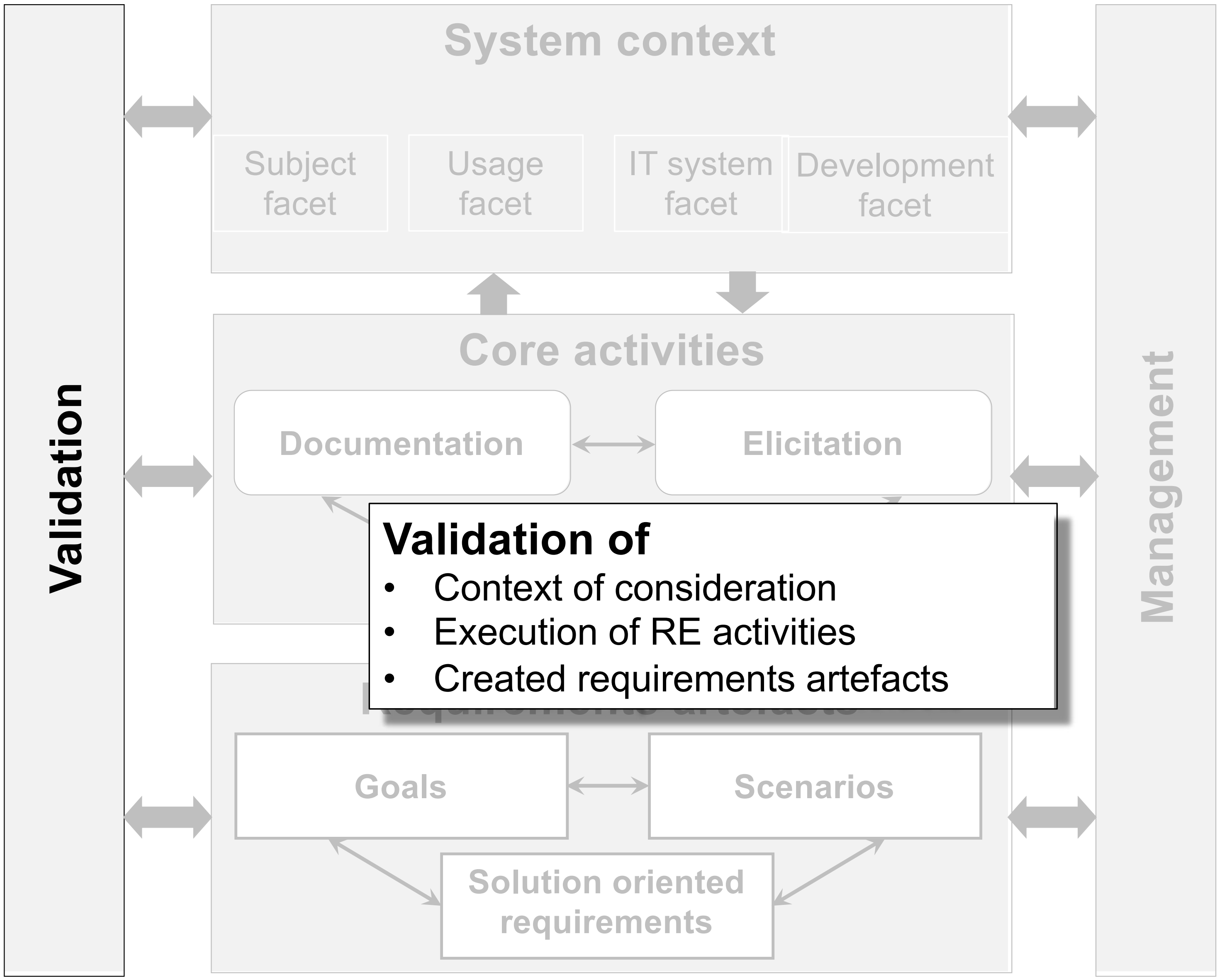


Validation

System context

Subject facet

Usage facet

IT system facet

Development facet

Core activities

Documentation

Elicitation

Validation of

- Context of consideration
- Execution of RE activities
- Created requirements artefacts

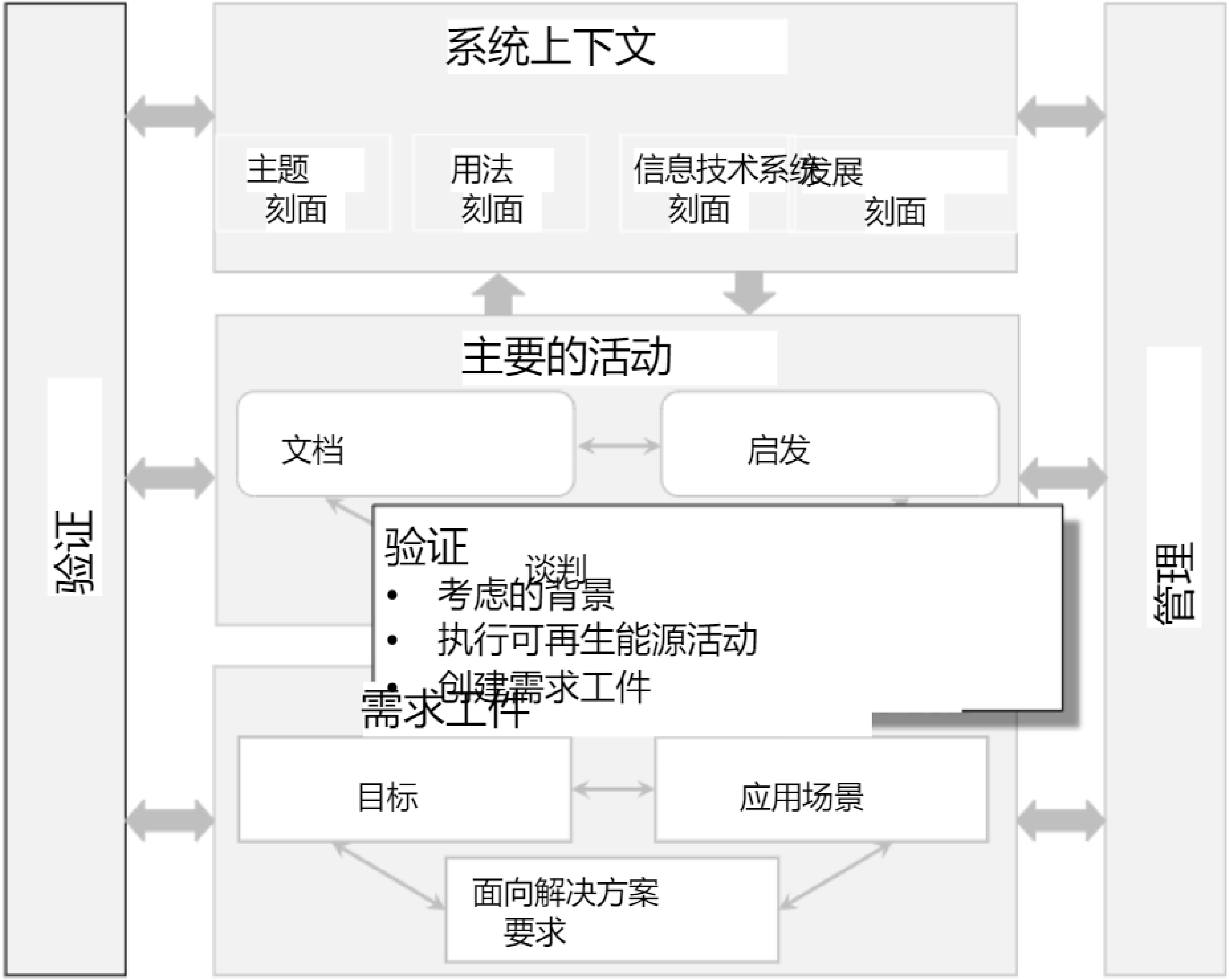
Requirements artefacts

Goals

Scenarios

Solution oriented requirements

Management



验证

系统上下文

主题 刻面

用法 刻面

信息技术系统 刻面

发展 刻面

主要的活动

文档

启发

谈判

验证

- 考虑的背景
- 执行可再生能源活动
- 创建需求工件

需求工件

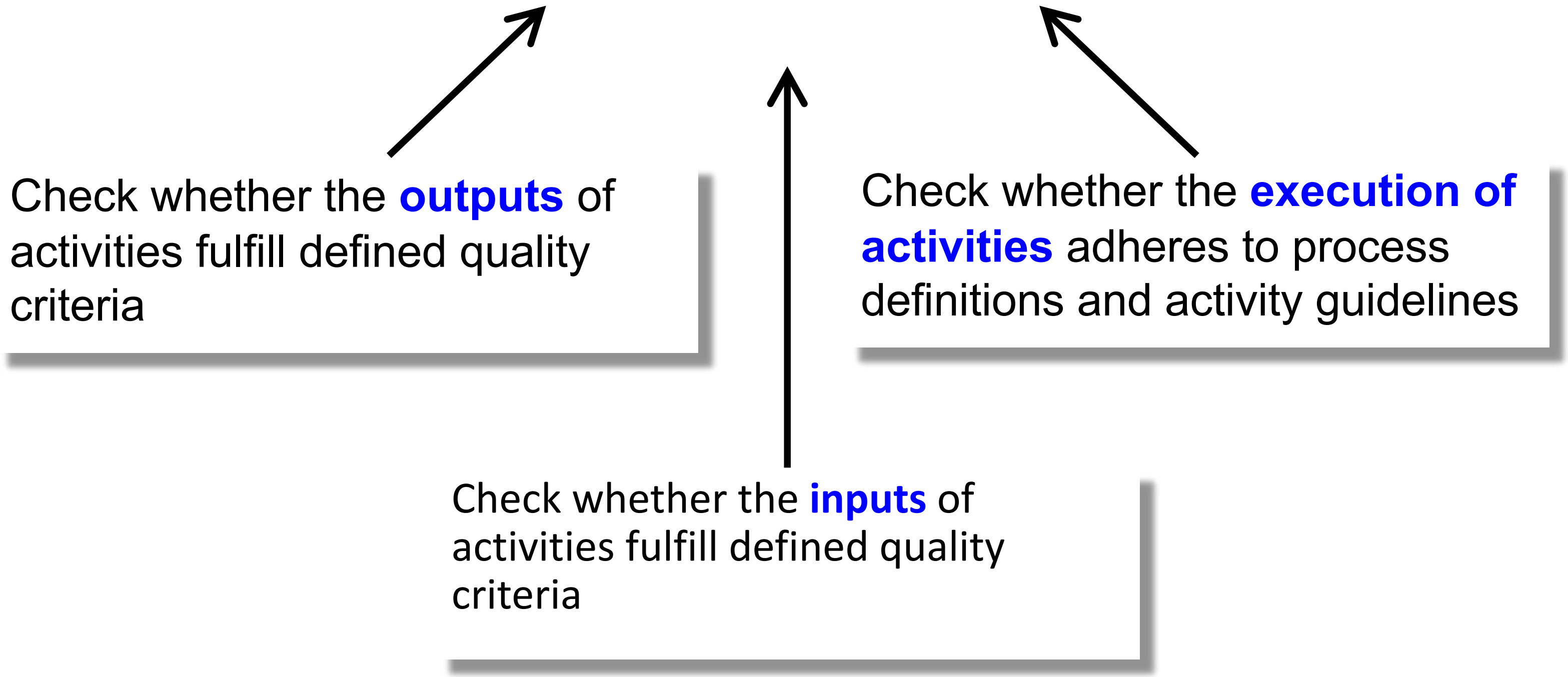
目标

应用场景

面向解决方案要求

管理

Validation Goals

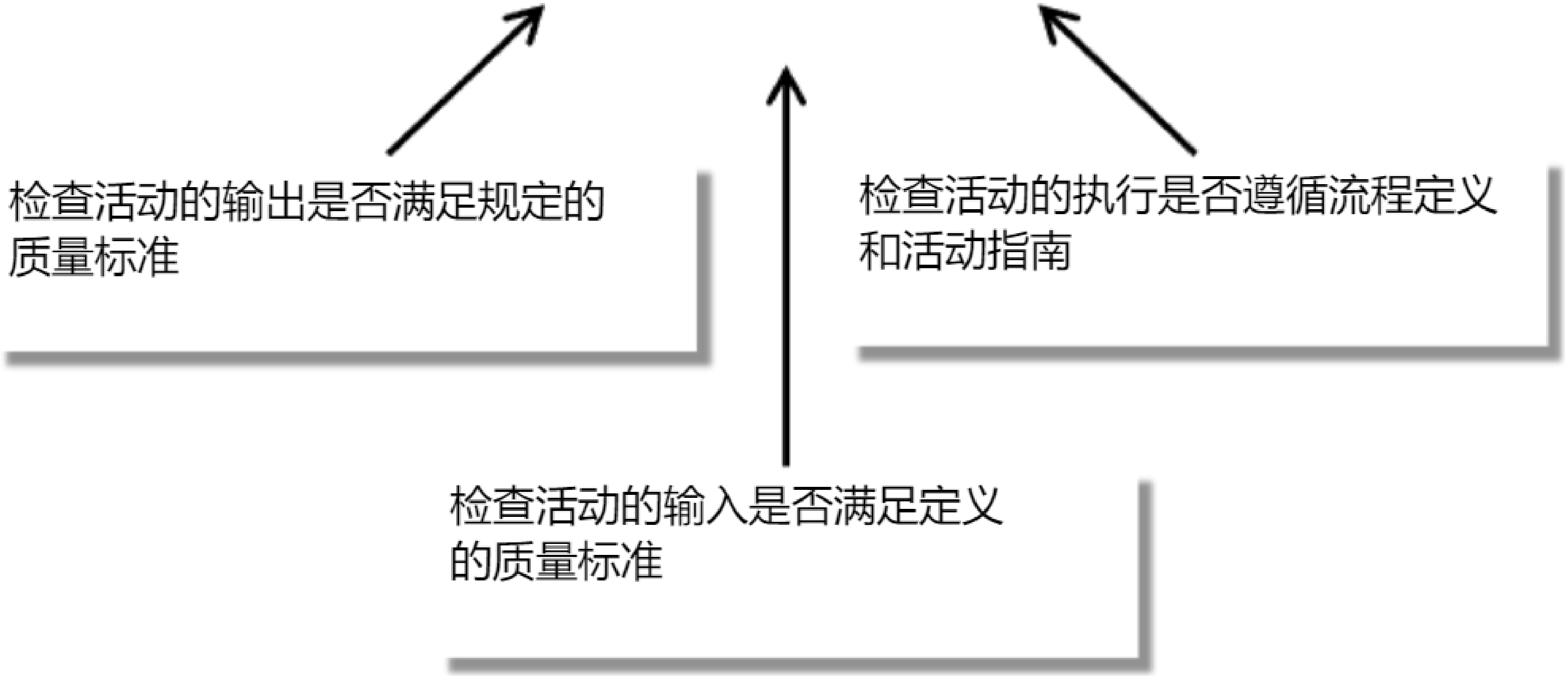


Check whether the **outputs** of activities fulfill defined quality criteria

Check whether the **execution of activities** adheres to process definitions and activity guidelines

Check whether the **inputs** of activities fulfill defined quality criteria

验证目标

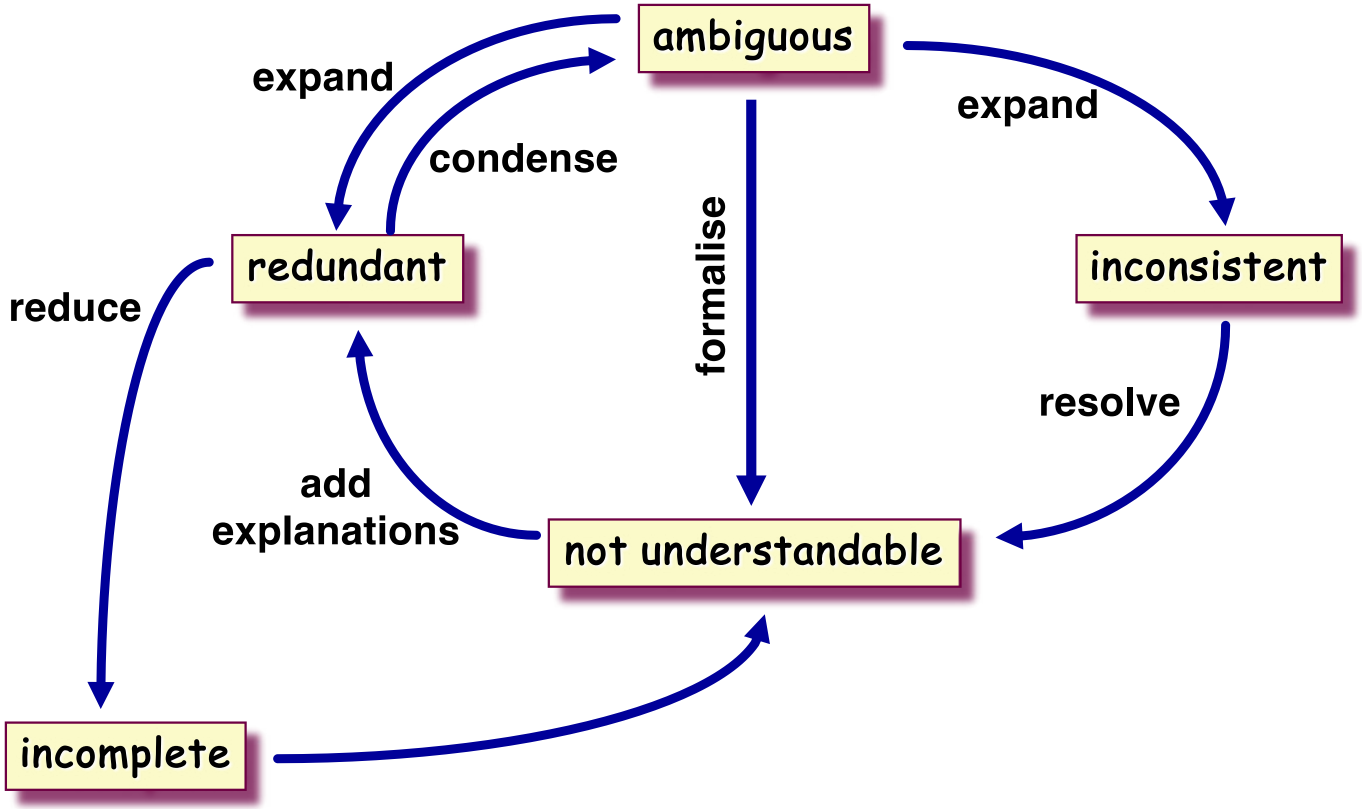


检查活动的输出是否满足规定的质量标准

检查活动的输入是否满足定义的质量标准

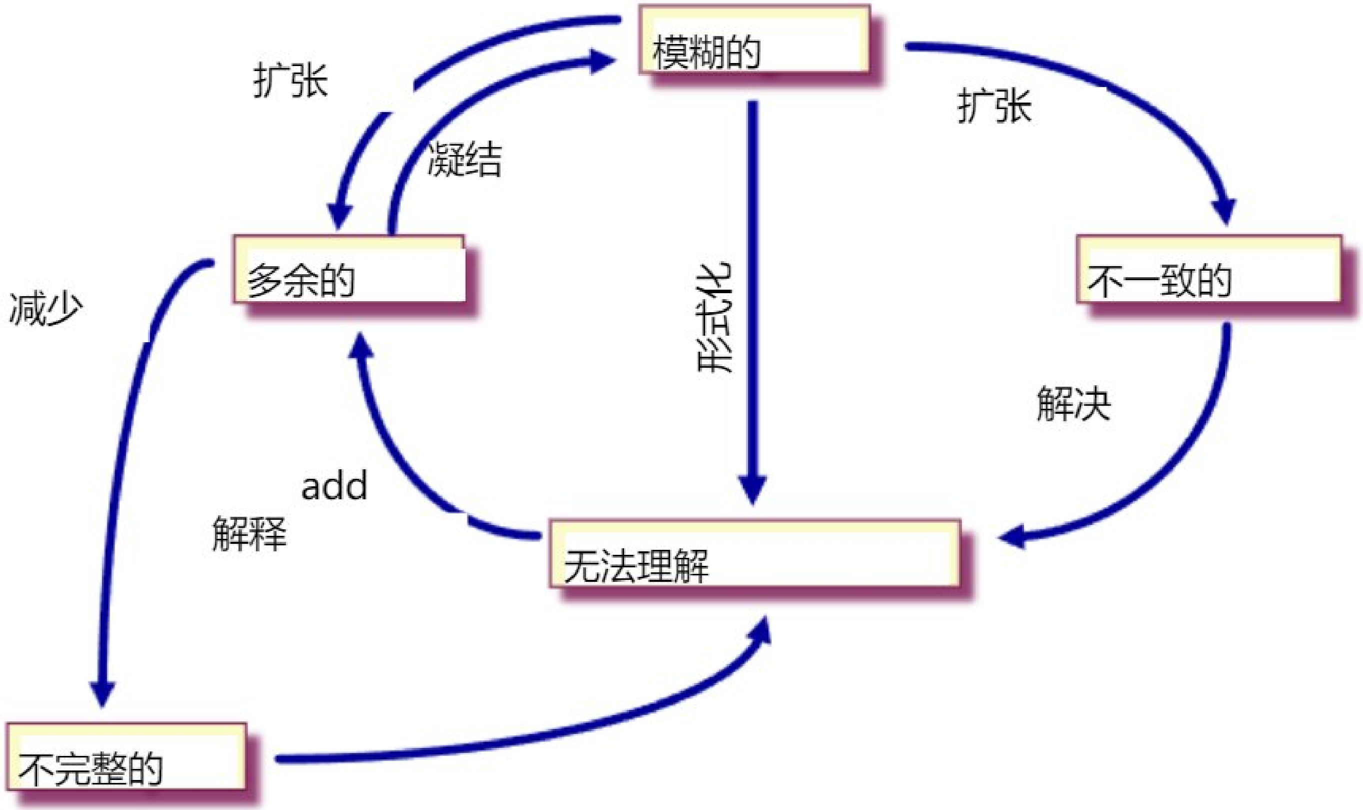
检查活动的执行是否遵循流程定义和活动指南

THERE IS NO SUCH THING AS A PERFECT REQUIREMENTS SPECIFICATION !



有 NO

A完美的 要求规范！



Validation Techniques

- What are goals of verification and validation?
- Checking quality
- **Model analysis**
- Prototyping

验证技术

- 验证和确认的目标是什么?
- 检查质量
- 模型分析
- 原型制作

Model Checking

- **Has revolutionized formal verification:**
  - ➢ emphasis on partial verification of partial models
    - E.g. as a debugging tool for state machine models
  - ➢ fully automated
- **What it does:**
  - ➢ Mathematically – computes the “satisfies” relation:
    - Given a temporal logic theory, checks whether a given finite state machine is a model for that theory.
  - ➢ Engineering view – checks whether properties hold:
    - Given a model (e.g. a FSM), checks whether it obeys various safety and liveness properties
- **How to apply it in RE:**
  - ➢ The model is an (operational) Specification
    - Check whether particular requirements hold of the spec
  - ➢ The model is (an abstracted portion of) the Requirements
    - Carry out basic validity tests as the model is developed
  - ➢ The model is a conjunction of the Requirements and the Domain
    - Formalise assumptions and test whether the model respects them

模型检验

- 彻底改变了形式验证：

Ø 强调部分模型的部分验证
例如。• 作为状态机模型的调试工具 Ø 完全自动化

- 它能做什么：

Ø 数学上——计算“满足”关系：

  - 给定时态逻辑理论，检查给定的有限状态机是否是该理论的模型。

Ø 工程视图——检查属性是否成立：

  - 给定一个模型（例如 FSM），检查它是否遵守各种安全性和活性属性

- 如何在 RE 中应用它：

Ø 该模型是一个（操作）规范
检查特定需求是否符合规范 Ø 模型是需求（的抽象部分）

在开发模型时进行基本的有效性测试 Ø 该模型是需求和领域的结合

  - 形式化假设并测试模型是否尊重它们

Model Analysis

- **Verification**
  - “Is the model well-formed?”
  - Are the parts of the model consistent with one another?

- **Validation**
  - Animation of the model on small examples
  - Formal challenges:
    - “if the model is correct then the following property should hold...”
  - ‘What if’ questions:
    - reasoning about the consequences of particular requirements;
    - reasoning about the effect of possible changes
    - “will the system ever do the following...”
  - State exploration
    - E.g. use a model checking to find traces that satisfy some property

模型分析

- 确认
  - Ø “模型结构是否良好？”
  - Ø 模型各部分是否一致？

- 验证
  - Ø 小例子上的模型动画Ø 形式挑战：
    - “如果模型正确，那么以下属性应该成立……”
  - Ø “如果”问题：
    - 对特定要求的后果进行推理；
    - 推理可能的变化的影响
    - “系统会执行以下操作吗……”
  - Ø 国家探索
    - 例如。使用模型检查来查找满足某些属性的痕迹

Requirements Specification

1 Introduction

- Purpose
- Scope
- Definitions, acronyms, abbreviations
- Reference documents
- Overview

2 Overall Description

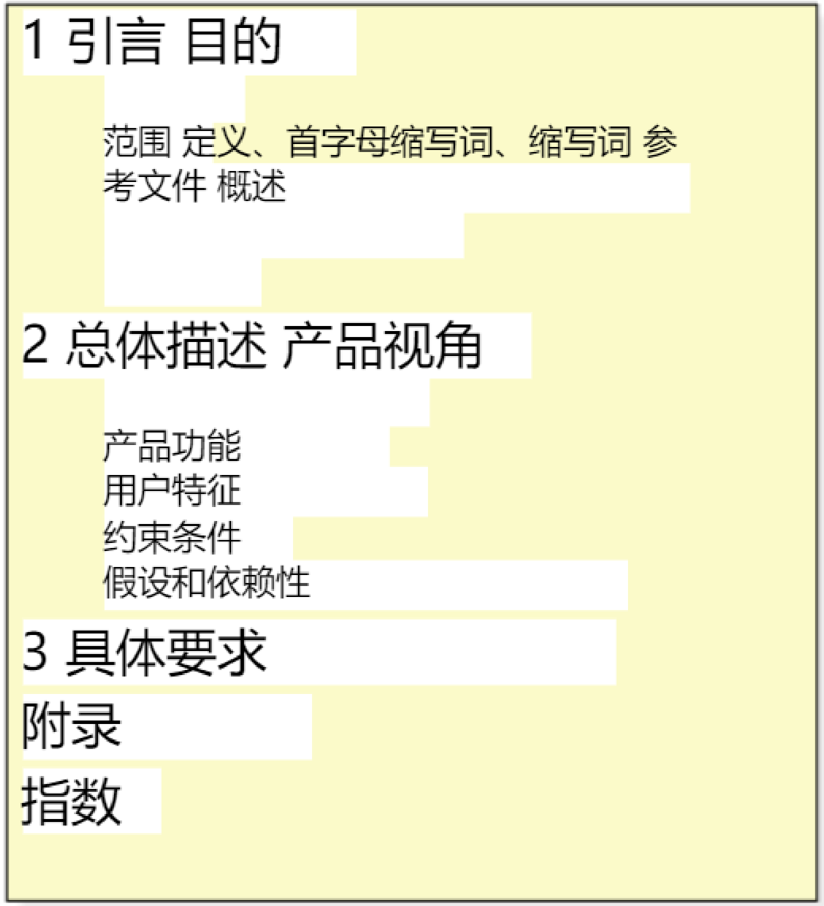
- Product perspective
- Product functions
- User characteristics
- Constraints
- Assumptions and Dependencies

3 Specific Requirements

Appendices

Index

要求规范



1 引言 目的

范围 定义、首字母缩写词、缩写词 参考文件 概述

2 总体描述 产品视角

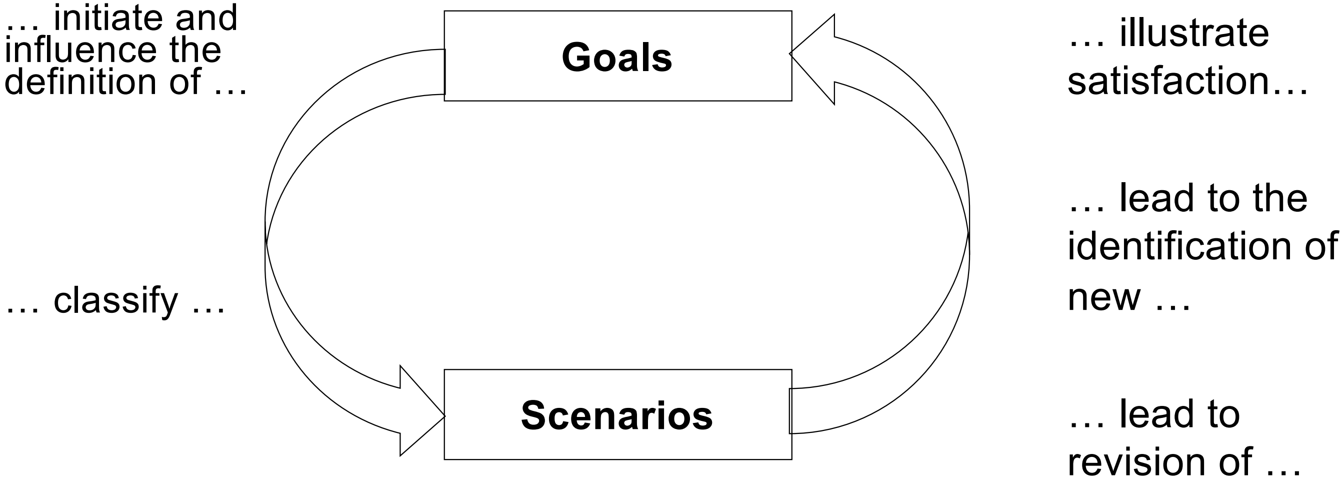
产品功能
用户特征
约束条件
假设和依赖性

3 具体要求

附录

指数

Goal-Scenario coupling



Goals

Scenarios

… initiate and influence the definition of …

… classify …

… illustrate satisfaction…

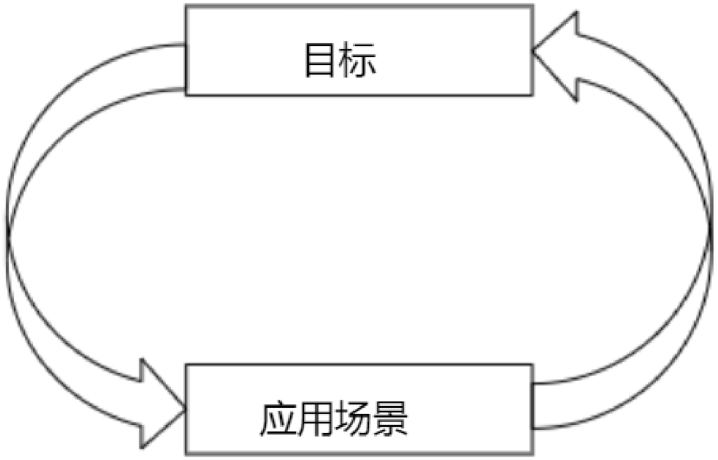
… lead to the identification of new …

… lead to revision of …

目标-场景耦合

...发起并影响
......的定义

... 分类 ...



目标

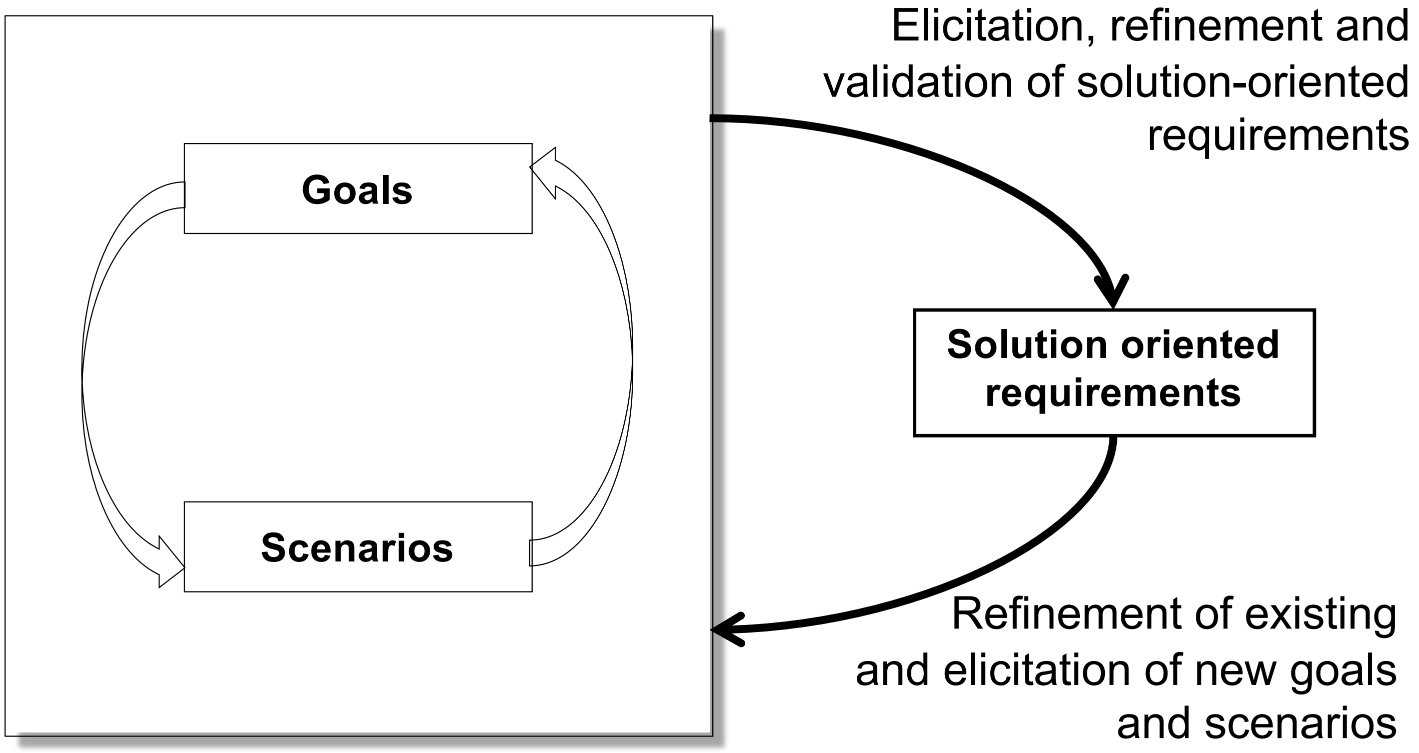
应用场景

... 阐明
满意...

......导致
识别
新的 ...

... 导致
修订...

Key Relationships



Goals

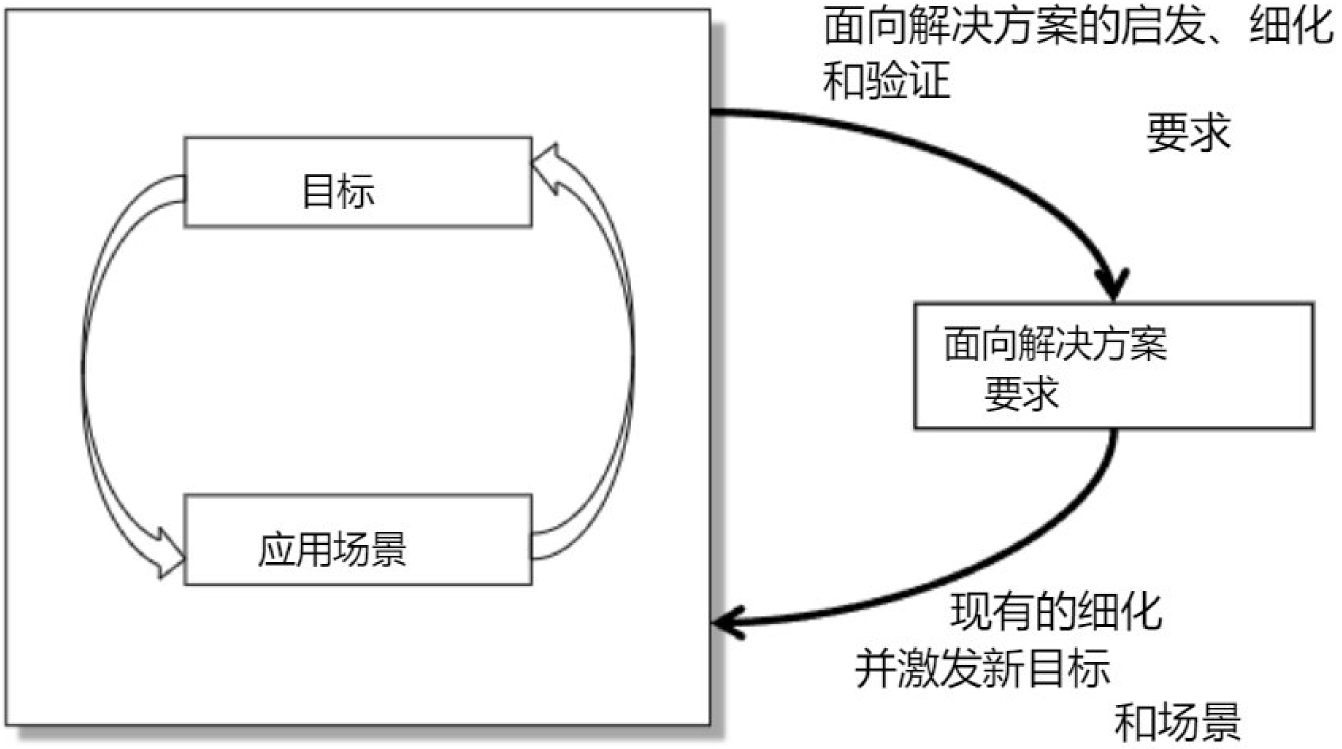
Scenarios

Elicitation, refinement and validation of solution-oriented requirements

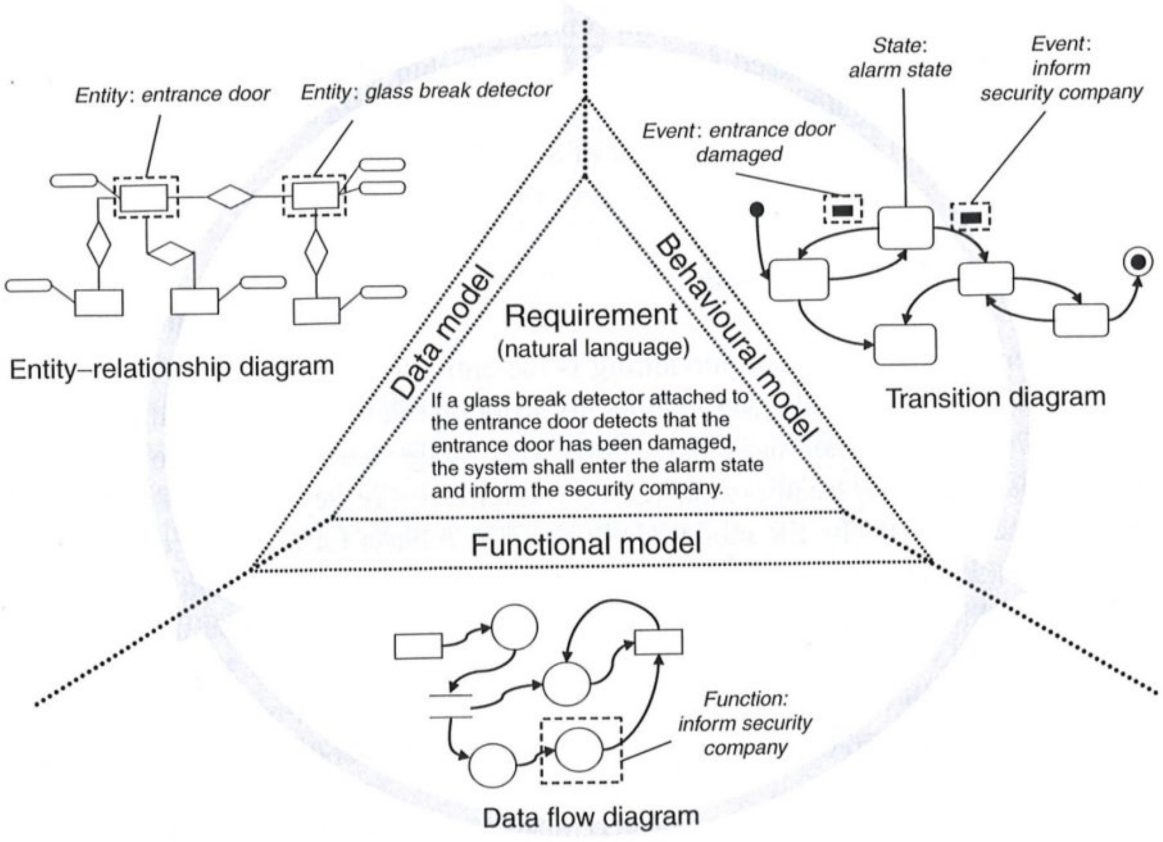
Solution oriented requirements

Refinement of existing and elicitation of new goals and scenarios

关键关系

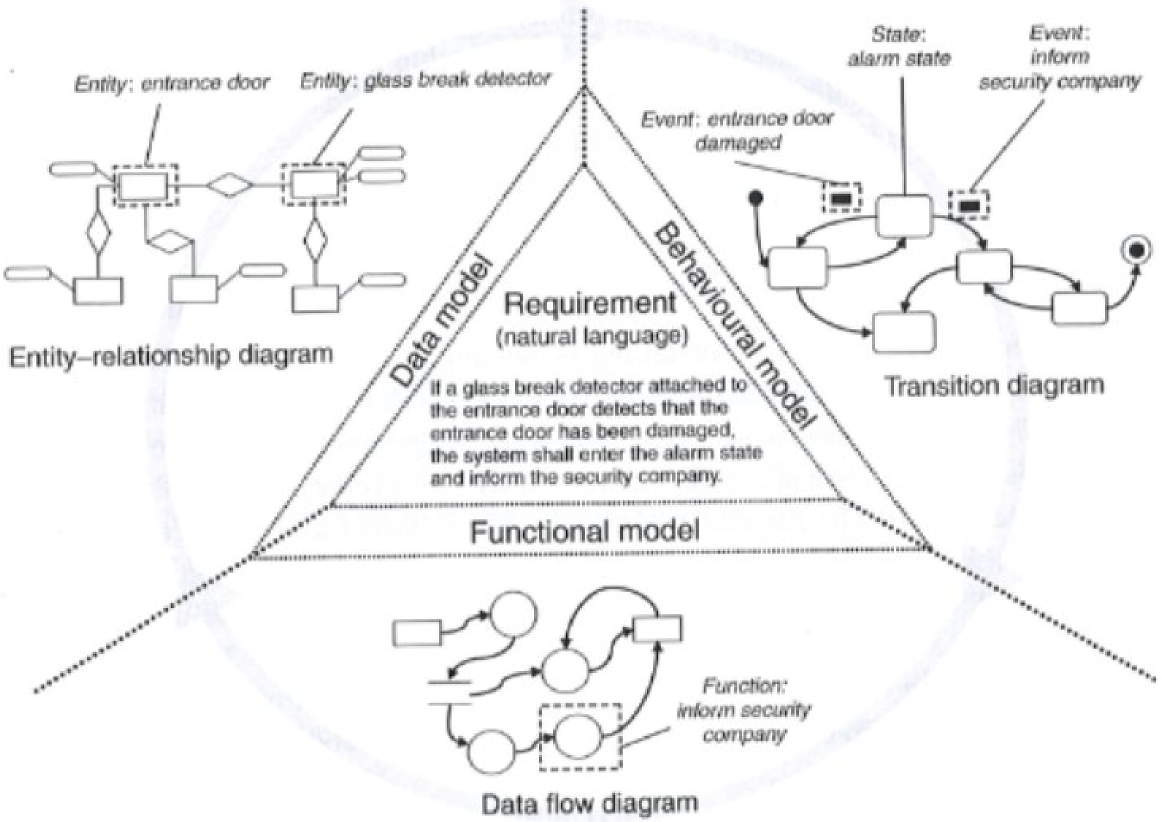


Documenting Solution-Oriented Requirements


Entity: entrance door
Entity: glass break detector
Entity–relationship diagram
State: alarm state
Event: inform security company
Event: entrance door damaged
Transition diagram
Data model
Behavioural model
Functional model
Requirement
(natural language)
Function: inform security company
Data flow diagram

If a glass break detector attached to the entrance door detects that the entrance door has been damaged, the system shall enter the alarm state and inform the security company.

记录面向解决方案的需求


Entity: entrance door
Entity: glass break detector
Entity–relationship diagram
State: alarm state
Event: inform security company
Event: entrance door damaged
Transition diagram
Data model
Behavioural model
Functional model
Requirement
(natural language)
Function: inform security company
Data flow diagram

If a glass break detector attached to the entrance door detects that the entrance door has been damaged, the system shall enter the alarm state and inform the security company.

We’ve looked at the following non-UML diagrams

- **Goal Models**
  - Capture strategic goals of stakeholders
  - Good for exploring ‘how’ and ‘why’ questions with stakeholders
  - Good for analysing trade-offs, especially over design choices

- **Strategic Dependency Models (*i**)**
  - Capture relationships between actors in an organisational setting
  - Helps to relate goal models to organisational setting
  - Good for understanding how the organisation will be changed

我们查看了以下非 UML 图

Ø目标模型

- 捕捉利益相关者的战略目标
- 适合与利益相关者探讨“如何”和“为什么”问题
- 适合分析权衡，尤其是设计选择

Ø战略依赖模型（i*)

- 捕获组织环境中参与者之间的关系
- 有助于将目标模型与组织设置联系起来
- 有助于了解组织将如何变革

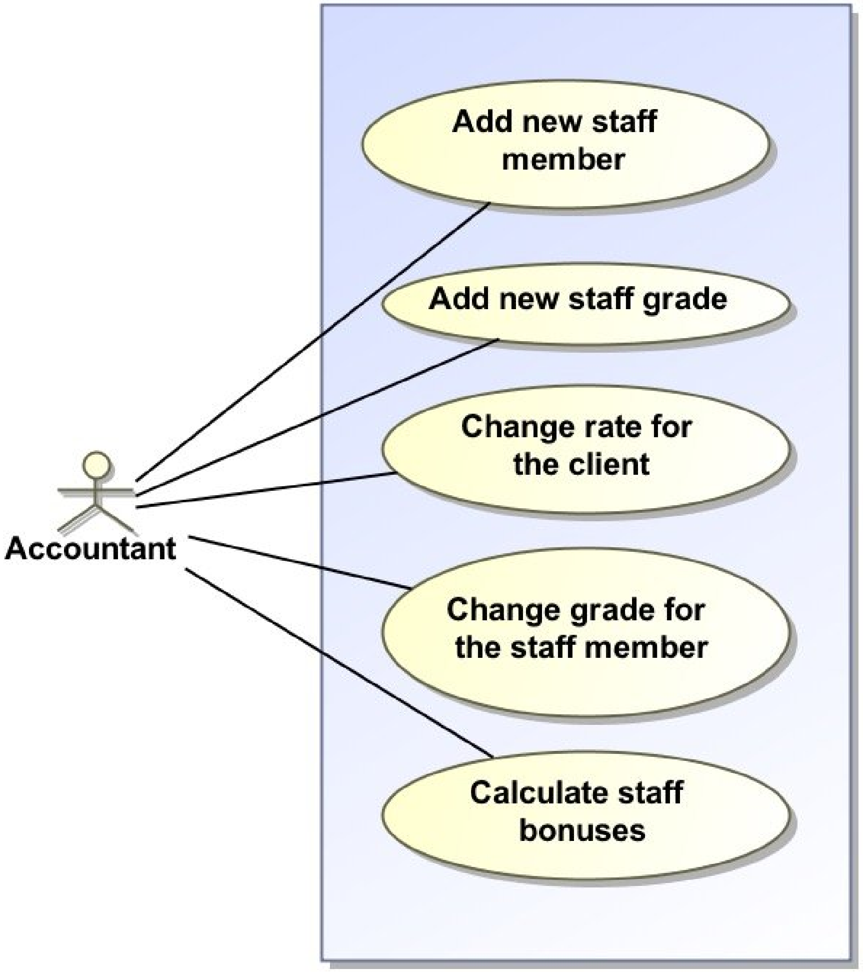
Use cases

➢**Use Cases**

- capture the view of the system from the view of its users
- good starting point for specification of functionality
- good visual overview of the main functional requirements

➢**Cross-checks:**

- Does each use case have a user?
  - Does each user have at least one use case?
- Is each use case documented?
  - Using sequence diagrams or use case template



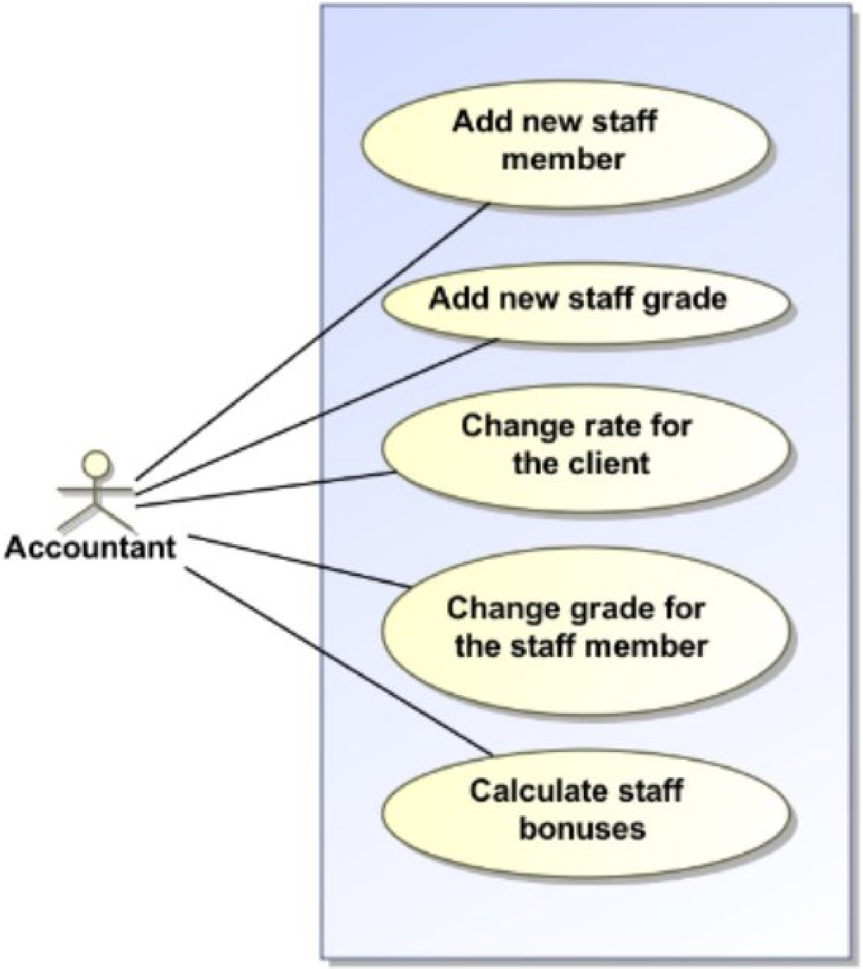
用例

Ø使用案例

- 从用户的角度捕捉系统的视图
- 功能规范的良好起点
- 主要功能需求的良好视觉概述

Ø 交叉检查:

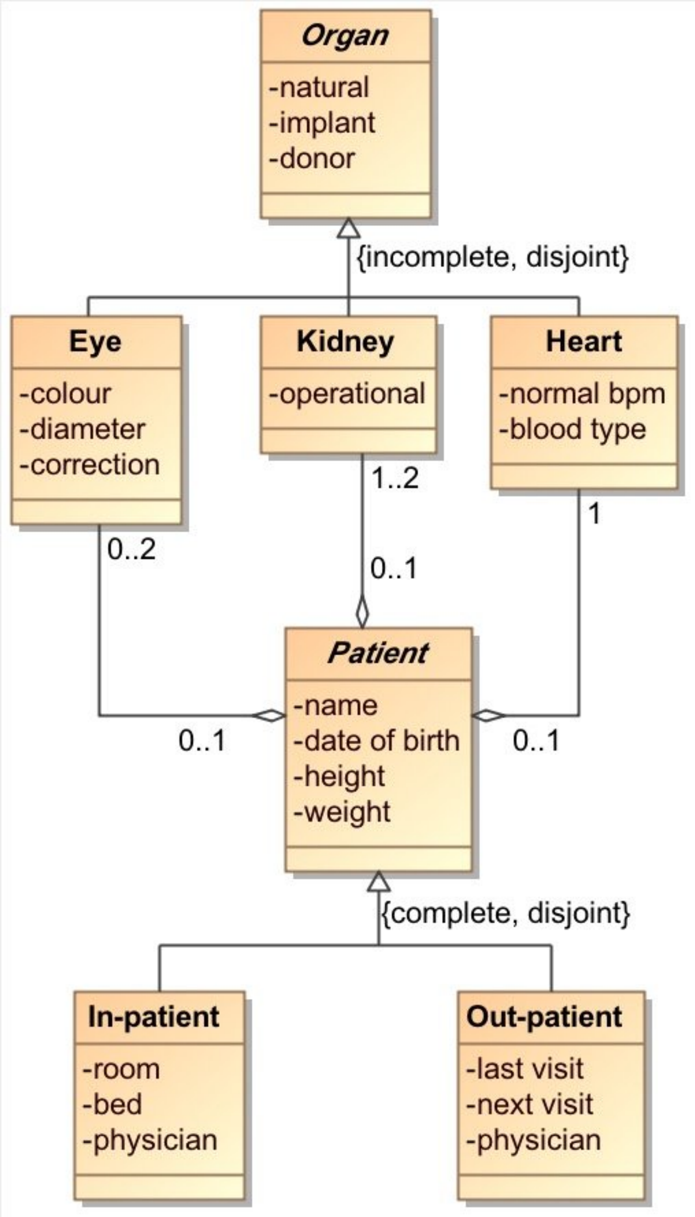
- 每个用例都有一个用户吗?
  - 每个用户是否至少拥有一个用例?
- 每个用例都有记录吗?
  - 使用序列图或用例模板



Class diagrams

- **Class Diagrams**
  - capture the structure of the information used by the system
  - good for analysing the relationships between data items used by the system
  - good for helping you identify a modular structure for the system

- **Cross checks**
  - Does the class diagram capture all the classes mentioned in
    - - other diagrams?
    - - specification glossary?
  - Does every class have methods to get/set its attributes?



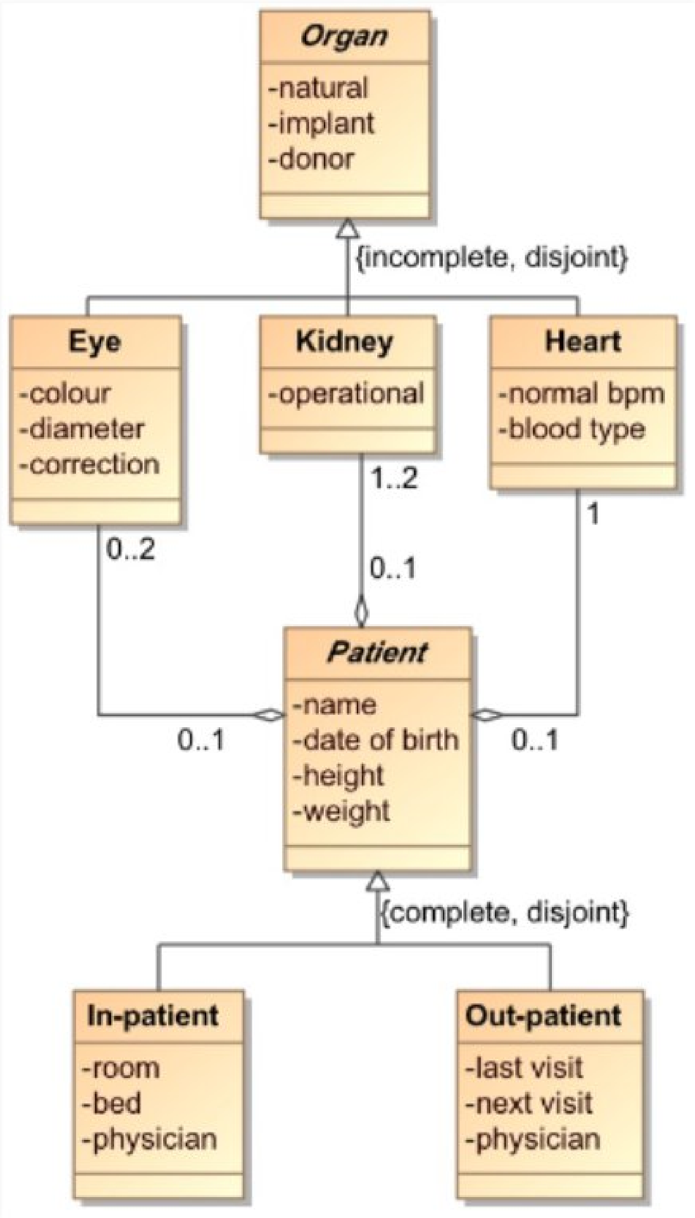
类图

Ø类图

- 捕获系统使用的信息的结构
- 有利于分析系统使用的数据项之间的关系
- 有助于帮助您确定系统的模块化结构

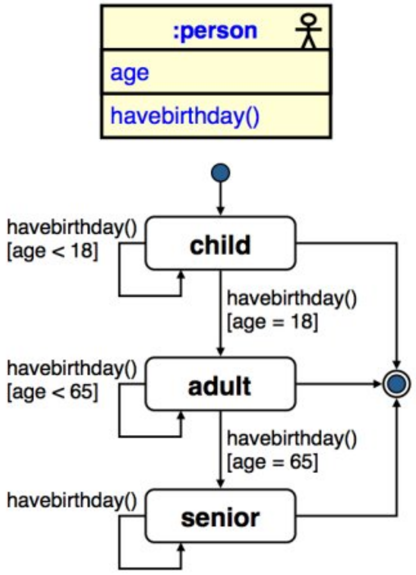
Ø交叉检查

- 类图是否捕获了中提到的所有类
  - - 其他图表?
  - - 规格术语表?
- 每个类都有获取/设置其属性的方法吗?



➢Statecharts

- capture all possible responses of an object to all uses cases in which it is involved
- good for modeling the dynamic behavior of a class of objects
- good for analyzing event ordering, reachability, deadlock, etc.

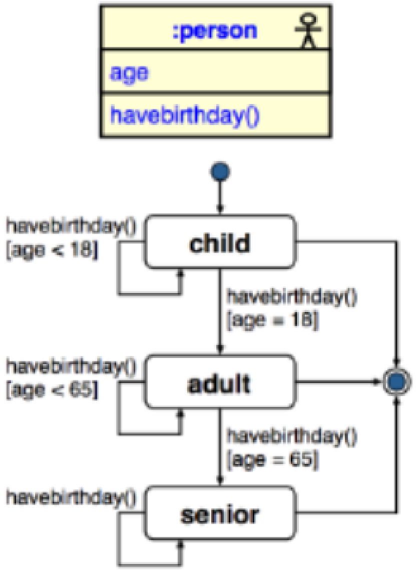


Cross-checks:

- ➢Does each statechart diagram capture (the states of) a single class?
  - ➢Is that class in the class diagram?
- ➢Does each transition have a trigger event?
  - ➢Is it clear which object initiates each event?
  - ➢Is each event listed as an operation for that object’s class in the class diagram?
- ➢Does each state represent a distinct combination of attribute values?
  - ➢Is it clear which combination of attribute values?
  - ➢Are all those attributes shown on the class diagram?
- ➢Are there method calls in the class diagram for each transition?
  - ➢…a method call that will update attribute values for the new state?
  - ➢…method calls that will test any conditions on the transition?
  - ➢…method calls that will carry out any actions on the transition?

Ø状态图

- 捕获对象对其涉及的所有用例的所有可能响应
- 适合对一类对象的动态行为进行建模
- 适合分析事件排序、可达性、死锁等。



交叉检查:

Ø每个状态图是否捕获单个类（的状态）？

Ø 类图中有这个类吗?

Ø每个transition是否都有触发事件?

Ø 是否清楚哪个对象发起每个事件?

Ø 每个事件是否在类图中被列为该对象的类的操作?

Ø每个状态是否代表不同的属性值组合?

Ø 是否清楚属性值是哪种组合?

Ø 所有这些属性都显示在类图上吗?

Ø 类图中是否有每次转换的方法调用?

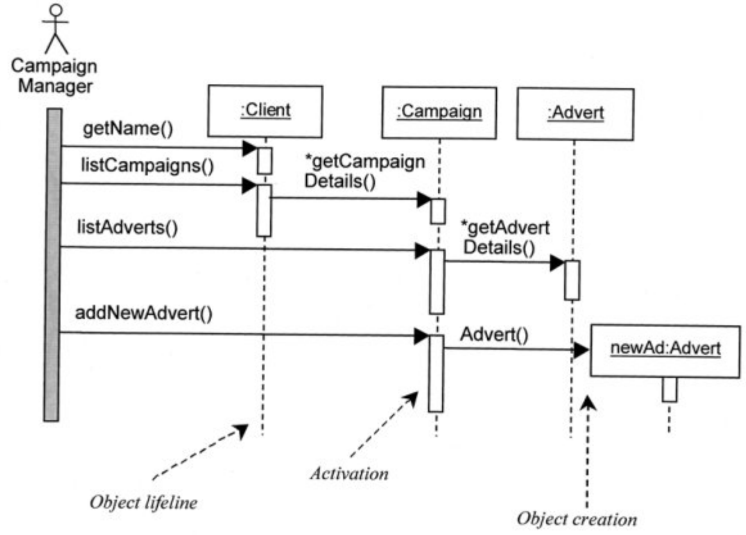
Ø ...将更新新状态的属性值的方法调用?

Ø ...将测试转换任何条件的方法调用?

Ø ...将在转换中执行任何操作的方法调用?

➢Sequence Diagrams

- capture an individual scenario (one path through a use case)
- good for modelling dialog structure for a user interface or a business process
- good for identifying which objects (classes) participate in each use case
- helps you check that you identified all the necessary classes and operations

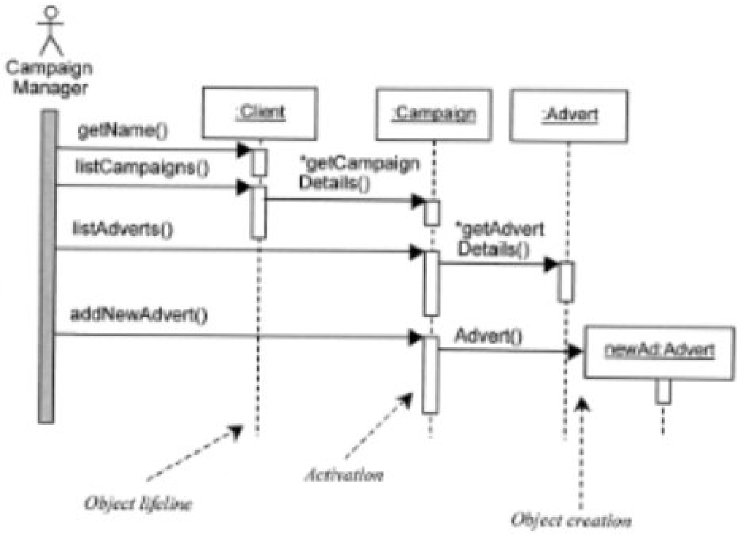


➢Cross-checks:

- Is each class in the class diagram?
- Can each message be sent?
  - Is there an association connecting sender and receiver classes on the class diagram?
  - Is there a method call in the sending class for each sent message?
  - Is there a method call in the receiving class for each received message?

Ø时序图

- 捕获单个场景（通过用例的一条路径）
- 适合为用户界面或业务流程建模对话框结构
- 有助于识别哪些对象（类）参与每个用例
- 帮助您检查是否识别了所有必要的类和操作



Ø 交叉检查：

- 每个类都在类图中吗?
- 每条消息都可以发送吗?
  - 类图上是否存在连接发送者类和接收者类的关联?
  - 发送类中是否有针对每条已发送消息的方法调用?
  - 接收类中是否有针对每个接收到的消息的方法调用?

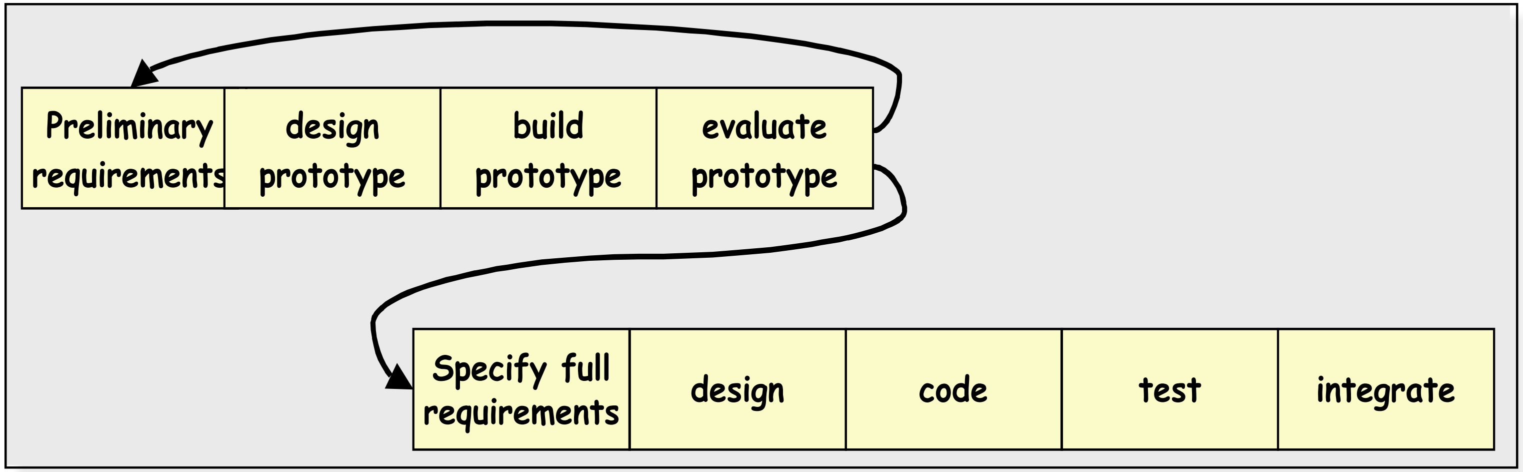
Validation Techniques

- What are goals of verification and validation?
- Checking quality
- Model analysis
- **Prototyping**

验证技术

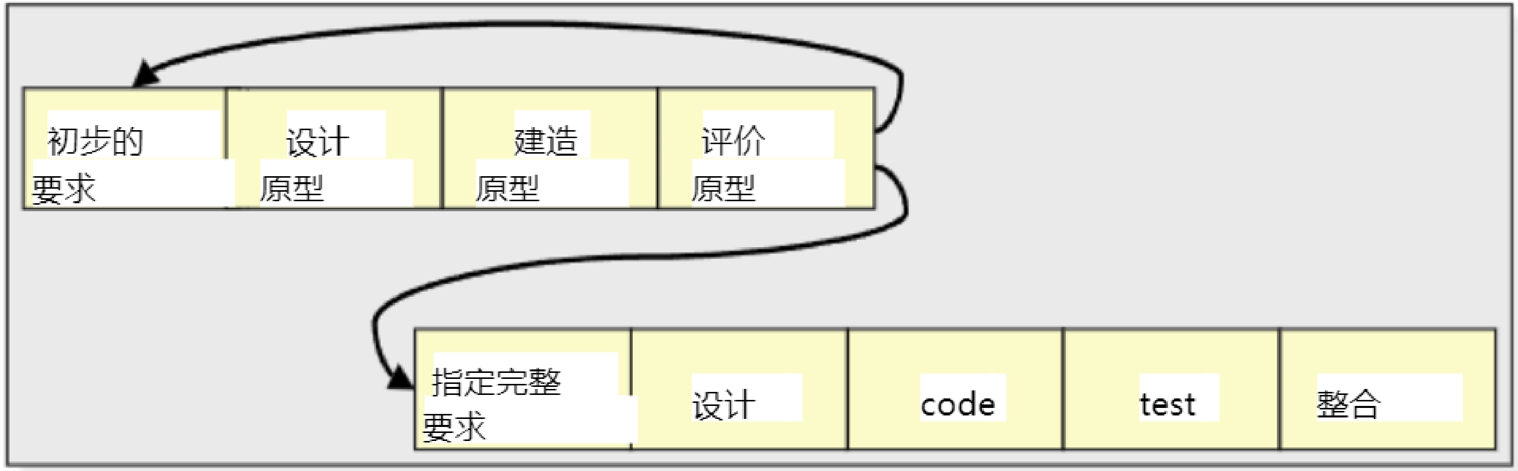
- 验证和确认的目标是什么?
- 检查质量
- 模型分析
- 原型制作

Prototyping lifecycle



- **Prototyping is used for:**
  - understanding the requirements for the user interface
  - examining feasibility of a proposed design approach
  - exploring system performance issues
- **Problems:**
  - users treat the prototype as the solution
  - a prototype is only a partial specification

原型生命周期



- 原型设计用于：
  - 了解用户界面的要求
  - 检查拟议设计方法的可行性
  - 探索系统性能问题
- 问题：
  - 用户将原型视为解决方案
  - 原型只是部分规格

Prototyping

> "A software prototype is a partial implementation constructed primarily to enable customers, users, or developers to learn more about a problem or its solution." [Davis 1990]
>
> "Prototyping is the process of building a working model of the system" [Agresti 1986]

- **Approaches to prototyping**
  - **Presentation Prototypes**
    - explain, demonstrate and inform – then throw away
    - e.g. used for proof of concept; explaining design features; etc.
  - **Exploratory Prototypes**
    - used to determine problems, elicit needs, clarify goals, compare design options
    - informal, unstructured and thrown away.
  - **Breadboards or Experimental Prototypes**
    - explore technical feasibility; test suitability of a technology
    - Typically no user/customer involvement
  - **Evolutionary (e.g. "operational prototypes", "pilot systems"):**
    - development seen as continuous process of adapting the system
    - "prototype" is an early deliverable, to be continually improved.

原型制作

“软件原型是主要为了使客户、用户、
或开发人员了解有关问题或其解决方案的更多信息。” [Davis 1990] “原型设计是构建系统工作模型的过程”[Agresti 1986]

- 原型设计方法
  - Ø 演示原型
    - 解释、演示和告知——然后扔掉
    - 例如用于概念证明；解释设计特点； ETC。
  - Ø 探索性原型
    - 用于确定问题、引发需求、明确目标、比较设计方案
    - 非正式的、无组织的、被丢弃的。
  - Ø 面包板或实验原型
    - 探索技术可行性；测试技术的适用性
    - 通常没有用户/客户参与
  - Ø 进化（例如“操作原型”、 “试点系统”）：
    - 发展被视为适应系统的持续过程
    - “原型”是早期交付的成果，需要不断改进。

- **Throwaway Prototyping**
  - **Purpose**:
    - to learn more about the problem or its solution…
    - discard after desired knowledge is gained.
  - **Use**:
    - early or late
  - **Approach**:
    - horizontal - build only one layer (e.g. UI)
    - “quick and dirty”
  - **Advantages**:
    - Learning medium for better convergence
    - Early delivery → early testing → less cost
    - Successful even if it fails!
  - **Disadvantages**:
    - Wasted effort if reqts change rapidly
    - Often replaces proper documentation of the requirements
    - May set customers’ expectations too high
    - Can get developed into final product

- **Evolutionary Prototyping**
  - **Purpose**
    - to learn more about the problem or its solution…
    - …and reduce risk by building parts early
  - **Use**:
    - incremental; evolutionary
  - **Approach**:
    - vertical - partial impl. of all layers;
    - designed to be extended/adapted
  - **Advantages**:
    - Requirements not frozen
    - Return to last increment if error is found
    - Flexible(?)
  - **Disadvantages**:
    - Can end up with complex, unstructured system which is hard to maintain
    - early architectural choice may be poor
    - Optimal solutions not guaranteed
    - Lacks control and direction

- 一次性原型设计Ø目的:
    - 了解有关问题或其解决方案的更多信息……
    - 获得所需知识后丢弃。

  Ø用途:
    - 早或晚

  Ø方法:
    - 水平 - 仅构建一层（例如 UI）
    - “又快又脏”

  Ø优点:
    - 更好融合的学习媒介
    - 早期交付 ® 早期测试 ® 更低的成本
    - 就算失败也能成功！

  Ø缺点:
    - 如果需求变化很快，就会浪费精力
    - 通常会取代适当的需求文档
    - 可能将客户的期望设定得太高
    - 可以开发成最终产品

- 进化原型

  Ø目的
    - 了解有关问题或其解决方案的更多信息……
    - …并通过尽早构建零件来降低风险

  Ø用途:
    - 增加的;进化的

  Ø方法:
    - 垂直 - 部分实现。所有层;
    - 旨在扩展/适应

  Ø优点:
    - 需求未冻结
    - 如果发现错误则返回到最后的增量
    - 灵活的（？）

  Ø缺点:
    - 最终可能会形成复杂、非结构化且难以维护的系统
    - 早期的架构选择可能很差
    - 无法保证最佳解决方案
    - 缺乏控制和方向

Validation Techniques

- What are goals of verification and validation?
- Checking quality
- Model analysis
- **Prototyping**

验证技术

- 验证和确认的目标是什么?
- 检查质量
- 模型分析
- 原型制作

Workshop 4

Build a prototype

Purpose:

- Illustrate the major functionality of the system
- Check the feasibility and validity of the requirements

- **Prototyping using**
  - **Pen, paper, post-its, markers, etc**
  - Develop the mockups, show scenario what the specified system should do

OR

- **Prototyping using the**
  - **proto.io**
  - 15 days trial

The prototype should support (a part of) the requirements specified in the requirements specification. Revise and complement the specification, if prototyping shows discrepancies

工作坊4

构建原型

目的：

- 说明系统的主要功能
- 检查需求的可行性和有效性

- 原型设计使用
  - Ø笔、纸、便利贴、记号笔等
  - Ø开发模型，展示指定系统应该做什么的场景

OR

- 原型设计使用
  - Ø 原型.io
  - Ø 15天试用期

原型应该支持需求规范中指定的（部分）需求。如果原型设计显示出差异，则修改并补充规范

Presentation

Requirements Specification & Prototype demo

- Important presentation points:
  - **Validation of the input**
    What is the problem and its scope (how you were managing the scope from workshop to workshop)
  - **Validation of the execution of activities**
    What was the RE process, what activities have you executed to reach the solution, how was it supported with the requirements management activities?
  - **Validation of the output**
    What are the most important results?

推介会

需求规格和原型演示

- 重要的演示要点：
  - 输入验证
    问题是什么及其范围（您如何管理各个研讨会的范围）
  - 活动执行的验证
    RE 流程是什么，您执行了哪些活动来达成解决方案，需求管理活动如何支持它?
  - 验证输出
    最重要的结果是什么?