UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

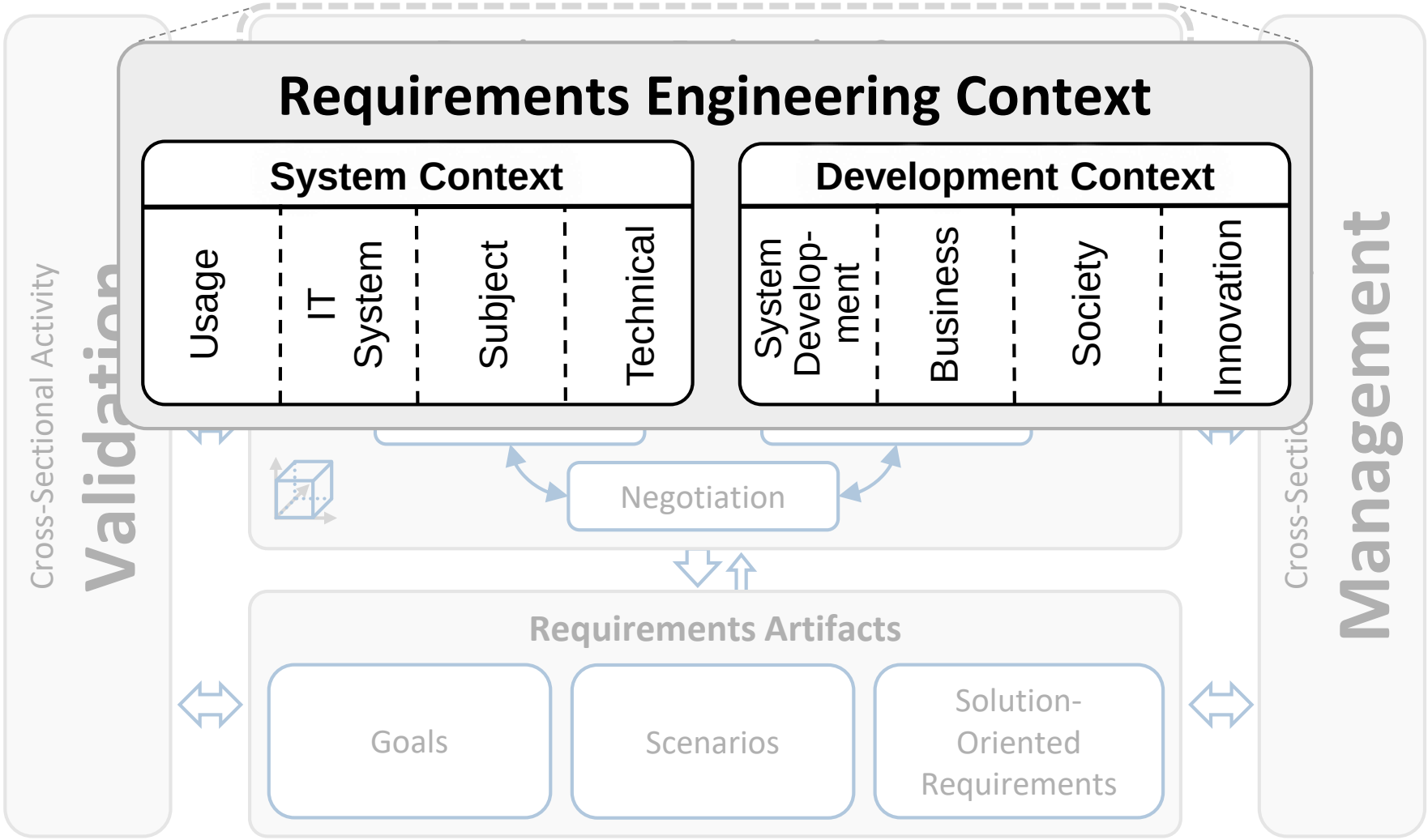
Requirements Engineering & Management

Context I – Foundations of Context Consideration

Prof. Dr. Klaus Pohl

PALUNO
The Ruhr Institute for Software Technology

Framework for Requirements Engineering



Agenda

1. Why Context Matters
2. The Requirements Engineering Context
3. Context Consideration in Requirements Engineering Activities
4. Context Scope and Decomposition of Context Information
5. Context and Scope of Development

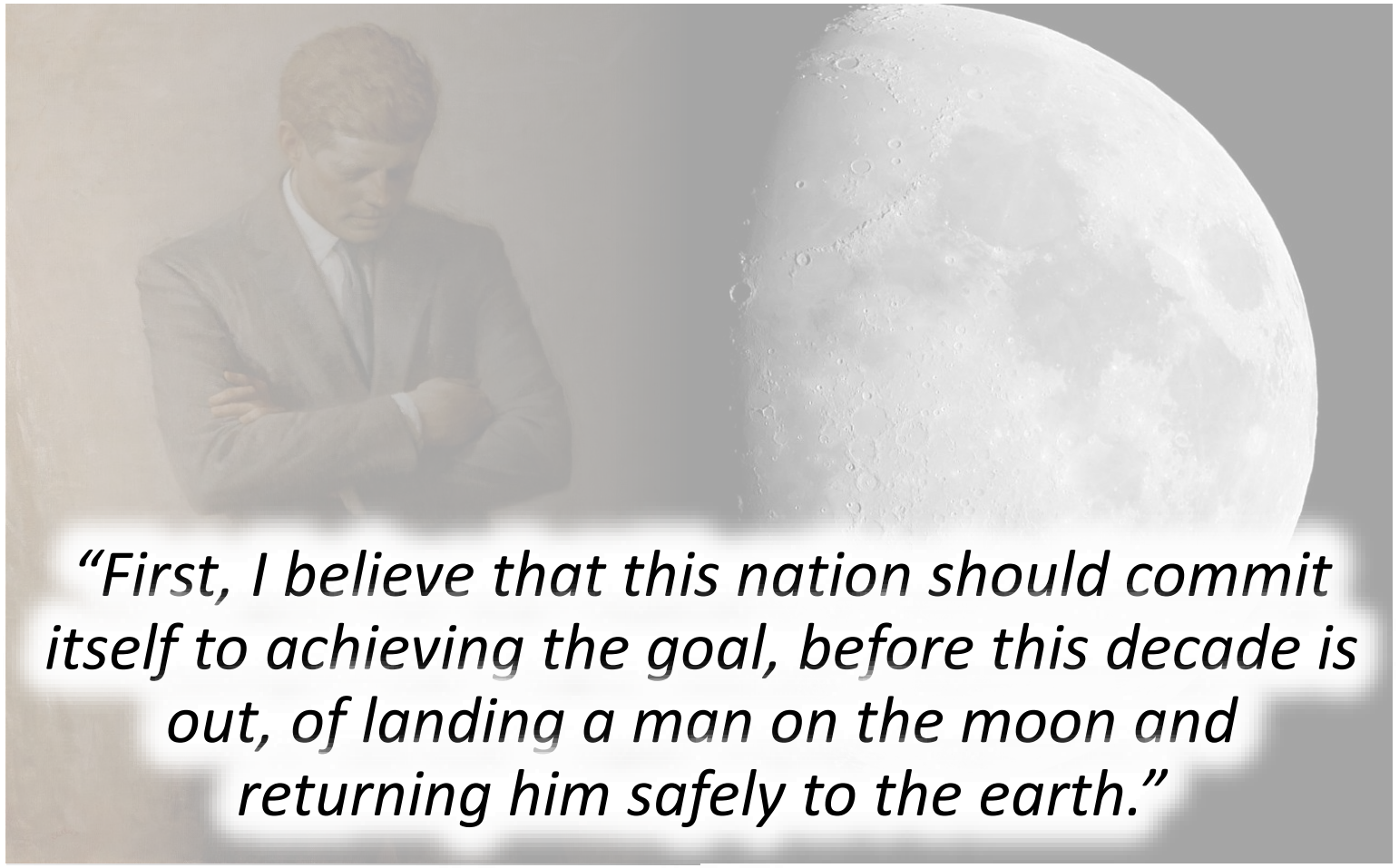
1. Why Context Matters

Context of a System

- The context has a **strong impact on the requirements** defined for a system to be developed!
- Inadequate consideration of the context (e.g., ignoring, overlooking, or misconceiving aspects of the context) → **requirements defects**!
- Without knowing the context, system requirements can neither be **correctly defined** nor **correctly interpreted**!

- System **vision** determines **initial framing** of the **context** to be considered for developing a system realising the vision
- Vision is refined into requirements based on information about the context

Establishing a Vision

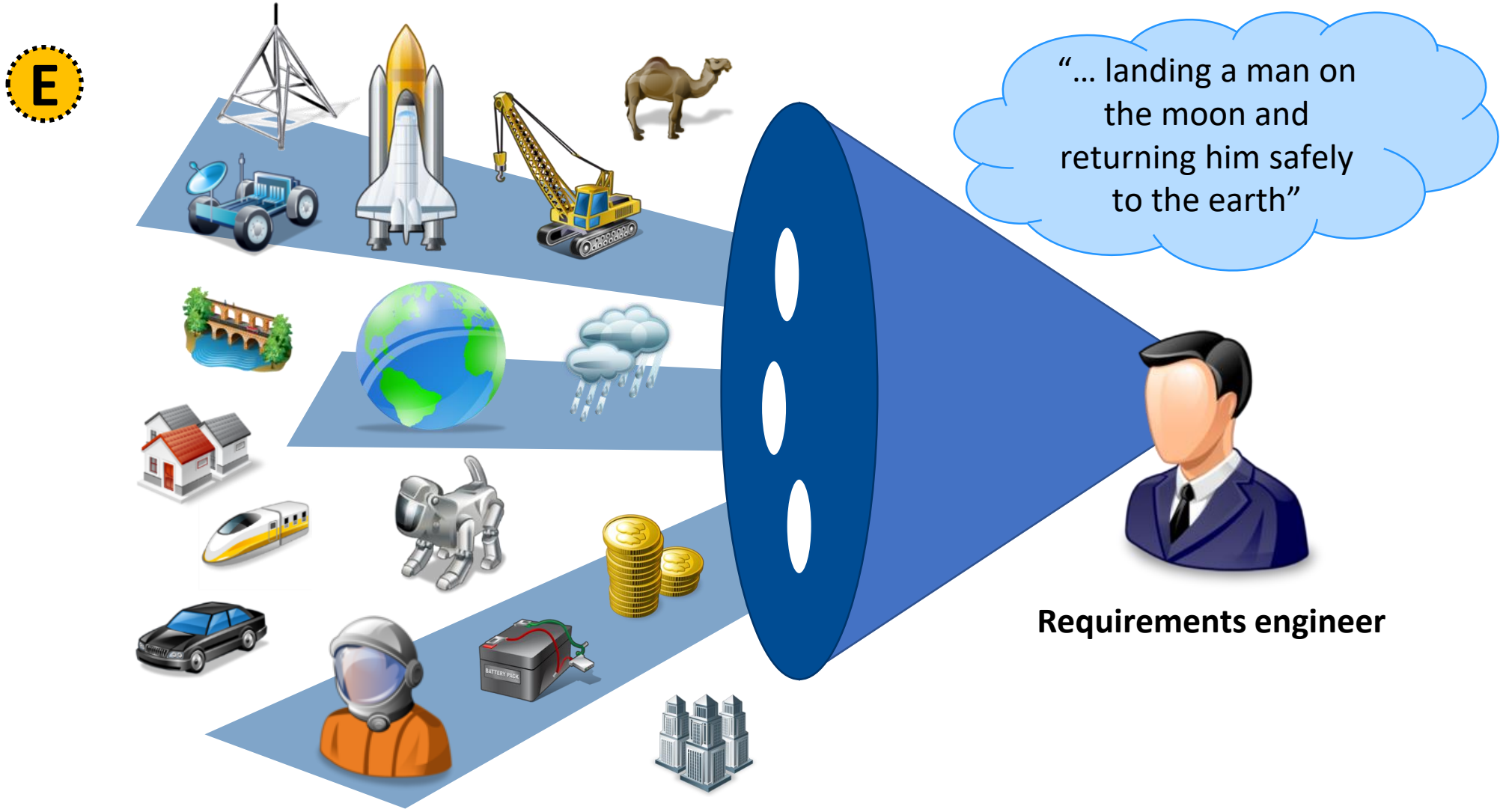


Vision of J.F. Kennedy in 1961

[Dudley 2000]

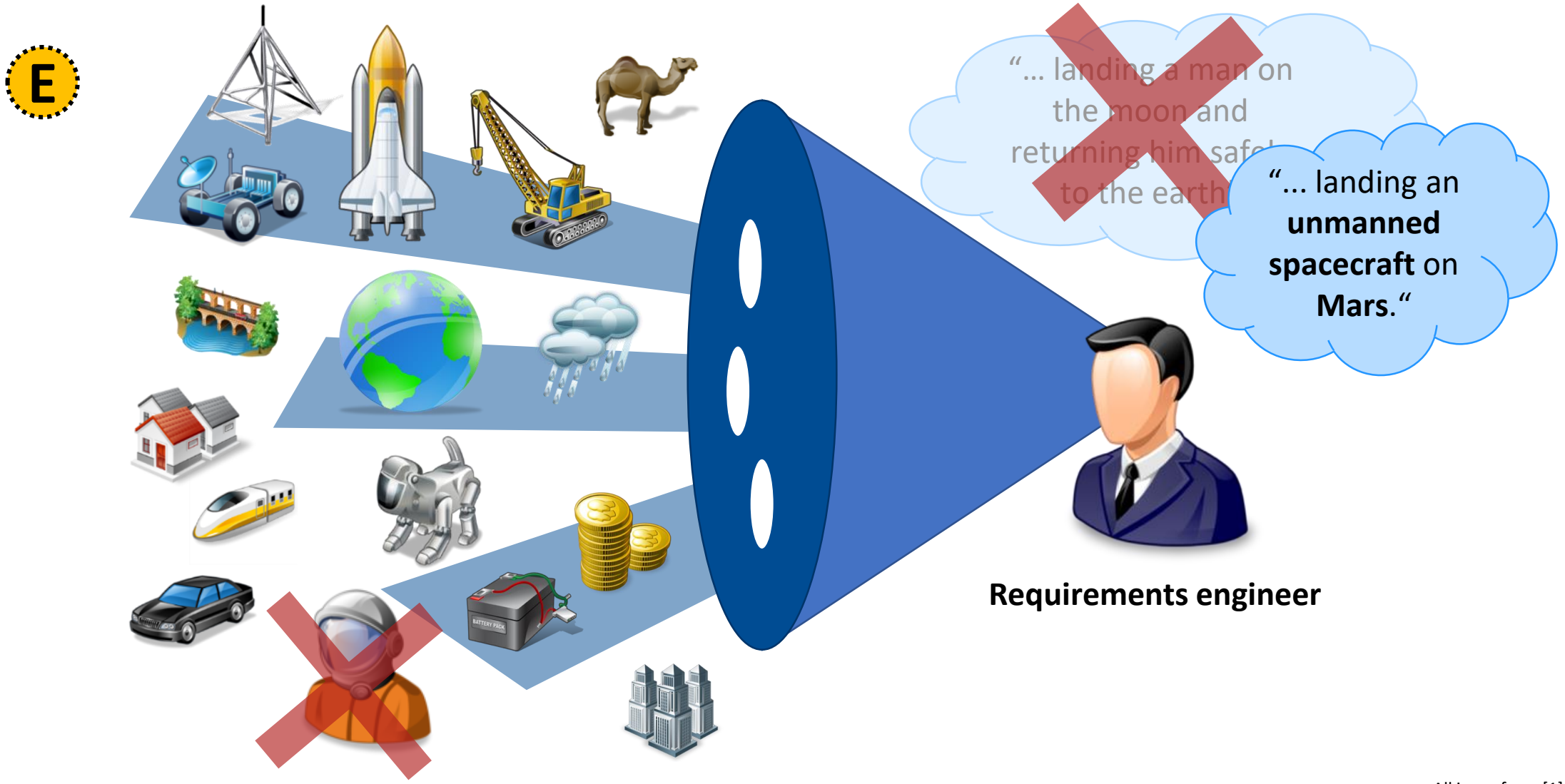
Pictures from [2, 3]

Vision Guides Initial Context Consideration



All icons from [1]

Vision Determines the Relevant Context (1)



Context Impact on Requirements

- A given **context** ...
  - Determines which **solutions** are **appropriate** for realising the vision and requirements

Example: Power Supply (1)

E **Vision**: Develop a power supply based on renewable energies.

Possible solution: Wind turbine

Context 1: Family home in a countryside

→ Appropriate and **good solution**, provided there is enough wind in the area

All icons from [1]

Example: Power Supply (2)

E **Vision** : Develop a power supply based on renewable energies.

Possible solution: Wind turbine

Context 2: Submarine under water

→ **Inappropriate solution**
for use under water, even if a battery is charged on land

All icons from [1]

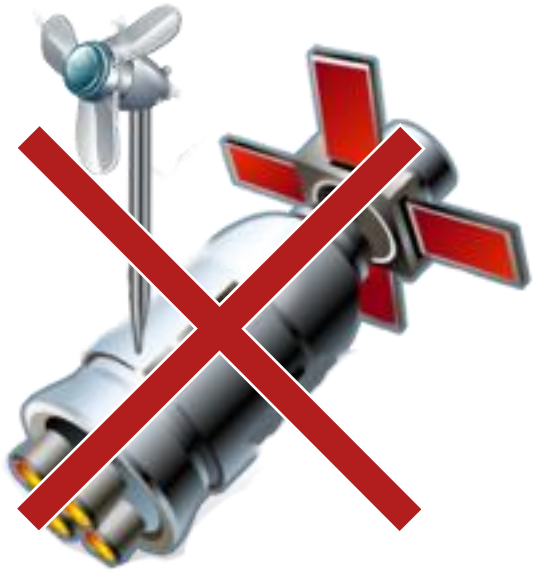
Example: Power Supply (3)

E **Vision** : Develop a power supply based on renewable energies.

Possible solution: Wind turbine

Context 3: Satellite in space

→ **Inappropriate (even infeasible solution)** due to physical laws (no wind in space)



All icons from [1]

Context Impact on Requirements

- A given **context** ...
  - Determines which **solutions** are **appropriate** for realising the vision and requirements
  - Can be successively **refined** by gathering more information, which gradually **constrains the solution space**.

Example: Transportation System (1)

"Establish a fast and safe transportation for people!"

All icons from [1]

Example: Transportation System (2)

"Establish a fast and safe transportation for people!"

"The transportation should be between the mainland and an island."

Potential solutions







"The island is located 5 km from the mainland."

"The island does not have an airfield."

All icons from [1]

Example: Transportation System (3)

"Establish a fast and safe transportation for people!"

E

"The transportation should be between the mainland and an island."

"The island has about 35 inhabitants."

"The island is no major tourist destination."

Potential solutions







"The island is located 5 km from the mainland."

"The island does not have an airfield."

All icons from [1]

Context Impact on Requirements

- A given **context** ...
  - Determines which **solutions** are **appropriate** for realising the vision and requirements
  - Can be successively **refined** by gathering more information, which gradually **constrains the solution space**.
  - Determines which **requirements** are **valid**

Example: Accounting System (1)

E **Vision**: Develop an accounting system

Context 1: Manufacturing industry in Germany

In this context we have to consider:

- relevant **German laws**
- business needs of **manufacturing companies**

All icons from [1]

Example: Accounting System (2)

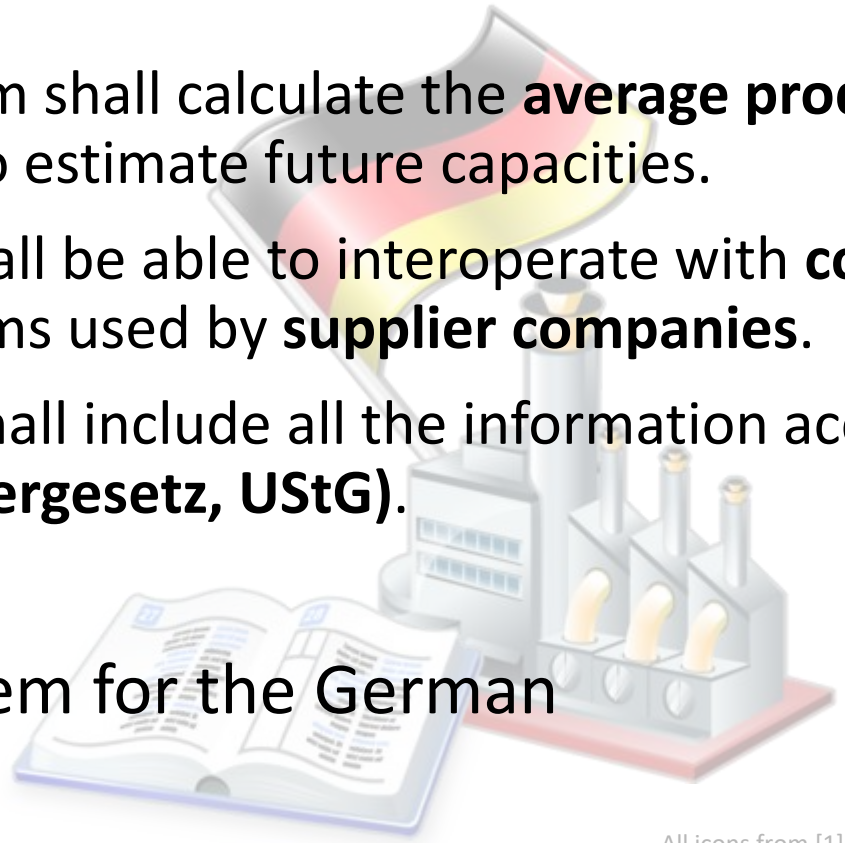
The **context impacts requirements**!

E

Functional-Req-121: The accounting system shall calculate the **average production capacity** utilisation over the last quarter to estimate future capacities.

Quality-Req-21: The accounting system shall be able to interoperate with **common Enterprise Resource Planning (ERP)** systems used by **supplier companies**.

Constraint-2: The generated **tax reports** shall include all the information according to the **German Sales Tax Act (Umsatzsteuergesetz, UStG)**.



→ Only **valid** for an accounting system for the German manufacturing industry

All icons from [1]

Example: Accounting System (3)

E **Vision**: Develop an accounting system.

Context 2: Financial companies in the USA

In this context we have to consider:

- relevant **US American laws**
- business needs of **financial companies**.

All icons from [1]

Example: Accounting System (4)

E **Vision**: Develop an accounting system.

Context 3: Shipping company in China

In this context we have to consider:

- relevant **Chinese laws**
- business needs of **shipping companies**.

All icons from [1]

Context Impact on Requirements

- A given **context** ...
  - Determines which **solutions** are **appropriate** for realising the vision and requirements
  - Can be successively **refined** by gathering more information, which gradually **constrains the solution space**.
  - Determines which **requirements** are **valid**
  - Influences how **requirements** are **interpreted** (i.e., how requirements are **refined** and **realised**)

Example: User Authentication (1)

Functional-Req-42: The system **shall authenticate the user** before he or she is able to access the system.

Context 1: Building access

- Rationale: Security issues w.r.t. physical entry
- Possible impact on realisation:
  - Physical locking mechanisms needed
  - No need to uniquely identify each individual person

All icons from [1]

Example: User Authentication (2)

Functional-Req-42: The system **shall authenticate the user** before he or she is able to access the system.

Context 1: **Point of sale terminal**

- Rationale: Theft prevention, tracking of employee performance
- Additional context information:
  - Each employee already has a unique ID card
  - Quickly serving the customers is a key goal documented in the business strategy
- Possible Impact on realisation:
  - Employee ID card might be used for authentication
  - ID card scanner suitable for quick user authentication

All icons from [1]

Example: User Authentication (3)

Functional-Req-42: The system **shall authenticate the user** before he or she is able to access the system.

Context 1: **Driver assistance system**

- Rationale: Select personal driving preferences
- Additional context information:
  - Driver must not be distracted from driving
  - Sleepiness of the driver can cause fatal accidents
- Possible Impact on realisation:
  - Visual face detection might be a suitable technology to also detect sleepiness while driving

All icons from [1]

Context Impact on Requirements – Summary

- A given **context** ...
  - Determines which **solutions** are **appropriate** for realising the vision and requirements
  - Can be successively **refined** by gathering more information, which gradually **constrains the solution space**.
  - Determines which **requirements** are **valid**
  - Influences how **requirements** are **interpreted** (i.e., how requirements are **refined** and **realised**)

2. The Requirements Engineering Context

Context Objects

D A real world object is a **tangible** or **intangible** object, which either **exists in the current real world** or is **assumed to exist in the future real world**.

D A context object is a real world object, which is **relevant for system development** and thus has to be **considered in requirements engineering**.

- Context objects are **characterised** by:
  - **Properties** (e.g., age, length, height,...)
  - **Relationships** with other objects (e.g., a vehicle can be owned and driven by a human)

Context Information (1)

D Context information are elicited **information about the context** relevant for requirements engineering.

- All **requirements engineering activities** depend on context information!
- Context information comprises, among others:
  - Information about **context objects** (e.g., availability of stakeholders)
  - **Physical laws** (e.g., gravity or electrical laws)
  - **Assumptions** (e.g., expected user behaviour, workload etc.)
  - Other information **relevant for requirements engineering and system development** (e.g., about development budget)

Examples of context information: see slides 5-25!

Context Information (2)

- Context information has to be elicited from **information sources**.
- These information sources are **context objects**!
  - Stakeholders, documents, existing systems
- Two specific **kinds of context information**:
  - **Context assumptions**: Context information expected to hold with a certain degree of **uncertainty** (i.e., there is **no proof/sufficient evidence** that the described subject is in fact valid, or holds as expected)
  - **Context constraints**: Context information expected to hold **without uncertainty** (i.e., information **taken for granted** and valid)

All kinds of context information need to be validated! (see Lecture L-19 Validation)

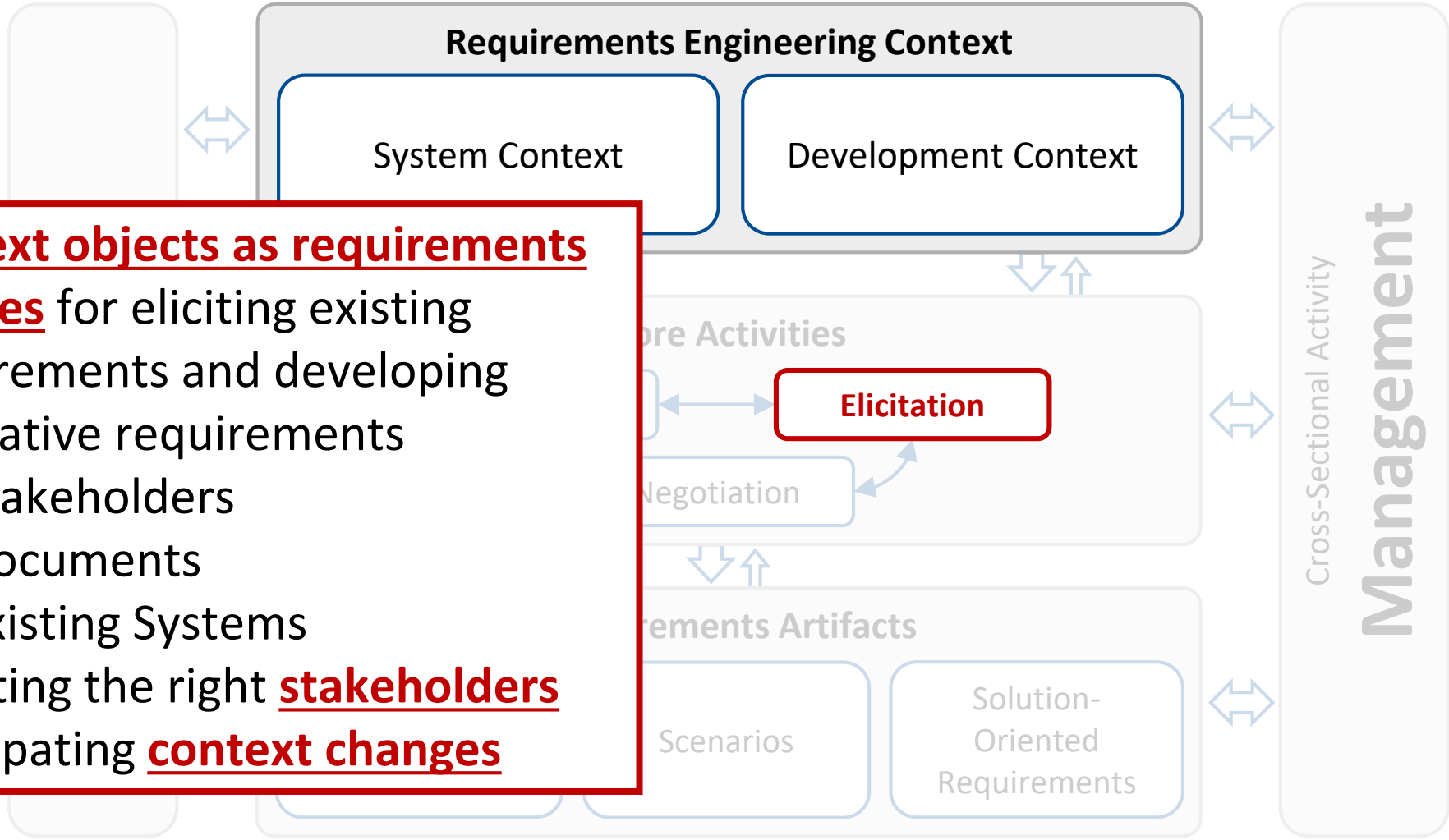
Requirements Engineering Context: Definition

D The **requirements engineering context** subsumes all **context objects** and **context information** relevant for requirements engineering.

- "*relevant for requirements engineering*" = relevant for **achieving the three essential goals** of requirements engineering!
  - **Not** only elicitation and definition of requirements!
  - However, there typically is also a high amount of **irrelevant information**.
- Context consideration is crucial **not only in requirements engineering**, but also in **other development activities** (e.g., design and testing)!

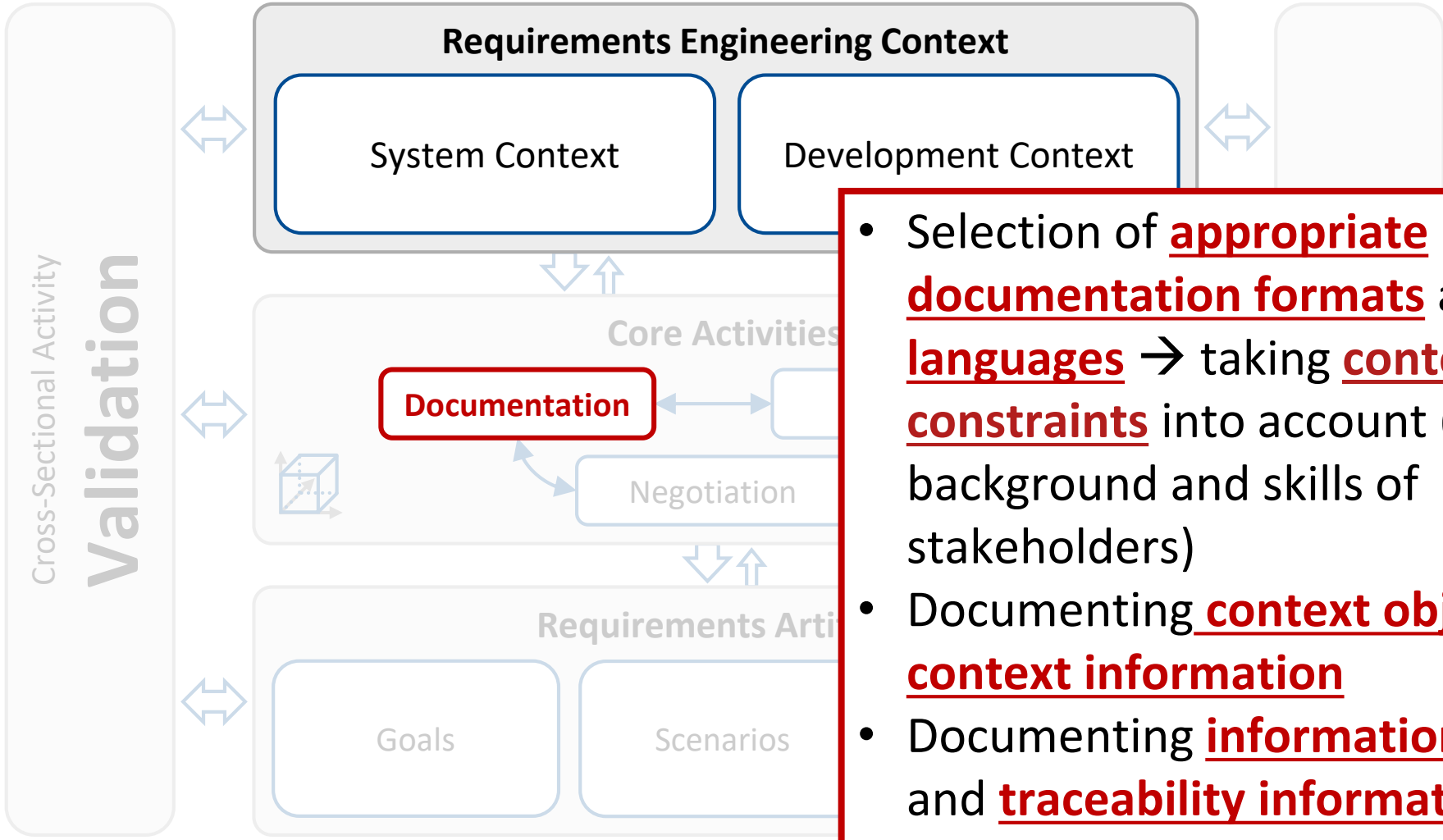
3. Context Consideration in Requirements Engineering Activities

Consideration in Elicitation



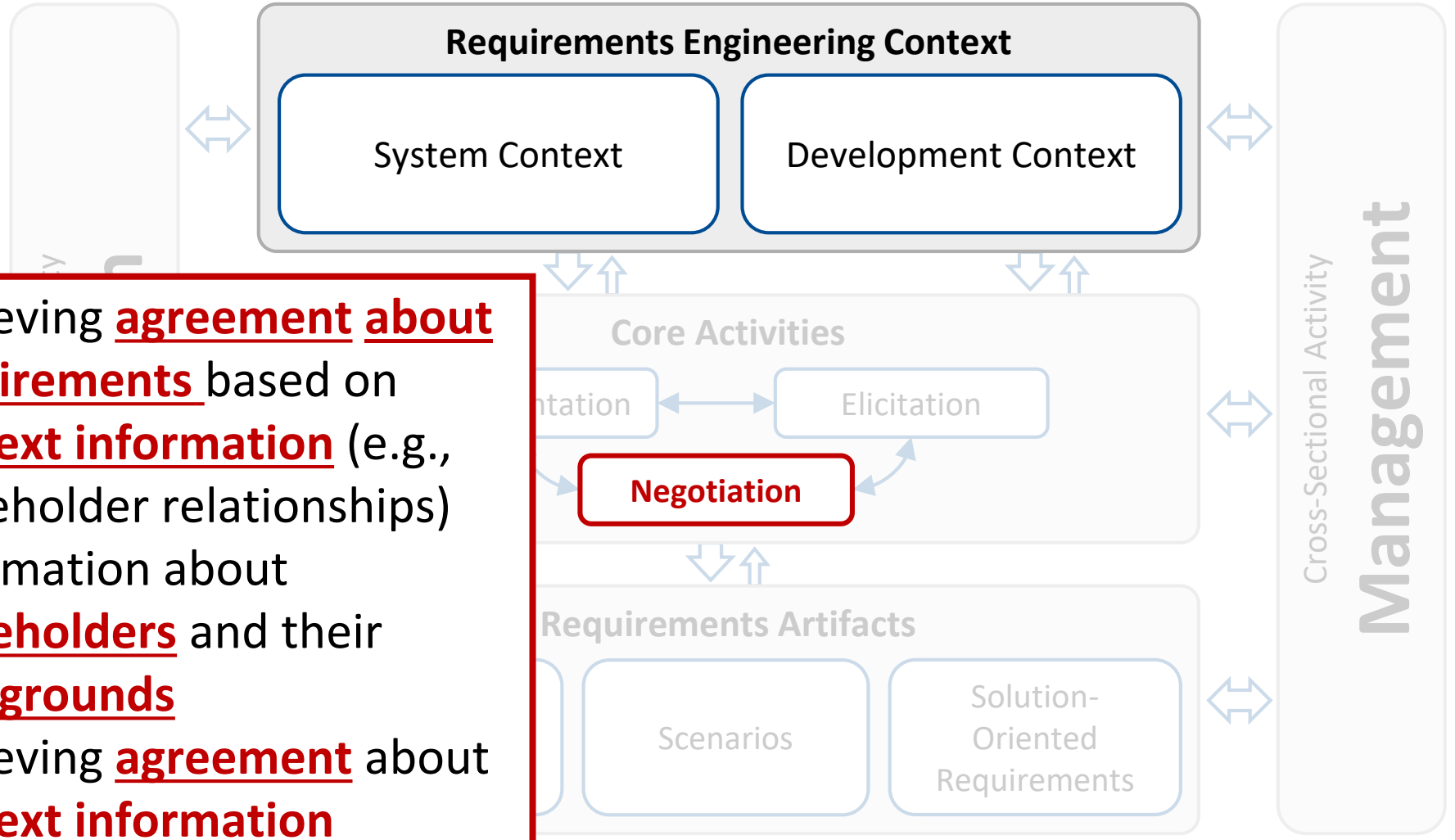
- **Context objects as requirements sources** for eliciting existing requirements and developing innovative requirements
  - Stakeholders
  - Documents
  - Existing Systems
- Selecting the right **stakeholders**
- Anticipating **context changes**

Consideration in Documentation



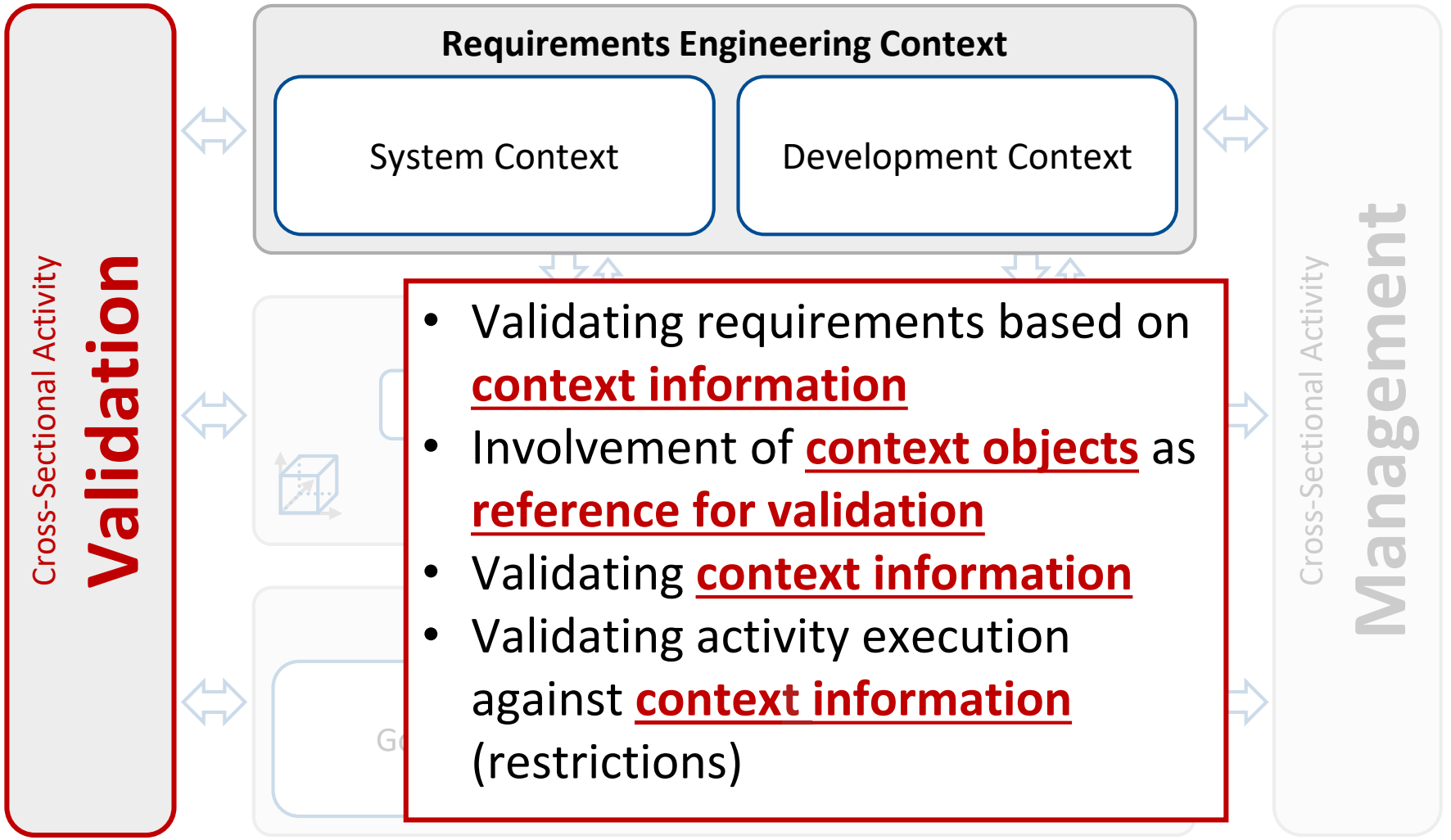
- Selection of **appropriate documentation formats** and **languages** → taking **context constraints** into account (e.g., background and skills of stakeholders)
- Documenting **context objects** and **context information**
- Documenting **information sources** and **traceability information**

Consideration in Negotiation

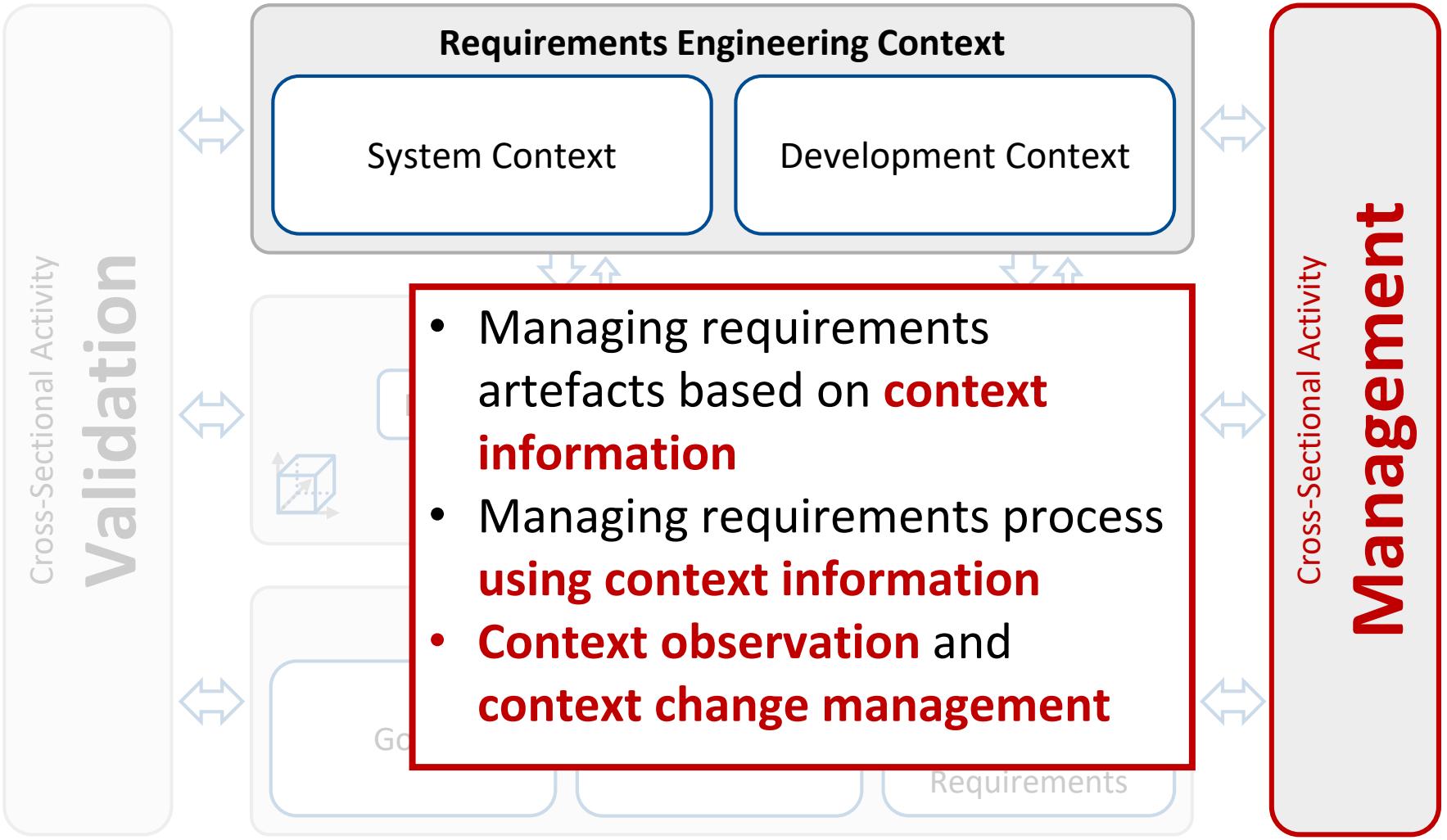


- Achieving **agreement about requirements** based on **context information** (e.g., stakeholder relationships)
- Information about **stakeholders** and their **backgrounds**
- Achieving **agreement** about **context information**

Consideration in Validation



Consideration in Management



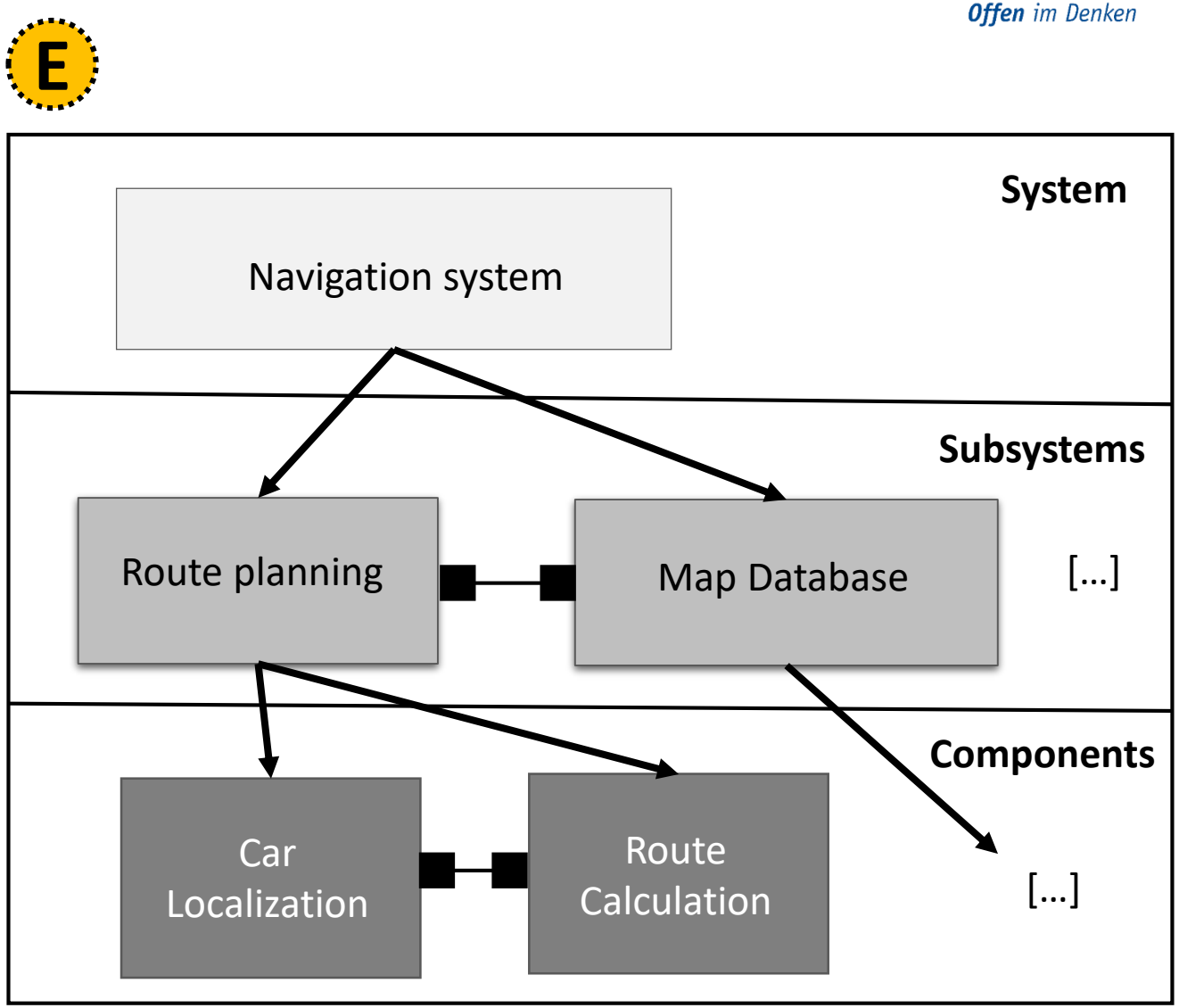
- Managing requirements artefacts based on **context information**
- Managing requirements process **using context information**
- **Context observation** and **context change management**

4. Context Scope and Decomposition of Context Information

System Decomposition

- To handle complexity, the system is typically **successively decomposed** and considered on different layers of **granularity**:
  - **Overall system**
  - **Subsystems**
  - **Components**

→ „Divide and conquer“ principle



Subsystems and Components as Context Objects

- **Focus** on the development a **constituent part of the system** (subsystem or component) determins **context scope**
  - → Considering **other parts** of the same system as **context objects**
- System parts **cannot be considered in isolation**
  - → Requirements of these other system parts = **context information**!
- System parts **interact** with each other via **interfaces**
  - **Context information** about interfaces of other subsystems and components is needed
  - Without context information about other system parts and their interfaces **requirements for subsystems/components cannot be defined!**

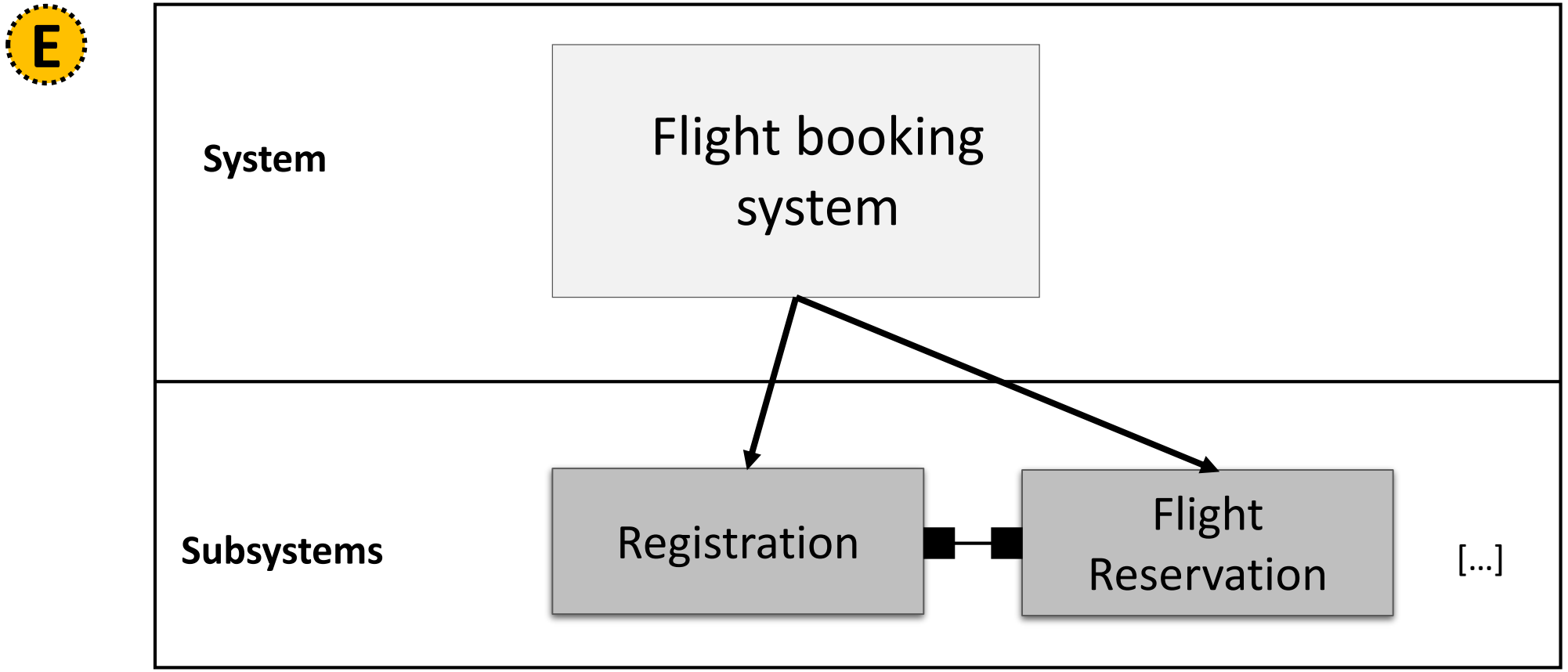
Context of Subsystems and Components (1)

- Each constituent part of the system is **subject of a dedicated development** → Deliberate **context consideration for each part**
- Context information needed to define and understand requirements differs on **different levels of granularity**
- Requirements are **successively refined** on lower levels of granularity, and **so is the respective context information**!

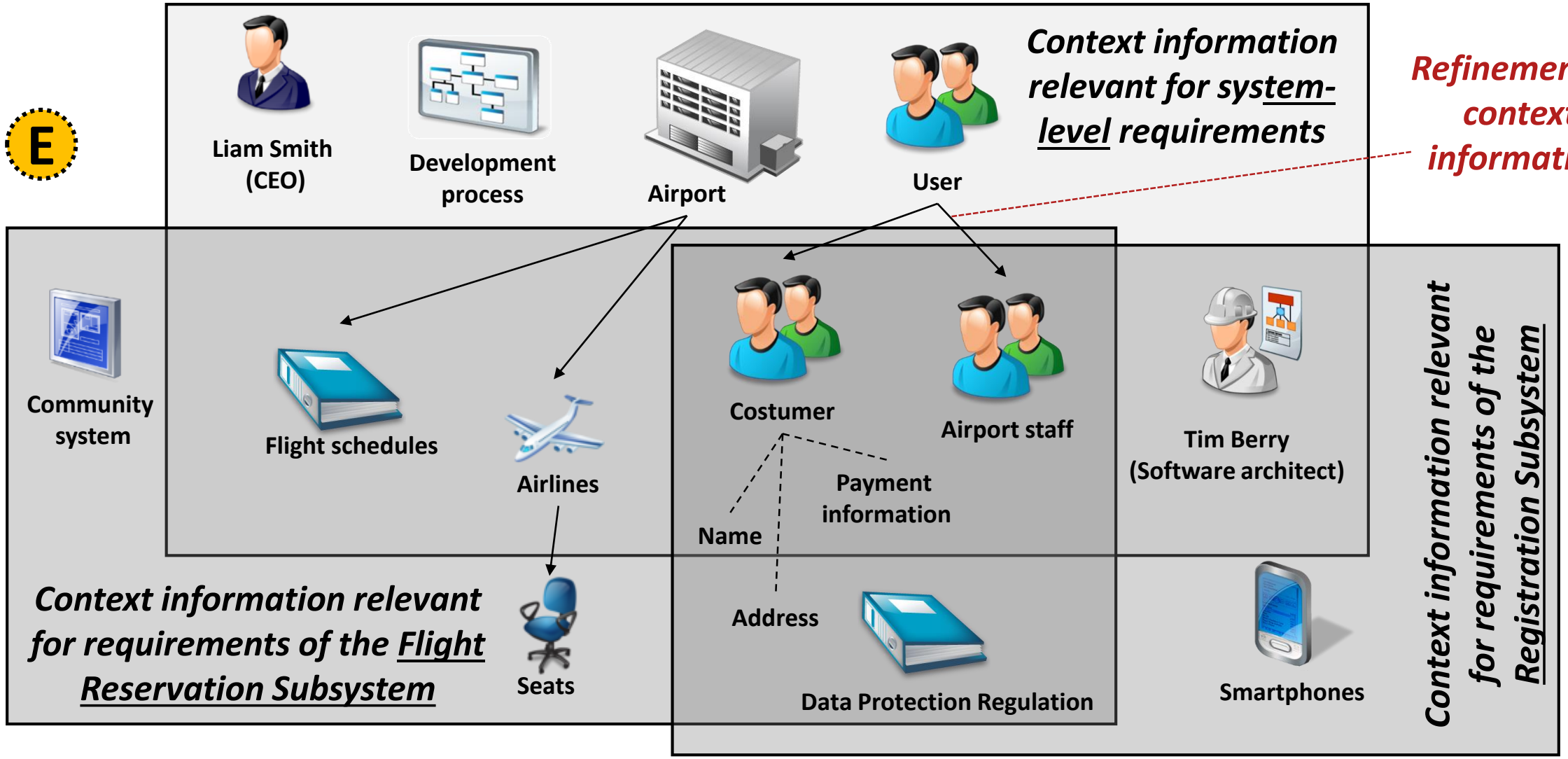
Context of Subsystems and Components (2)

- **Impacts of context consideration** on different layers of granularity:
  - **Refinement of context objects and context information** necessary to specify/refine requirements for a specific part of the system
  - **New context objects and context information**, which is specifically relevant only for a specific system part (and not for the more abstract consideration of the overall system)
  - **Irrelevant context objects/context information**, which do not need to be considered for developing a specific system part

Flight Booking System: System Decomposition



Flight Booking System: Context Decomposition



All icons from [1]

5. Scope of Development

System Boundary: Traditional Conception

D

"The system boundary is **the boundary between a system and its surrounding context**.

[…]

The system boundary delimits the **system as it shall be after its implementation and deployment**.

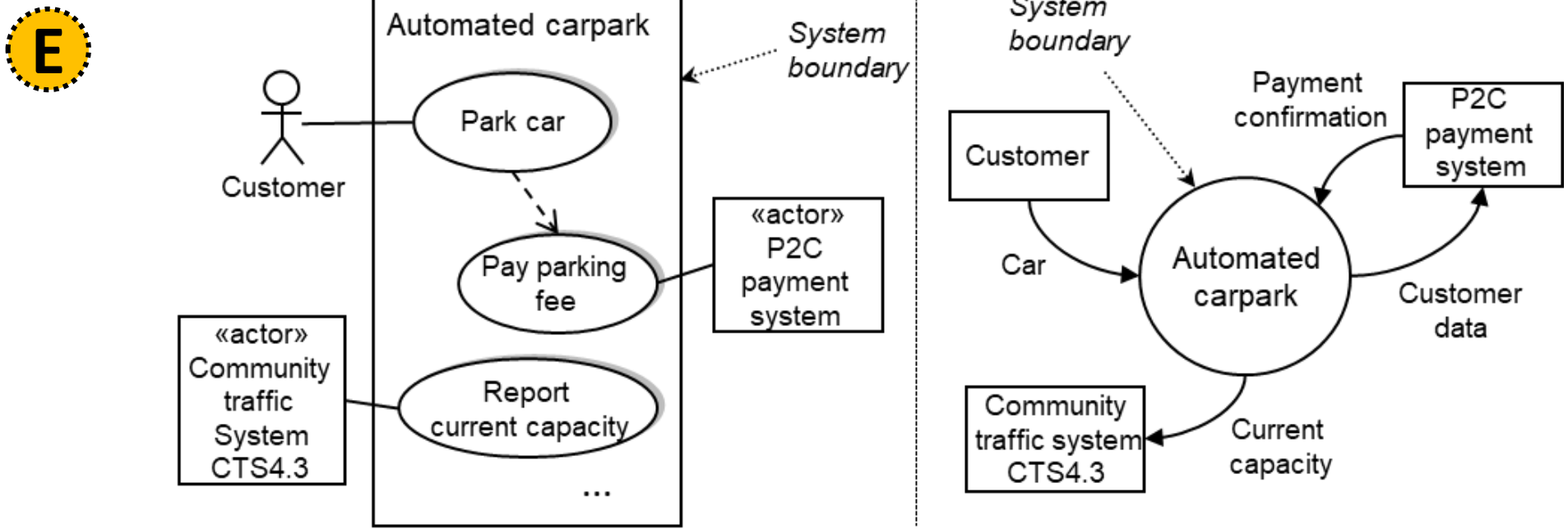
At the system boundary, the **external interfaces** between the system and its context have to be defined."

[…]

[Glinz 2020, p. 21]

System Boundary in Requirements Models

Modeling languages for requirements artifacts **explicitly** or **implicitly capture the system boundary**



UML Use Case Diagram **Data Flow Diagram**

Scope of Development: Changeable vs. Unchangeable Real World Objects

D The **scope of development** subsumes all real world objects, which **can be changed** during development.

Based on [Glinz 2020, p.21]

- Scope of development is (implicitly) determined by **explicitly classifying real world objects as either**:
  - **Changeable**: Can be shaped, designed, created, modified (e.g., extended), or removed.
  - **Unchangeable**: It cannot be influenced and is considered as given.

System Boundary vs. Scope of Development

- **System boundary**: Separating the **system in operation** from the **context objects interacting with the system**
- **Scope of development**: Distinguishing objects, which are **changeable** in development, from objects, which are **not**
- Using both concepts can be **misleading** and **causing confusion**!
- Thus, we suggest using only the term "**system boundary**" to distinguish **changeable** from **unchangeable** real world objects, i.e., to define the **scope of development**.

Summary

- A requirement is always defined for a particular context.
- A requirement can only be correctly interpreted and validated based on information about the context.
- The context impacts the appropriate solutions for realizing requirements.
- The requirements engineering context subsumes all context objects and context information relevant for requirements engineering.
- Context objects are described by context information, and are also sources of context information.
- Two kinds of context information can be distinguished: Context assumptions and context constraints.
- Context objects and context information are considered in all requirements engineering activities, and in all subsequent development phases as well.
- The context is decomposed along with system decomposition: Context consideration for the overall system, as well as individual subsystems and components.
- The system boundary separates the system in operation from its context.
- The scope of development distinguishes changeable from non-changeable objects.

Literature

[Pohl 2010] K. Pohl: Requirements Engineering - Fundamentals, Principles and Techniques. 1st edition, Springer, 2010.

[Glinz 2020] A Glossary of Requirements Engineering Terminology, IREB, 2020. URL: https://www.ireb.org/content/downloads/1-cpre-glossary-2-0/ireb_cpre_glossary_en_2.0.pdf

[Zave and Jackson 1997] P. Zave, M. Jackson: Four Dark Corners of Requirements Engineering. In: ACM TOSEM, Vol. 6, No. 1, January 1997, pp. 1–30.

[Robertson and Robertson 2006] S. Robertson, J. Robertson: Mastering the Requirements Engineering Process. 2nd edition, Addison-Wesley, Amsterdam, 2006.

Literature for Further Reading

[Gunter et al. 2000] C. A. Gunter, E. L. Gunter, M. Jackson, P. Zave: A Reference Model for Requirements and Specifications. IEEE Software, Vol.17, NO. 3, IEEE Press, Los Alamitos, 2000, pp. 37-43.

[McMenamin and Palmer 1984] S. M. McMenamin, J. F. palmer: Essential Systems Analysis. Prentice Hall, London, 1984.

[Parnas and Madey 1995] D. L. Parnas, J. Madey: Functional Documents for Computer Systems. Science of Computer Programming, Vol. 25, No. 1, Elsevier, North-Holland, Amsterdam, 1995, pp. 41-61.

[Sutcliffe 2002] A. Sutcliffe: The Domain Theory – Patterns for Knowledge and Software Reuse. L. Erlbaum, Mahwah, 2002.

Image References

[1] Licensed by http://www.iconshock.com/

[2] Provided by Microsoft Office

Legend

Definition

Example

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Requirements Engineering & Management

Vielen Dank für Ihre Aufmerksamkeit

PALUNO
The Ruhr Institute for Software Technology