课程：软件工程方法与实践

Requirements Engineering

—Fundamentals, Framework and Context

董　威

wdong@nudt.edu.cn

Department of Computer Science

National University of Defense Technology

Textbook: Klaus Pohl. “Requirements Engineering: Fundamentals, Principles, and Techniques”. Springer, 2010

1 MOTIVATION

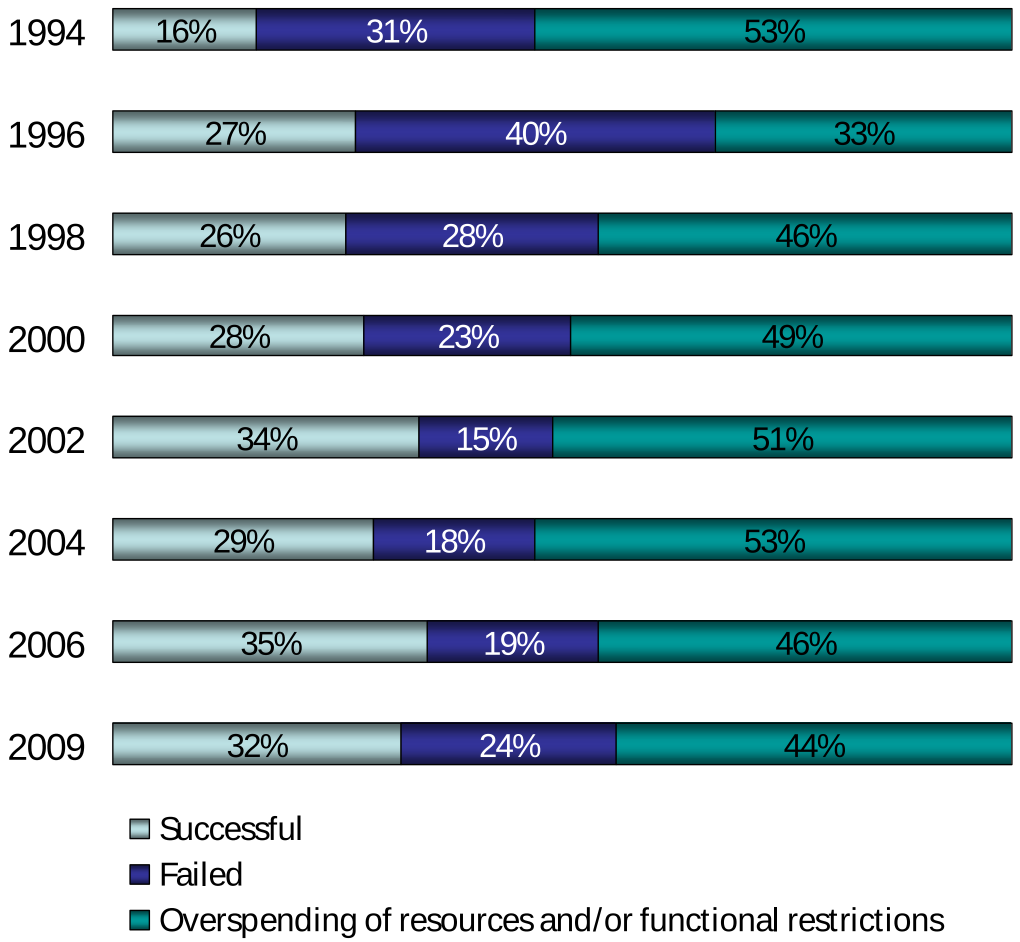
1.1 Software-Intensive Systems

- Software-intensive systems are systems mainly driven by software
  - Information systems: software on general-purpose computers
  - Embedded systems: integration of hardware and software, importance of software growing
- Challenges in the development of software-intensive systems
  - Software-based innovations
  - Increasing complexity
  - Pressure to reduce costs
  - Shorter development times
  - Higher quality demands

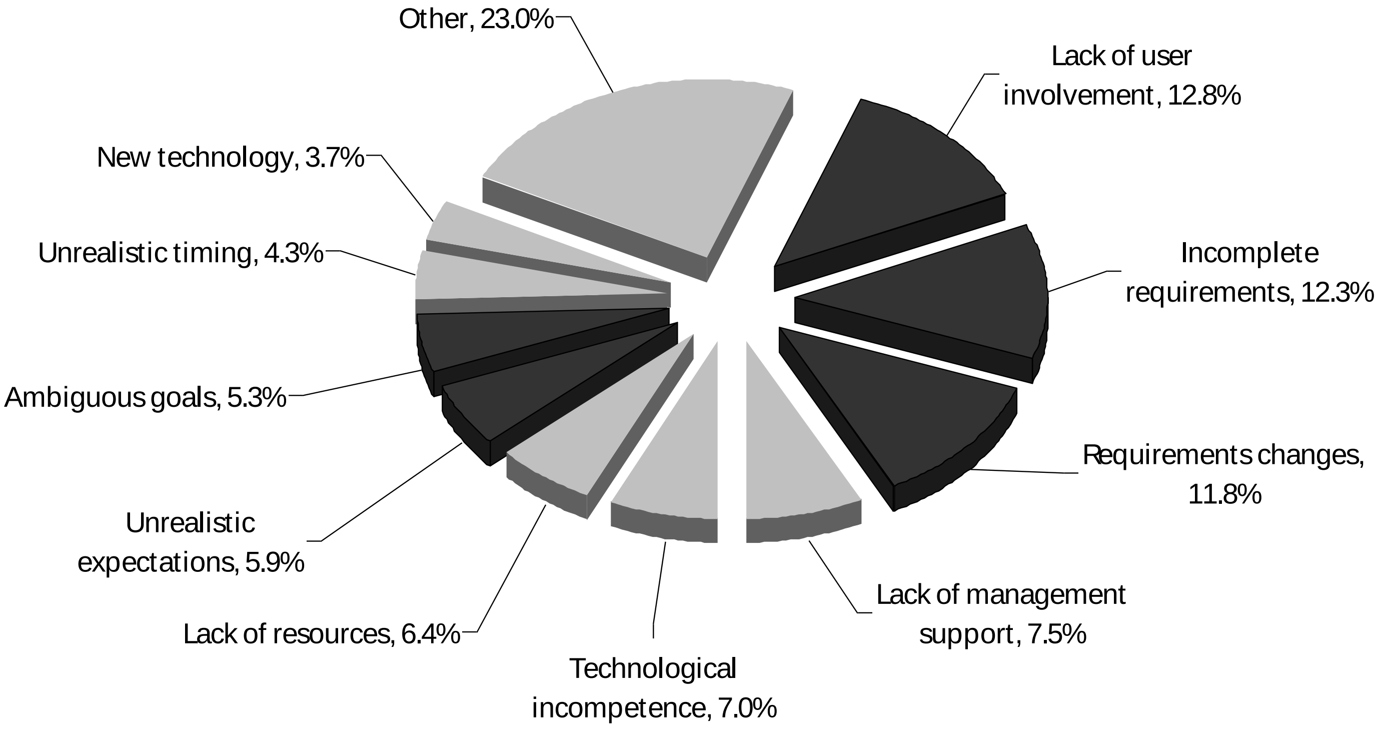
1.2 Importance of Requirements Engineering

- High impact on project success
  - Many projects failed or finished with overspend and/or restricted implemented functionality
  - Requirements defects are reason for approximately 50 % of the cases
- Example: London Ambulance Service
  - Poor requirements process that did not involve ambulance crews
  - Result: system was not able to operate under realistic conditions (i.e. sending too many ambulances to an incident) which endangered patients lives
- Defects in Requirements Engineering cause high costs
  - Defect found during programming increases costs by factor 20
  - Defect found during acceptance test increases costs by factor 100!

Project success rates from 1994 to 2009 taken from the Standish Group (CHAOS) studies



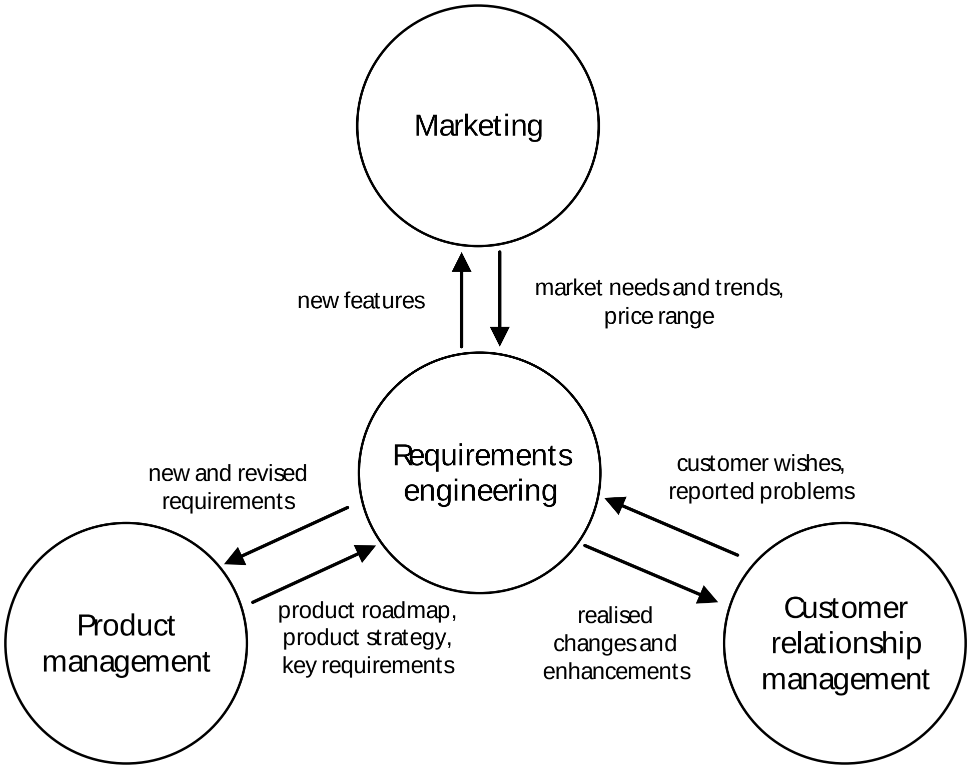
Reasons for resource overspend and/or functional restrictions (based on data from The Standish Group)



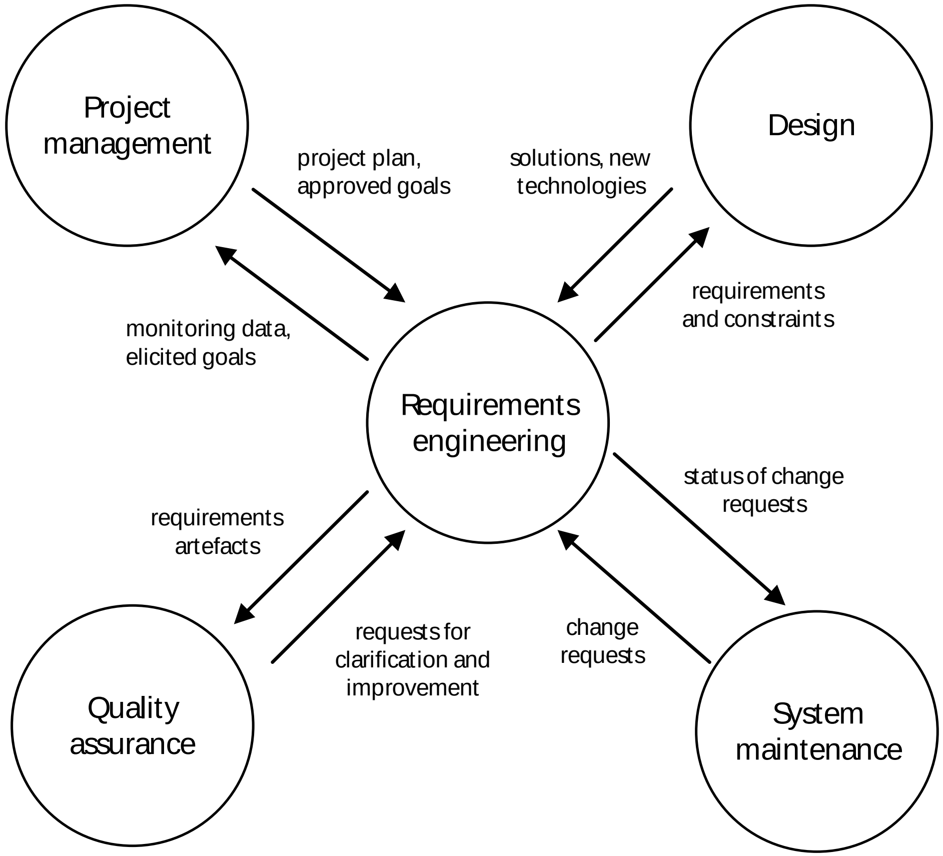
1.3 Embedding of Requirements Engineering in the Organisational Context

- Requirements engineering is embedded within a specific organisational context
  - Organisational processes
    - Marketing
    - Product management
    - Customer relation management
  - Development process in an organisation
    - Project management
    - Design
    - Quality assurance
    - System maintanance
- Involving relevant stakeholders from other organisational processes and other software development phases reduces requirements defects significantly

Interrelations between requirements engineering and other processes in the organisation



Interrelations between requirements engineering and other processes in the organisation


Project management
project plan, approved goals
solutions, new technologies
Design
requirements and constraints
monitoring data, elicited goals
Requirements engineering
status of change requests
requirements artefacts
requests for clarification and improvement
change requests
Quality assurance
System maintenance

2 REQUIREMENTS

2.1 The Term ”Requirement”

- A condition or capability
  1. ... needed by a user to solve a problem or achieve an objective
  2. ... that must be met or posessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document
  3. A documented form of 1 or 2
- A documented requirement is also called ”requirements artefact”

Example

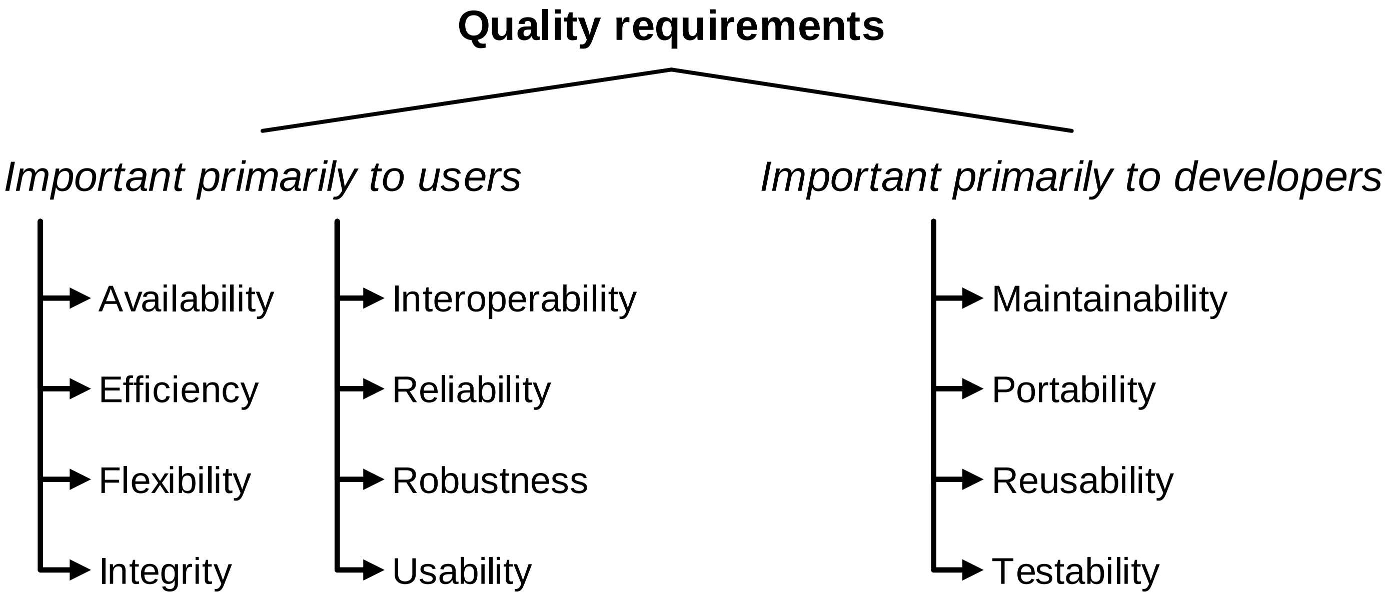
一个房屋信息系统的需求：

- R1：系统应该提供一个用户友好的界面。
- R2：系统能够生成包含所有被允许进入及被拒绝进入房屋的记录的月报表。
- R3：如果用户的PIN码是正确的，系统应开门并记录此次信息，包括日期、时间、用户姓名等。
- R4：系统应该在2020年12月20日交付。
- R5：门应该在PIN码正确输入后0.8秒打开。

2.2 Requirement Types

- Functional requirements
  - Define services the system should provide, behaviour of the system and in some cases also what the system should not do
- Quality requirements
  - Define quality properties of the system, a component, a service or a function (e.g. performance)
- Constraints
  - An organisational or technological requirement that restricts the way in which the system shall be developed
- Non-functional requirements
  - Term used in the literature, but do not really exist
    - Mainly underspecified functional requirements
    - Few of them are quality requirements

A quality requirements taxonomy



Quality requirements

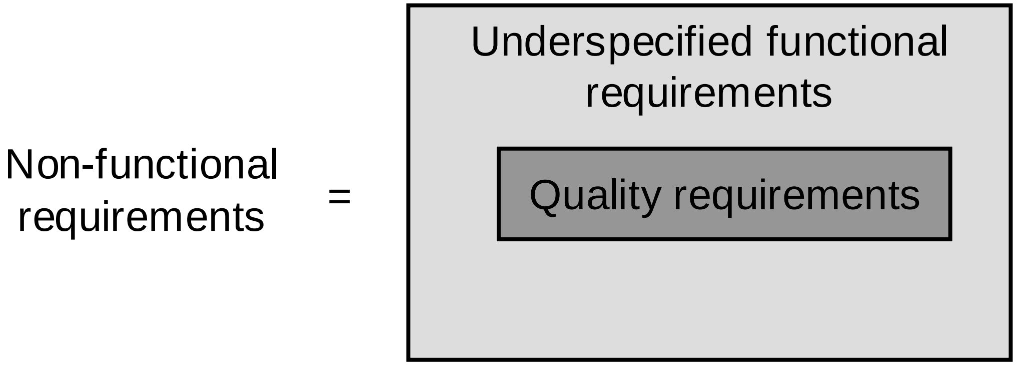
Important primarily to users

- Availability
- Efficiency
- Flexibility
- Integrity
- Interoperability
- Reliability
- Robustness
- Usability

Important primarily to developers

- Maintainability
- Portability
- Reusability
- Testability

Separation of ”non-functional” requirements into underspecified functional requirements and quality requirements

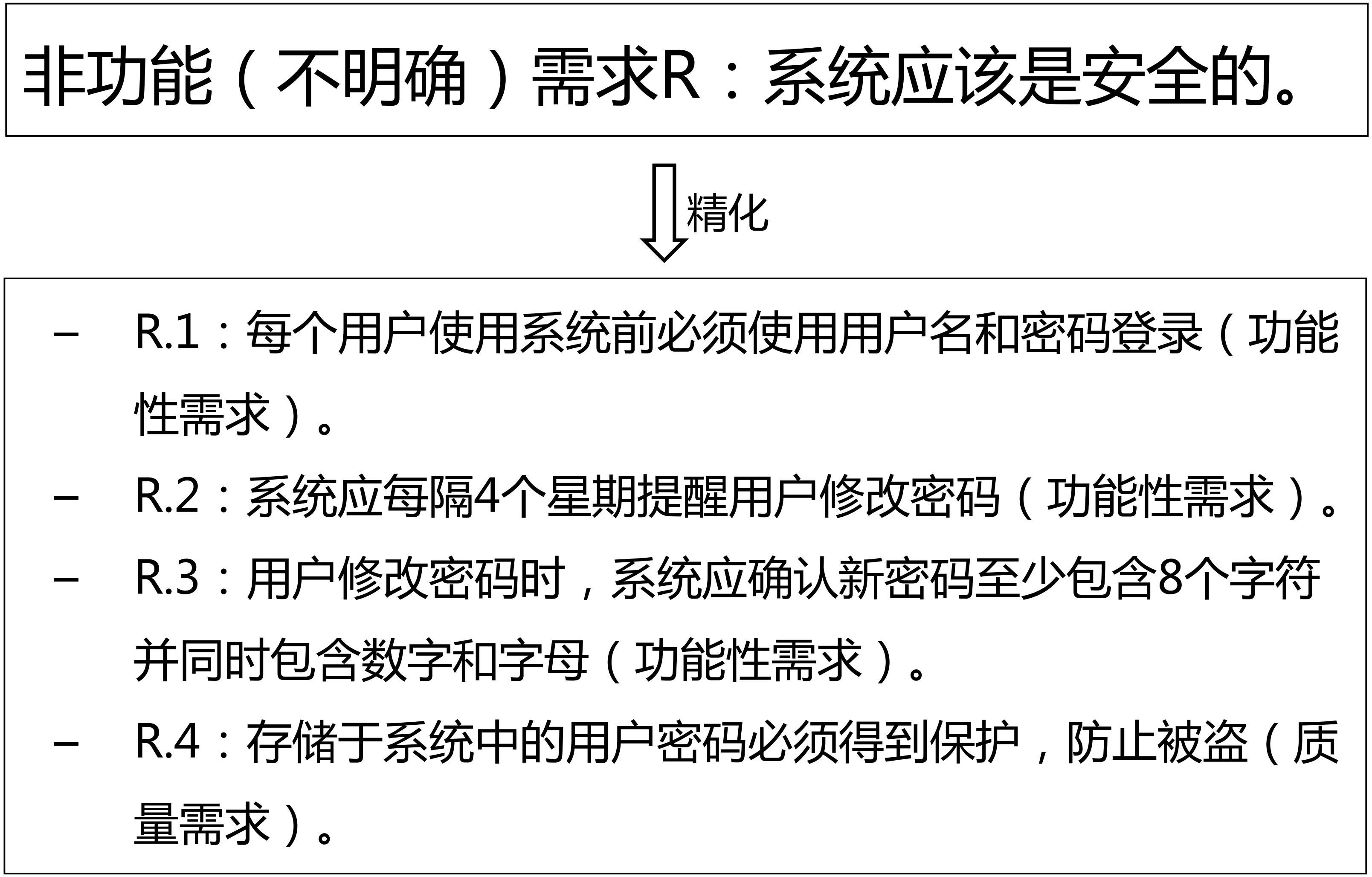


Non-functional requirements =

Underspecified functional requirements

Quality requirements

Example



非功能（不明确）需求R：系统应该是安全的。

精化

- R.1：每个用户使用系统前必须使用用户名和密码登录（功能性需求）。
- R.2：系统应每隔4个星期提醒用户修改密码（功能性需求）。
- R.3：用户修改密码时，系统应确认新密码至少包含8个字符并同时包含数字和字母（功能性需求）。
- R.4：存储于系统中的用户密码必须得到保护，防止被盗（质量需求）。

Example

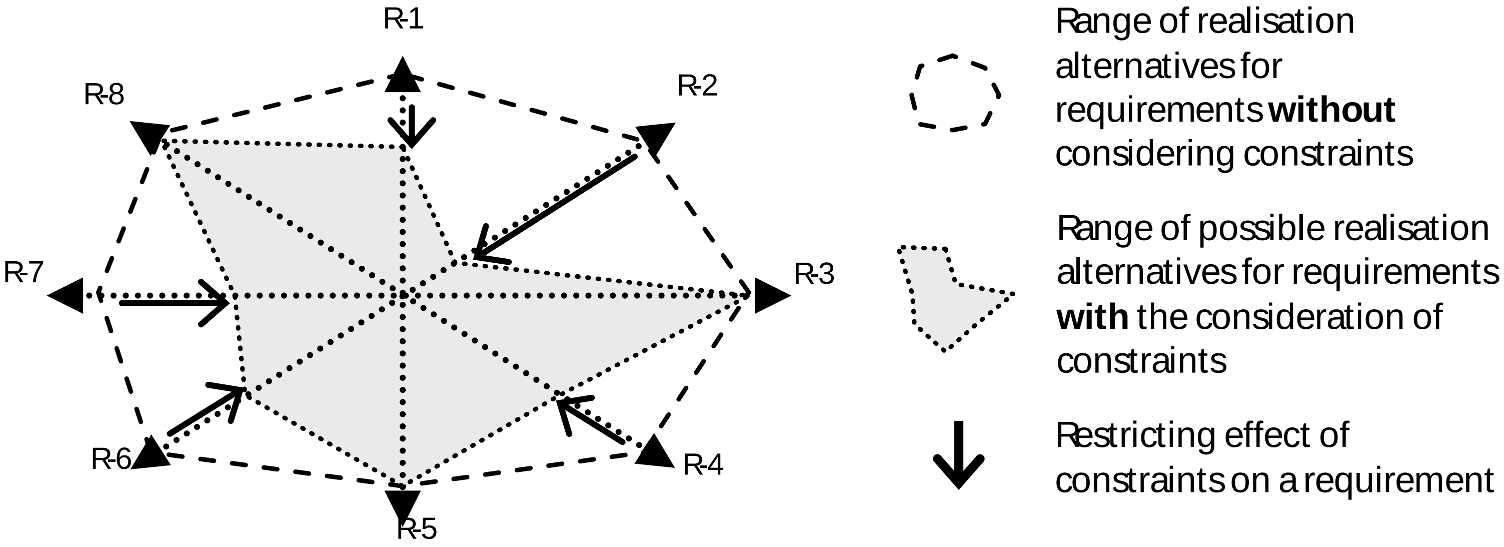
影响系统的约束：

- R1：由于当前保险公司所规定的条件，只有安全专家才允许停止系统的控制功能。
- R2：消防要求规定系统终端的尺寸不能超过120cm×90cm×20cm。

影响开发过程的约束：

- R1：系统开发的工作量不能超过480人月。
- R2：系统必须使用RUP过程进行开发。

Restrictions imposed by constraints



Range of realisation alternatives for requirements **without** considering constraints

Range of possible realisation alternatives for requirements **with** the consideration of constraints

Restricting effect of constraints on a requirement

Example

约束的影响：

- 初始需求R：该建筑的安全系统采用个人密码用于访问控制系统中对每个人的身份验证。

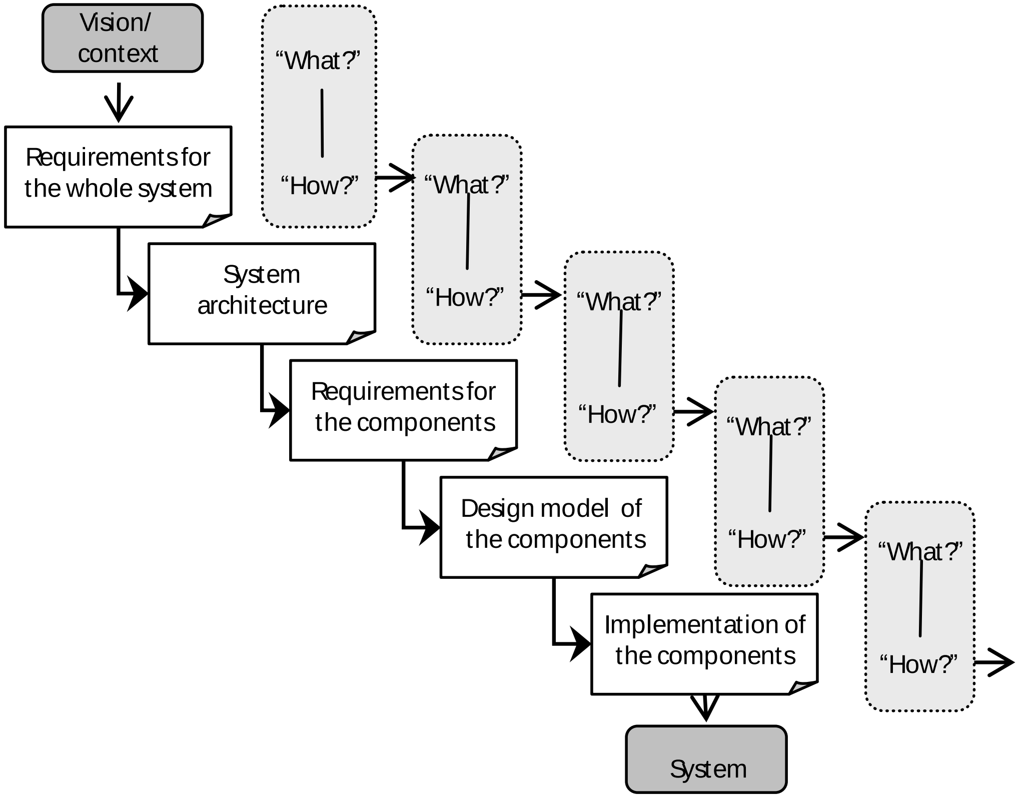
在需求访谈中，领域专家提到有政府法律规定该类重要建筑的访问控制必须符合某标准，该标准中提到禁止此类建筑使用个人密码进行身份验证，要求使用指纹传感器。

- R.1：指纹传感器应当用于访问控制系统中对每个人的身份验证。
- R.2：身份验证必须以至少60个细节的精度执行。

2.3 Problem vs. Solution

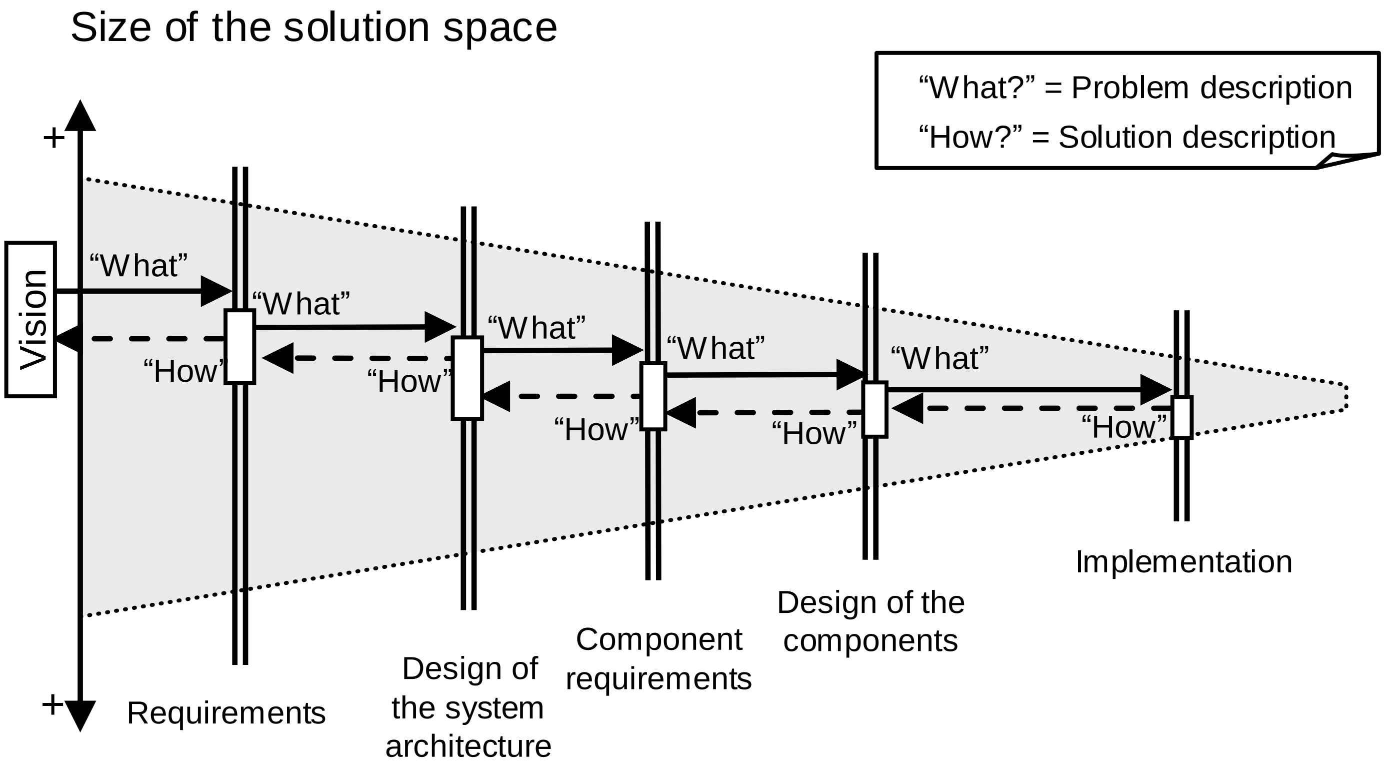
- ”What?” vs. ”How?”
  - Different views for different stakeholders, e.g.
    - Requirements engineering:
      - ”What?” – system vision
      - ”How?” – specified requirements
    - System architect:
      - ”What?” – system requirments
      - ”How?” – resulting system
- Solution space decreases during the development process
- Interactions between requirments and design stage possible

”What?” versus ”how?” in the system development process



Vision/ context
Requirements for the whole system
System architecture
Requirements for the components
Design model of the components
Implementation of the components
System
“What?”
“How?”
“What?”
“How?”
“What?”
“How?”
“What?”
“How?”
“What?”
“How?”

Reduction of the solution space during the development process



Example

问题定义——“做什么”：

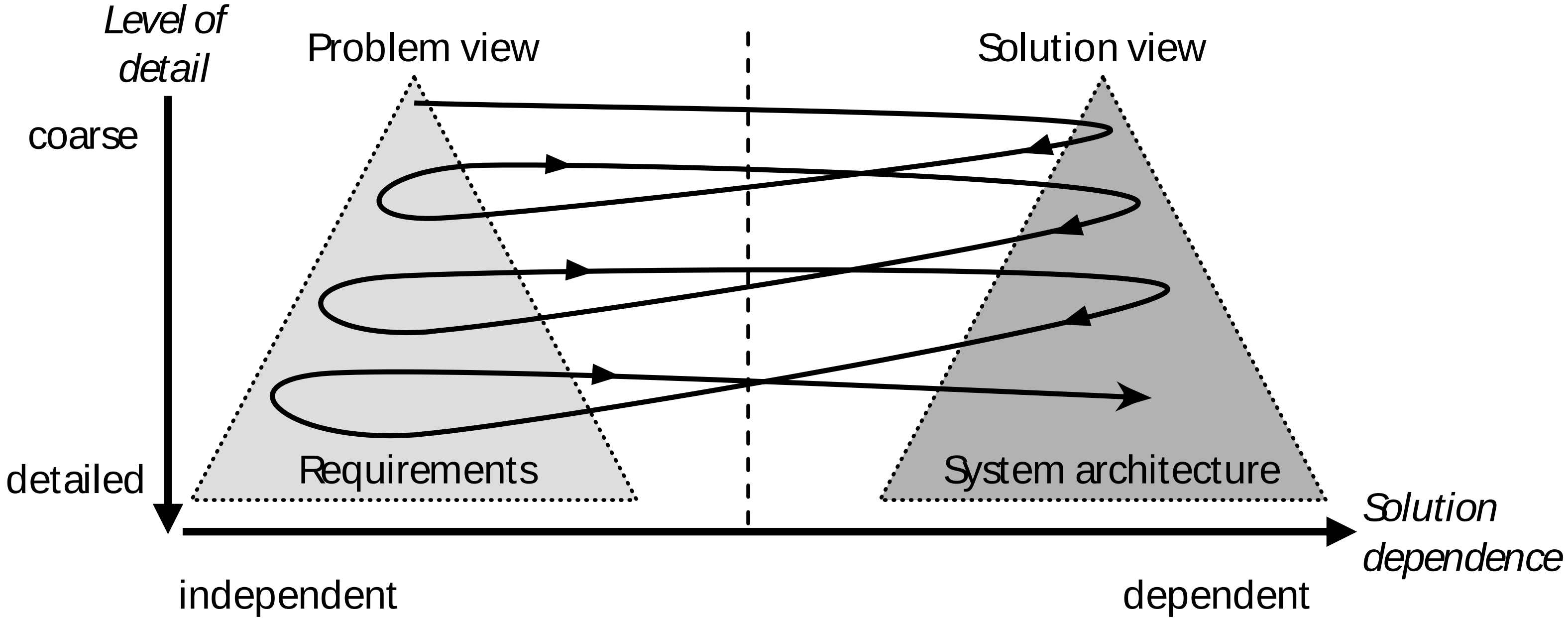
- R：导航系统应该方便地让司机输入行程目的地。

⇓ 高层需求精化为详细需求

解决方案——“怎么做”：

- R.1：当司机开始一个新行程时，导航系统应当显示以当前位置为中心的区域路线图。
- R.2：导航系统应当允许司机滚动和缩放路线图。
- R.3：当司机在路线图上选择了目的地后，系统应允许司机编辑详细的目的地信息（街道和门牌号码）。

Interactions between requirements engineering and architectural design

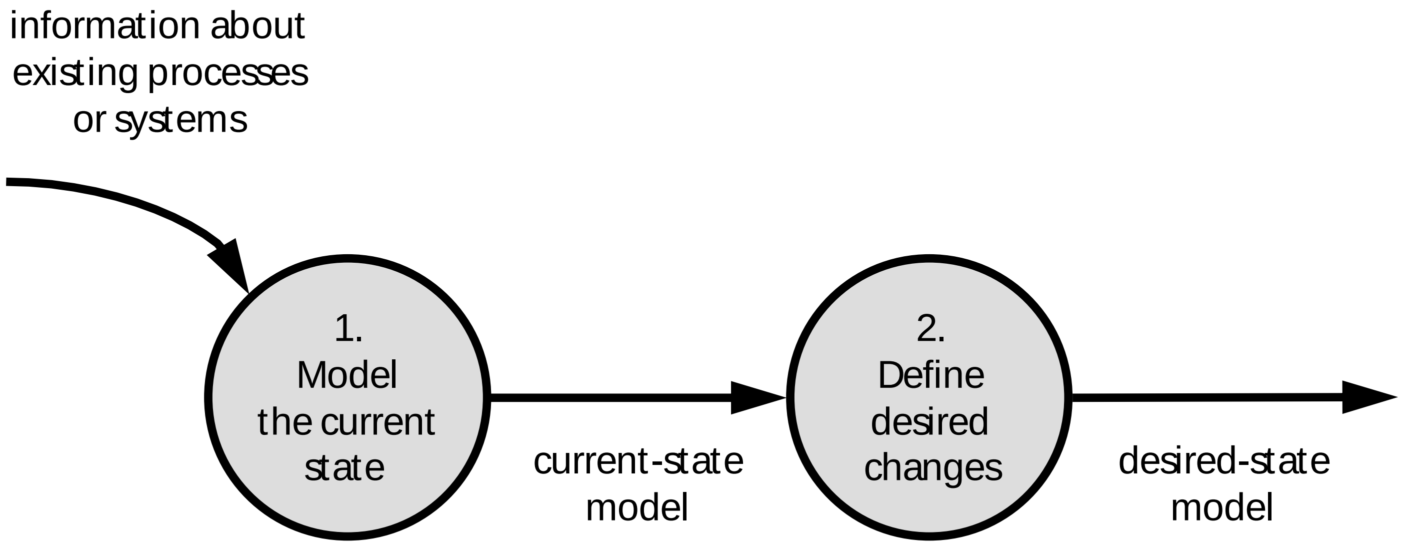


3. CONTINUOUS REQUIREMENTS ENGINEERING

3.1 Traditional Systems Analysis

- Structured Analysis
  - Analysis of the current state
    - Reflects the current realisation of the system
  - Definition of the desired state
    - Defines the requirements for the system developed
  - Integration of the new system into the existing environment
  - But: No definition how to define a model!

Current-state and desired-state models in systems analysis



information about existing processes or systems

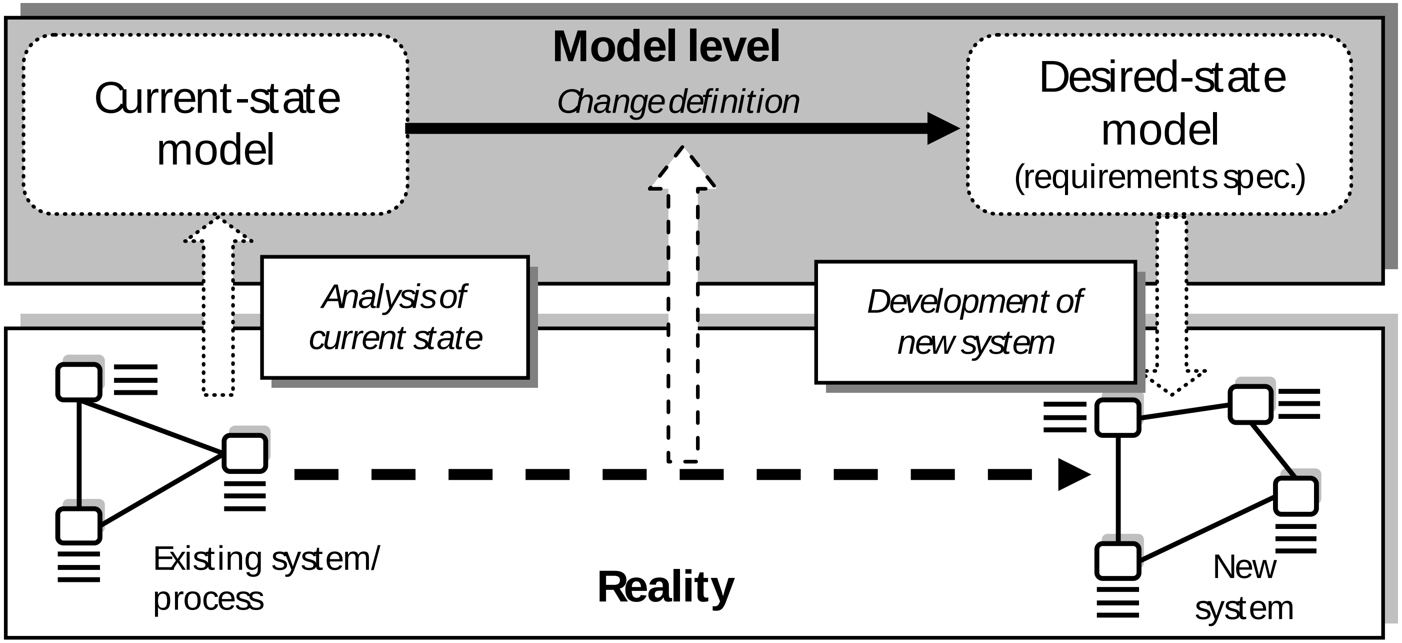
1. Model the current state

current-state model

2. Define desired changes

desired-state model

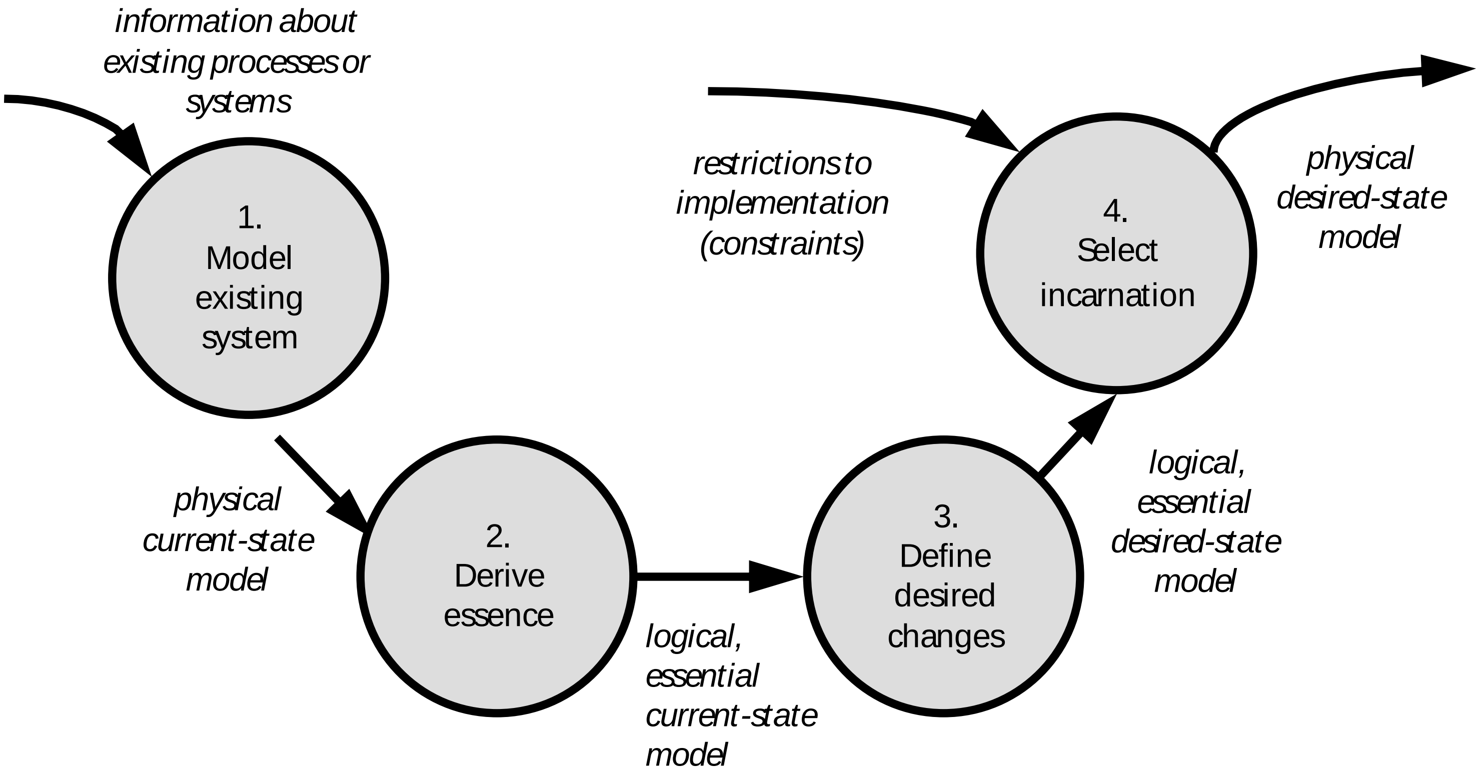
Current-state and desired-state models



3.2 Essential Systems Analysis

- Differentiate between essence and incarnation of a system
- Essence: a planned system that would exist if the technology to implement it was perfect
  - Essential models define the essence of a system
  - Essential models are stable and significantly smaller, there are no design restrictions
- Incarnation: sum of real-world items used to implement a system

Current and desired-state models in Essential Systems Analysis


information about existing processes or systems
1. Model existing system
physical current-state model
2. Derive essence
logical, essential current-state model
3. Define desired changes
logical, essential desired-state model
restrictions to implementation (constraints)
4. Select incarnation
physical desired-state model

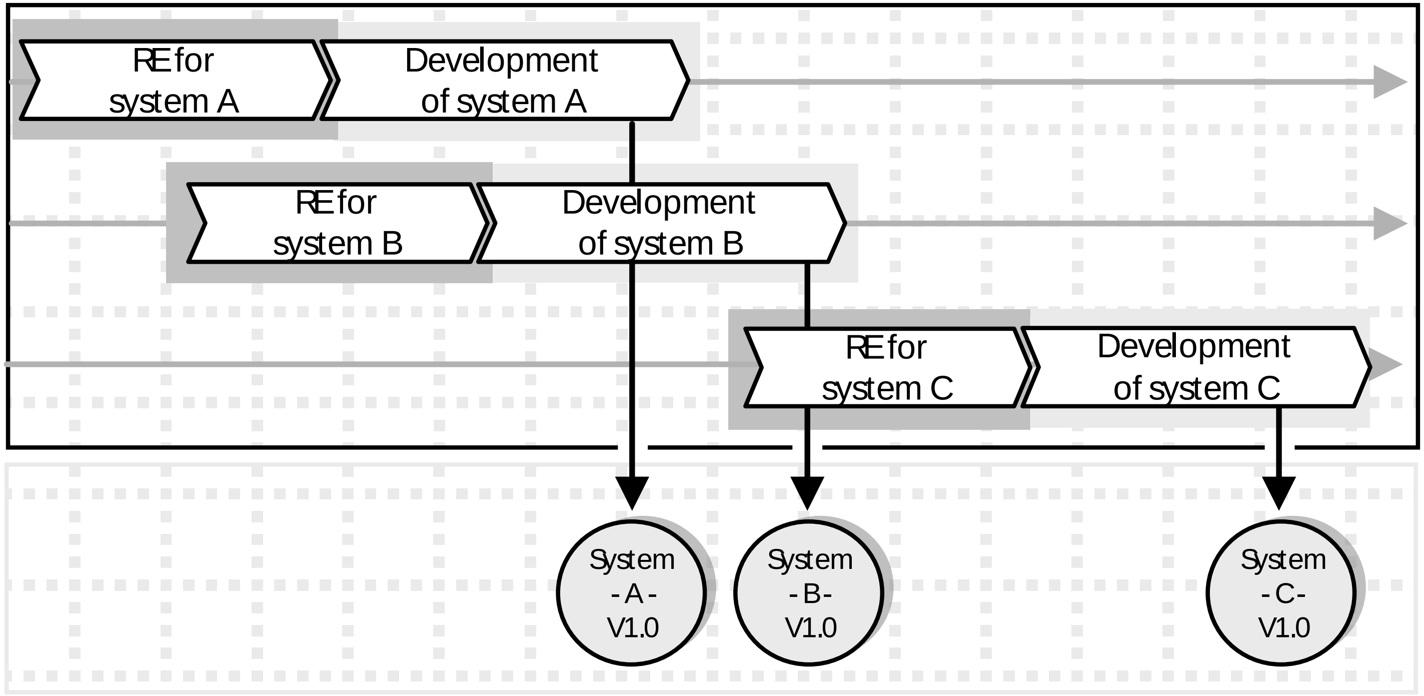
Advantages of Essential System Analysis

- More stable models
- Smaller than physical models
- Without design restrictions
- No much differences between new and current systems
- Independent with development methods

3.3 Requirements Engineering as an Early Development Phase

- Often, requirements engineering is seen as an early, single development phase
- RE is performed for each project at the beginning of the development process
- Elicitation is independent for each project

Requirements engineering as an early phase in development projects



3.4 Shortcomings of Systems Analysis and Phase Oriented Requirements

- Types of systems to be developed changed in the last decades
- Significant challenges arise due to new trends
- Wide range of technology
  - e.g. convergence of embedded and information systems
- Traditional requirements engineering methods are not capable to cope with the new challenges
- Requirements engineering as an early development phase has several disadvantages, for example:
  - Requirements are not updated during the development process
  - Each new project requires a time consuming current state analysis
  - Requirements reuse is not provided in a systematic manner
  - Narrow focus on one product leads to ignoring relevant information

Example

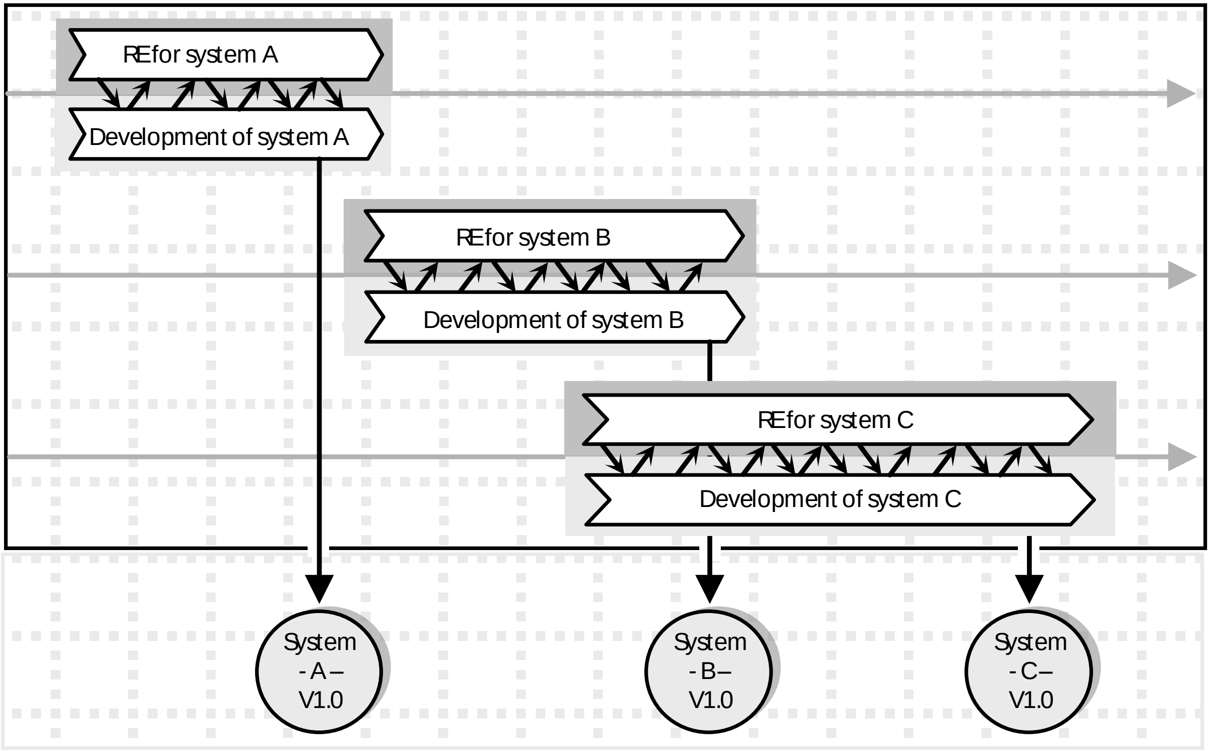
汽车工业中，各种创新性功能不断出现：

- ABS，自动防抱死刹车系统
- ACC，自适应巡航系统
- APS，自动驾驶系统
- BAS，制动辅助系统
- ESP，电子稳定装置
- PDC，驻车距离控制
- ……

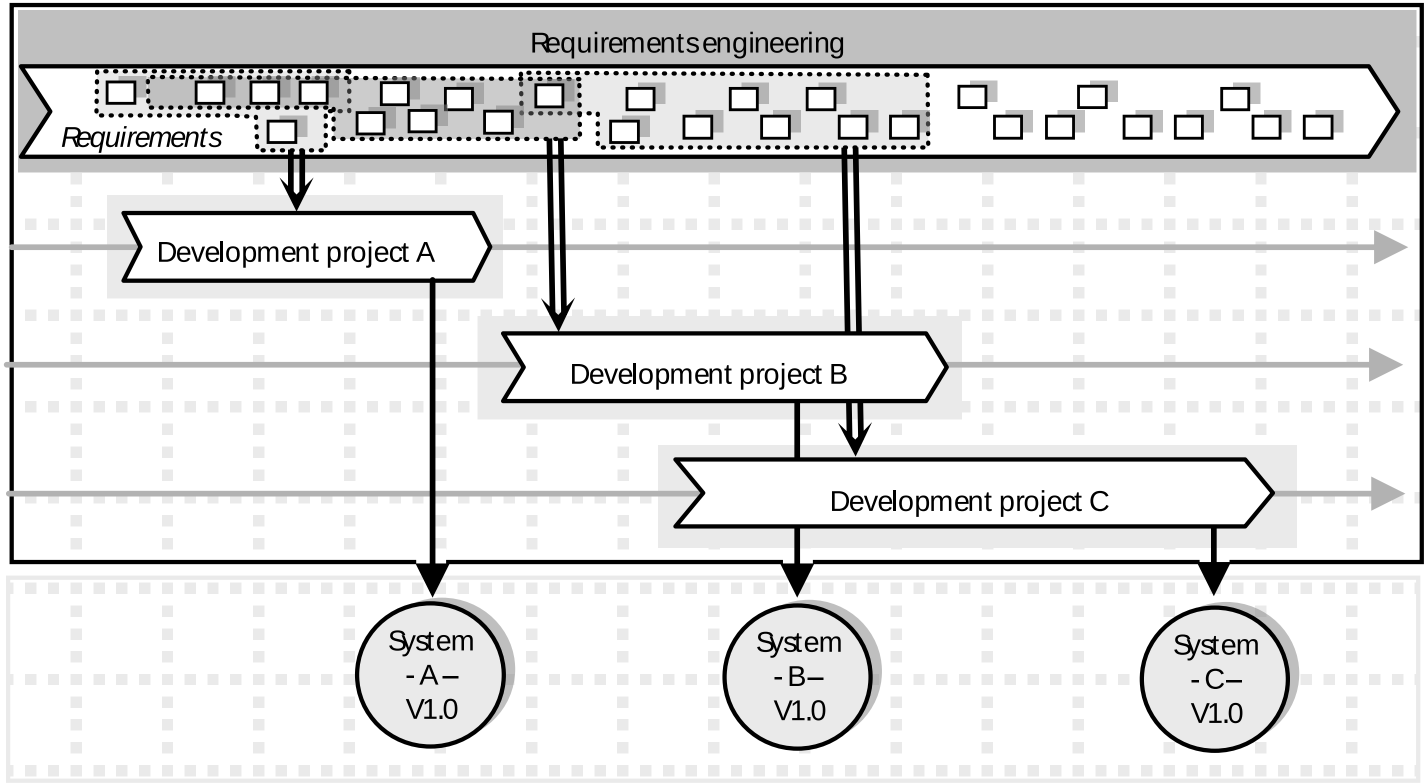
3.5 Continuous Requirements Engineering

- Requirements engineering as a cross-lifecycle, cross-project and cross-product activity
  - Requirements engineering is a continuous process
- Advantages for short product lifecycles
  - Established requirements base
  - Requirements stay up to-date
  - Shorter development times
  - Systematic reuse of requirements
  - Clear responsibilities

Requirements engineering as a cross-lifecycle activity



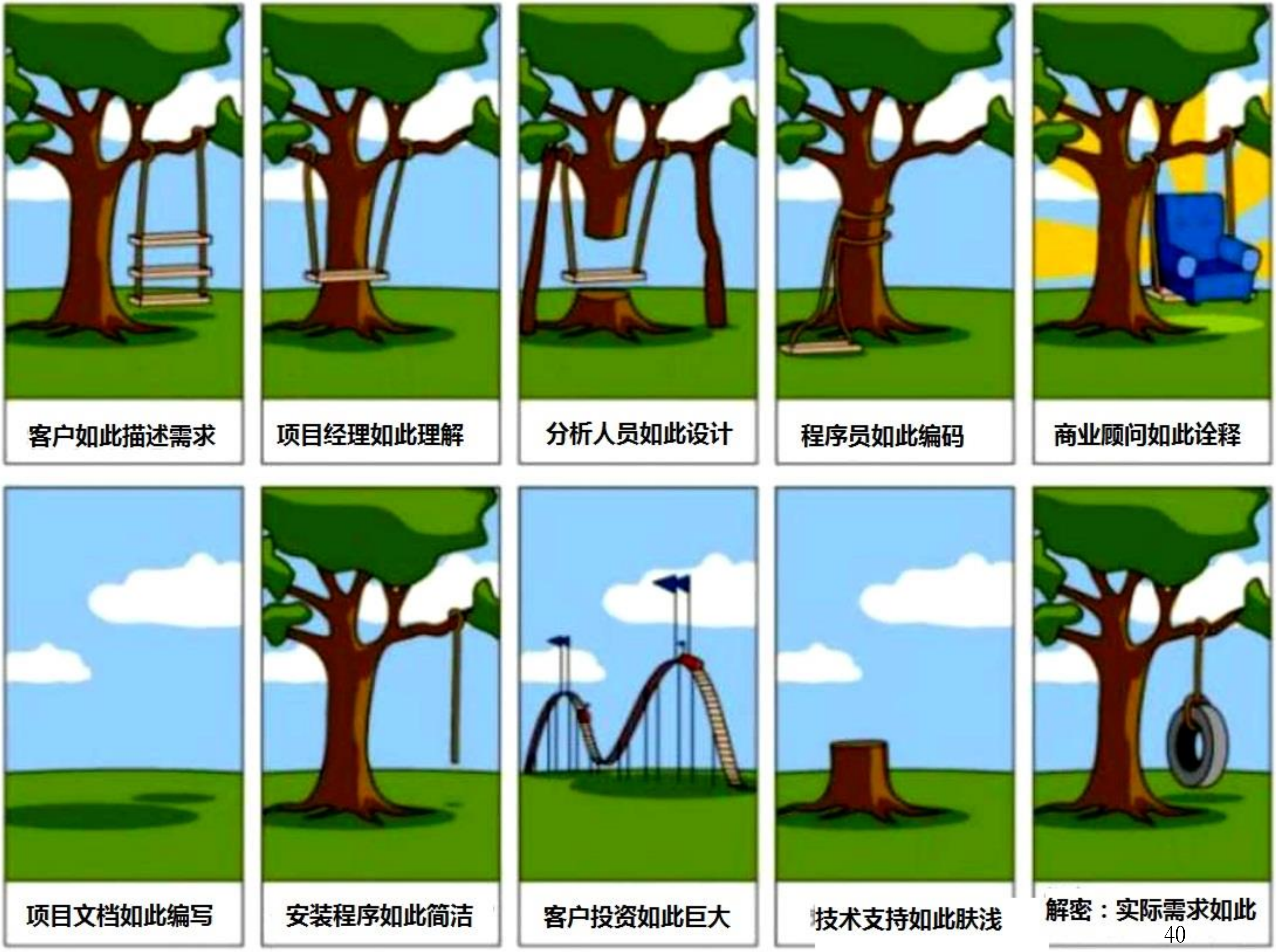
Requirements engineering as a cross-project and cross-product activity



4. THE REQUIREMENTS ENGINEERING FRAMEWORK

4.1 Goal of Requirements Engineering: Establishing a Vision in Context

- Vision: defines intended change to current reality
- Vision states a goal, not how to achieve it
- Vision is a guidance througout the development process
- System context essential for the requirements and the requirements engineering process


客户如此描述需求
项目经理如此理解
分析人员如此设计
程序员如此编码
商业顾问如此诠释
项目文档如此编写
安装程序如此简洁
客户投资如此巨大
技术支持如此肤浅
解密：实际需求如此

Definition of Requirements Engineering

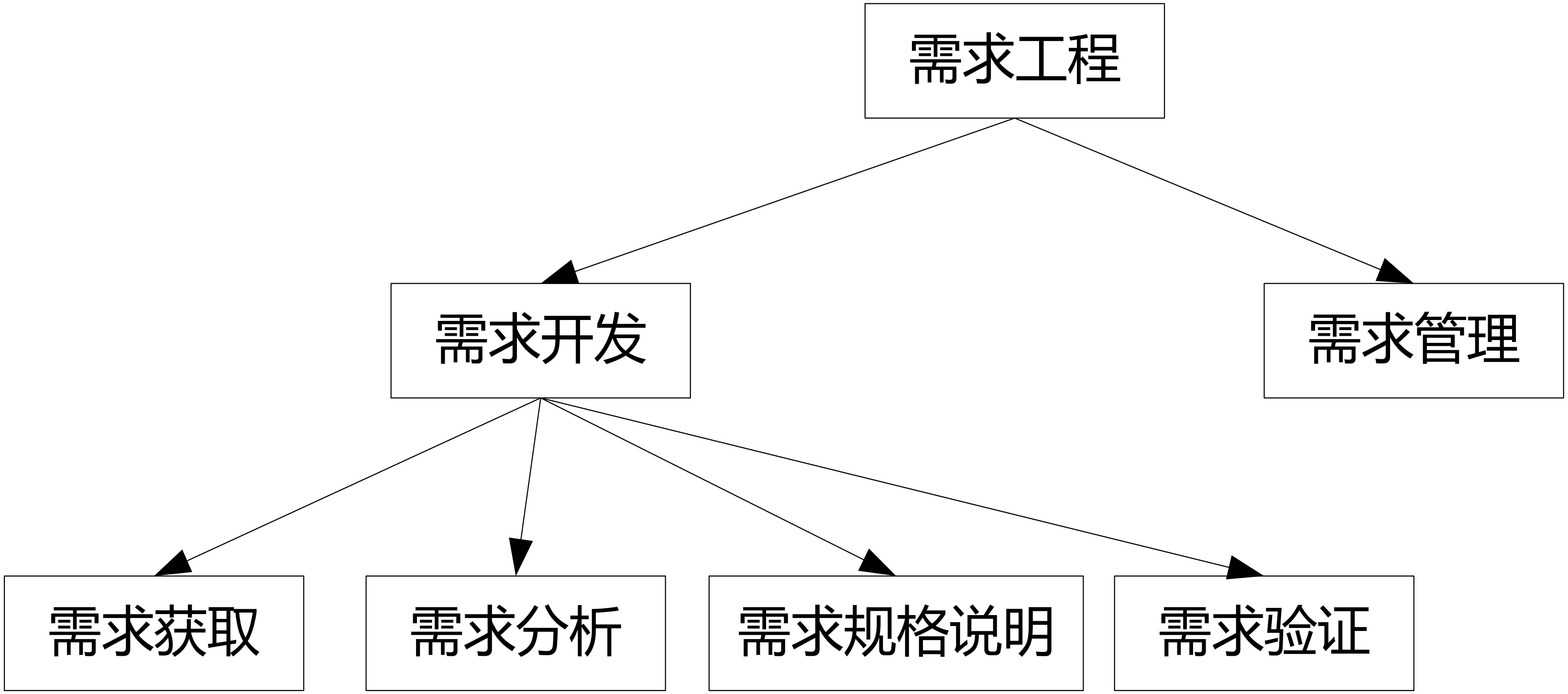
[Classic] –The application of a systematic, disciplined, quantifiable approach to the specification and management of requirements; that is the application of engineering to requirements.

—[Adapted from the definition of Software Engineering in IEEE 610.12-1990]

[Customer-oriented] –Understanding and documenting the customers' desires and needs.

[Risk-oriented] –Specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs.

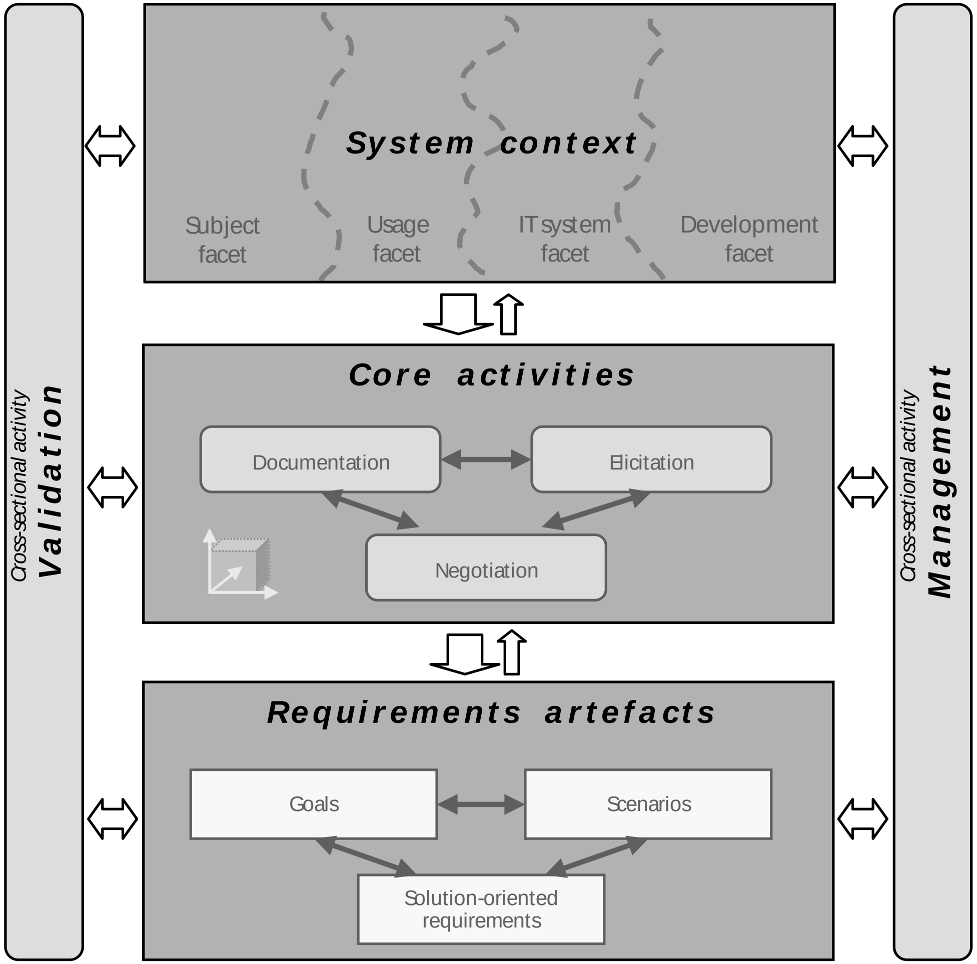
Traditional RE Framework



4.2 Overview of the Framework

- Four context facets
  - Subject, usage, IT system and development facet
- Three core requirements engineering activities
  - Documentation, elicitation and negotiation
- Two cross-sectional activities
  - Validation and management
- Three types of requirements artefacts
  - Goals, scenarios and solution-oriented requirements

The requirements engineering framework



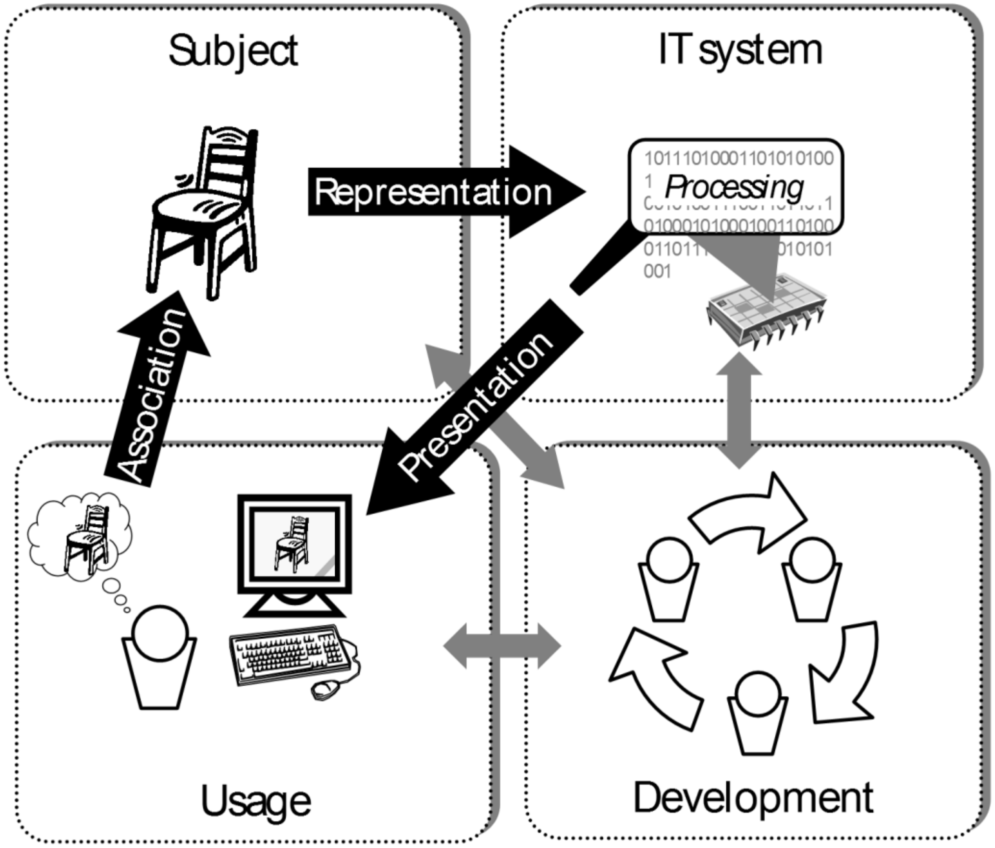
4.3 Four Context Facets

- Subject facet
  - Objects and events relevant for the system
  - For example elements the system must store or process information about
- Usage facet
  - Aspects concerning the usage by people or other systems
- IT system facet
  - Objects and elements of the IT system environment of the system
  - For example existing hardware and software components to be used
- Development facet
  - Aspects concerning the development process of the system
  - For example process guidelines, development tools

4.3 Four Context Facets (cont‘d)

- Relationship between subject and IT system facet
  - Representation of information about real-world objects such as the speed of a car or the name of a customer
- Relationship between IT system and usage facet
  - System processes represent information according to the functionality
- Relationship between usage and subject facet
  - System user interprets output of the system and associates it with the real world objects of the subject facet
- Role of the development facet
  - Consider relevant aspects of the other three context facets and their relationships, is therefore related to all other facets
- Use of facets improves quality of requirements documents

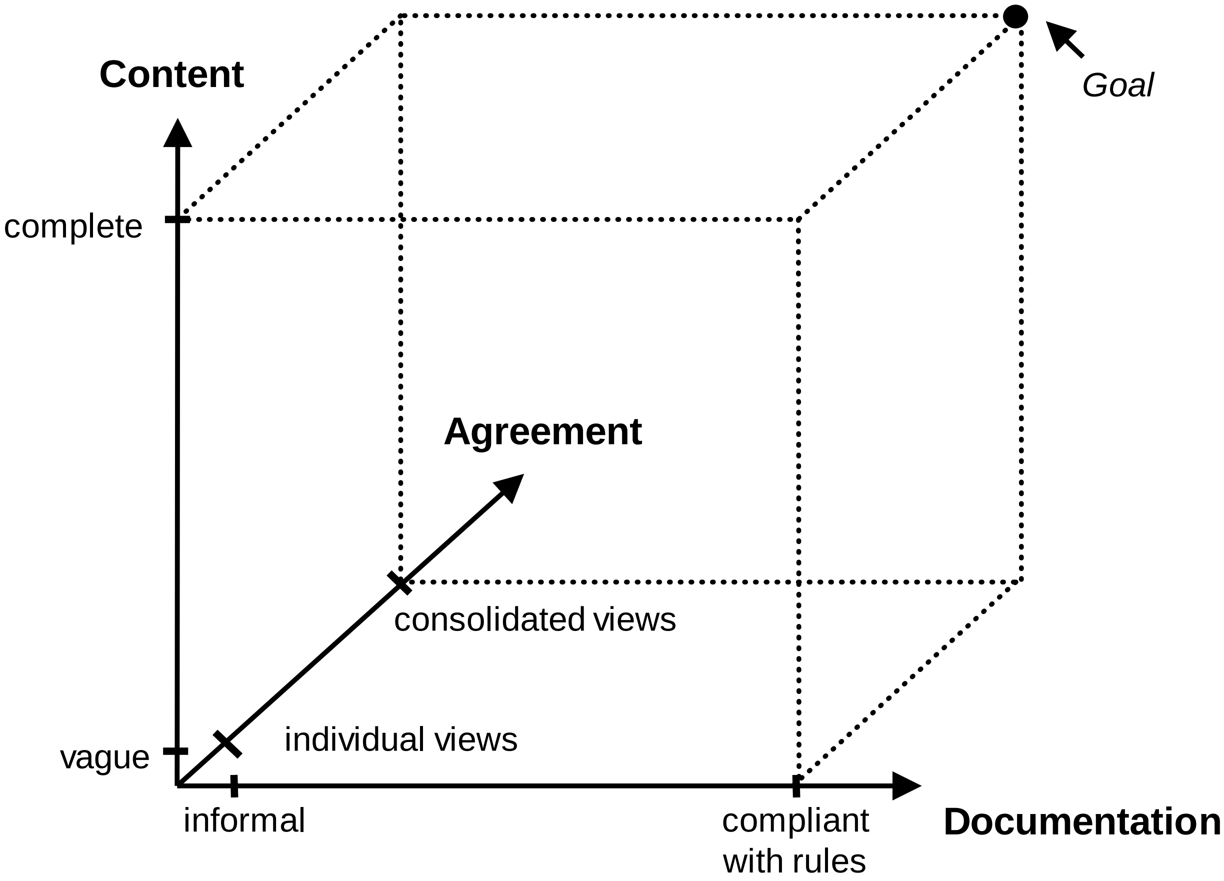
Logical relationships between the four context facets



4.4 Three Core Activities

- Three dimensions of requirements engineering
  - Content dimension
  - Agreement dimension
  - Documentation dimension
- Three core activities
  - Documentation (progress in documentation dimension)
  - Elicitation (progress in content dimension)
  - Negotiation (progress in agreement dimension)

The three dimensions of requirements engineering



4.5 Two Cross Sectional Activities

- Validation
  - Validation of the requirements artefacts
  - Validation of the core activities
  - Validation of the consideration of the system context
- Management
  - Management of artefacts
  - Management of activities
  - Observation of the system context
- Activities are interrelated, for example performing one activity may require the execution of additional activities

4.6 The Three Kinds of Requirements Artefacts

- Goals
  - Intention with regard to the objectives, properties or use of the system
- Scenarios
  - Describes a concrete example of satisfying or failing to satisfy a goal (or a set of goals)
- Solution-oriented requirements
  - Define the data, functions, behaviour, quality and constraints
  - Often imply a conceptual solution
- The three types are used complementarily

Example

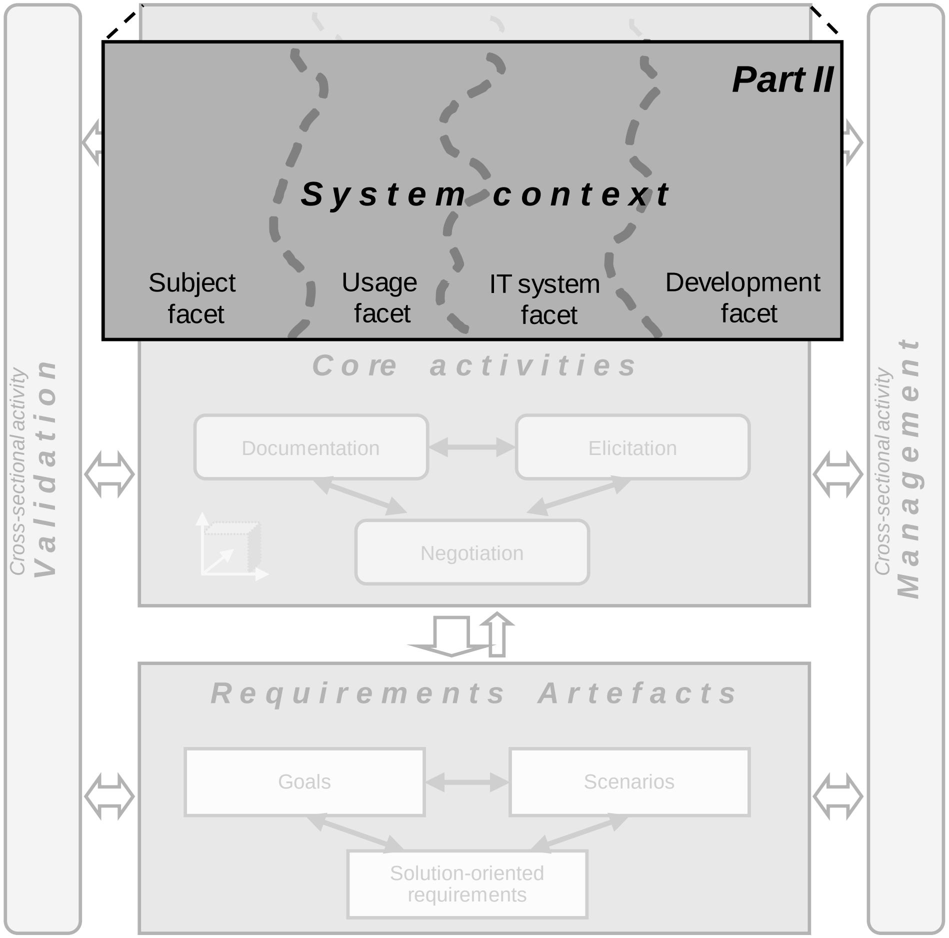
汽车导航系统的目标：

- G1：系统应自动引导驾驶员到达指定地点。
- G2：系统的响应时间应比先前的系统至少降低20%。

“自动刹车操作”场景：

- Carl正驾驶汽车以每小时50英里的速度在高速公路上行驶。Peter驾驶另一辆汽车在Carl前方行驶并开始减速。当Carl发现Peter减速后也踩下刹车踏板。此时Carl的车载系统检测到与前方车辆的距离已不在安全距离内，因此向驾驶员发出警告。接着，两车距离继续拉近。此时，为了帮助驾驶员，车载系统会启动自动全刹车，并通知Carl开启了自动全刹车。当两辆车距离停止减少后，系统会终止全刹车动作。但是，系统会继续控制Carl车辆的速度，直到与前车的距离拉大到安全距离，然后终止此次动作并通知Carl。

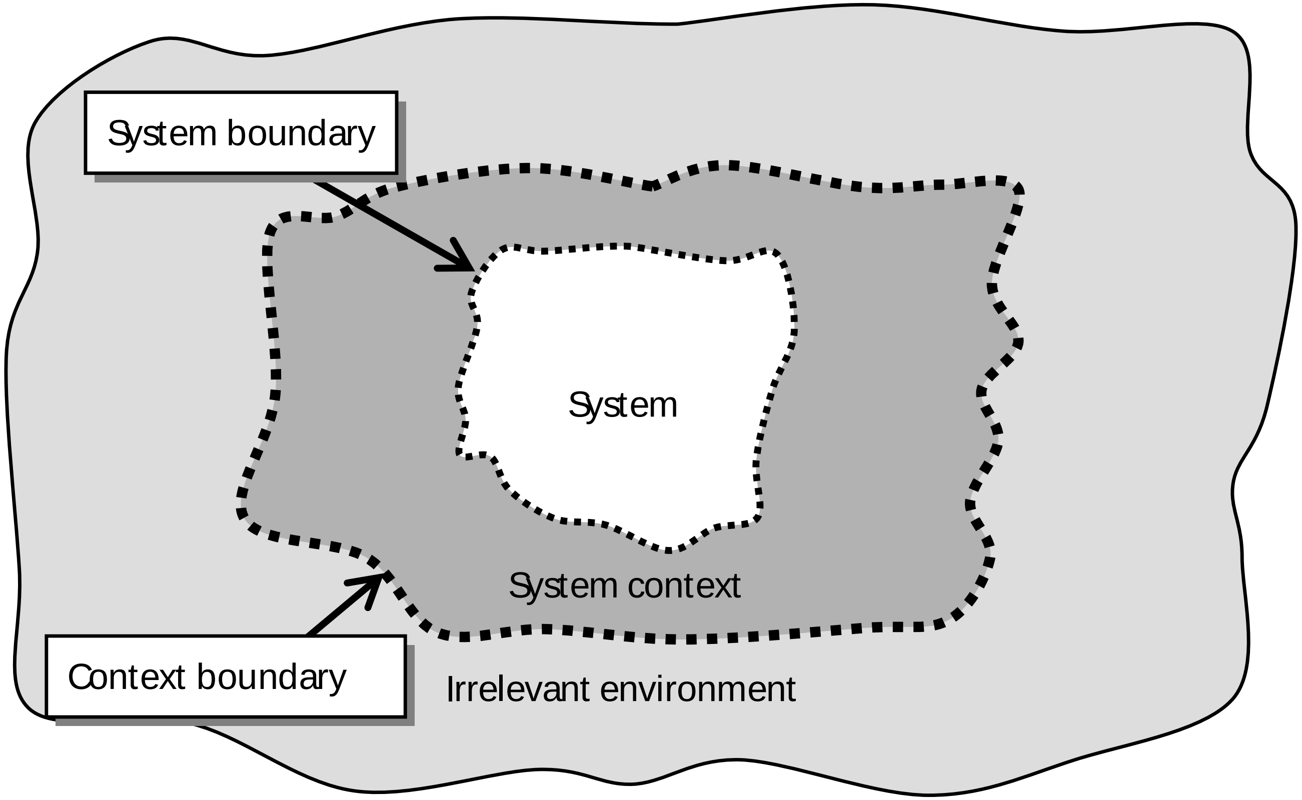
5 SYSTEM AND CONTEXT BOUNDARIES



5.1 The Term ”Context”

- Software-intensive system is always embedded in enviroment
- System environment significantly influences requirements for the system
- System context is the part of the environment relevant for the system requirements
- Rquirements only exist in a context

System boundary and context boundary separating the system, the context, and the irrelevant environment



Context Aspects

- 上下文方面是系统上下文中的各种物质和非物质对象，例如人、技术与非技术系统、过程、物理规律等。
- 每个上下文方面都与系统有某种特定的关系
  - 上下文方面与系统直接交互，影响系统需求的定义。
  - 上下文方面完全不与系统发生交互，但通过某种方式影响系统需求。

Example

与系统存在直接交互关系的上下文方面：

- 银行客户使用ATM机取款，如果ATM可能被国际用户使用，那么接口必须支持多种语言。

与系统没有交互关系但影响系统需求的上下文方面：

- 法律要求进入到银行系统中以及在系统中使用的敏感数据项必须使用某些加密标准加密。

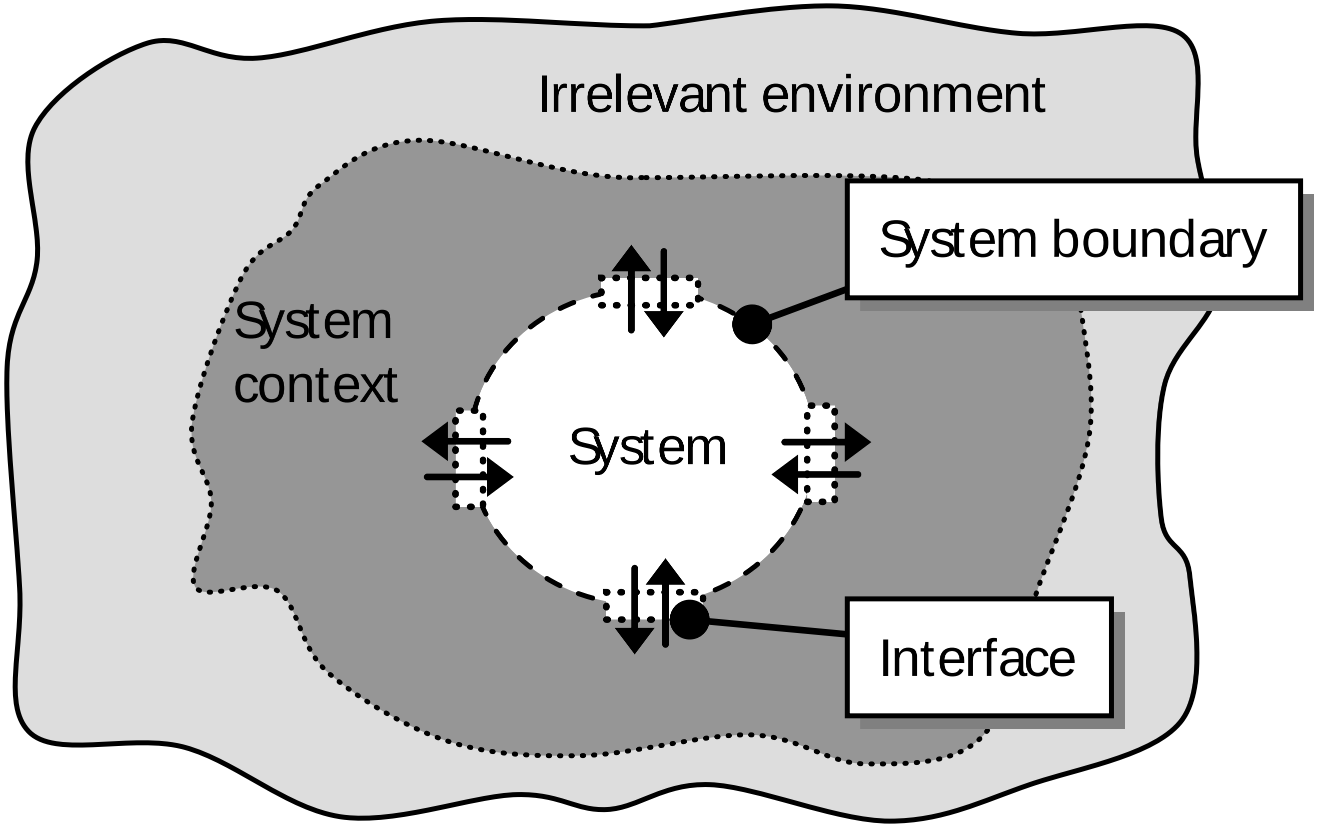
5.2 System Boundary

- System boundary separates changeable from stable artefacts
  - Artefacts within the system boundary are changed during the development process
  - Artefacts outside of the system boundary remain constant
- System boundary is initially only vaguely defined
  - ”Grey zone”
- Later adjustments during the development process change the system boundary

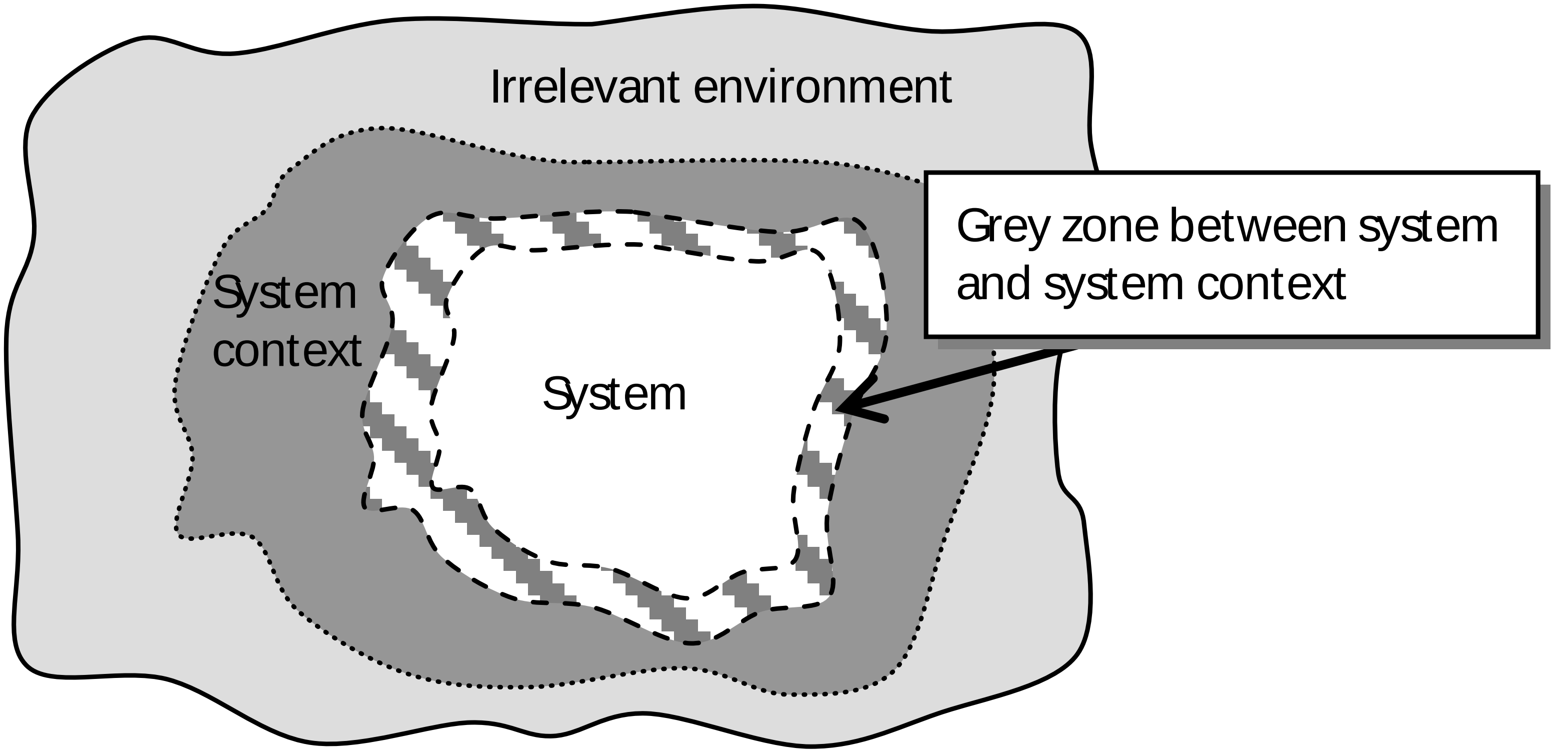
Interface

- 为了识别系统和上下文边界，可以考虑系统的信息源（Source）和接收单元（Sink）。
- 信息源和接收单元包括
  - 人
  - 其他技术或非技术系统
  - 传感器和执行器
- 信息源和接收单元通过接口与系统交互。

System boundary and system interfaces



Grey zone between the system and the context



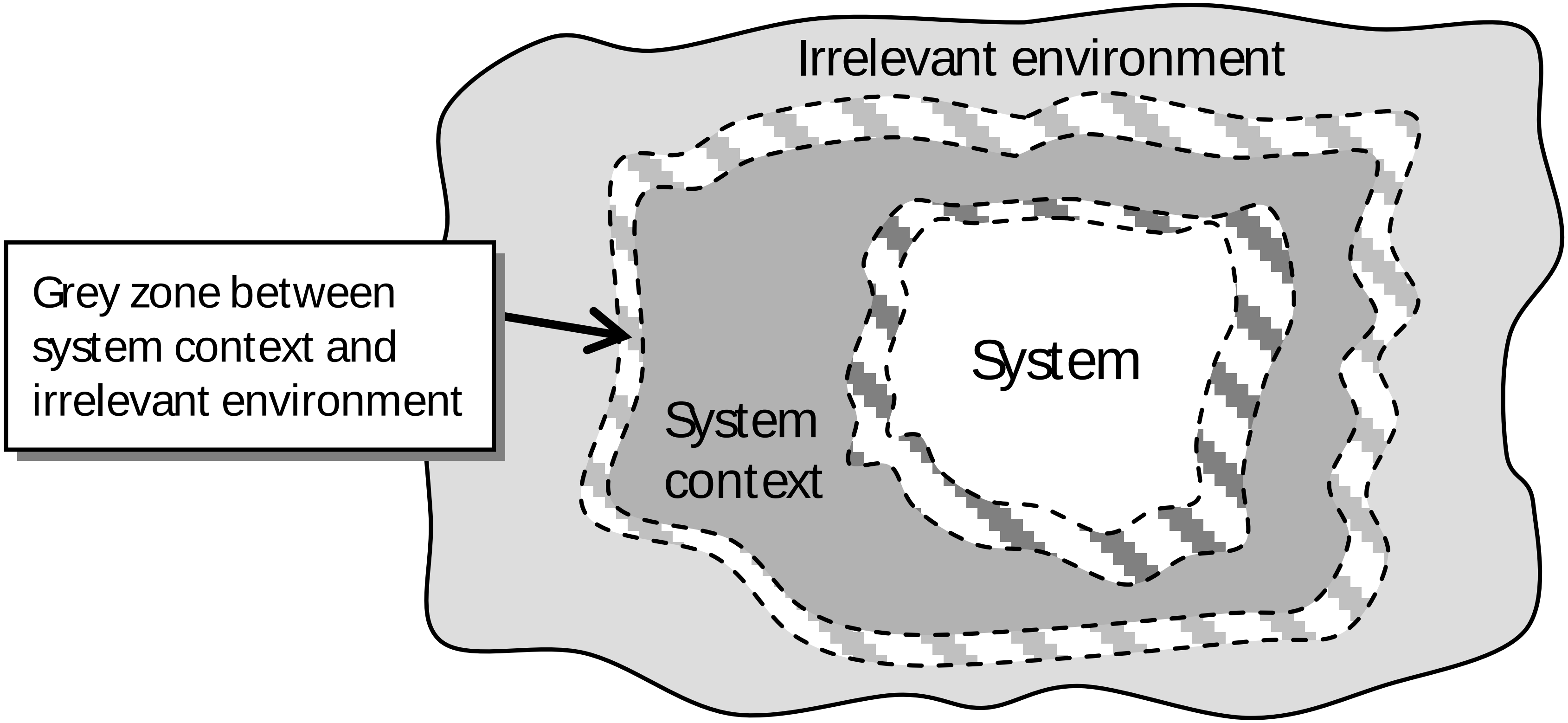
定义系统边界

- 明确哪些方面属于系统；
- 确定哪些方面属于系统边界之外；
- 定义系统边界时应让所有相关的涉众参与其中；
- 设法就系统边界达成一致，当无法确定某一对象是否属于系统时，把其放入到灰色区域中；
- 经常检查已定义的系统边界是否仍然有效，注意系统边界的扩大或缩小；
- 如果系统边界需要进行调整，验证所做的调整是否影响已定义的需求。

5.3 Context Boundary

- Context boundary divides context into
  - Relevant context aspects
  - Irrelevant environment
- Relevant aspects need to be considered during system development
- Beginning of requirments engineering process
- Context boundary definition typically changes during the requirements engineering process
  - Only a fraction of the system environment is known at the beginning
  - Grey zone
  - Knowledge about relevance of certain context aspects changes

The grey zone between the system context and the irrelevant environment



定义上下文边界

- 通过各种上下文刻面的组织模式，逐步将系统上下文和无关环境分离开；
- 将不确定的上下文方面归入灰色区域；
- 若发现某些上下文方面与系统无关，将它们归入无关环境；
- 定义新需求时，检查归入无关的上下文方面是否会变成相关上下文方面；
- 使用目标和场景来检查系统环境中各个方面是否与系统相关，相关的上下文方面至少会影响一个目标或场景；
- 重复执行上述步骤。

5.4 Need to Document Context Aspects

- Requirements depend on the context
- Context information needs to be documented
  - Knowledge about context reduces the risk of wrong interpretations
- Document context information independently of the requirements
  - No redundant documentation
  - Requirements are changed during the development process, context is unchangeable
- Use project-specific guidelines for documenting context information
  - Smaller projects tpically require a smaller amount of context information do be documented

Example

仅带有模糊（几乎没有）上下文信息的需求描述：

- R1：所规划的运输方式应为旅客到达目的地提供便携的行程。
- 上下文信息“将人从海岛运送到大陆，且岛上没有飞机跑道”将显著减少对该需求的误解。

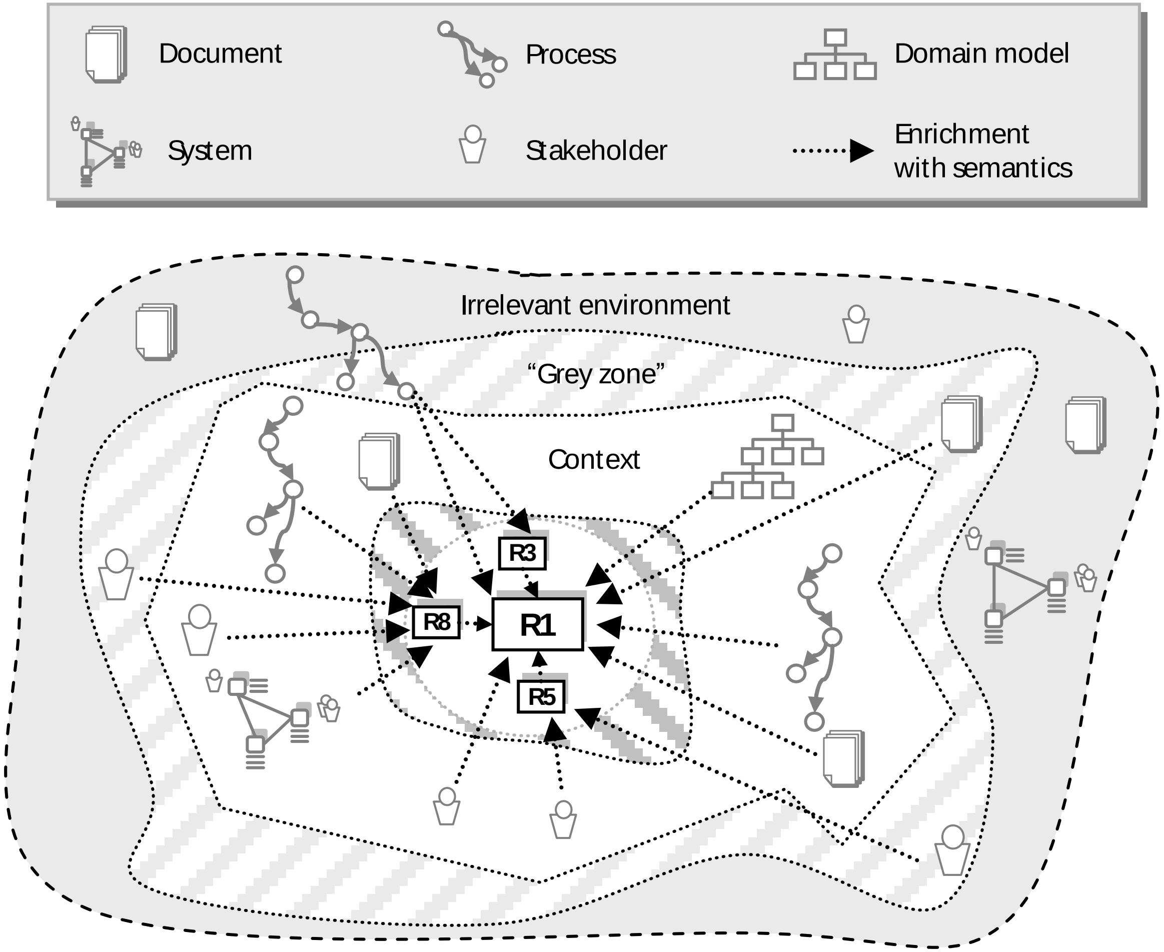
上下文信息直接影响需求：

- 该地是一个没有机场的海岛。

上下文信息间接影响需求：

- R3：海岛上的永久居住的居民应给予车辆轮渡票的折扣。

Context of a requirement

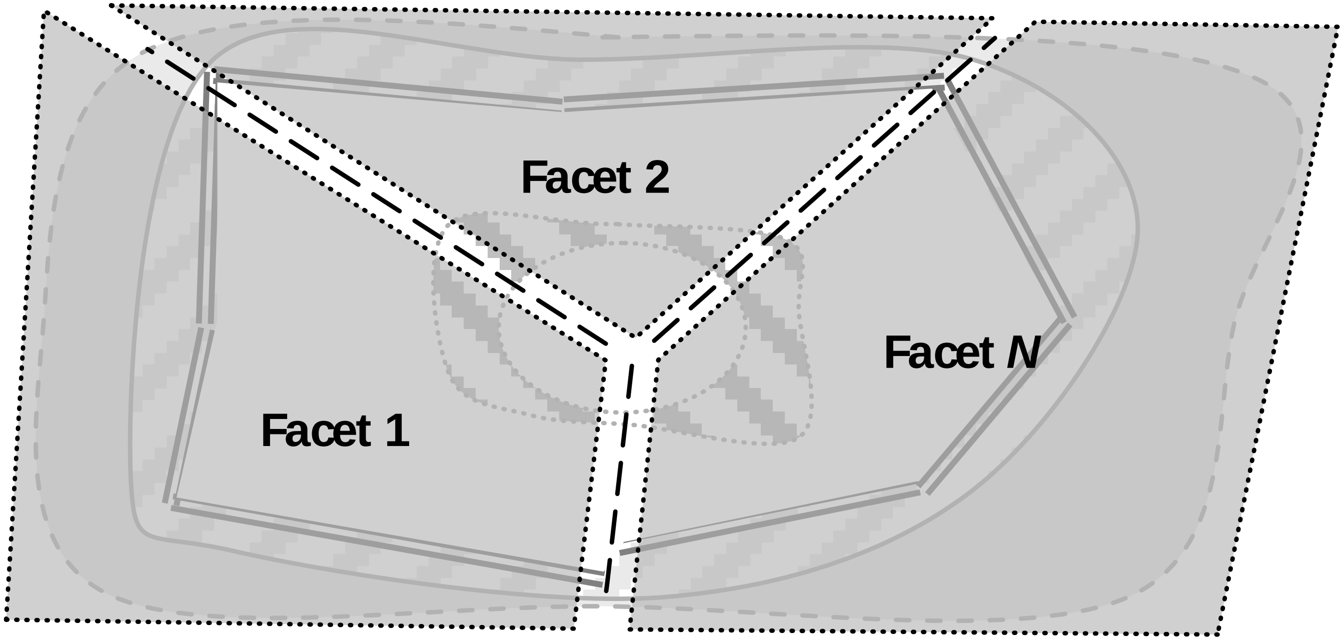


6 STRUCTURING THE SYSTEM CONTEXT

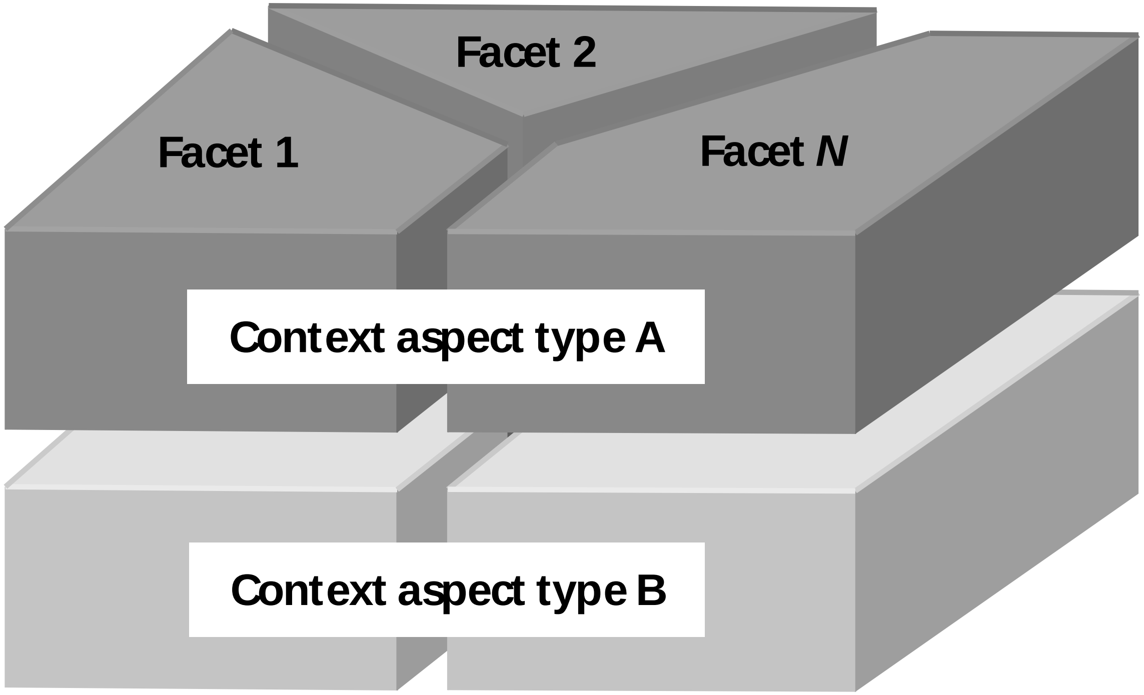
6.1 Structuring Principles

- Divide the context into four different facets
  - Stakeholders can focus on one facet at a time
- Classify the relevant context aspects
  - Context aspects can be addressed individually and systematically

Structuring the context into facets



Classification of context aspects in each facet



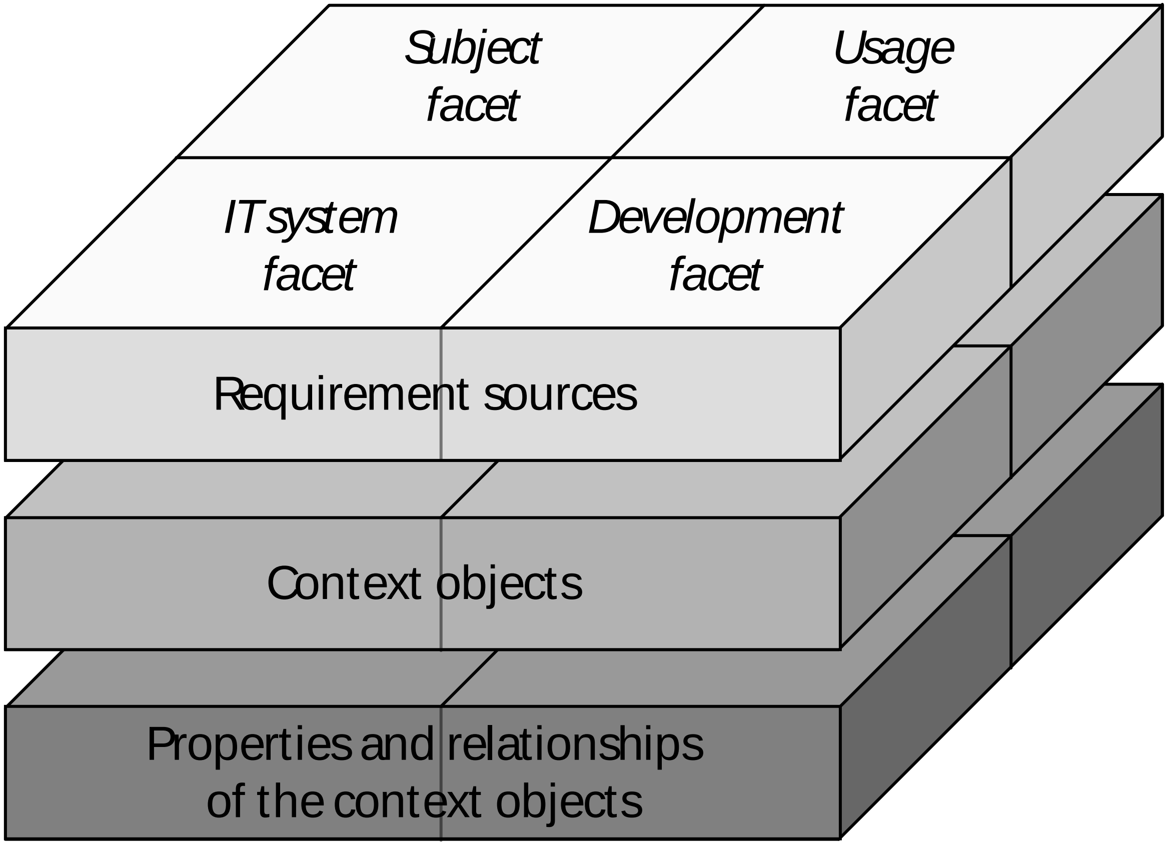
6.2 Four Context Facets and Three Types of Context Aspects

- Subject facet: objects and events that must be represented in the system
- Usage facet: all aspects concerning the usage of the system by people and other systems
- IT system facet
  - All aspects concerning the operational or technical environment
  - Software and hardware components
  - IT strategies and policies
- Development facet: all aspcects influencing the development process of the system

6.2 Four Context Facets and Three Types of Context Aspects (cont‘d)

- Requirement sources:
  - origins of the requirements defined for the systems
  - Stakeholders, existing documentation and existing systems
- Context objects
  - people or objects in a context that need to be involved in or considered during requirements engineering
- Properties and relationships of context objects
  - represent additional information about the context objects and characterise a context object more precisely

Three context aspects and four context facets



Requirement Sources 1/3

定义：Stakeholders

- 涉众是在待开发系统中存在潜在利益的人或组织。
- 涉众对于系统通常有着他们自己的需求。
- 一个人可以代表不同涉众的利益，即有多个角色、代表多个涉众

典型涉众：

- 客户、系统开发者、系统用户、架构师、领域专家、软件开发人员、测试人员、维护人员等
- 隐私保护官员、法律专家、专利代理人、劳工委员会成员等

Requirement Sources 2/3

现有文档

- 市场分析报告
- 客户问询信件
- 遗留系统的需求文档、体系结构文档、源代码、测试文档、用户手册、变更请求报告、错误报告、错误修正报告等
- 业务过程文档
- 技术过程描述
- 法律
- 网站上的产品规格说明
- 保密指南
- 标准
- 其他

Requirement Sources 3/3

现有系统：

- 遗留系统和原有系统
- 竞争对手的系统
- 类似系统

例：汽车安全系统的类似系统

- 为新型汽车安全系统定义需求，通过参加航空领域关于撞击预防系统的一个演示，以期获得相关启发。

如何利用现有系统：

- 复制现有系统的特性
- 改进现有系统的特性
- 避免现有系统已知的缺陷
- 避免原系统中已经被修正的错误

Context Objects

- 人，如用户、客户、管理员等
- 法律主体，如项目开发所涉及的组织机构
- 物质对象，如商品或构成元件
- 非物质对象，如物理变量、数学公式、业务过程、法律或生产过程

Properties and relationships of context objects

例：

- 上下文对象“交通工具”：“当前速度”属性
- 上下文对象“顾客”：“姓名”属性

关系可以存在于相同或不同上下文刻面的对象之间：

- 用户（使用刻面的上下文对象）向系统中输入相关信息（主体刻面的上下文对象）

6.3 Relevant Context Aspects within the Four Context Facets

- Relevance of aspects depends on vision
  - Even for two systems with the same relevant context objects, there may be totally different properties and relationships of these objects
  - But: a generic set of context aspects for each context facet can be identified

6.3 Relevant Context Aspects within the Four Context Facets (cont‘d)

- Subject facet: Requirement Sources
  - Domain experts, relevant stakeholders (e.g. data privacy officers)
  - Models of the subject domain, textbooks (e.g. laws)
  - Existing systems
- Subject facet: Context Objects
  - Persons about which data is stored (e.g. customers)
  - Material objects (e.g. goods), immaterial objects (e.g. temperature)
  - Processes (e.g. production process to be supported by the system)
- Subject facet: Properties and Relationships
  - Relevant properties of the identified context objects
  - Mapping function defining the representation of context objects in the system (e.g. the accuracy of the representation)

Example(汽车安全系统): subject facet

主体刻面的需求来源

- 汽车安全系统领域专家：汽车安全系统应当监控驾驶员的注意力，并采取相应措施来防止驾驶员在驾驶过程中打盹，系统必须了解驾驶员当前的注意力。为了在系统中表示“注意力”这个属性，医生和事故调查员这些领域专家必须参与进来作为需求来源。制动系统和驱动控制系统方面的专家也要参与进来，以明确发现驾驶员在开车打盹的情况下该如何应对。

主体刻面的上下文对象

- 汽车本身以及相关的部件，如引擎、轮胎和刹车
- 汽车使用者，如驾驶员和副驾驶员
- 行驶过程的其他参与者，如前面行驶的车辆和行人
- 汽车所处的外部环境状况，如温度、路况等

Example(汽车安全系统): subject facet

主体刻面的属性和关系

- 上下文对象“驾驶员”：属性“注意力”
- 上下文对象“车胎”：属性“压力”
- 上下文对象“前面行驶的汽车”：属性“距离”

表示的精确性

- 系统应当通过GPS定位汽车当前位置，精度在30米以内（数据精度）。
- 在二维空间内进行定位即可，不需要考虑海拔高度。

表示的现实性

- 至少每秒更新一次汽车当前位置。

表示的法律要求

- 基于安全和可控的考虑，法律规定每辆车的当前位置和速度必须每隔10秒进行一次记录并至少保存2个月，记录的位置分辨率要达到50-100米，记录的速度误差小于3公里/小时。

6.3 Relevant Context Aspects within the Four Context Facets (cont‘d)

- Usage facet: Requirement Sources
  - Direct users, indirect users, experts for user interfaces
  - Standards, laws and regulations for user interfaces, domain models in the usage domain
  - Existing systems useed in the same or a similar way
- Usage facet: Context Objects
  - User groups
  - Input modality of the user interface, usage workflows
  - Usage by other systems
- Usage facet: Properties and Relationships
  - Relationship between usage workflows, between user groups and usage workflows
  - Relationship to the IT system facet and the subject facet

Example(汽车安全系统): usage facet

使用刻面的需求来源

- 汽车安全系统中的直接用户和间接用户：驾驶员是直接用户之一。副驾驶员是间接用户，虽然不直接控制汽车，但会影响驾驶员的决定，例如提示驾驶员注意警告信息。
- 标准和法规：欧洲汽车人机接口设计原则目录中规定“系统设计应当避免分散驾驶员的注意力，同时也不允许向驾驶员提供任何形式的视觉娱乐”。
- 类似系统：一个工作组在为2015年投产的汽车新型人机接口设计进行构想，为获得启发，他们参加了一次展示新型人机接口的未来飞机驾驶的仿真活动。

使用刻面的上下文对象

- 汽车的用户组：区分运动型驾驶员和安全导向型驾驶员的差异是很重要的，系统应该可以为这两种驾驶员作出相应调整，避免向运动型驾驶员提供过多警告以及向安全导向型驾驶员提供太晚的警告和干预。
- 汽车安全系统的人机接口：当前使用经典的显示面板向驾驶员提供提示信息。通过投影装置把信息投影到挡风玻璃的系统正在测试，计划下一年投入生产。设计人员必须同时考虑这两种显示方式，这将引入额外的需求。

Example(汽车安全系统): usage facet

使用刻面中的属性和关系

- 汽车安全系统的关闭：系统使用的一个重要方面是驾驶员可能关闭系统，关闭系统的人机交互是使用刻面中的一个上下文对象。另一种系统关闭可以在IT系统刻面中识别，因为系统失效而导致系统自动关闭。因此，使用刻面和IT系统刻面中的上下文对象（及属性）之间存在关联。
- 和主体刻面的关联关系：驾驶员必须能够自然地将所提示的警告信息和显示世界中的某种具体威胁联系在一起。例如，威胁“与前车距离小于安全距离”和上下文对象“前车”相关。这样，使用刻面和主体刻面的关系就建立起来了。

6.3 Relevant Context Aspects within the Four Context Facets (cont‘d)

- IT system facet: Requirement Sources
  - Persons who deal with planning design and operation of the IT system environment, technology consultants, suppliers
  - IT infrastructure documents, reference architecture
  - Analysis of existing system designs
- IT system facet: Context Objects
  - Hardware and software components to be considered during system development,
  - operation and maintainance of the system to be developed
  - IT strategies of the involved companies
- IT system facet: Properties and Relationships
  - Technical data of hardware and software, operation profiles, update policies

Example(汽车安全系统): IT system facet

IT系统刻面中的需求来源

- IT系统刻面中的涉众：对于汽车安全系统，所有相关的电子器件（如传感器、执行器、刹车控制单元和安全气囊控制单元）之间准确协作非常重要。因此，IT系统刻面的涉众包括这些器件的开发人员以及汽车使用的通信协议的专家。
- 分析现有系统：为了分析一个移动通信设备终端操作系统的需求，涉众分析已有的通信终端设备的IT系统体系结构以及这些设备的操作系统属性。分析结果被用来识别和定义待开发系统的需求。

IT系统刻面中的上下文对象

- 汽车中的软硬件部件
  - 车内传感器和执行器，例如车轮速度传感器、加速传感器等
  - 车内的总线系统，例如CAN、LIN、MOST、FlexRay等
  - 操作系统，例如汽车工业常用的嵌入式操作系统
- IT策略，例如系统必须遵循AUTOSAR体系结构标准

IT系统刻面中的属性和关系

- 相关软硬件组件的技术性能、成本、可用性、成约人、供应商的责任义务等

6.3 Relevant Context Aspects within the Four Context Facets (cont‘d)

- Development facet: Requirements Sources
  - Process stakeholders, like process engineers, process managers and process executors (e.g. requirements engineers, architects)
  - Guidelines for the development process
- Development facet: Context Objects, Properties and Relationships
  - Role definitions, artefact definitions, activity definitions, tools, resource availability and resource restrictions

6.4 Different Roles of a Context Aspect

- A specific context aspect may be relevant for more than one context facet
- Differentiation is particularly helpful in such cases, it assures that all relevant aspects are considered

Example：一个上下文方面作为多个刻面需求来源

用户提供主体刻面和使用刻面的信息

- 需求工程师针对一个保险公司软件系统开发与保险代理进行访谈。保险代理描述了其典型工作流程，提供了输入到系统中的用户数据的详细信息。此外，他还给出工作流程的一些可能改进措施。
- 该保险代理对于主体刻面（如客户信息）和使用刻面（如工作流程）都提供了信息。

一份用户手册包含所有4个上下文刻面的信息

- 其中一章描述原有系统的软硬件环境，即IT系统刻面的信息
- 有一章提供系统数据库的详细说明，包含了主体刻面的信息
- 有一章描述了开发用户自定义的系统扩展时必须考虑的约束，如编程语言和支持的编译器等，因此包含了开发刻面的信息
- 有一章描述了用户接口的详细信息，因此包含了使用刻面的信息

Example：一个上下文方面作为多个刻面上下文对象

传感器同时作为主体刻面和IT系统刻面的上下文对象

- 汽车安全系统应当根据传感器提供的数据侦测到震动，为此必须对传感器接口和通信能力进行考虑。因此，传感器是IT系统刻面的上下文对象。
- 汽车安全系统还会检查传感器是否正常工作，因此，传感器必须在软件中进行某些属性的设定和周期性的更新，它也是主体刻面的相关上下文对象。
- 应该分别分析这些角色，避免出现分属不同刻面的传感器属性和关系的混杂，以及在这两个刻面中对传感器角色的不完整的考虑。

The end, thanks!