Main Bibliography:

Requirements Engineering: Fundamentals, Principles, and Techniques. Springer Verlag, Klaus Pohl, 2010. (Chapters 2 and 4)

AMS

Requirements Engineering: Fundamentals & Processes

Recalling: systems are not “natural things”, but are always the emergences of concerns expressed by stakeholders!!!

- **System of Interest:** The system that is being understood, created, utilized or managed due to specific concerns.
- **Enabling System:** Systems that are outside the boundary of the **system of interest**, which are required to assure its functionality **but are out of the scope of the concerns of the stockholders**.
- **System of Systems (SoS):** A system that is understood as made of several independent systems, for whose specific and independent concerns can be expressed.

A System of Systems as the Sistem of Interest (a computer, made of a power supply, memory, processors, discs, …)

http://bp0.blogger.com/_Wj1Rt3Yxk08/RkoKPEiIi3I/AAAAAAAAAAU/rSo0nDV8kGA/s320/01fig02.png

System of Systems (SoS)

An airport involves aircraft, support trucks, baggage-handling equipment, control towers, emergency facilities, and many other systems that can and do operate independently of each other. But, for the airport to function, it needs to have the right mix of these independent systems, and these systems need to cooperate with each other.

Stressing again: A system is *always* a relative concept!

- The definitions of system of interest, enabling system and system of systems are always dependent on the **concern** and of the **context** related to that (the environment of the system).
  - An enabling system can be a system of interest or vice-versa.
  - A system can always be included as part of a "larger" SoS or decomposed as a "smaller" SoS.

- The definition of the **context of a system** depends on the **purpose** and on the **concerns** of the **stakeholders** of the system.
  (cf. IEEE 1471:2000, ISO 42010).

ISO 42010 on “Environment”

“A **system** is situated in an **environment**.

The environment determines the totality of influences upon the system throughout its life cycle, including its interactions with that environment. **The environment of a system can contain other systems**.

(…) the environment of a system is bounded by and understood through the identification and analysis of the system’s stakeholders and their concerns (see 4.2.3).”

- The environment can include other systems that interact with the system of interest, either directly via interfaces or indirectly in other ways.
- The environment determines the boundaries that define the scope of the system of interest relative to other systems.

Starting simple

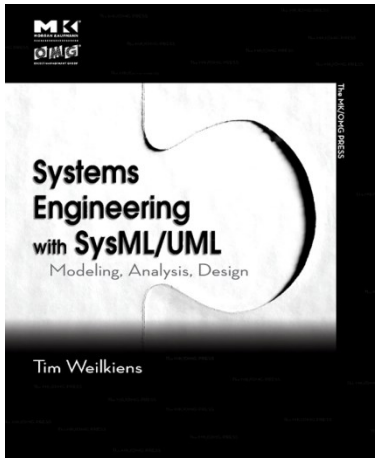
The system context model (environment)

"The **system context model** represents the direct **environment of the system** and gives initial information about the **communication flowing from and to** the system.

The external interaction partners are the **system actors**.

The communication itself is described by **information flows**."

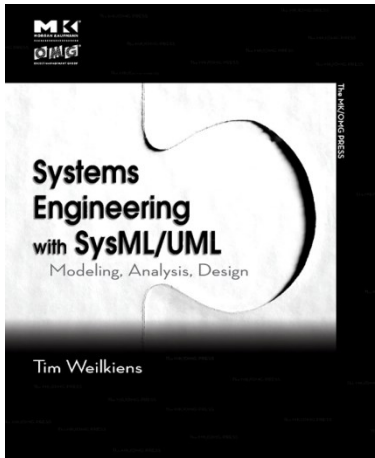
Systems engineering with SysML/UML: modeling, analysis, design/Tim Weilkiens; page 45)



The system context model (environment)

- The system context model is not specified on SysML or UML but as part of the SYSMOD methodology.
- This model can be visualized using different notations.

Note: we recommend this book, but the SYSMOD methodology in (Wielkiens 2008) **is not part** of this course's syllabus!



The system context model

definition to be used on this course

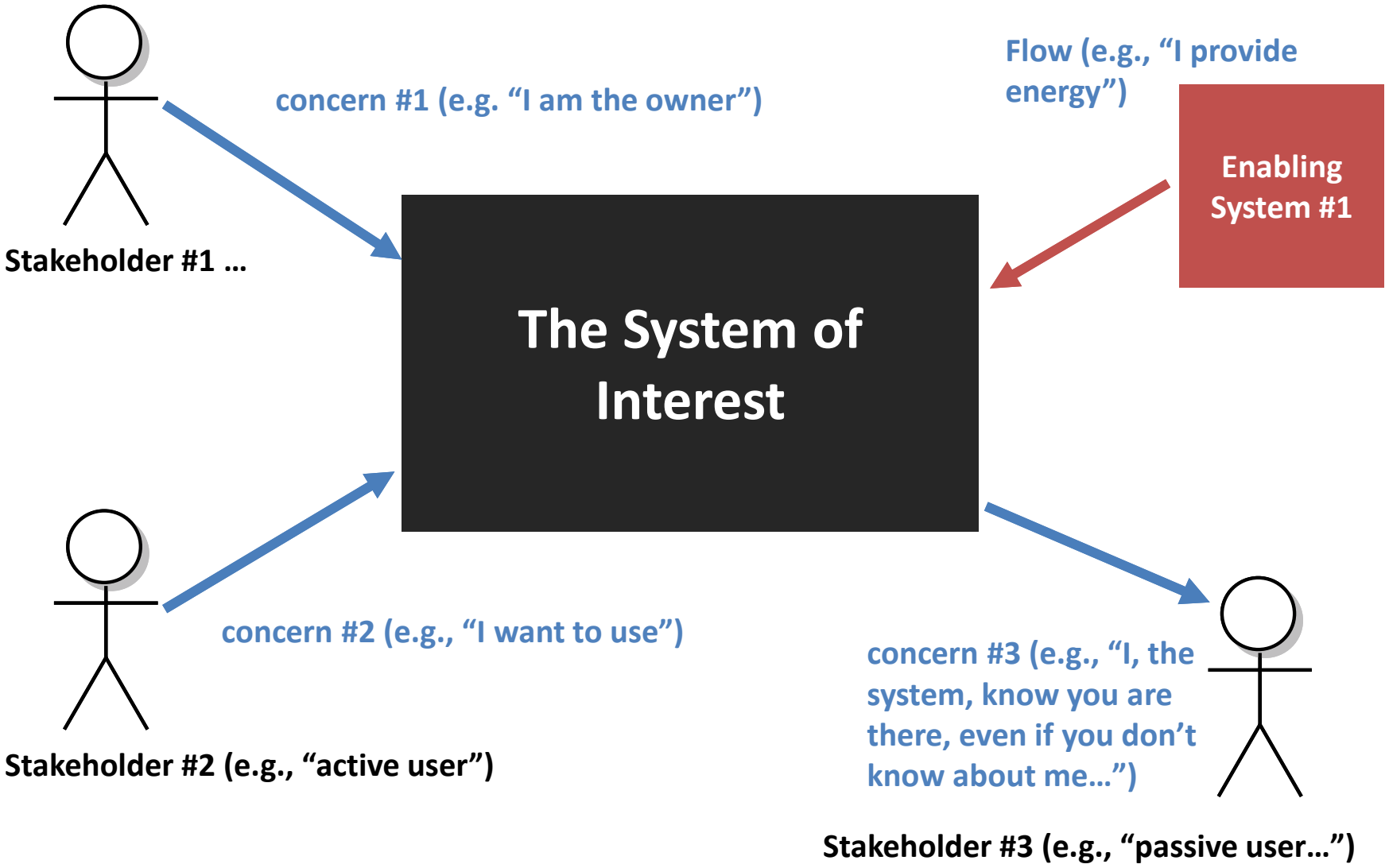
- The **system context model** (aka **context diagram**) is a specification of the environment of the system of interest.
- Context diagrams describe the following **concepts**:
  - The **system (of interest)**, represented as a named black-box entity.
  - The **stakeholders** (an individual, team, or organization) that have concerns over the system. Each stakeholder (or class thereof) is uniquely identified by a <noun>.
  - The **enabling systems** that directly exchange information with the system. Each enabling system is uniquely identified by a <noun>.
- Context diagrams describe the following **relationships**:
  - Associations between **stakeholders** and the **system of interest** describe the concerns of each stakeholder (or class thereof).
  - Associations between **enabling systems** and the **system of interest** describe the corresponding directed information flows.
  - Associations can be described by <verb>, <noun > or <verb noun>.

The system context model

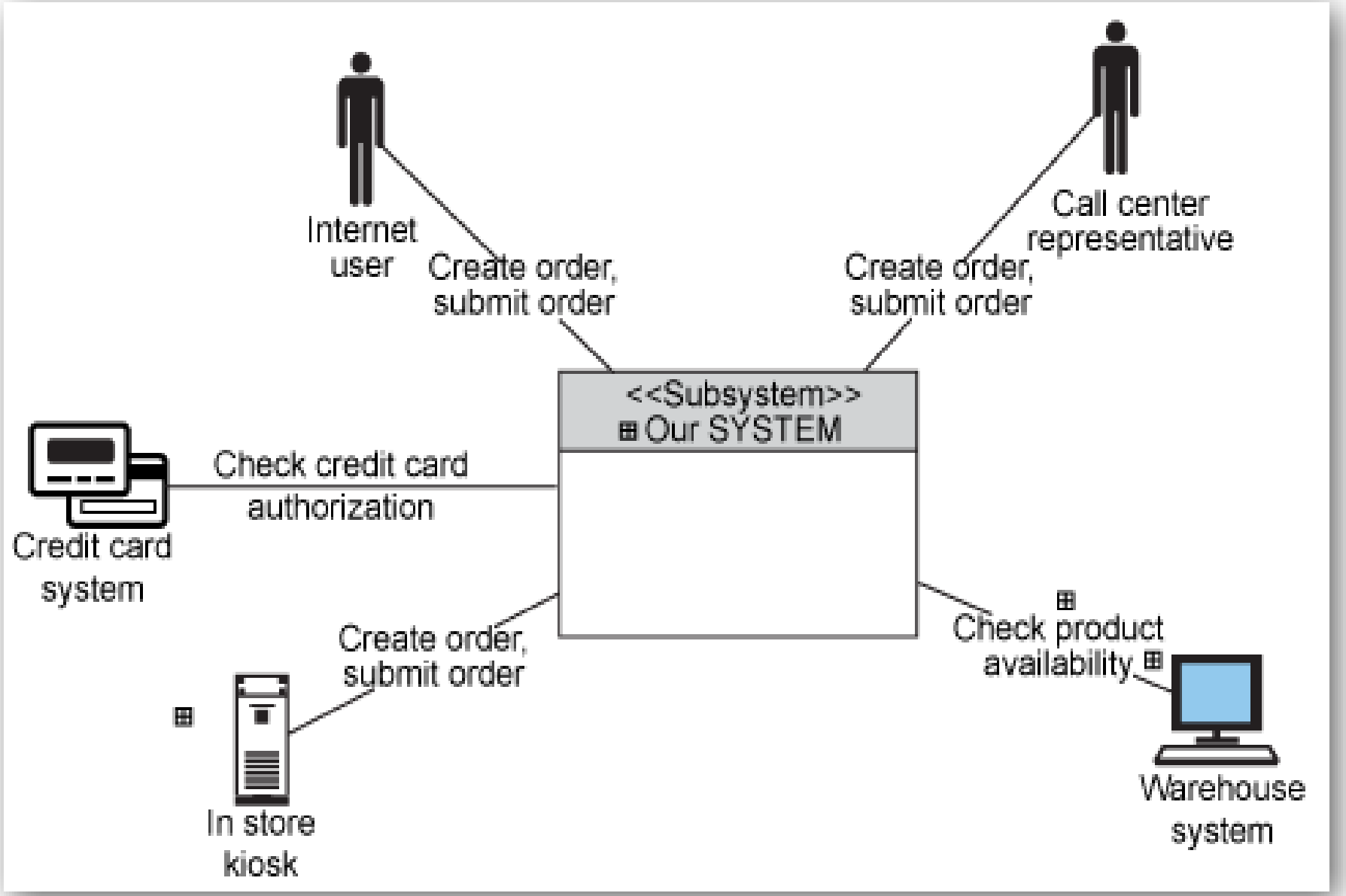
- The context model should clearly and non ambiguously define the system's boundary, stakeholders and other eventual enabling systems.

- The purpose is to provide a **simple black-box** depict of the **system of interest** along with its external dependencies...

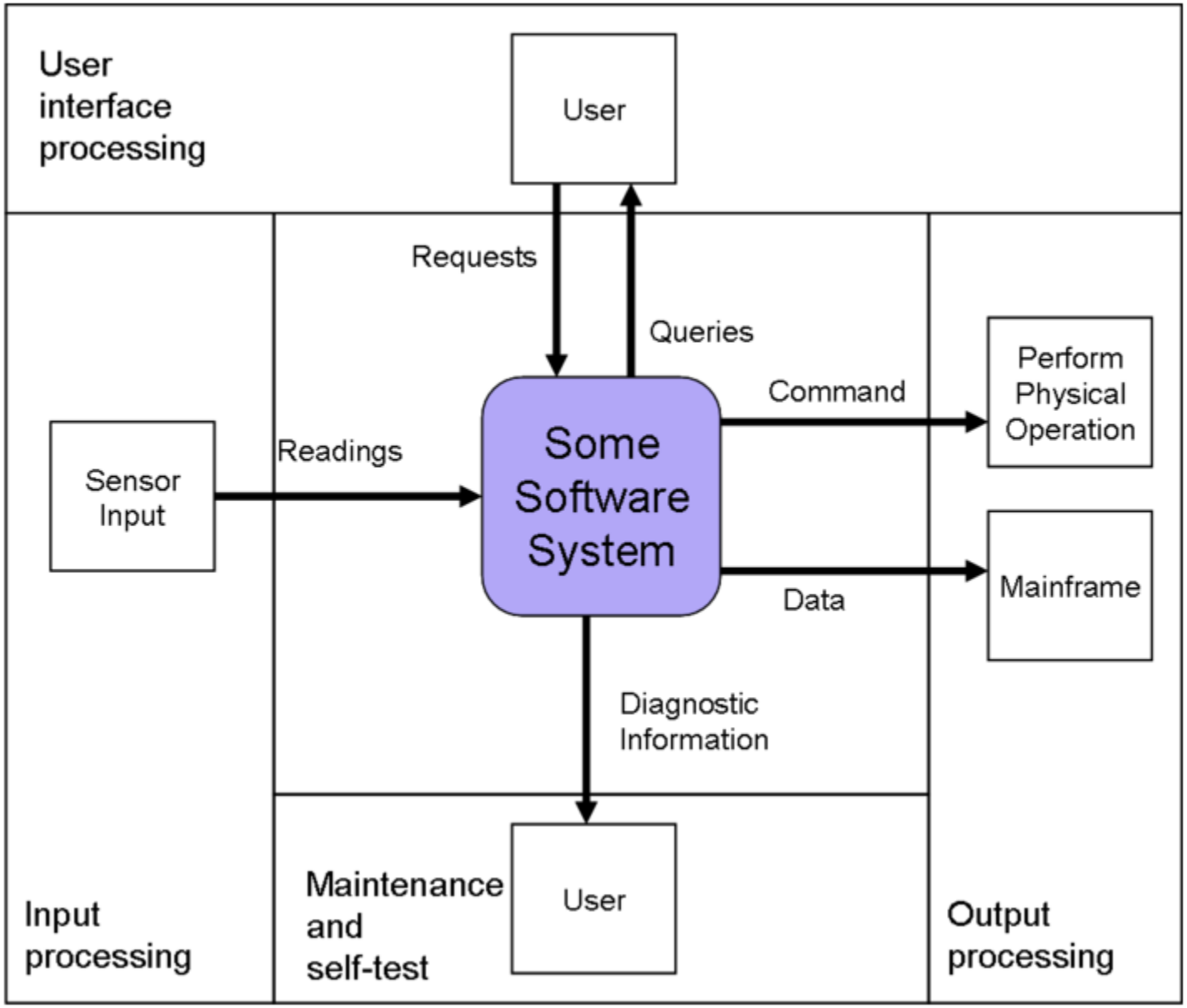
Context Diagram



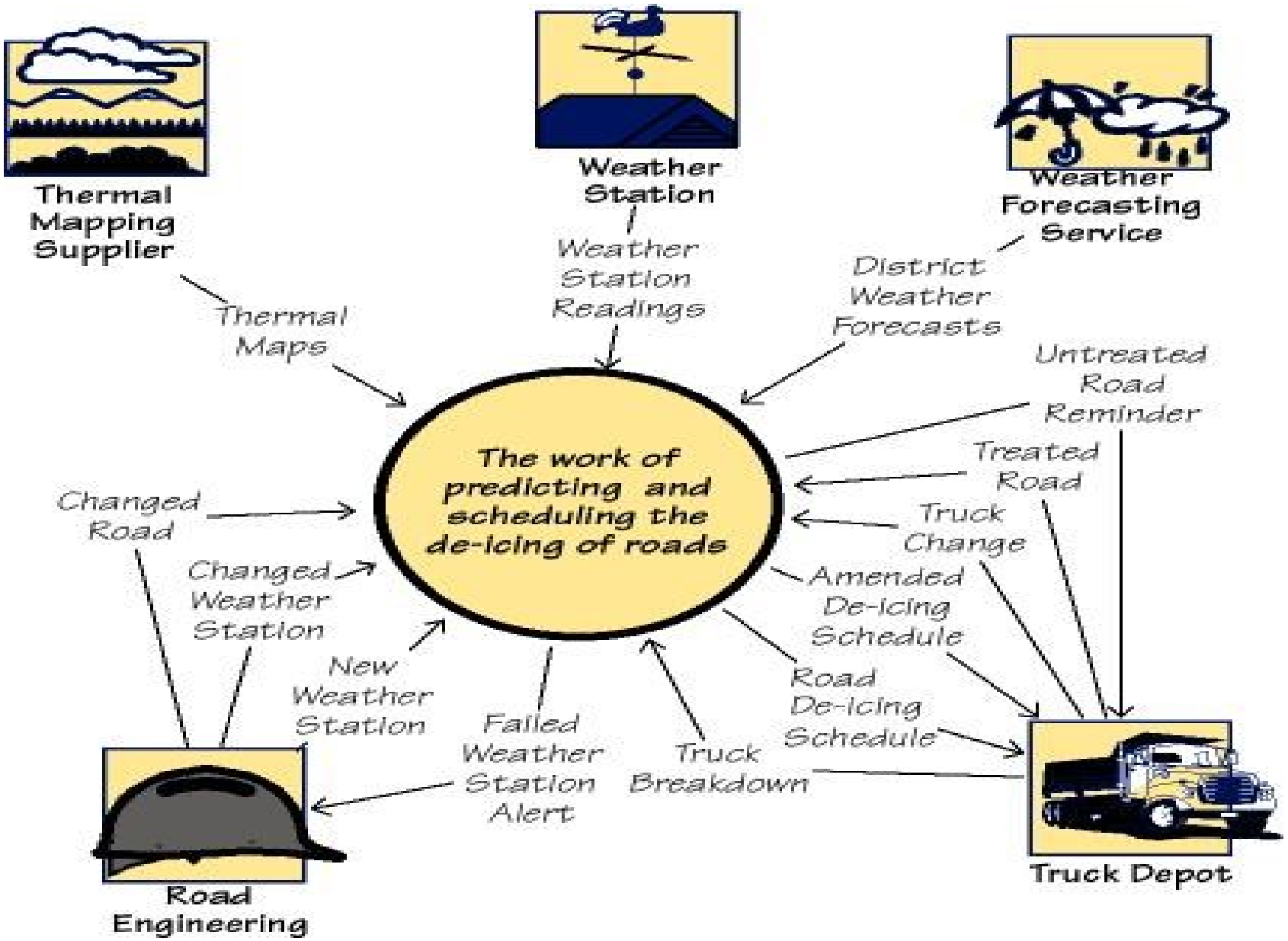
Context Diagram: Example #1



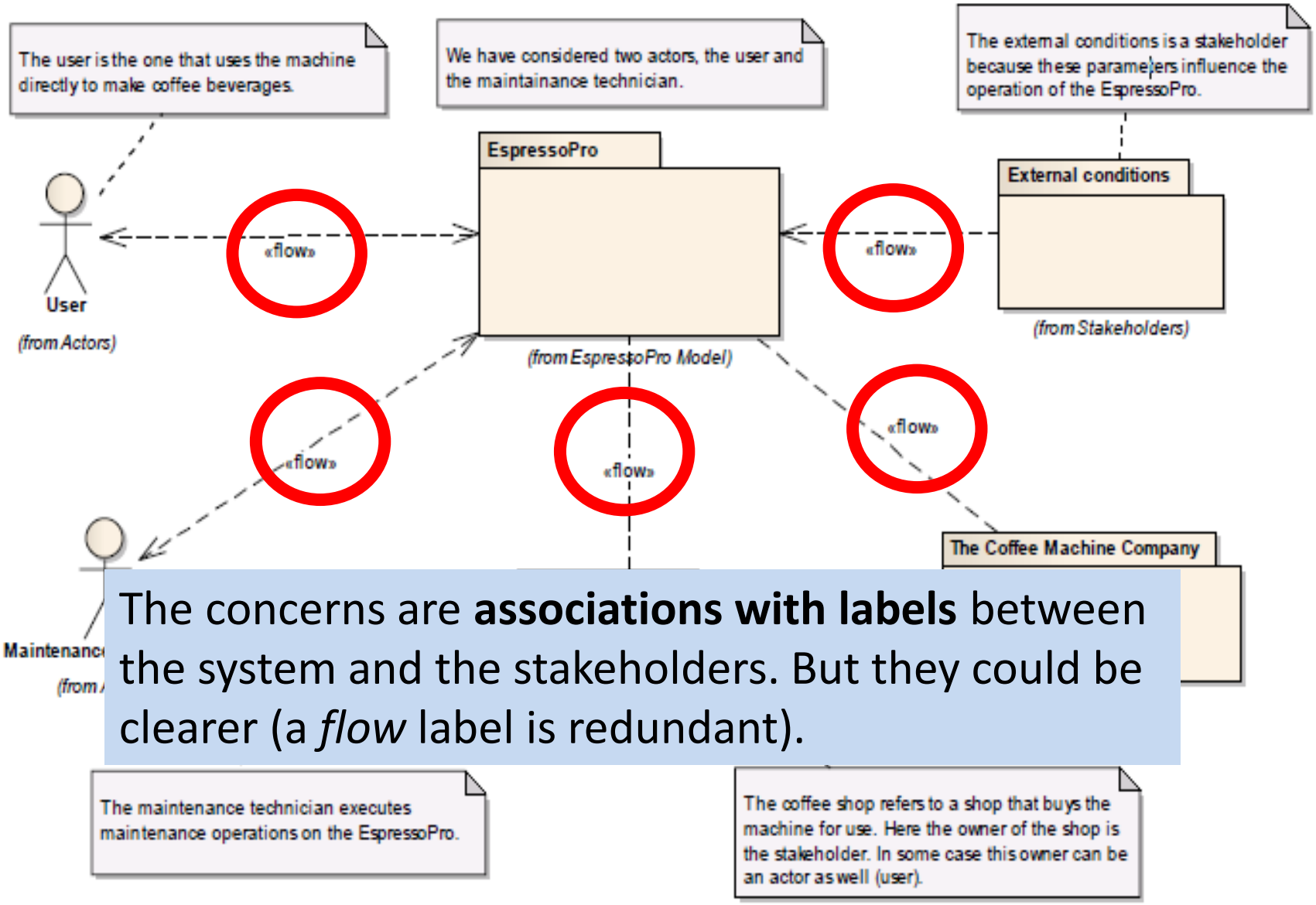
Context Diagram: Example #2



Context Diagram: Example #3



Context Diagram: Example #4



AMS

Requirements Engineering Fundamentals…

…What are requirements?

Requirements

- A **requirement** is a condition or capability to which a system must conform to.

- Must derive from a well-defined source.
  - The source may be internal or external.
  - Include stakeholders, requests, standards, regulations, strategy, etc.
- Are elicited (captured) using different techniques depending on context.
  - Elicitation techniques include document analysis, interviews, brainstorming sessions, workshops, etc.

What is a Requirement?

Definition 2-1: *Requirement*

(1) A condition or capability needed by a user to solve a problem or achieve an objective.
(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
(3) A documented representation of a condition or capability as in (1) or (2).

[IEEE Std 610.12-1990]

Definition 2-2: *Requirements artefact*

A requirements artefact is a documented requirement.

What is a Requirement?

Example 2-1: Requirements for a house information system[4]

R1 The system shall offer a user-friendly interface.

R2 The house information system shall generate a monthly report containing all granted and denied admittances to the house.

R3 If the PIN (personal identification number) the user enters at the keypad is correct, the system shall open the door and record the granted access, i.e. it should record the date and time, and the name of the PIN owner.

R4 The system shall be available on the market by 1 May 2006.

R5 The unlocking of the door shall happen within 0.8 s after the PIN has been entered correctly.

Requirements in a business context…

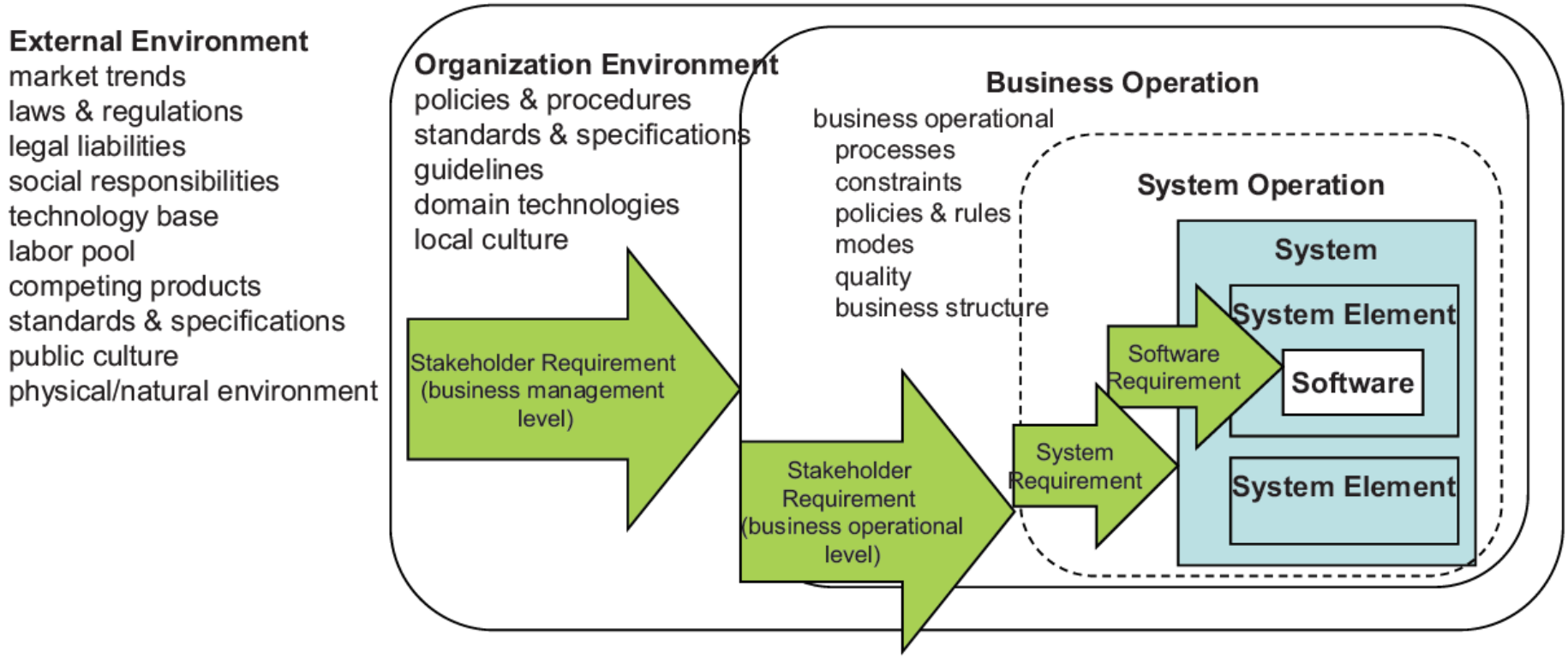


Figure 4 — Example of requirements scope in a business context

Main Types of Requirements:

- Functional Requirements
- Quality Requirements
- Constraints

Main Types of Requirements

- Functional Requirements

Definition 2-3: *Functional requirements*

"These [functional requirements] are statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. In some cases, the functional requirements may also state what the system should not do. [...]

[...] When expressed as user requirements, the requirements are usually described in a fairly abstract way. However, functional system requirements describe the system function in detail, its inputs and outputs, exceptions, and so on."

[Sommerville 2007]

Example 2-2: Functional requirement

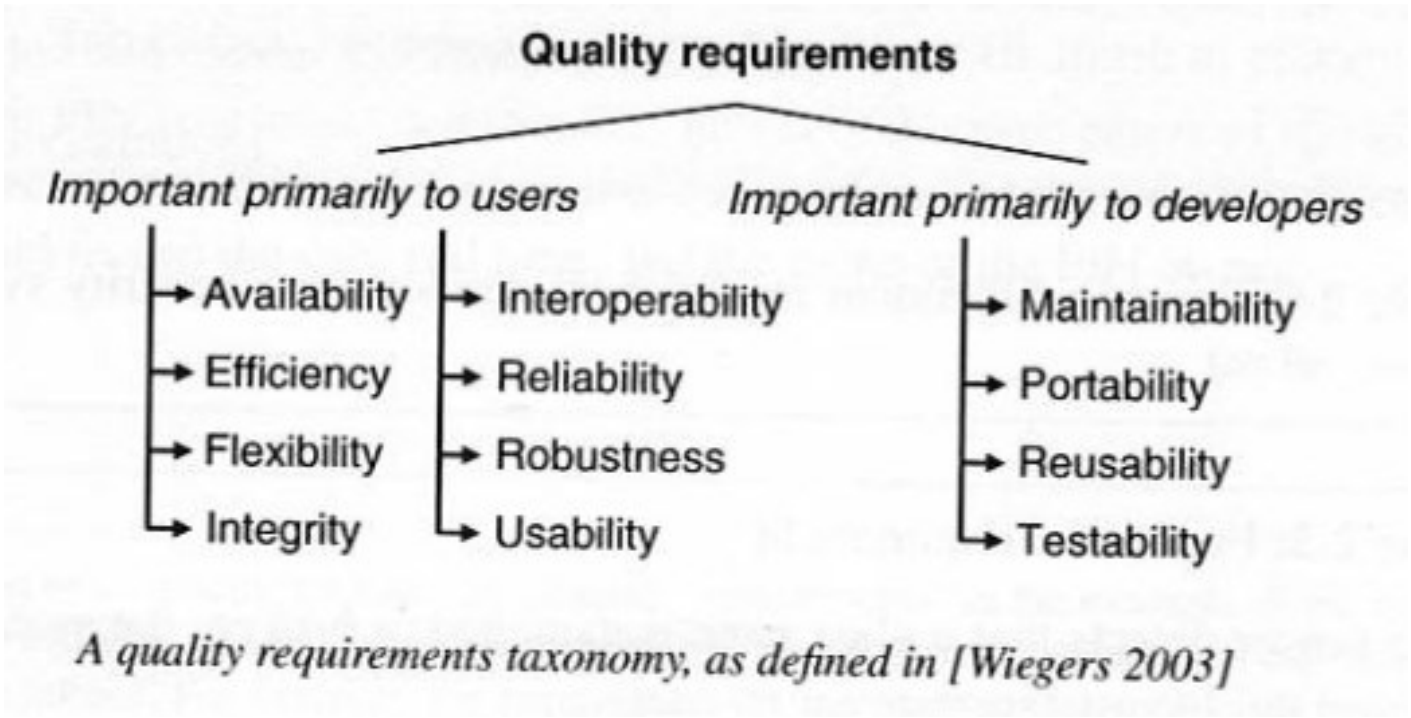
R31 If a sensor detects that a glass pane is damaged or broken, the system shall inform the security company.

Main Types of Requirements

- Quality Requirements

Definition 2-4: *Quality requirement*

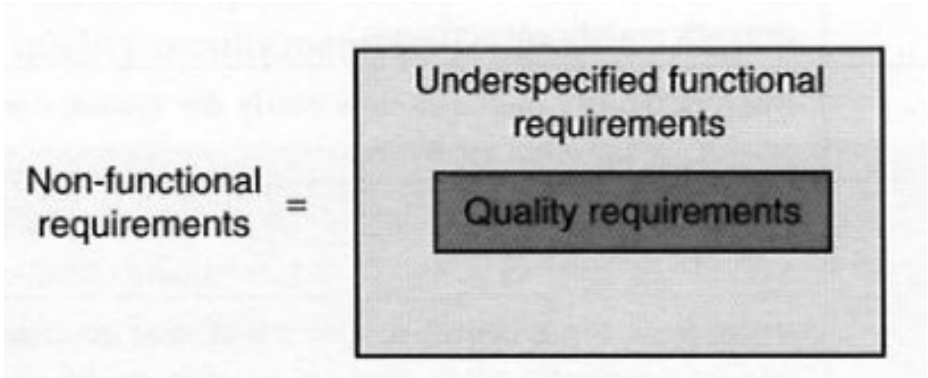
A quality requirement defines a quality property of the entire system or of a system component, service, or function.



A quality requirements taxonomy, as defined in [Wiegers 2003]

Main Types of Requirements

- Non-Functional Requirements



Example 2-3: Non-functional requirement/underspecified functional requirement

R12 The system shall be secure.

On looking at requirement R12 from Example 2-3 in slightly more detail, several questions arise due to the underspecification of this requirement, such as:

- What does the adjective "secure" mean?
- Which properties shall the system provide in order to be "secure"?
- How can one check whether the implemented system is "secure"?

Main Types of Requirements

Example 2-4: Refinement of the underspecified requirement from Example 2-3 – "R12: The system shall be secure."

R12.1 Each user must log in to the system with his user name and password prior to using the system. (*functional requirement*)

R12.2 The system shall remind the user every four weeks to change the password. (*functional requirement*)

R12.3 When the user changes his password, the system shall validate that the new password is at least eight characters long and contains alphanumeric characters. (*functional requirement*)

R12.4 The user passwords stored in the system must be protected against password theft. (*quality requirement - integrity*)

Main Types of Requirements

- Non-Functional Requirements

Hint 2-1: *How to deal with "non-functional" requirements*

- Avoid the category "non-functional" requirements in the requirements specification.
- Instead, differentiate between functional requirements and (specific types of) quality requirements (see Fig. 2-1).
- "Non-functional" requirements often conceal underspecified functional requirements.
- Only a few of the so-called non-functional requirements are actually quality requirements.
- For each "non-functional" requirement, check whether it represents an underspecified functional requirement and, if so, refine it adequately.

Main Types of Requirements

- Constraints

Definition 2-5: *Constraint*

A constraint is an organisational or technological requirement that restricts the way in which the system shall be developed.

based on [Robertson and Robertson 2006]

Example 2-5: Constraints affecting the system

C1 Due to current conditions defined by the insurance company, only the security technician is allowed to deactivate the control function of the system.

C2 A fire protection requirement demands that the terminals in the sales rooms do not exceed the size 120 cm (height) × 90 cm (width) × 20 cm (depth).

Example 2-6: Constraints affecting the development process

C3 The effort for the development of the system must not exceed 480 person-months.

C4 The system must be developed using the Rational Unified Process.

The three dimensions of Requirements Engineering

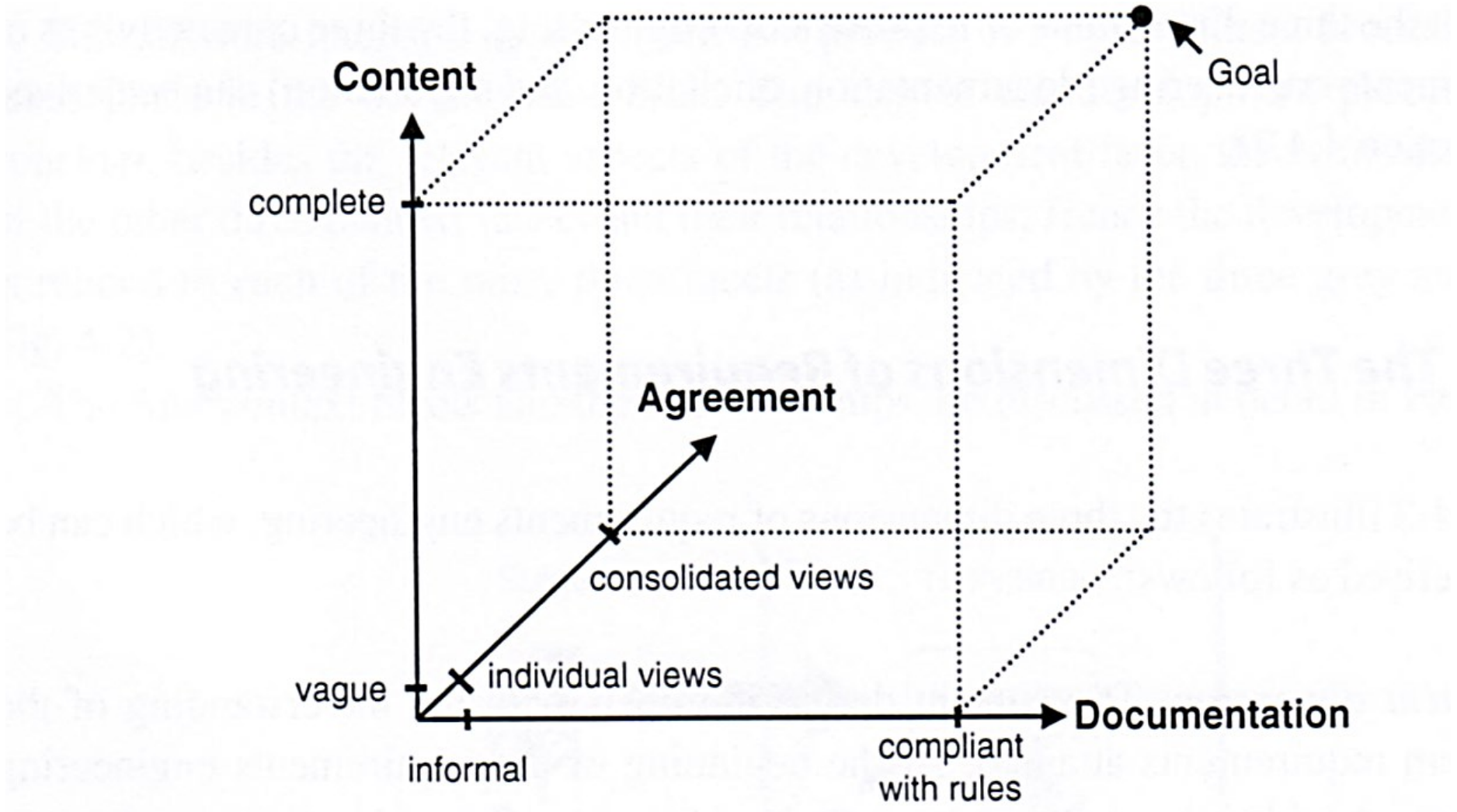


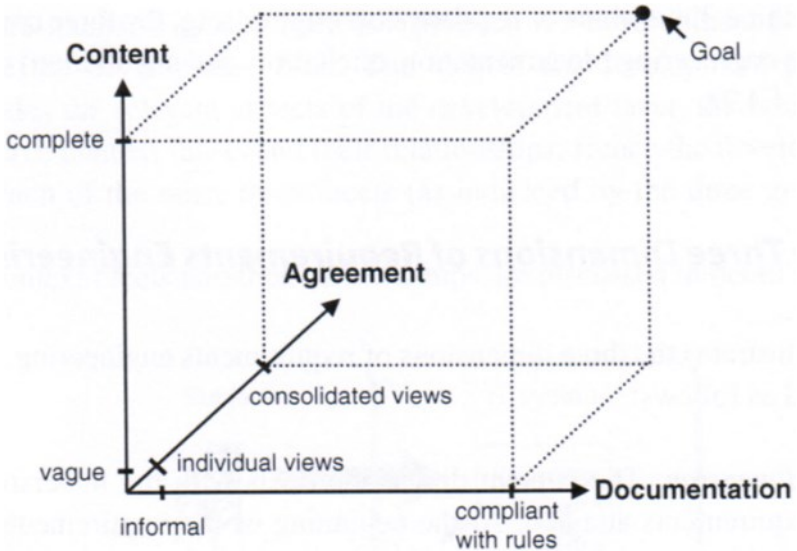
Fig. 4-3 *The three dimensions of requirements engineering (based on [Pohl 1994])*

Requirements Engineering is…

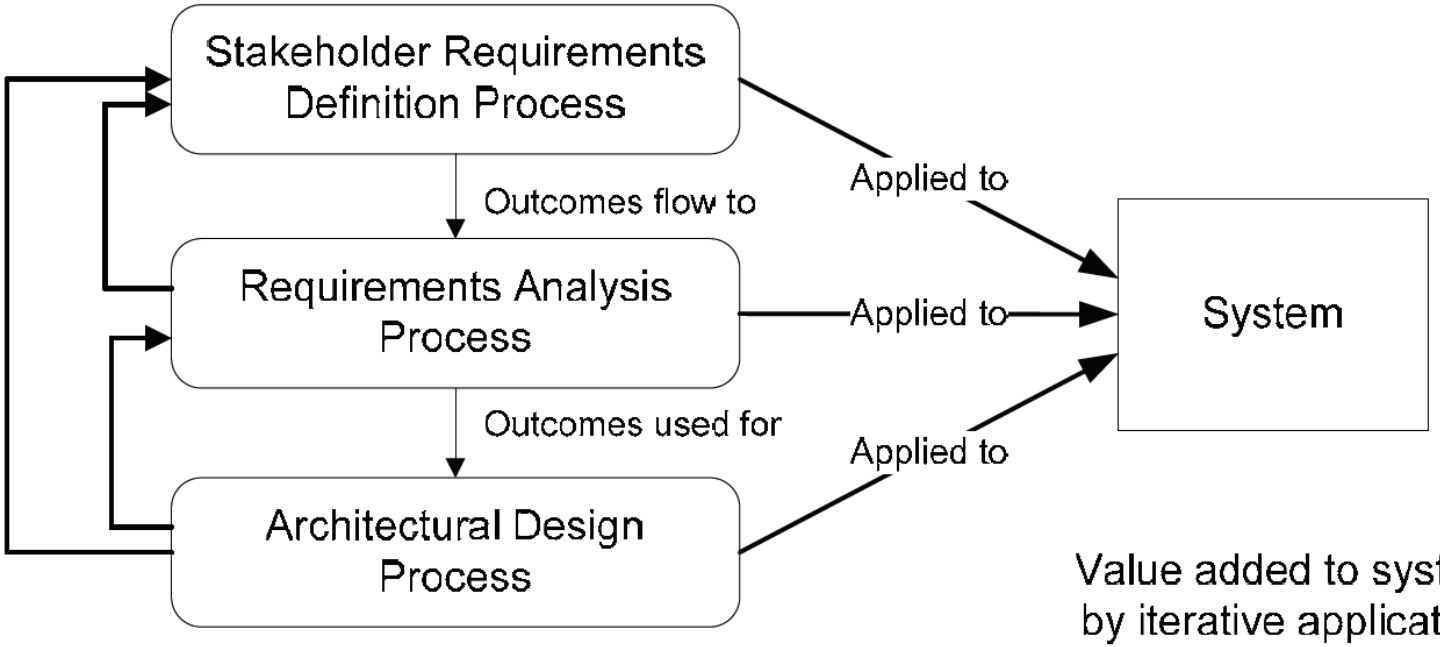
Definition 4-1: *Goals of requirements engineering*

Requirements engineering is a cooperative, iterative, and incremental process which aims at ensuring that:

(1) All relevant requirements are explicitly known and understood at the required level of detail.
(2) A sufficient agreement about the system requirements is achieved between the stakeholders involved.
(3) All requirements are documented and specified in compliance with the relevant documentation/specification formats and rules.



Iterative requirements engineering…



System goals as “high level requirements”

Requirements engineers need to understand the stakeholders' intentions with regard to the system to be developed. In requirements engineering, the stakeholders' intentions are documented as goals. Antón states in [Antón 1996] that goals "are high-level objectives of the business, organisation, or system". According to [Van Lamsweerde 2001], a goal is "an objective the system under consideration should achieve". We define a goal (in requirements engineering) as follows:

Definition 4-2: *Goal*

A goal is an intention with regard to the objectives, properties, or use of the system.

System goals as “high level requirements”

Goals have a prescriptive nature, i.e. a goal states what is expected or required from the system. Thereby, goals differ from descriptive statements such as statements about the domain of the system (e.g. the description of a physical law). Example 4-4 depicts two goals for a navigation system.

Example 4-4: Goals for the car navigation system example

G_1: The system shall guide the driver to a desired destination automatically.

G_2: The response times of the system shall be 20% lower compared with the predecessor system.

System Scenarios in Requirement Engineering…

A scenario typically documents a concrete example of system usage. It thus illustrates the fulfilment (or non-fulfilment) of a goal (or set of goals). Thus, a scenario describes a concrete example of either how the system satisfies a goal or how it fails to satisfy a goal. A scenario may define an interaction sequence at different levels of abstraction. For example, a scenario can describe the interactions in detail and thus very close to reality or only document the essential interactions (and thereby abstract from the incarnation). We define a scenario as follows:

Definition 4-3: *Scenario*

A scenario describes a concrete example of satisfying or failing to satisfy a goal (or set of goals). It thereby provides more detail about one or several goals. A scenario typically defines a sequence of interaction steps executed to satisfy the goal and relates these interaction steps to the system context.

System Scenarios in Requirement Engineering…

Example 4-5: Scenario "Automatic braking manoeuvre"

Carl drives his car on the motorway at a speed of 50 mph. Peter, the driver of the car ahead of Carl, steps on the brake pedal firmly. After recognising that the car in front is braking, Carl pushes on the brake pedal as well. The on-board computer of Carl's vehicle detects that the safety distance to Peter's car is no longer maintained and issues a warning to the driver. The distance between the two cars continuously decreases. In order to support the driver, the on-board computer initiates an automated full braking. The computer informs Carl about the automatic braking manoeuvre. After the distance between the two cars stops decreasing, the on-board computer terminates the full braking manoeuvre. The on-board computer continues controlling the speed of Carl's car until the safety distance to Peter's car is maintained and informs Carl about the end of this "manoeuvre".

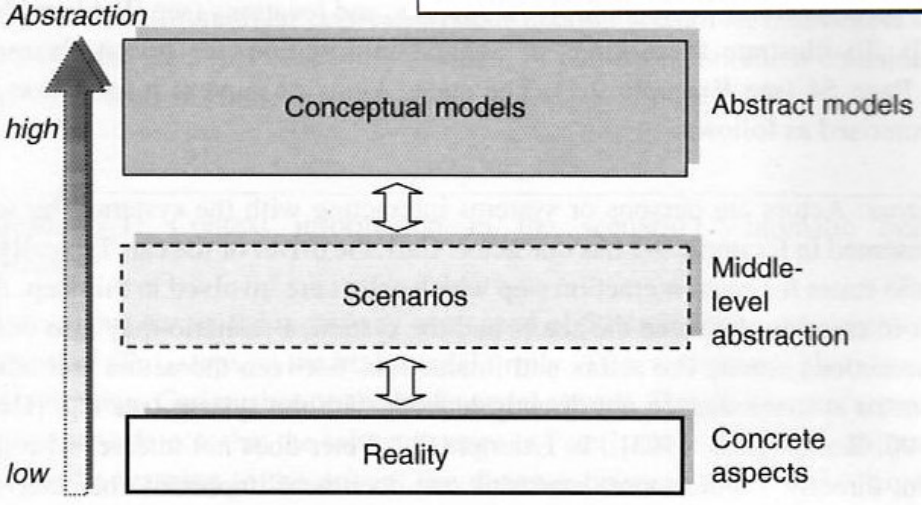
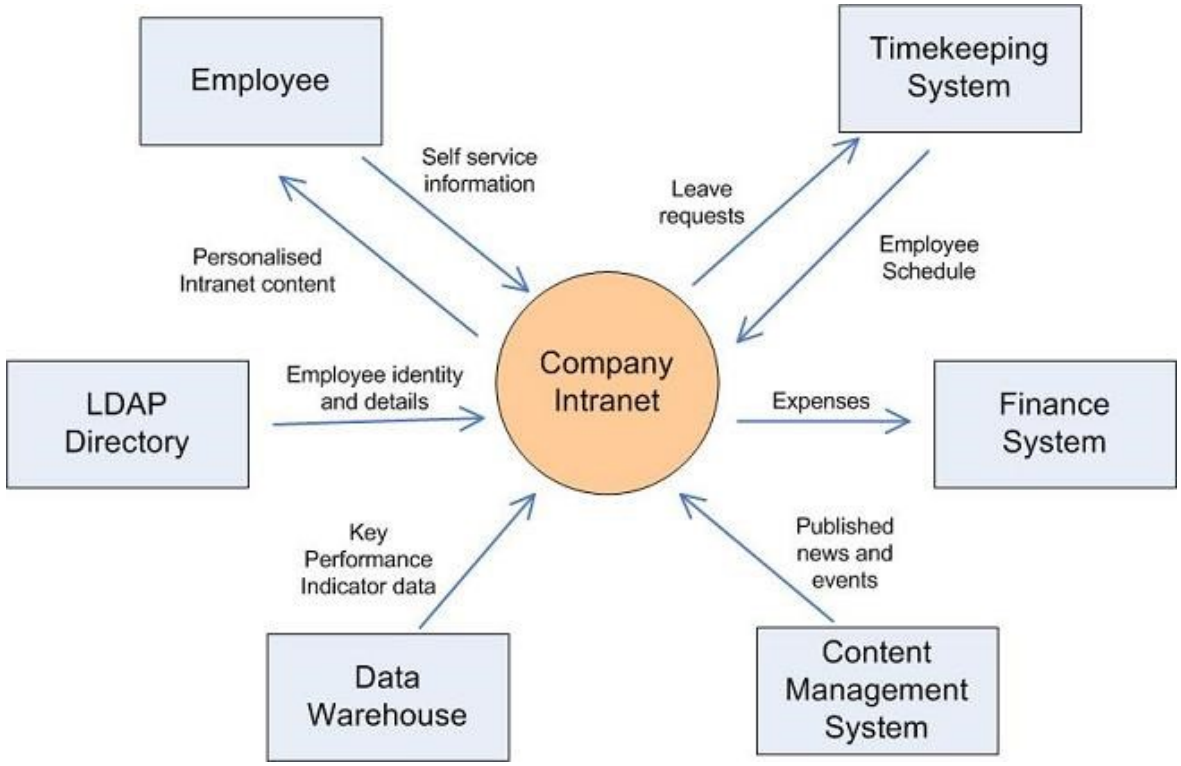


Fig. 9-1 *Scenarios as middle-level abstractions*

System Context in Requirement Engineering…

Requirements are defined and interpreted within a given context. A change in the context can change the meaning of a requirement dramatically. For example, the way a requirement is interpreted is significantly influenced by the documented context information. If insufficient or even no context information is documented, the same requirement can be interpreted in fairly different ways.



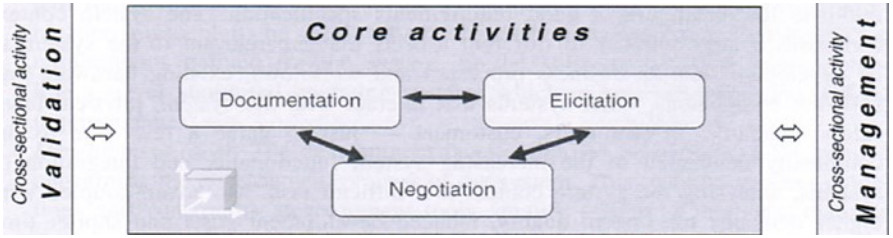
System Context in Requirement Engineering…

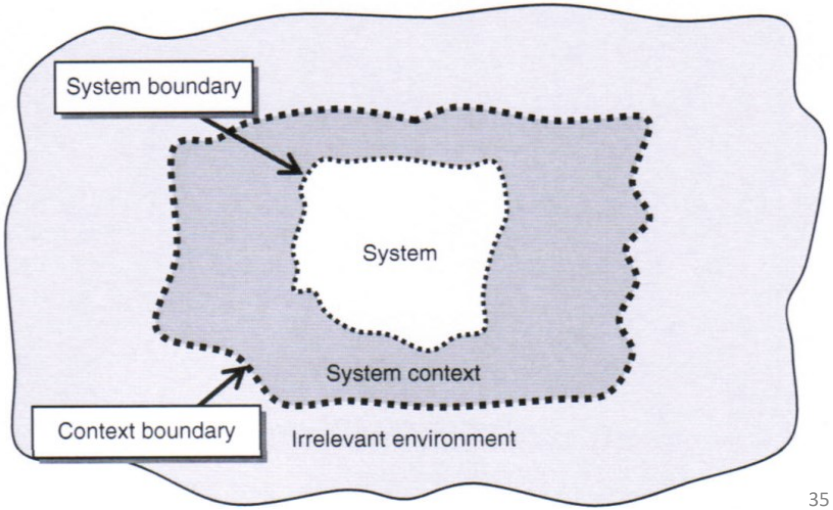
Requirements are defined and interpreted within a given context. A change in the context can change the meaning of a requirement dramatically. For example, the way a requirement is interpreted is significantly influenced by the documented context information. If insufficient or even no context information is documented, the same requirement can be interpreted in fairly different ways.

Example 5-4: Documented requirement with vague (almost no) context information

R23: The planned means of transportation shall offer travellers a fast journey to their destination.

...
Planned means of transportation ???
...
Fast journey to destination ???
...





AMS

Requirements Engineering Fundamentals…

…**What are the properties of requirements**?

Examples of FR

C. Functional Requirements

1. Printing

1.1. The user can select which pages to print
1.2. The user can view a preview of the pages before printing
1.3. The user can change the margins, paper size (e.g., letter, A4) and orientation on the page

2. Spell Checking

2.1. The user can check for spelling mistakes; the system can operate in one of two modes as selected by the users
2.1.1. Mode 1 (Manual): The user will activate the spell checker and it will move the user to the next misspelled word
2.1.2. Mode 2 (Automatic): As the user types, the spell checker will flag misspelled words so the user immediately see the misspelling
2.2. The user can add words to the dictionary
2.3. The user can mark words as not misspelled but not add them to the dictionary

(“Systems analysis and design with UML” – ISBN: 978-0471348061)

Examples of NFR

D. Nonfunctional Requirements

1. Operational Requirements
- 1.1. The system will operate in Windows and Macintosh environments
- 1.2. The system will be able to read and write Word documents, RTF, and HTML
- 1.3. The system will be able to import Gif, Jpeg, and BMP graphics files

2. Performance Requirements
- 2.1. Response times must be less than 7 seconds
- 2.2. The Inventory database must be updated in real time

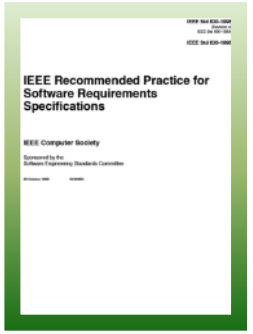
3. Security Requirements
- 3.1. No special security requirements are anticipated

4. Cultural and Political Requirements
- 4.1. No special cultural and political requirements are anticipated

(“Systems analysis and design with UML” – ISBN: 978-0471348061)

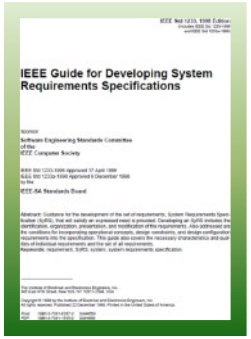
Properties of requirements (IEEE 830)

1. Correct
2. Unambiguous
3. Complete
4. Consistent
5. Ranked for importance
6. Verifiable
7. Modifiable
8. Traceable



Properties of requirements (IEEE 1233)

1. **Unique set** - Each requirement should be stated only once.
2. **Normalized** - Requirements should not overlap (i.e, they shall not refer to other requirements or the capabilities of other requirements).
3. **Linked set** - Explicit relationships should be defined among individual requirements to show how the requirements are related to form a complete system.
4. **Complete** - An SyRS should include all the requirements identified by the customer, as well as those needed for the definition of the system.
5. **Consistent** - SyRS content should be consistent and non contradictory in the level of detail, style of requirement statements, and in the presentation of material.
6. **Bounded** - The boundaries, scope, and context for the set of requirements should be identified.
7. **Modifiable** - The SyRS should be modifiable. Clarity and non overlapping requirements contribute to this.
8. **Configurable** - Versions should be maintained across time and across instances of the SyRS.
9. **Granular** - This should be the level of abstraction for the system being defined.


IEEE Guide for Developing System Requirements Specifications

Properties of requirements (other)

1. **Early/Premature Design**: information about too early design or implementation?
2. **Detail**: is it a unique requirement, or should it be split in more than one?
3. **Need**: is it really a requirement, or just a “cosmetic” wish?
4. **Non standard technology**: detect as soon as possible if the requirement implies non standard hardware, software, or other technology
5. **Conformity with the business objectives**: is the requirement consistent with the objectives exposed in the beginning?
6. **Ambiguities**: is it only one possible interpretation, or is the requirement susceptible of multiple interpretations?
7. **Realism**: it is a realistic requirement, especially concerning the technology, costs, and other assumptions ort constraints?
8. **Testability**: can it be designed a test to assess if the system accomplishes the requirement?

Properties of SMART Requirements

Specific (requirement is concrete, detailed, well defined)

Measurable (numbers, quantity, comparison to verify it)

Achievable (the system can realistically exhibit the requirement)

Realisable (can the requirement be satisfied? Also, is it **relevant**, does it considers goals that matter?)

Traceable (can we trace it through the system's lifecycle?)

Specific

- **Clear** there is no ambiguity.
- **Consistent** the same terminology has been used throughout the specification to describe the same system element or concept (implies system vocabulary).
- **Simple** avoid double requirements, e.g. X *and* Y, X *or* Y.
- Of an **appropriate level of detail.**

Specific

1. Avoid phrases such as: "obviously", clearly", "certainly"
2. Avoid ambiguities such as "some", several", "many"
3. Avoid list terminators such as: "etc", "and so on", "...such as"
4. Ensure pronouns are clearly referenced e.g. "When module A calls B its message history file is updated"
5. When numbers are specified identify the units
6. Ensure all possible elements in a list are described
7. Use pictures to clarify understanding
8. Ensure all system or project terms are defined in a glossary
9. Consider placing individual requirements in a separate paragraph and individually numbered
10. Ensure verbs such as "transmitted", "sent", "downloaded", "processed" are qualified by precise explanations.

Measureable

Measurable means that it is possible to verify that a requirement has been met once the system has been constructed.

- What other requirements need to be verified before this requirement?
- Can this requirement be verified as part of the verification for another requirement ? If so, which one?
- How much data or what test cases are required?
- How much processing power is required?
- Can the test be conducted on one site?
- Can this requirement be tested in isolation?

Achievable

A requirement is achievable if it is physically possible for the system to exhibit that requirement under given conditions.
Some requirements may be beyond the bounds of human knowledge. Others may have theoretical solutions but be beyond what is currently achievable.

- Is there a theoretical solution to the problem ?
- Has it been done before ? If not, why not ?
- Has a feasability study been done ?
- Is there an overiding constraint which prohibits this requirement ?
- Are there physical constraints on the size of the memory, processor or peripherals ?
- Are there environmental constraints such as temperature?

Realisable

A requirement is realisable if it is possible to achieve it given what its known about the constraints under which the system and the project are developed.

Determining whether a requirement is realisable or not is the most difficult part of creating a SMART requirement.

Realisable

- Determine who has responsibility for satisfying the requirement.
  - Can they deliver?
  - Can we afford to manage them?
- How badly is it needed?
- Are there sufficient resources?
  - staff with the right skill set;
  - space and desks;
  - hardware and software for development;
  - hardware and software for testing
- Is there sufficient time?
- Is there sufficient budget?
- Are we constrained to a particular package which does not support this requirement?
- Will we have to develop it ourselves?
- Can we reuse from other projects?

Traceable

Traceability is the ability to follow (forward and backward) a requirement from its conception through its specification to its subsequent design, implementation and test.

It is important for the following reasons:

- To know and understand the reason why each requirement was included on the system.
- To verify if and how each requirement has been implemented.
- To facilitate consistent and complete modifications.

Examples

- **R1 “The Mission Planning System shall support several planning environments for generating the mission plan.”**
- **R2 “The system shall support 50 simultaneous users.”**
- **R3 “The real-time system will have no memory leaks.”**
- **R4 “The system shall produce a plan optimised for time.”**

Examples

- **R1 “The Mission Planning System shall support several planning environments for generating the mission plan.”**
- **R2 “The system shall support 50 simultaneous users.”**
- **R3 “The real-time system will have no memory leaks.”**
- **R4 “The system shall produce a plan optimised for time.”**

- **R1 is not specific.**
  - What is meant by “several”?
  - Is “planning environment” defined?
  - Is “mission plan” defined?
  - Is “generating” clear in this context?
- **R2 is not specific.**
  - R2 is clear, but… what actions can the users be doing? That is not specified.
- **R3 is possibly not measurable.** Such measurement may interfere with the real-time concern.
- **R4 is possibly not measurable.** What is the baseline plan?

Examples

- **R1 “The system shall have six-nines* availability.”**
- **R2 “The system shall have a maximum response time of 10ms to any query irrespective of system load.”**

- 99.999% => downtime 5.26 min/year; 99.9999% => 31.5 sec/year

- **Attainable?** The likely consequence of attempting to meet these two requirements is that the system may never be accepted or may be prohibitively expensive or both.

Examples

- **R1 “The system software should have five-nines reliability”**
- **R2 “The system shall have a chair for each of the ten operators”**
- **R3 “The operator chairs should be red“**

- R1 may be attainable but **may not be realisable** if the project does not allow for it.
  - **essential** (shall, must) vs. **desirable** (should)
- R2 is essential; R1, R3 are desirable.

Back to the Types of Requirements

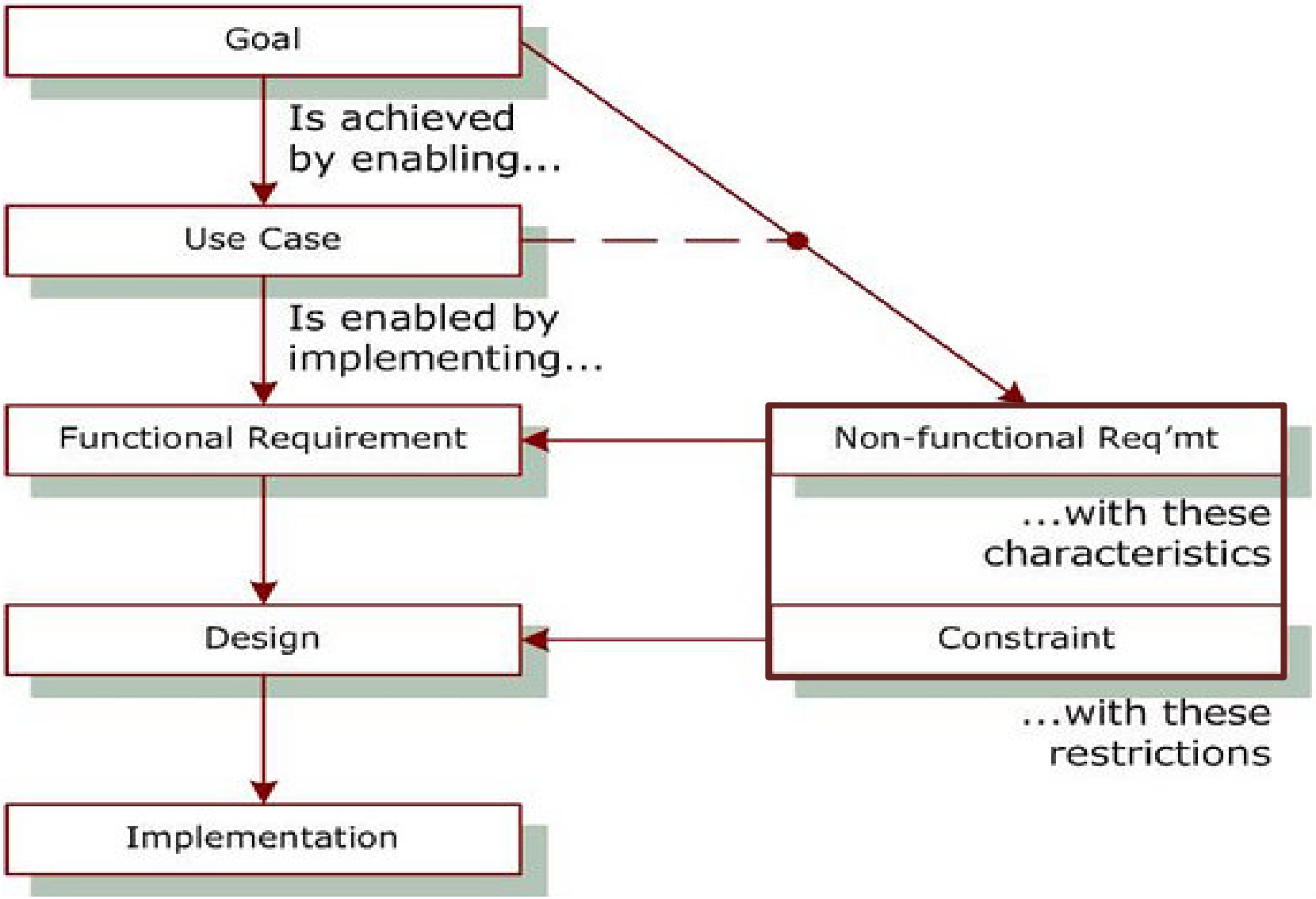
- **Functional Requirements (FR)**
  - What the system is supposed *to do* (the system shall do…)
  - The specification of a function (behaviour) that the system must be able to perform.
  - A FR affects the design of a system.
- **Non-Functional Requirements (NFR)**
  - What the system is supposed *to be* (the system shall be…)
  - The specification of the qualities of a system (the “ilities”).
  - A NFR affects the architecture of a system.
  - Execution (e.g. security, usability)
  - Evolution (e.g. scalability, availability, reliability, maintainability, serviceability, interoperability, manageability)
  - Other constraints (e.g. space, time, regulations, standards)

Back to the Types of Requirements

- **User Requirement (Use Case)**
  - Represents a set of actions that an actor (user) requests from a system in order to obtain a tangible result.
- **Business Requirement (Goal)**
  - The specification of an intention with regards to the objectives, use or properties of a system.
  - A goal is a high-level objective of business, organization or system,
  - Goals represent the intentions of stakeholders.

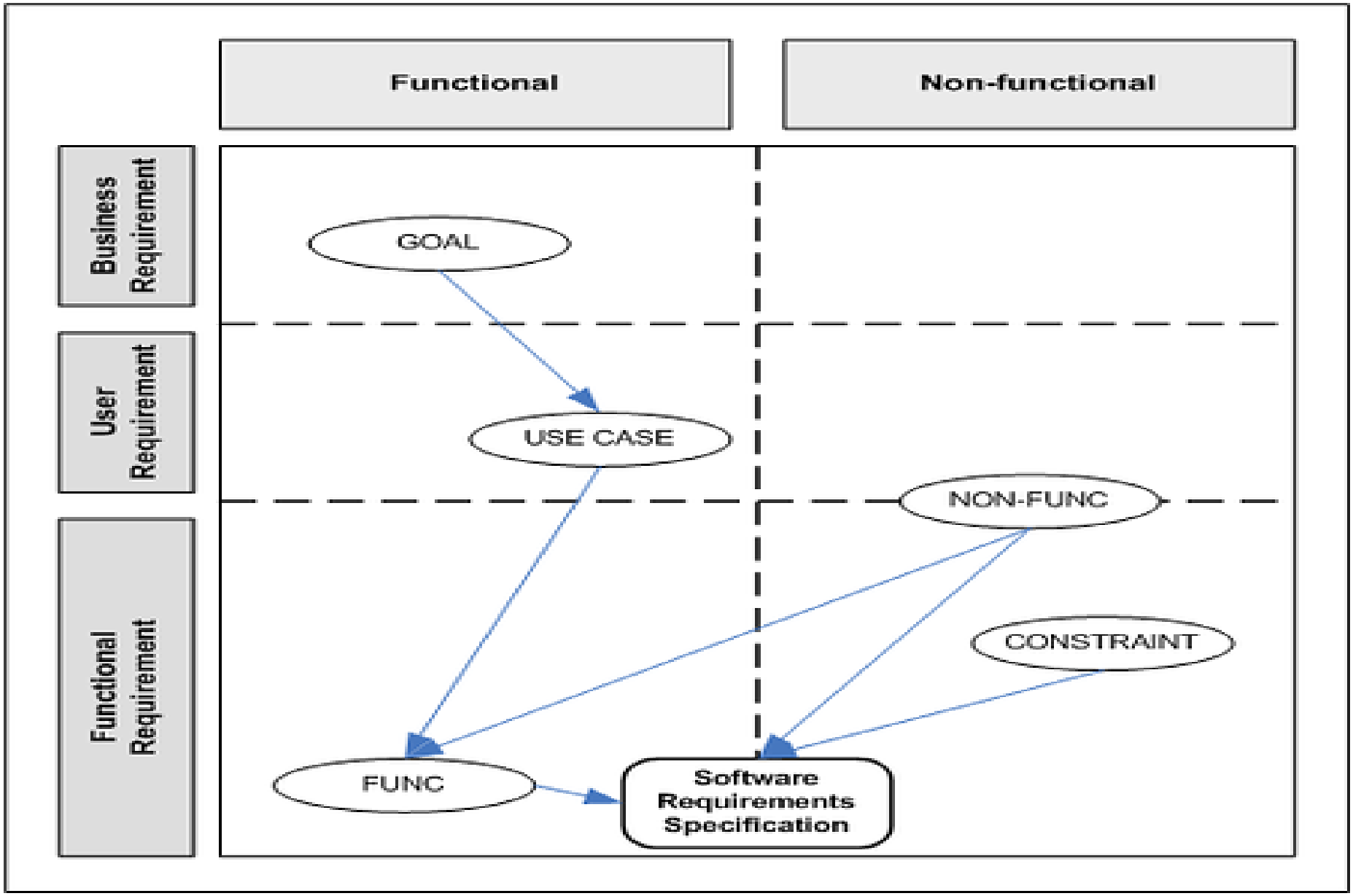
Note: the specific requirement ontology depends on nature of the project.

An Overview of the Core Types of Requirements



http://tynerblain.com/blog/2006/01/04/foundation-series-structured-requirements/

One Other Overview of the Types of Requirements



http://tynerblain.com/blog/2006/01/04/foundation-series-structured-requirements/

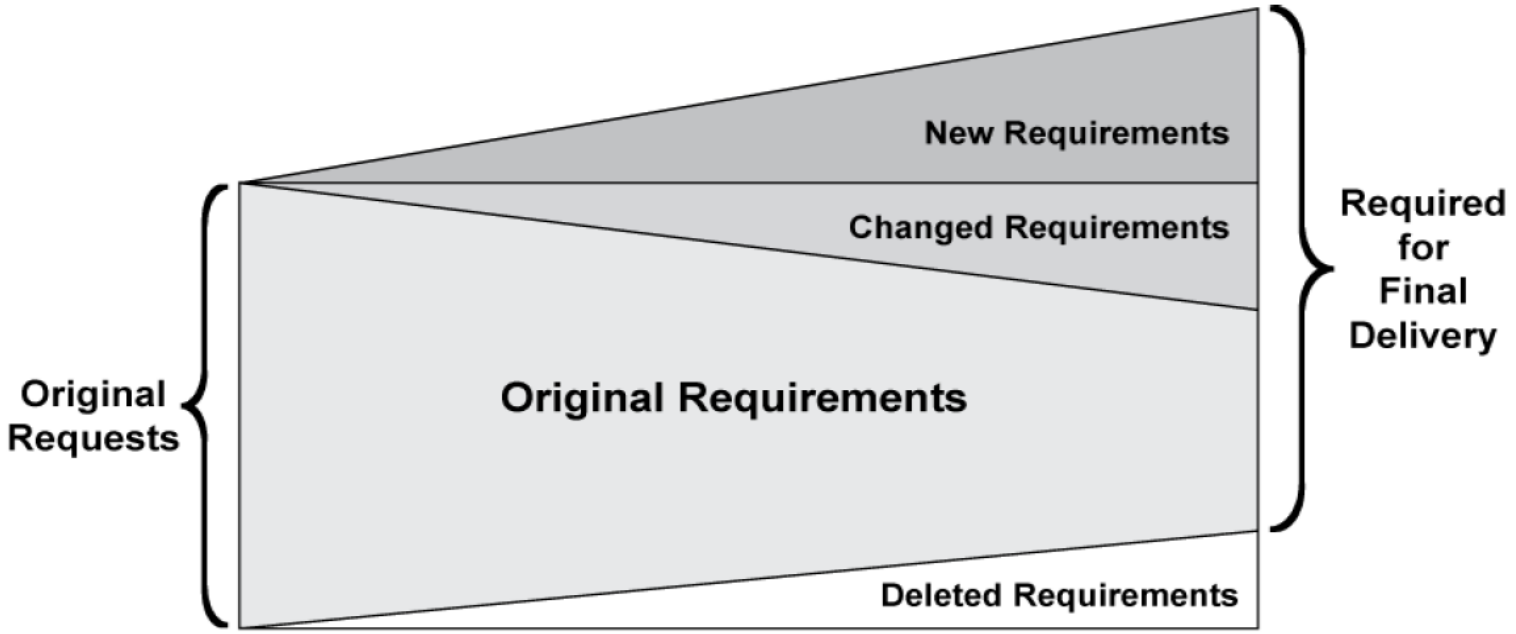
AMS

Requirements Engineering Fundamentals…

…the Requirements Engineering Process…

Requirements change over time

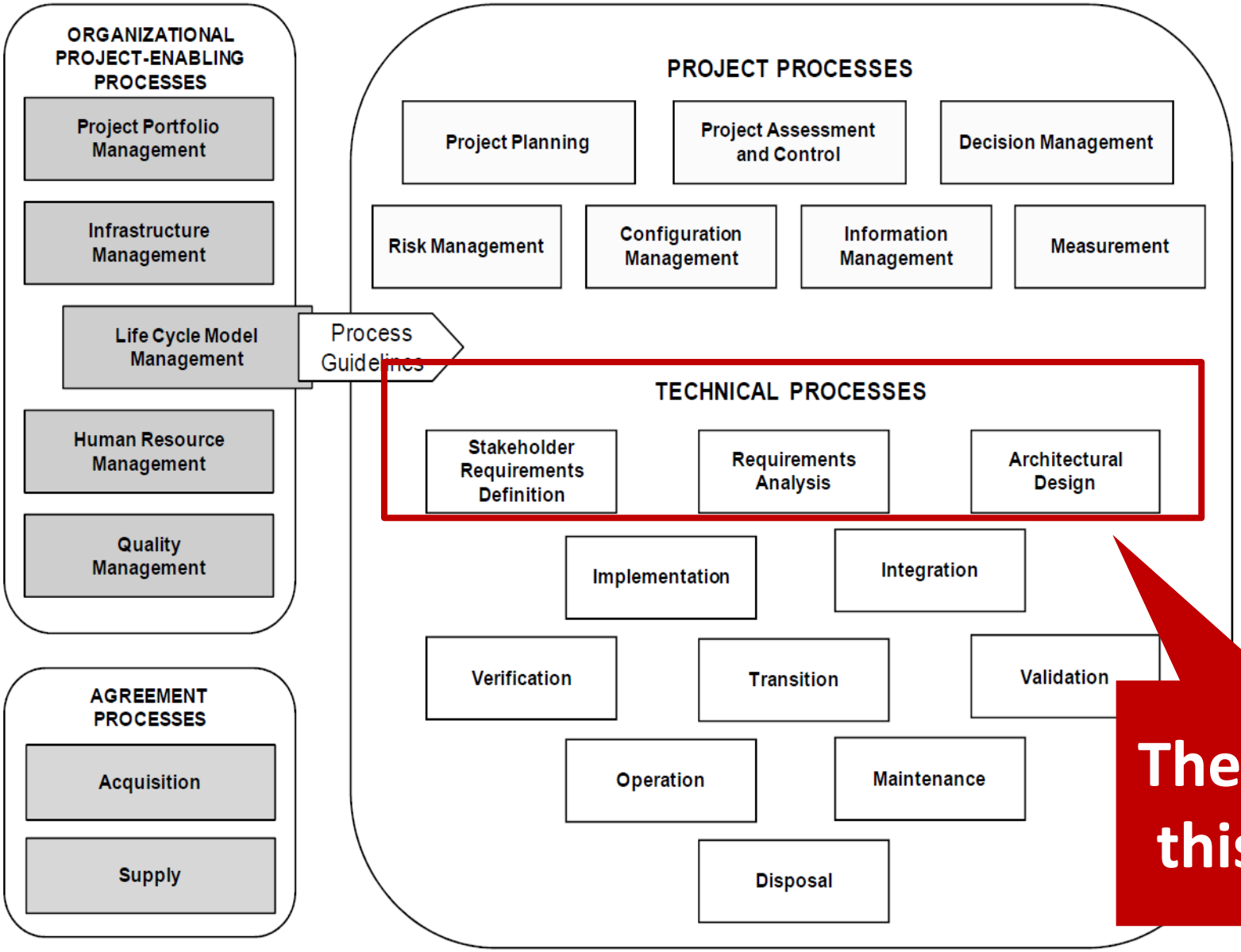
"**Change is inevitable.** The purpose of **requirement configuration management** is to establish and maintain control of requirements, documentation, and artifacts produced throughout the system's life cycle and to manage the impact of change on a project. Systems engineers ensure that the change is necessary, and that the most cost-effective recommendation has been proposed."



INCOSE Systems Engineering Handbook v. 3.2
INCOSE-TP-2003-002-03.2
January 2010 INCOSE

System Life-cycle Process Overview
(according to ISO/IEC 15288:2008)

INCOSE Systems Engineering Handbook v. 3.2
INCOSE-TP-2003-002-03.2
January 2010



The focus of this course

Requirements Engineering

Requirements engineering is a systematic approach to develop and manage the requirements life cycle of a system.

It is the branch of systems engineering concerned with the real-world goals for, functions of, and constraints of systems.

It is also concerned with the relationship of these factors to precise specifications of system behaviour, and to their evolution over time and across system families.

"Conceptual Modeling of Information Systems"[Olivé2007]

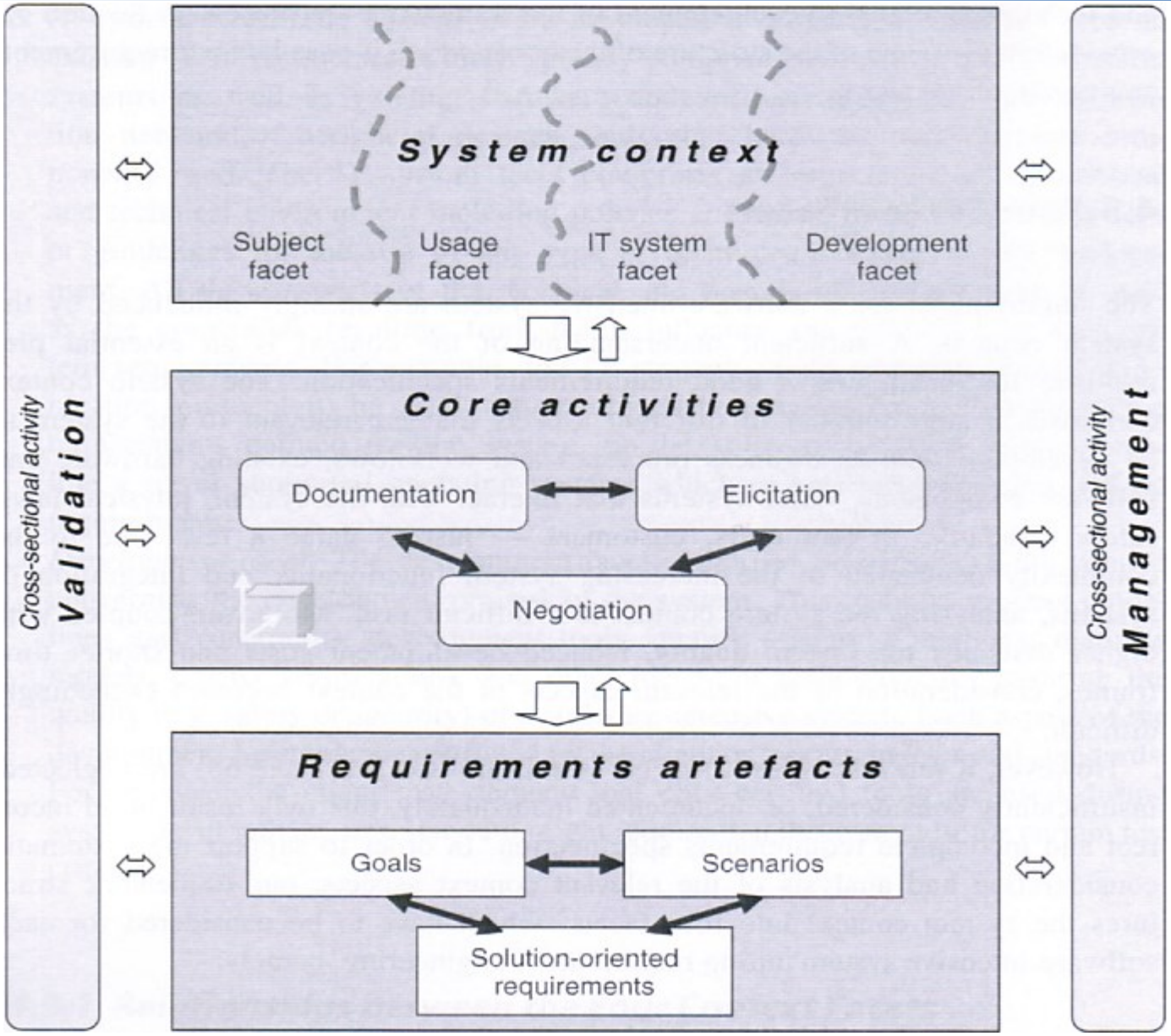
Requirements Engineering Key Concepts…

- **requirement:** (A) A condition or capability needed by a user to solve a problem or achieve an objective. (B) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, speciation, or other formally imposed document. (C) A documented representation of a condition or capability as in definition (A) or (B). (IEEE Std 610.12-1990)
- **raw requirement:** An environmental or customer requirement that has not been analyzed and formulated as a well-formed requirement. (IEEE 1233)
- **derived requirement:** A requirement deduced or inferred from the collection and organization of requirements into a particular system configuration and solution. (IEEE 1233)
- **well-formed requirement:** A statement of system functionality (a capability) that can be validated, and that must be met or possessed by a system to solve a customer problem or to achieve a customer objective, and is qualified by measurable conditions and bounded by constraints. (IEEE 1233)

Requirements Engineering Key Concepts…

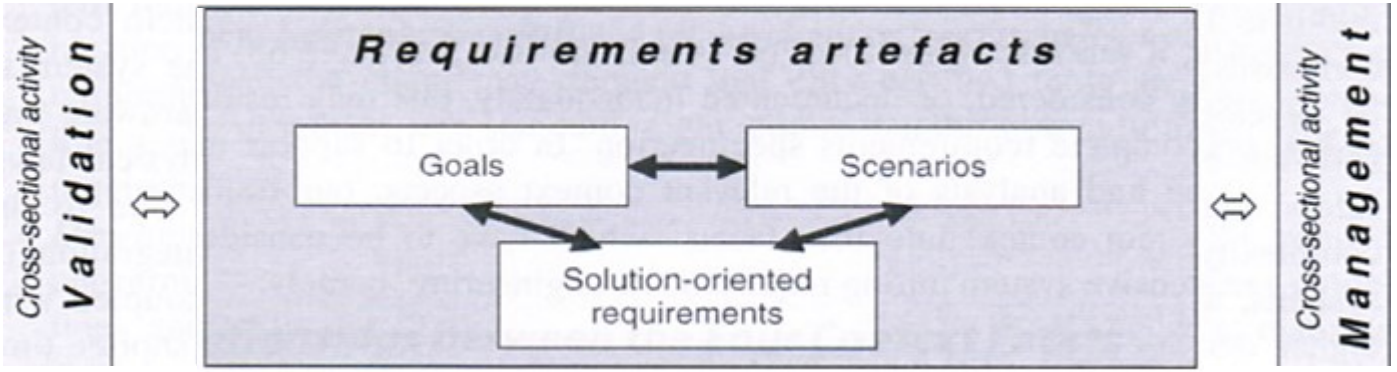
- **constraint**: A statement that expresses measurable bounds for an element or function of the system. That is, a constraint is a factor that is imposed on the solution by force or compulsion and may limit or modify the design changes. (IEEE 1233)
- **traceability**: The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; e.g., the degree to which the *requirements* and *design* of a given system element match. (IEEE Std 610.12-1990)
- **testability**: The degree to which a requirement is stated in terms that permit establishment of test criteria and performance of tests to determine whether those criteria have been met. (IEEE Std 610.12-1990)
- **verification**: The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. (IEEE Std 610.12-1990)
- **validation**: The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers (IEEE Std 610.12-1990)

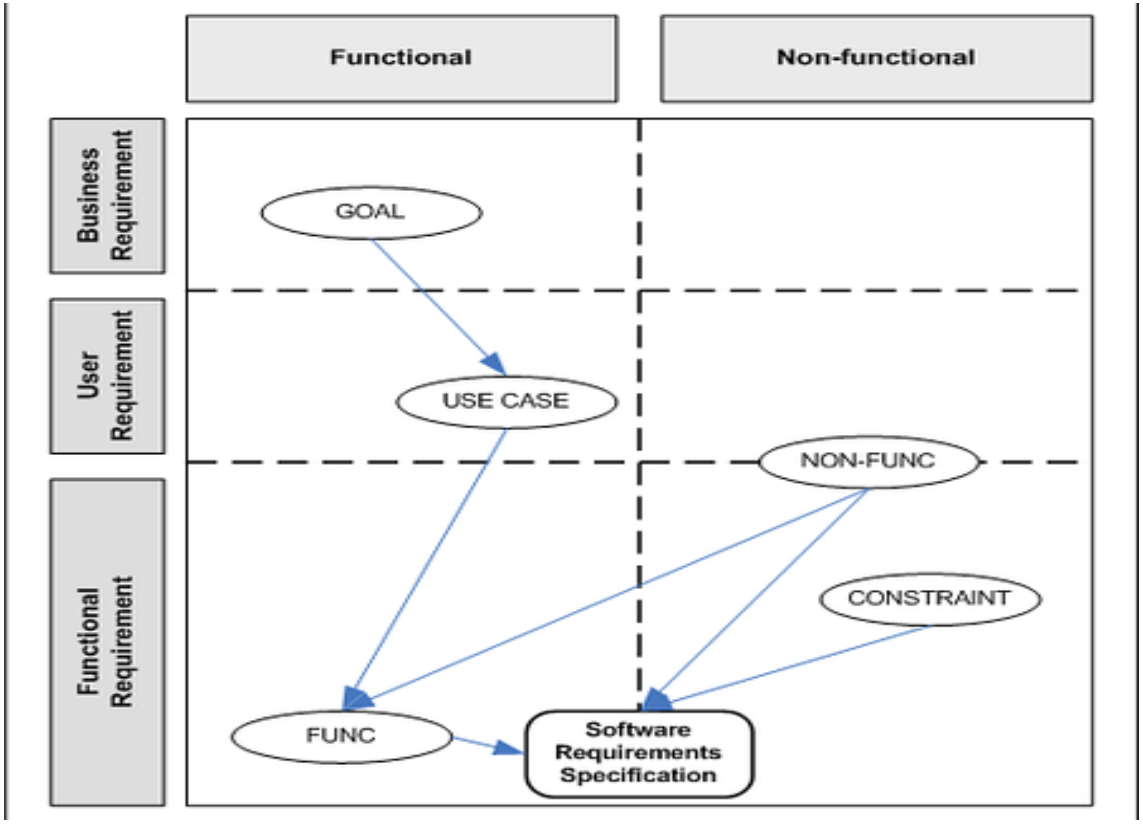
A Requirements Engineering Framework…



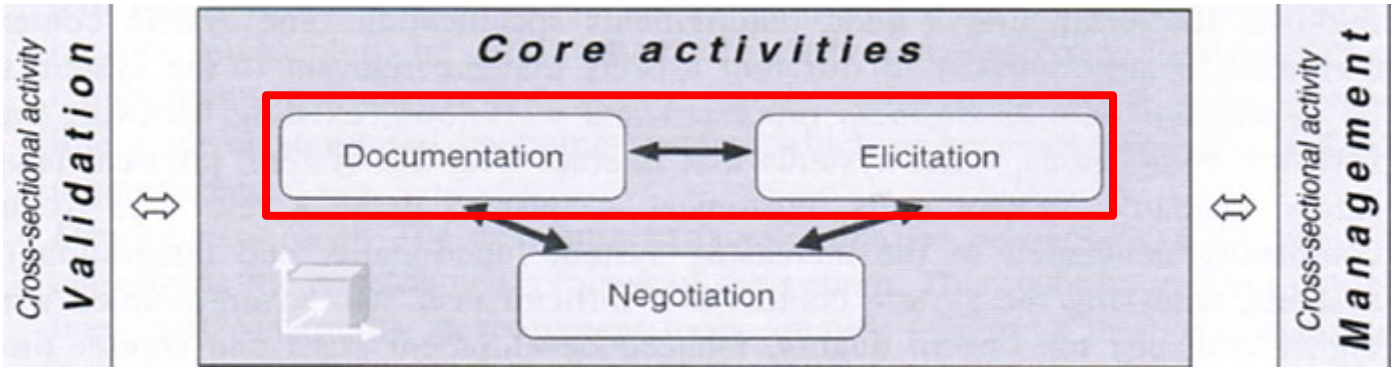
Facet ≈ Viewpoint

Requirements Artefacts





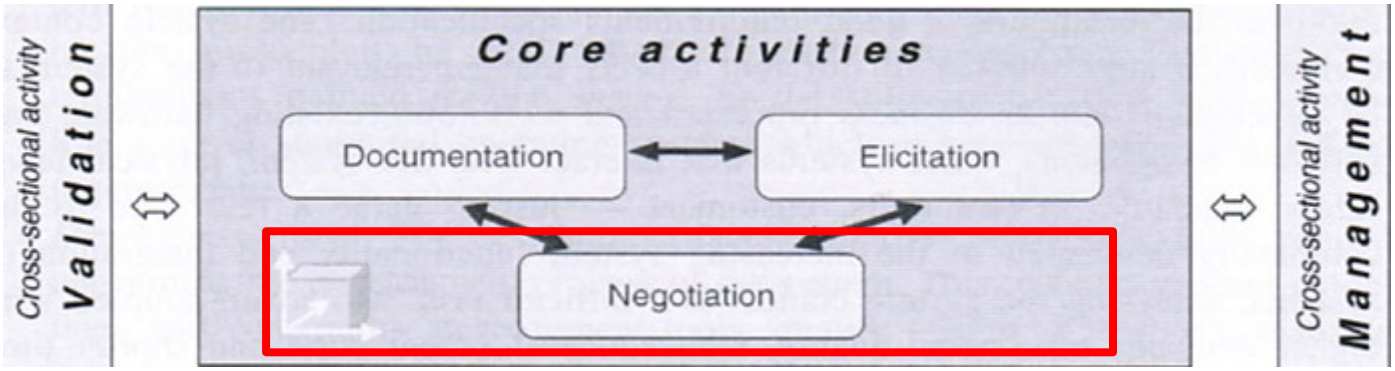
Requirements Engineering core activities



The goal of the elicitation activity is to improve the understanding of the requirements, i.e. to achieve progress in the content dimension. During the elicitation activity, requirements are elicited from stakeholders and other requirement sources. In addition, new and innovative requirements are collaboratively developed.

The focus of the documentation activity is the documentation and specification of the elicited requirements according to the defined documentation and specification rules. In addition, other important types of information such as rationale or decisions must be documented

Requirements Engineering core activities

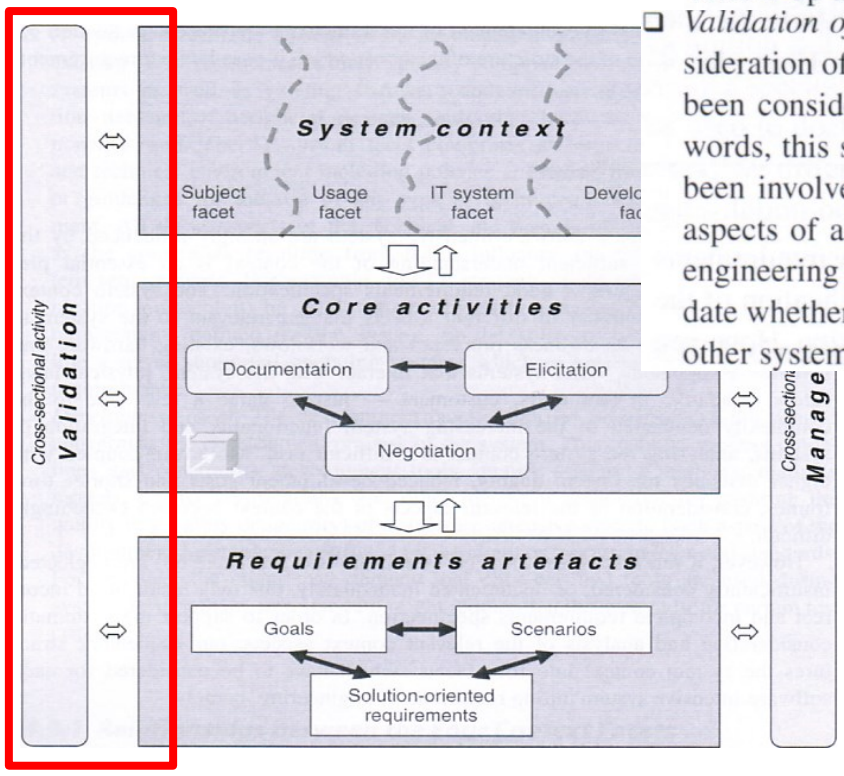


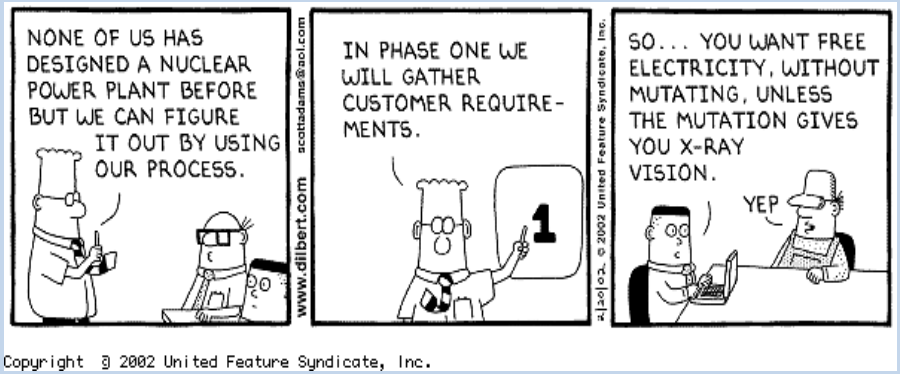
The goal of the negotiation activity is therefore twofold: First, all conflicts between the viewpoints of the different stakeholders have to be detected and made explicit. Second, the identified conflicts should be resolved (as far as possible). Depending on the cause of the conflict, different strategies can be applied for resolving it. At the beginning of the requirements engineering process, typically the viewpoints of the different stakeholders differ significantly. Ideally, at the end of the requirements engineering process, the negotiation activity has identified and resolved all conflicts which exist between the different stakeholders involved. The negotiation activity is described in more detail in Part IV.c.

Requirements Engineering core activities

The validation activity is a cross-sectional activity which consists of three sub-activities:

- *Validation of the requirements artefacts:* The validation of the requirements artefacts aims at detecting defects in the requirements. Only requirements with high quality provide a sound basis for the architectural design, the implementation of the system, and the development of test artefacts. Defects in requirements entail defects in the architecture, in the implementation, and in test artefacts.
- *Validation of the core activities:* The validation of the core activities (documentation, elicitation, and negotiation) has the goal of checking the compliance between the activities performed and the process and/or activity specifications. For example, one should validate whether the steps defined for an activity and the defined follow-up activities have been performed.
- *Validation of the consideration of the system context:* The validation of the consideration of the system context aims at validating whether the system context has been considered in the intended way during requirements engineering. In other words, this sub-activity aims at validating whether all relevant stakeholders have been involved in the process at the right time and whether the relevant context aspects of all four context facets have been considered during the requirements engineering process. For example, with respect to the usage facet, one should validate whether all required interactions between the system and its actors (users and other systems) have been elicited.





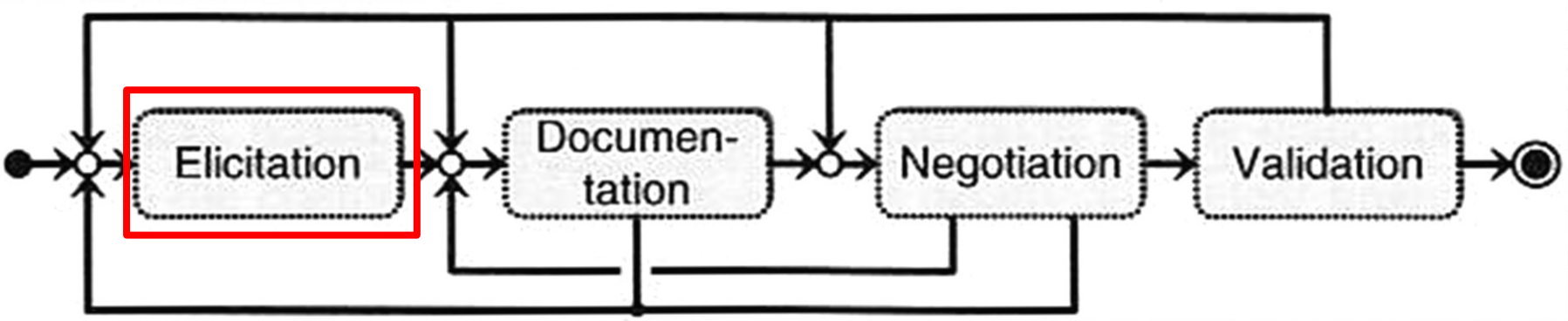
AMS

Requirements Engineering Fundamentals…

…the Requirements Engineering Process…

…requirements elicitation:

Requirements Elicitation



The goal of the elicitation activity is to improve the understanding of the requirements, i.e. to achieve progress in the content dimension. During the elicitation activity, requirements are elicited from stakeholders and other requirement sources. In addition, new and innovative requirements are collaboratively developed.

Definition 21-1: *Requirements elicitation*

Requirements elicitation is one of the three core requirements engineering activities. The goal of the elicitation activity is to:

(1) Identify relevant requirement sources
(2) Elicit existing requirements from the identified sources
(3) Develop new and innovative requirements

Requirements Elicitation

Many projects are driven by eager **project champions** who want "to get on with it." They succumb to the temptation to cut short the Concept Stage, and they use exaggerated projections to support **starting detailed design without adequate understanding of the challenges involved**.

Many commissions reviewing failed systems after the fact have identified insufficient or superficial study in the Concept Stage as a root cause of failure.

Requirements Elicitation

One of the biggest challenges in this activity is the identification of the set of stakeholders from whom requirements should be elicited. Customers and eventual end-users are relatively easy to identify, but regulatory agencies and other interested parties that may reap the consequences of the system-of-interest should also be sought out and heard. Stakeholders can include the interoperating systems and enabling systems themselves as these will usually impose constraints that need to be identified and considered. In sustainable development this includes finding representation for future generations.

Requirements Elicitation

- Requirements elicitation is the process of discovering the requirements for a system by communication with the stakeholders.
- This process can employ several techniques, such as:
  - Questionnaires
  - Analysis of documents
  - Interviewing
  - Joint Application Design workshops
  - Ethnography
  - Prototyping
  - Use Cases

Techniques for Requirements Elicitation

Suitability of the technique for the sub-activities		Identifying requirement sources	Eliciting existing requirements	Developing new and innovative requirements
Technique	**Effort**			
Interview	Medium to high	✓	✓	✓
Workshop	High to very high	✓	✓	✓
Focus groups	Medium to high		✓	✓
Observation	High to very high		✓	
Questionnaire	Low to medium	✓	✓	
Perspective-based reading	Medium to high		✓	

Techniques for Requirements Elicitation

Suitability of the technique for the sub-activities		*Identifying requirement sources*	*Eliciting existing requirements*	*Developing new and innovative requirements*
Technique	**Effort**			
Brainstorming	very low	✓		✓
Prototyping	depends on realisation technology		✓	✓
KJ method	very low	✓	✓	(✓)
Mind mapping	very low	✓	✓	✓
Elicitation checklists	very low	✓	✓	✓

KJ-Method (Basic Cycle)

1. **Card making**: all relevant facts and information are written on individual cards and collated (Post-its would do). In a group-work version, this step could be adapted to use BrainStorming or Constrained Brainwriting, to generate a supply of ideas on cards. The KJ-Method tends to place emphasis on the ideas being relevant, verifiable and important.
2. **Grouping and naming**: The cards are shuffled, spread out and read carefully. Cards that look as though they belong together should be grouped, ignoring any 'oddities'. For each group write an apt title and place it on top of its group of cards. Repeat the group making, using new titles and any 'oddities' to create higher-level groups. If you have more than about 10 groups, repeat this iterative process at yet higher levels.
3. **Redistribution**: At this stage in the group-work version, the cards are collected and reallocated in order than no one is given their own cards. One card is read out, and all contributors look through the cards in their own 'hand' of cards, and find any that seem to go with the one read out, so building a 'group'. A name is selected for the set that clearly portrays the contents of the cards in the set, but is neither too broad nor a simple aggregation of the cards in the group
4. **Chart making**: Now that you have less than 10 groups, some of which may contain sub-groups, sub-sub-groups, etc. arrange them carefully on a large sheet of paper in a spatial pattern that helps you to appreciate the overall picture.
5. **Explanation**: Now try to express what the chart means to you, writing notes as you go and being careful to differentiate personal interpretations from the facts contained in the chart. Ideas for the solution are often developed whilst explaining the structure of the problem.

Techniques for Requirements Elicitation – KJ method

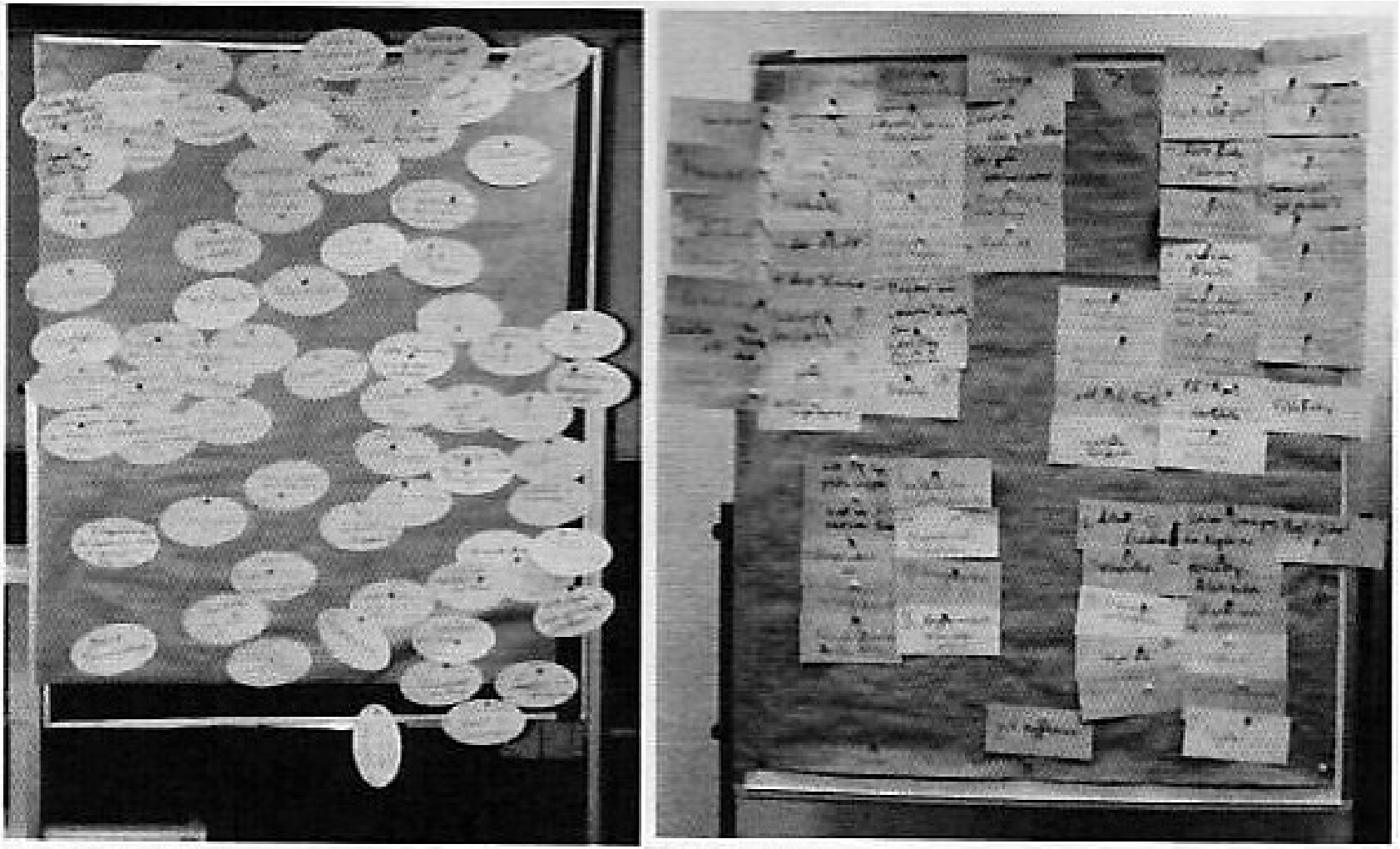
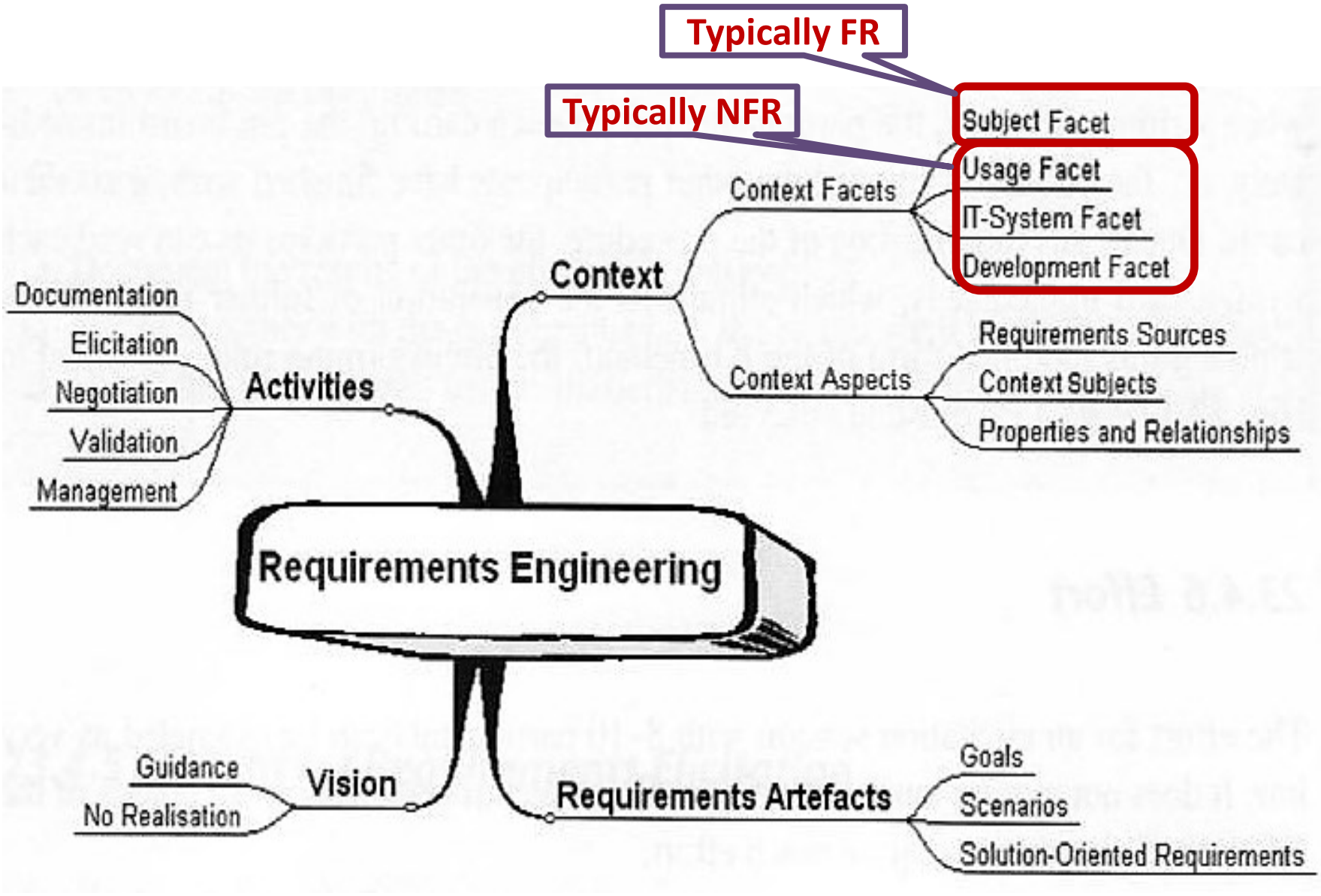


Fig. 23-6 *Documented results of an elicitation session using the KJ method*

Techniques for Requirements Elicitation – Mind maps



Questionnaires

- Relevant for
  - High number of stakeholders of the same kind, with and already established sense of the needs
  - Main focus on functional requirements
- Process
  - Select participants
  - Define questions
  - Define strategies to assure a high number of quality responses
  - Follow-up of the results to the stakeholders (for information, validation, engagement…)

Analysis of Documents

- Relevant in scenarios where systems or processes are already in place (scenario “as-is”) and there is a perceived need to change (scenario “to-be”, for the envisaged system)

- Typical documents
  - Forms
  - Reports
  - User manuals
  - …

Interviews

- Relevant for
  - Low number and well identified of stakeholders, especially if with the power to decide, highly specialized and with differentiated perspectives
  - Main focus on functional requirements
- Process
  - Prepare yourself carefully
  - Define clear objectives for the interviews
  - Select stakeholders and provide them details in advance, so they can prepare themselves
  - Define the questions carefully
  - Pay attention to “never”, “always”, “ever”, “easier”, “better” “simpler”,…, terms

JAD – Joint Application Design

- Relevant for
  - It is intended a new system where the requirements depend from a heterogeneous group of stakeholders representing different classes of users of decision makers.
  - Requires a relaxed organizational environment
- Process
  - Prepare several (2, 3, …) workshops in a functional place, with plenty of time (1 full day). Do it only if everyone can be present!!! 8 to 12 participants is the ideal size…
  - One of the participants must play a role of scribe.
  - One of the participants must play the role of moderator (special communication skills are required)
  - 1 or 2 members of the design team must be present but playing a passive role (just listening)
  - One high level executive must sponsor the workshop, for credibility.
- Discuss advantages and risks…

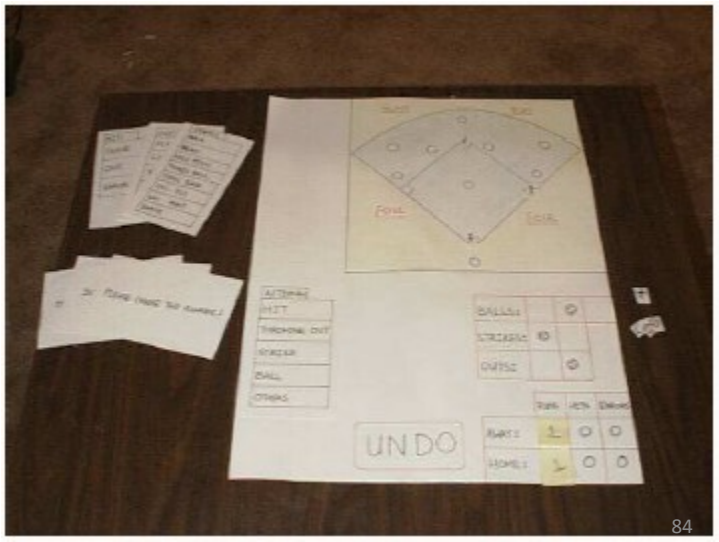
Ethnography

- Ethnography it is the practice of observing all aspects of a culture from within the culture. The challenge is to do that as an internal observer but without directing the outcome.

- It is a relevant observational technique to elicited social and organizational requirements, especially those derived from the way in which people actually work rather than the way in which documents, assumptions of process definitions say they ought to work.

Prototyping

- Prototypes can be built early, to provide insight into the general look and feel of the system.
- Prototypes can be:
  - Throw-away artefacts (especially in the case of physical systems) or
  - Evaluative, as early versions of the final systems themselves (especially in the case of software systems, developed with agile methodologies)
- Prototypes can be relevant because:
  - Requirements can be tested
  - Design alternatives can be explored
  - (Potential) Users can be involved
  - Problems can be identified early, saving money and time.



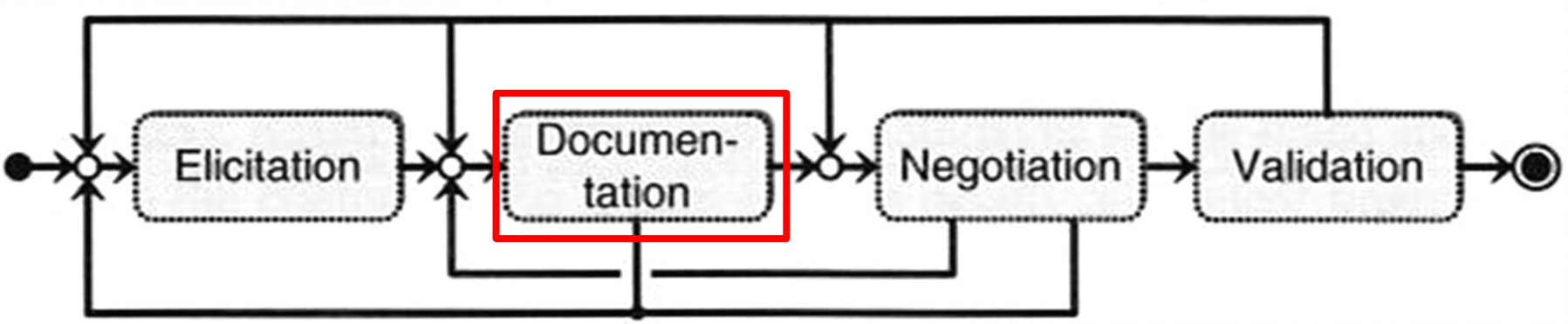
AMS

Requirements Engineering Fundamentals…

…the Requirements Engineering Process…

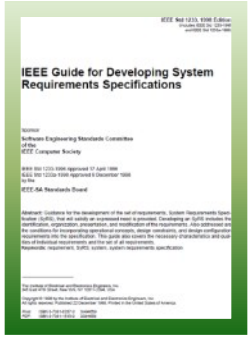
…requirements documentation:

Requirements Documentation



The focus of the documentation activity is the documentation and specification of the elicited requirements according to the defined documentation and specification rules. In addition, other important types of information such as rationale or decisions must be documented

Documenting requirements according to the IEEE 1233


IEEE Guide for Developing System Requirements Specifications

From IEEE 1233

An SyRS Outline

Table of contents
List of figures
List of tables

1. Introduction

1.1 System purpose
1.2 System scope
1.3 Definitions, acronyms, and abbreviations
1.4 References
1.5 System overview

2. General system description

2.1 System context
2.2 System modes and states
2.3 Major system capabilities
2.4 Major system conditions
2.5 Major system constraints
2.6 User characteristics
2.7 Assumptions and dependencies
2.8 Operational scenarios

3. System capabilities, conditions, and constraints

NOTE—System behavior, exception handling, manufacturability, and deployment should be covered under each capability, condition, and constraint.

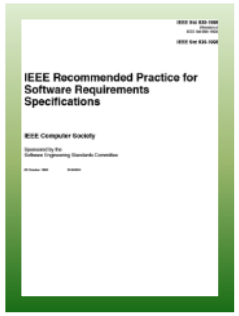
3.1 Physical
  3.1.1 Construction
  3.1.2 Durability
  3.1.3 Adaptability
  3.1.4 Environmental conditions
3.2 System performance characteristics
3.3 System security
3.4 Information management
3.5 System operations
  3.5.1 System human factors
  3.5.2 System maintainability
  3.5.3 System reliability
3.6 Policy and regulation
3.7 System life cycle sustainment

4. System interfaces

Documenting requirements according to the IEEE 830

Table of Contents

Figure 1—Prototype SRS outline



From IEEE 830

AMS

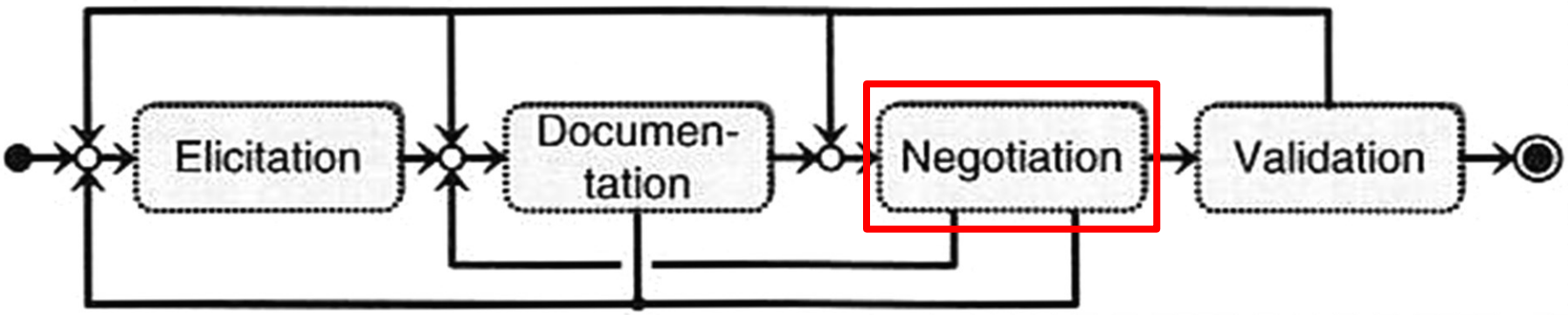
Requirements Engineering Fundamentals…

…the Requirements Engineering Process…

…requirements negotiation,

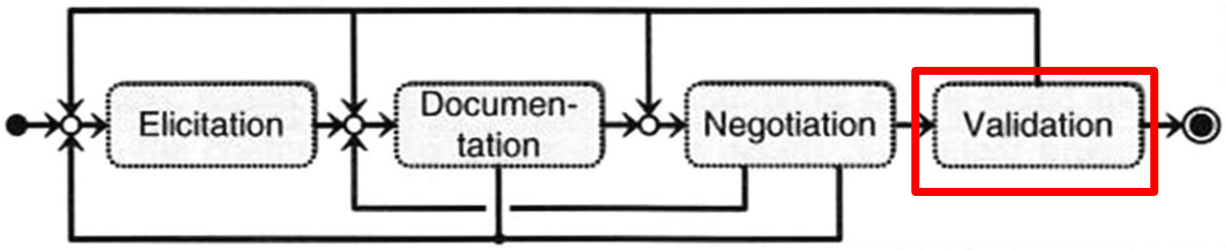
and requirements validation:

Requirements Negotiation



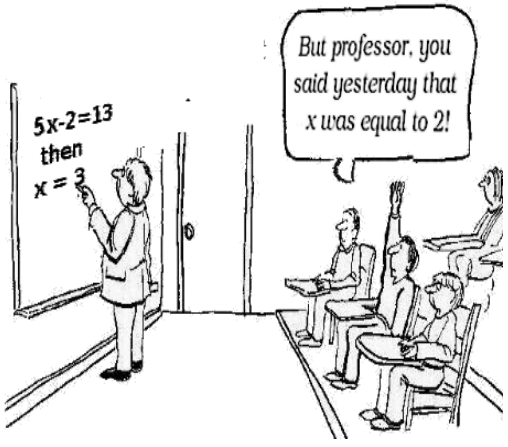
The goal of the negotiation activity is therefore twofold: First, all conflicts between the viewpoints of the different stakeholders have to be detected and made explicit. Second, the identified conflicts should be resolved (as far as possible). Depending on the cause of the conflict, different strategies can be applied for resolving it. At the beginning of the requirements engineering process, typically the viewpoints of the different stakeholders differ significantly. Ideally, at the end of the requirements engineering process, the negotiation activity has identified and resolved all conflicts which exist between the different stakeholders involved. The negotiation activity is described in more detail in Part IV.c.

Requirements Validation



The validation activity is a cross-sectional activity which consists of three sub-activities:

- *Validation of the requirements artefacts:* The validation of the requirements artefacts aims at detecting defects in the requirements. Only requirements with high quality provide a sound basis for the architectural design, the implementation of the system, and the development of test artefacts. Defects in requirements entail defects in the architecture, in the implementation, and in test artefacts.
- *Validation of the core activities:* The validation of the core activities (documentation, elicitation, and negotiation) has the goal of checking the compliance between the activities performed and the process and/or activity specifications. For example, one should validate whether the steps defined for an activity and the defined follow-up activities have been performed.
- *Validation of the consideration of the system context:* The validation of the consideration of the system context aims at validating whether the system context has been considered in the intended way during requirements engineering. In other words, this sub-activity aims at validating whether all relevant stakeholders have been involved in the process at the right time and whether the relevant context aspects of all four context facets have been considered during the requirements engineering process. For example, with respect to the usage facet, one should validate whether all required interactions between the system and its actors (users and other systems) have been elicited.



Requirements Validation

- Requirements validation: to show that the requirements actually define the intended system.

- Techniques:
  - **Requirements review**: analyze requirements systematically by one or a group of reviewers
  - **Prototyping**: validate requirements with potential users
  - **Test-case generation**: assure that requirements are testable
  - **Automated analysis**: requirements are expressed in tools with functions to check consistency…

Requirements Analysis

- The purpose is to find problems, errors, vagueness, inconsistencies, etc.

- The requirements analysis phase must be interlaced with the elicitation phase.
  - Check lists must be maintained

Requirements interaction matrix

- A technique to show interactions between requirements, stressing conflicts and overlaps:
  - 0 => Independent requirements
  - 1 => Conflicting requirements
  - 1000=> Overlapping requirements (they state the same, totally or partially)

Requirement	R1	R2	R3	R4	R5	R6
R1	0	0	1000	0	1	1
R2	0	0	0	0	0	0
R3	1000	0	0	1000	0	1000
R4	0	0	1000	0	1	1
R5	1	0	0	1	0	0
R6	1	0	1000	1	0	0

AMS

Requirements Engineering Fundamentals…

…**modelling requirements**,

…and **traceability:**

Requirements Traceability

Definition 31-1: *Requirements traceability*

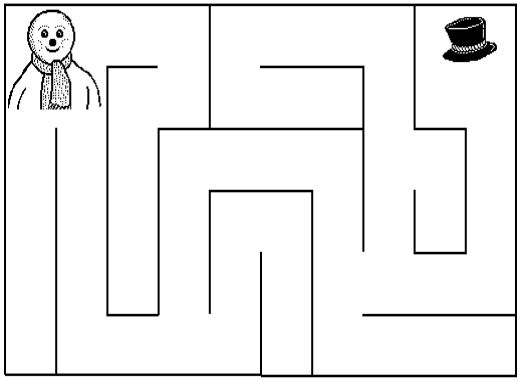
"Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases)."

[Gotel and Finkelstein 1994]

Requirements Traceability

According to [IEEE Std 610.12-1990] traceability denotes, in general, the degree to which a relationship can be established between different development artefacts, especially artefacts having a predecessor–successor– or master–subordinate–relationship to one another.

A requirement is considered to be traceable if the origin of the requirement as well as its further use in the development process can be traced (see [IEEE Std 830-1998; Edwards and Howell 1992; Hamilton and Beeby 1991; Gotel and Finkelstein 1994; Pohl 1996b]). A prerequisite for the traceability of requirements throughout the development process is that each requirement has a unique identifier. We adopt the following definition of requirements traceability from [Gotel and Finkelstein 1994]:



Help Frosty find his hat

Requirements Traceability

Motivation for Requirements Traceability

Recorded requirements traceability information supports various system development activities. For example, an impact analysis of a requirements change depends on the quality of the recorded traceability information. Table 31-1 provides some examples of the use of requirements traceability information in different system development activities (see, e.g., [Pohl 1996a; Ramesh 1998]).

The quality of the traceability information influences the quality of the system developed. Establishing a sophisticated traceability approach is therefore an important aspect of maturity models such as CMMI [CMMI 2006] or SPICE [Van Loon 2004; Hörmann et al. 2006].

AMS

Requirements Engineering Fundamentals…

…tools…

…languages…

…and templates:

Requirements Management Tools

45 Best Requirements Management Tools (Complete List)

45 Best Requirements Management Tools:

As the word states 'Requirements Management' means the process of managing requirements or the needs of any product. For a successful delivery of the best quality product, the requirements should be in its best place.

For a successful delivery of the best quality product, the requirements should be in its best place.

Similarly, in IT industry requirements place a very important role for successful user satisfaction and software/application delivery with minimum issues.

TOP 45 MOST IMPORTANT REQUIREMENT MANAGEMENT TOOLS

By www.SoftwareTestingHelp.com

To handle requirements efficiently to make its full and accurate use, the industry has emphasized on the requirement management system. This is a process in which the expectations from the customers/ stakeholders are documented, analyzed, traced, agreed upon, monitored, versioned and prioritized.

Requirement Management also plays a vital role in notifying the stakeholders about the changes in the requirement, if any! Managing Requirement functions as long as the Project is alive and functioning.

In other words, this is a continuous process which happens throughout the project cycle. In requirement management system, all the needs of the user are considered for the core and managed in a proper

Capterra® The Smart Way to Find Business Software

Software Categories

Requirements Management Software

Find the best Requirements Management Software for your business. Compare product reviews and fea

Learn Why Capterra is Free

Sort by: Sponsored

Learn about sort options

Filter software products: 48 products found

Product Rating | Users | Deployment | Features

Orcanos ALM and QMS

by Orcanos

Visit Website

(0 reviews)

Orcanos Requirements management tool is an affordable one-stop-shop cloud solution, for tracking and managing requirements and testing, as part of Orcanos integrated ALM and QMS platform. Orcanos ALM provides Requirements Management, Test Management, FMEA Risk Management, Document Management, and Quality Management procedures, on a single repository. The great power of Orcanos is in the integration between R&D, Quality, and Regulation. Learn more about Orcanos ALM and QMS

Add to Compare

Related C

Product Life Software

Project Mana

Task Management Software

Process Street

by Goodwinds

View Profile

(282 reviews)

Process Street is the easiest way to manage your teams workflows,recurring processes (BPM) and standard operating procedures (SOPs). Learn more about Process Street

Add to Compare

codeBeamer

by Intland Software

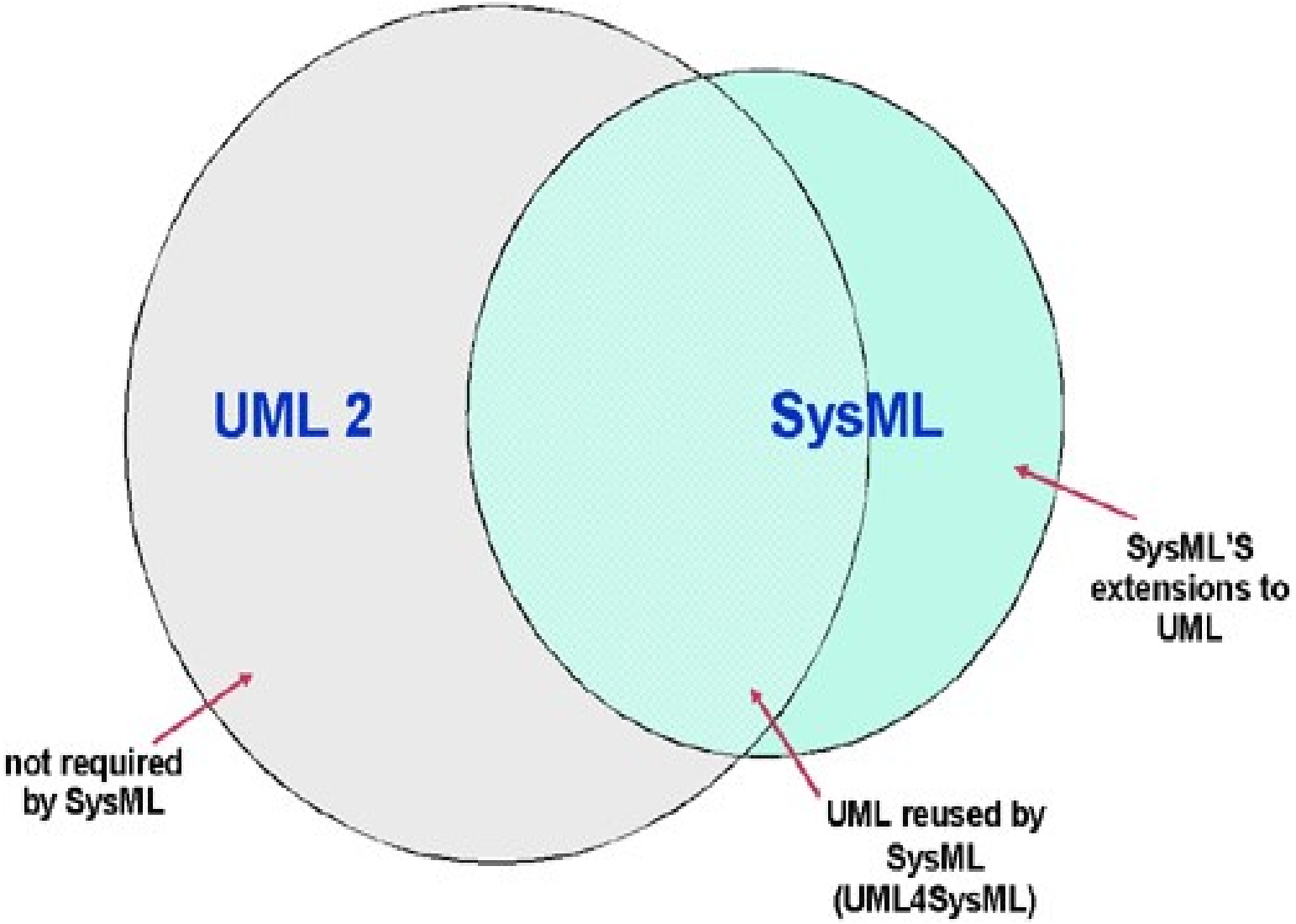
View Profile

(7 reviews)

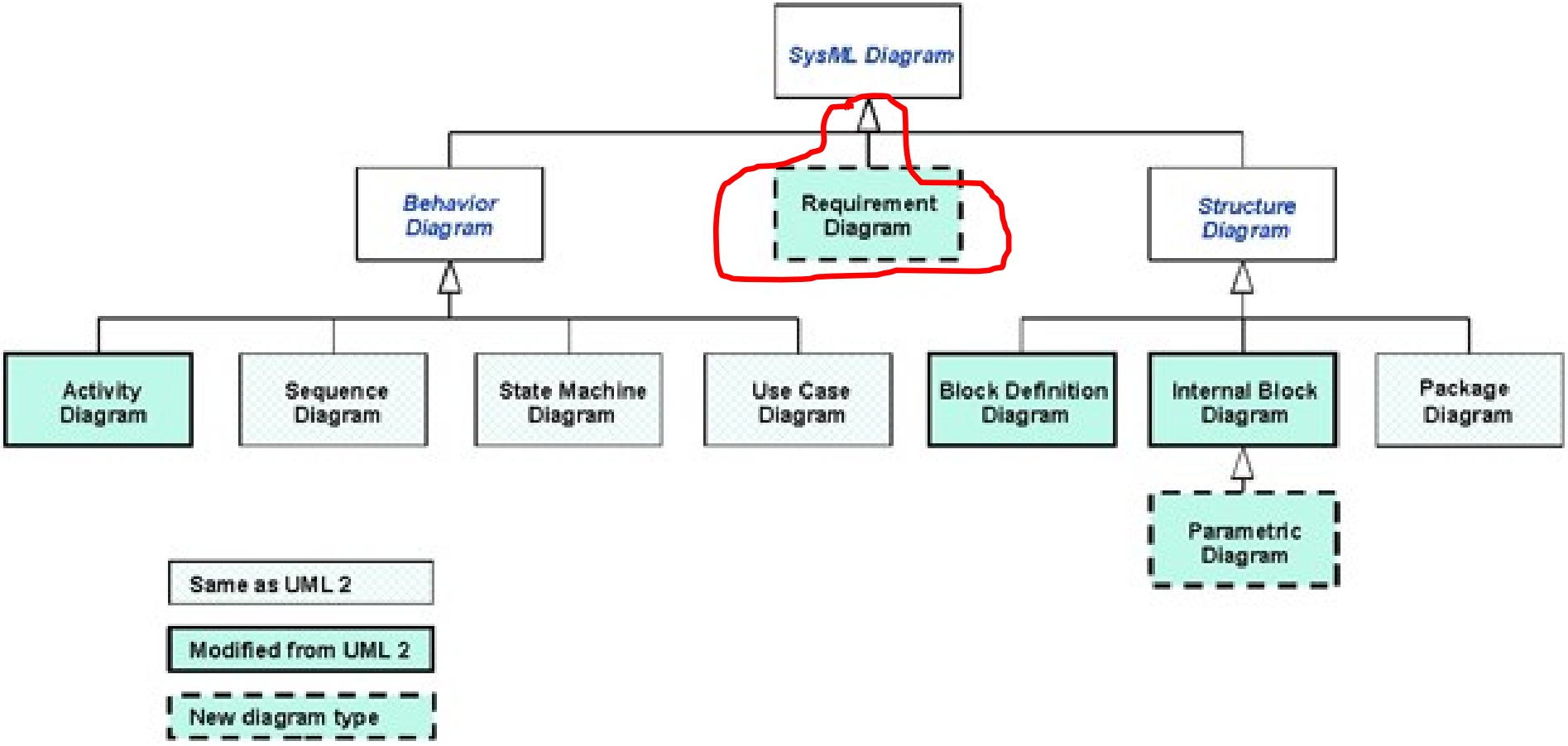
codeBeamer ALM helps you develop better products faster, and easily comply with safety-critical regulations.

Learn more about codeBeamer

UML and SysML

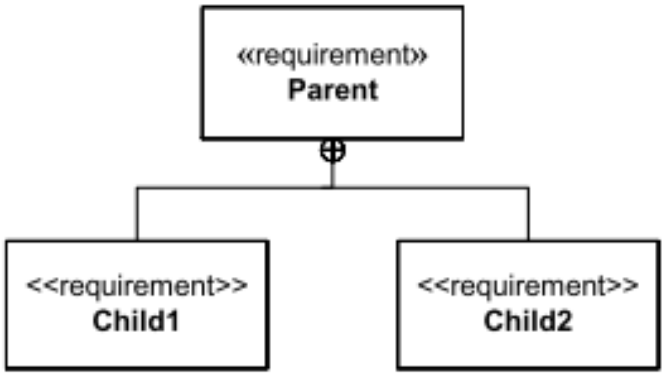


SysML


SysML Diagram
Behavior Diagram
Requirement Diagram
Structure Diagram
Activity Diagram
Sequence Diagram
State Machine Diagram
Use Case Diagram
Block Definition Diagram
Internal Block Diagram
Package Diagram
Parametric Diagram
Same as UML 2
Modified from UML 2
New diagram type

Requirements relationships in SysML

The **containment** relationsl refers to the practice of decomposing a complex requirement into simpler, single requirements.

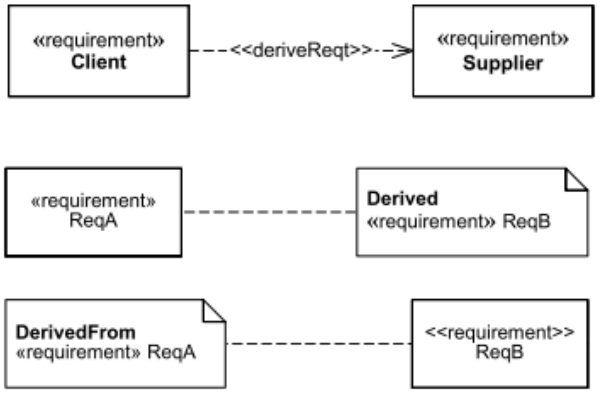


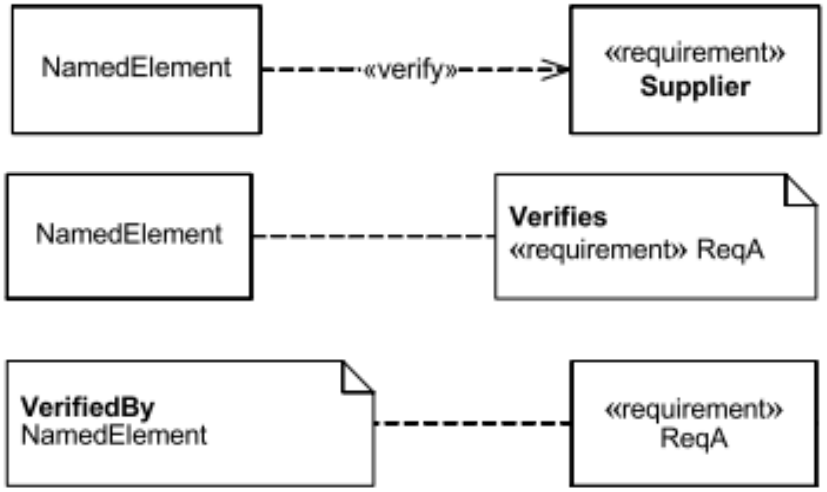
A **callout notation** can be used to represent derive, satisfy, verify, refine, copy, and trace relationships.

Requirements relationships in SysML

A **deriveReqt** relationship is a dependency between two requirements in which a client requirement can be derived from the supplier requirement.

- For example, a system requirement may be derived from a business need, or lower-level requirements may be derived from a system requirement.
- As with other dependencies, the arrow direction points from the derived (client) requirement to the (supplier) requirement from which it is derived.



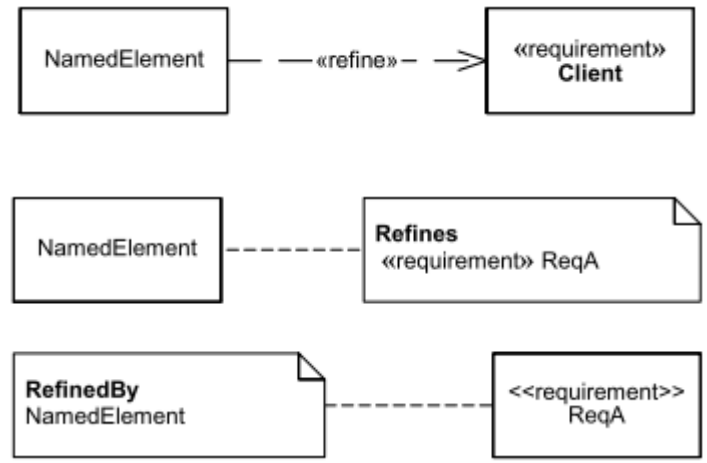


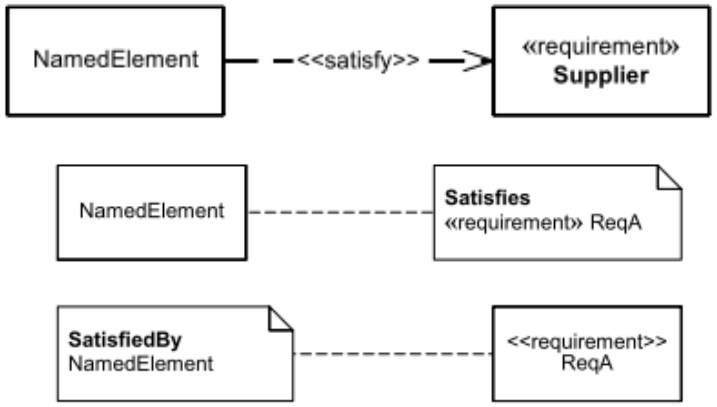
A **verify** relationship is a dependency between a requirement and a test case or other model element that can determine whether a system fulfills the requirement.

Requirements relationships in SysML

The **refine** relationship can be used to describe how a model element or set of elements can be used to further refine a requirement.

For example, a use case or activity diagram may be used to refine a text-based functional requirement. Alternatively, it may be used to show how a text-based requirement refines a model element. In this case, some elaborated text could be used to refine a less fine-grained model element.





A **satisfy** relationship is a dependency between a requirement and a model element that fulfills the requirement.

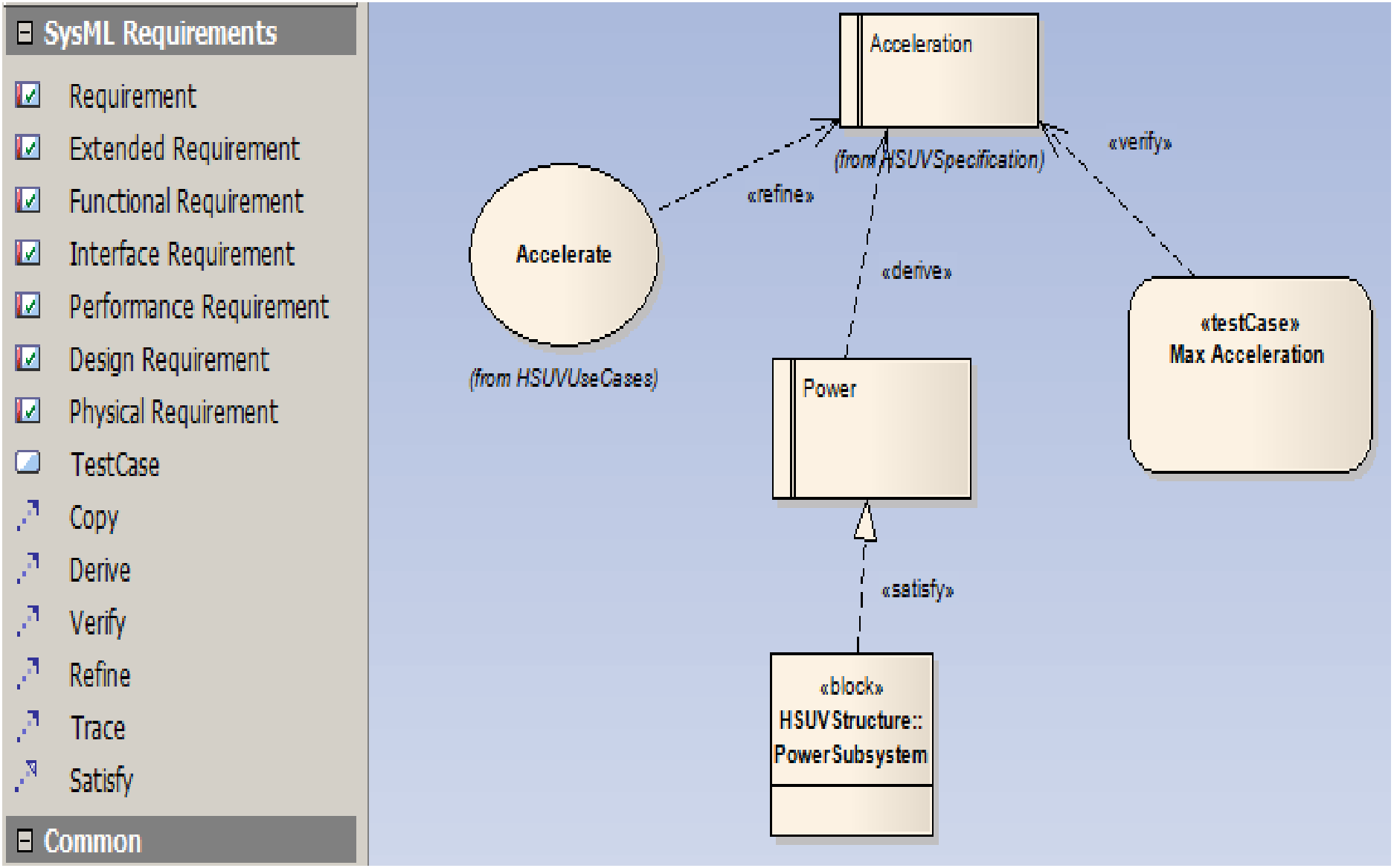
Requirement Relationships

The aim of the relationships is to simplify the model and not to obfuscate it!

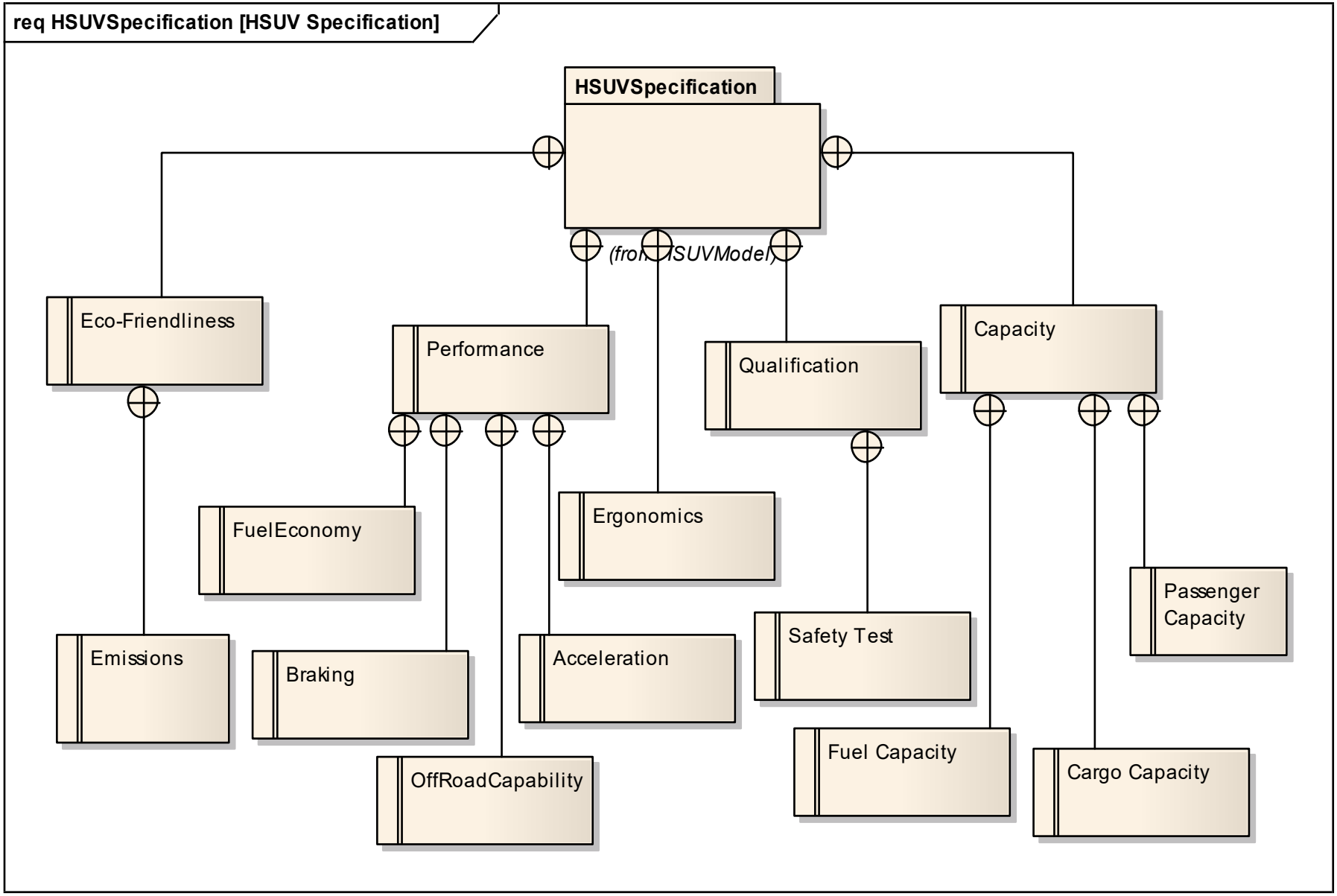
The relationships must be used to facilitate communication and must always add value to the model!

Complex relationships must be used with utmost care.

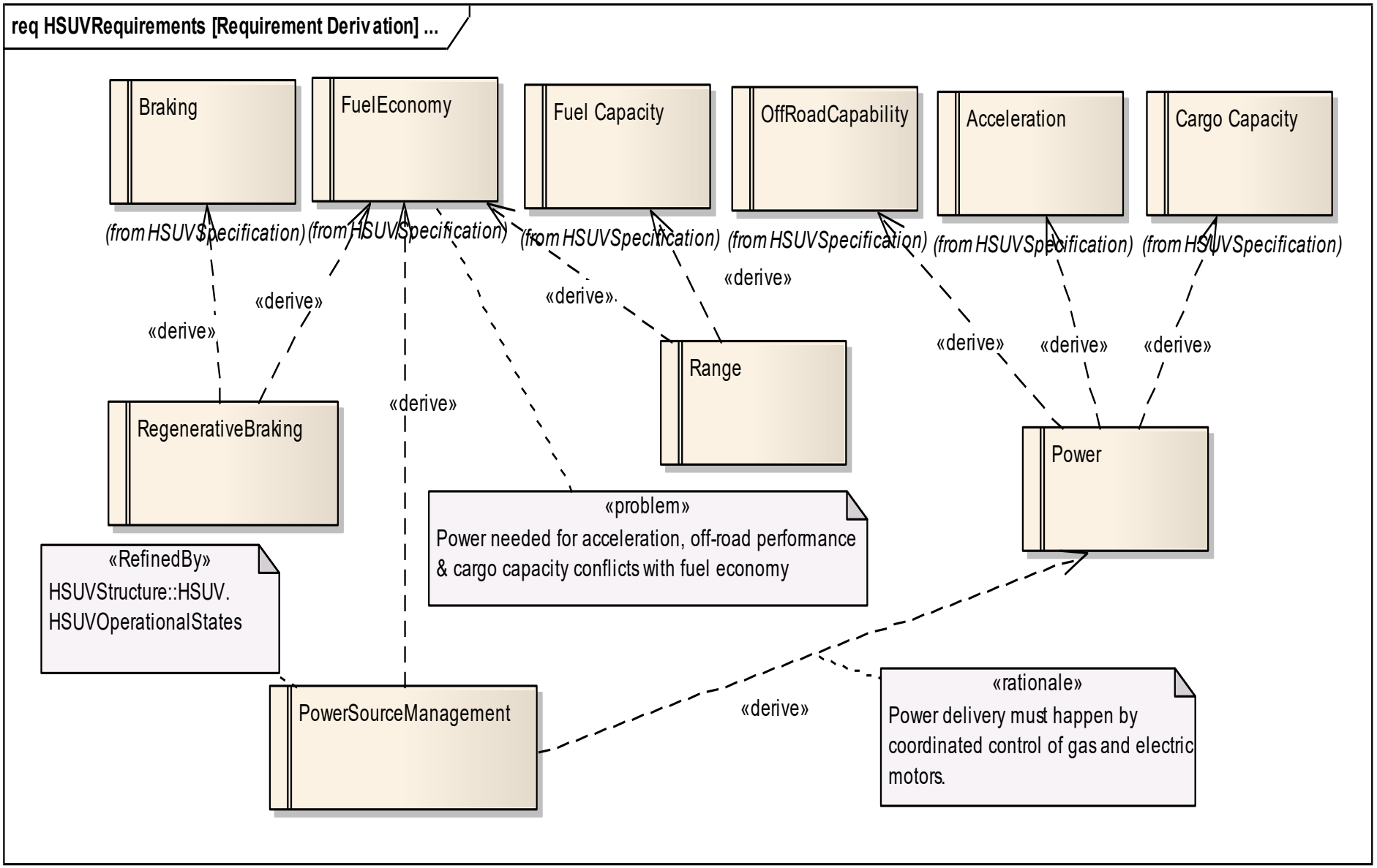
Relating requirements with other elements



Requirements specification in SysML



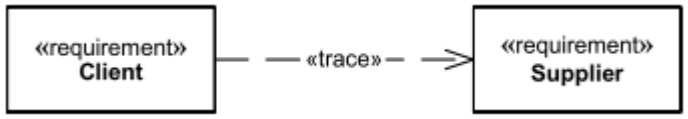
Requirements derivation in SysML

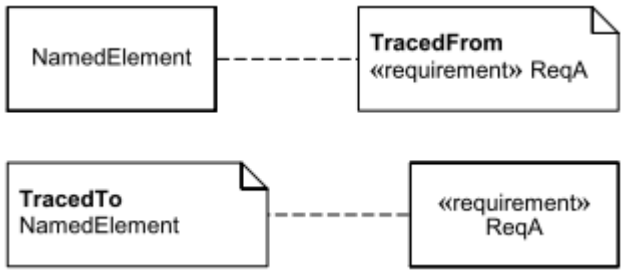


Requirements relationships in SysML

A generic **trace** relationship provides a general-purpose relationship between a requirement and any other model element. The semantics of trace include no real constraints and therefore are weak.

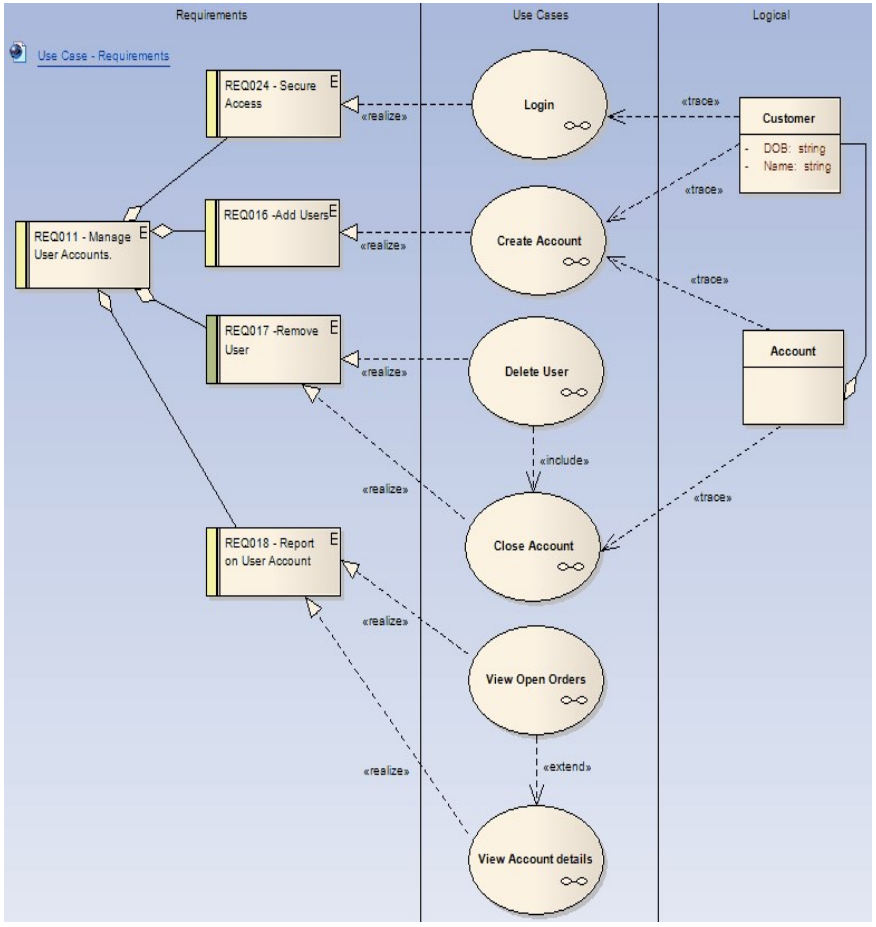
As a result, it is recommended that the trace relationship not be used in conjunction with the other requirements relationship.

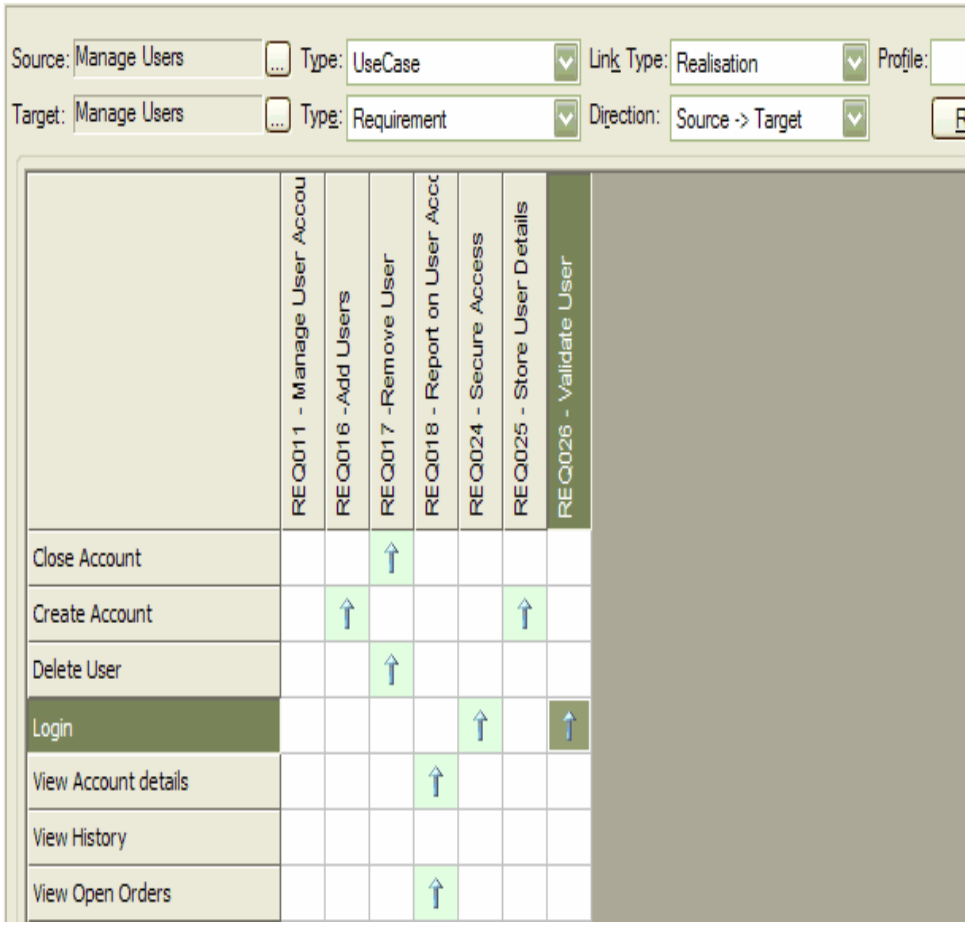




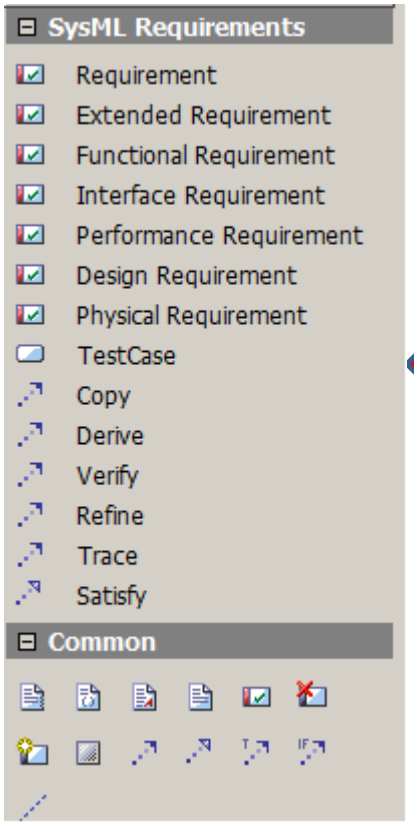
Traceability

A fundamental technique to relate decisions with their origin and/or implications.





Requirements in the Enterprise Architect Tool



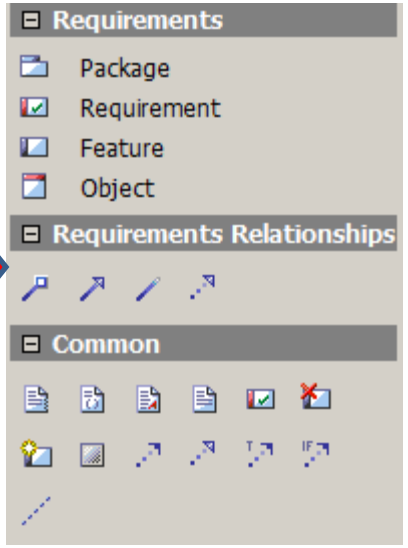
IMPORTANT

Be aware that EA tool supports requirements diagrams as

formal SysML Requirements Diagrams

or as

informal requirements diagrams as extensions to UML



User Stories (in Agile approaches)

User Stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

Common Template:

As a *<type of user>* ***I want*** *<some goal>* ***so that*** *<some reason>*

User Stories (in Agile approaches)

Examples

Jobs

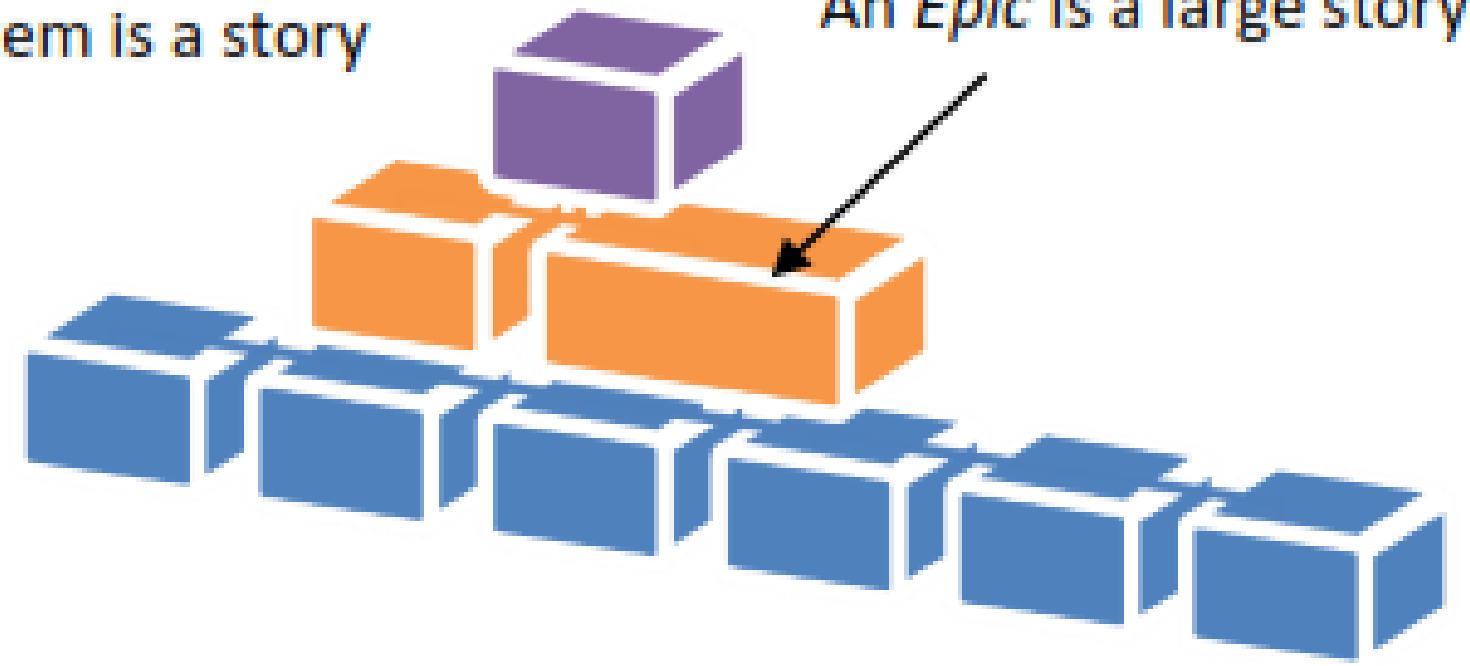
- As a site member (?), I can scroll through a listing of jobs. (There won't be enough at first to justify search fields.)
- As someone who wants to hire, I can post a "help wanted ad".
- As a site admin, I need to approve each help wanted ad before it gets to the site.
- As a site admin, I am emailed whenever a job is submitted (so that I am aware of it and can decide if I want to post it).
- As a site member, I can subscribe to an RSS feed of jobs available.
- As a site admin, I can edit and delete help wanted ads.
- As a site admin, I want jobs to stop publishing on the site 30 days after being posted. (Note: 30 days doesn't need to be configurable at this point. Hardcoding is fine for now.)
- As someone who wants to hire, I want to be able to extend an ad for another 30 days (repeatedly) by visiting the site and updating the posting. (Note, I can't update it 10 times today and extend the posting 300 days today.)

User Stories (in Agile approaches)

More Concepts

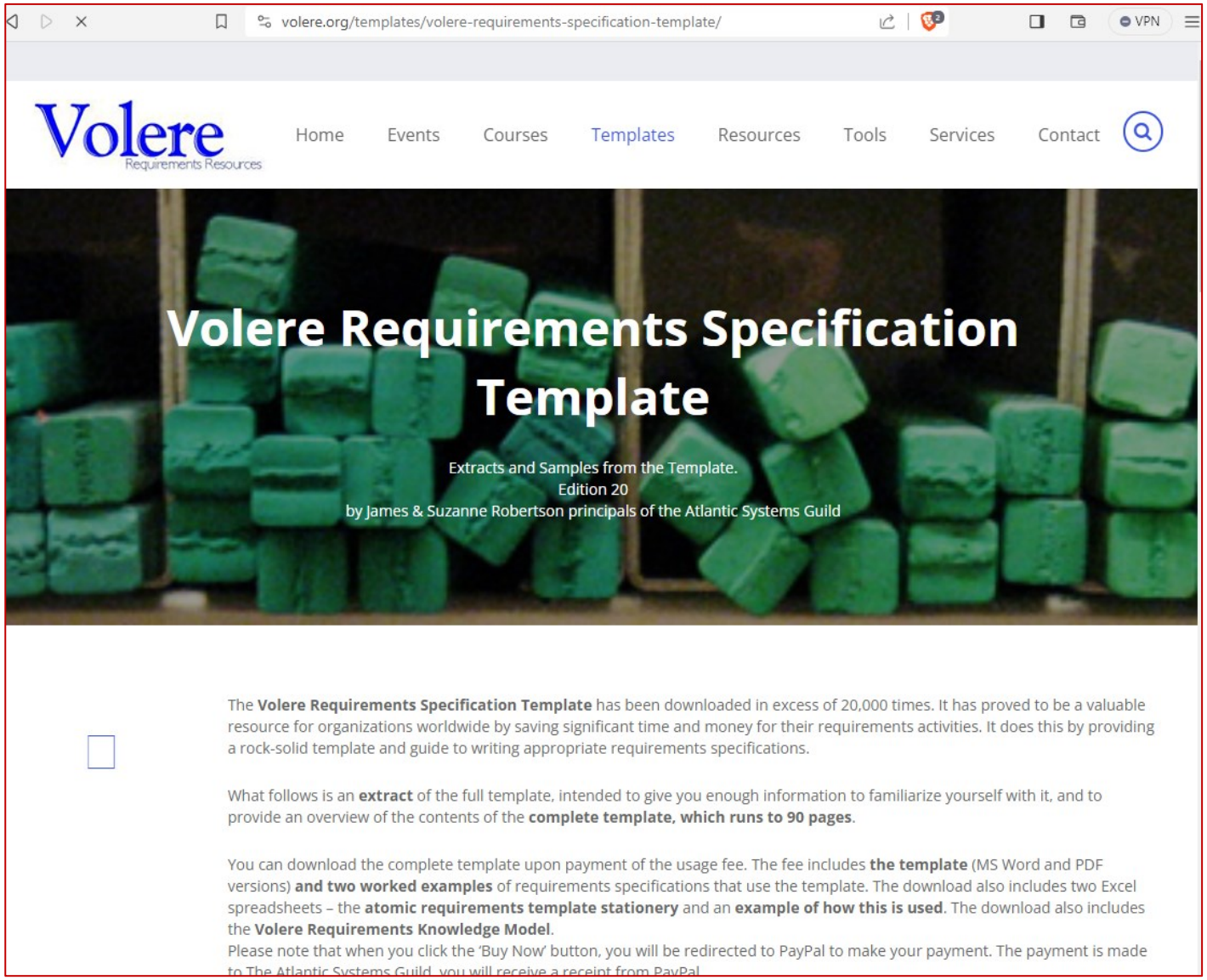
Each backlog item is a story

An *Epic* is a large story

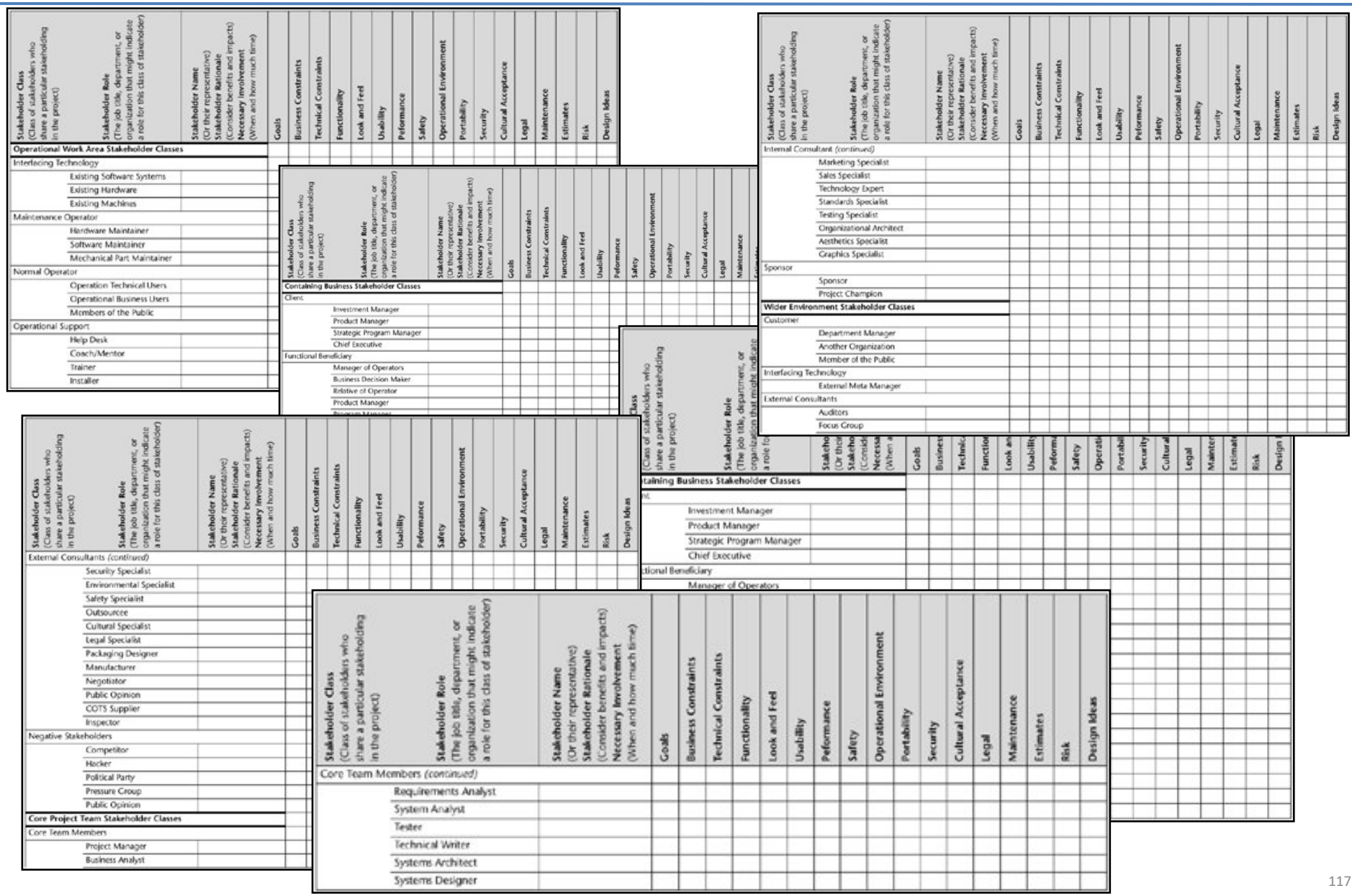


A *Theme* is a collection of related stories

Templates for requirements...



Volere template



Stakeholder Class (Class of stakeholders who share a particular stakeholding in the project)	Stakeholder Role (The job title, department, or organization that might indicate a role for this class of stakeholder)	Stakeholder Name (Or their representative) Stakeholder Rationale (Consider benefits and impacts) Necessary Involvement (When and how much time)	Goals	Business Constraints	Technical Constraints	Functionality	Look and Feel	Usability	Peformance	Safety	Operational Environment	Portability	Security	Cultural Acceptance	Legal	Maintenance	Estimates	Risk	Design Ideas
Operational Work Area Stakeholder Classes																			
Interfacing Technology																			
	Existing Software Systems																		
	Existing Hardware																		
	Existing Machines																		
Maintenance Operator																			
	Hardware Maintainer																		
	Software Maintainer																		
	Mechanical Part Maintainer																		
Normal Operator																			
	Operation Technicel Users																		
	Operational Business Users																		
	Members of the Public																		
Operational Support																			
	Help Desk																		
	Coach/Mentor																		
	Trainer																		
	Installer																		

Stakeholder Class (Class of stakeholders who share a particular stakeholding in the project)	Stakeholder Role (The job title, department, or organization that might indicate a role for this class of stakeholder)	Stakeholder Name (Or their representative) Stakeholder Rationale (Consider benefits and impacts) Necessary Involvement (When and how much time)	Goals	Business Constraints	Technical Constraints	Functionality	Look and Feel	Usability	Peformance	Safety	Operational Environment	Portability	Security	Cultural Acceptance	Legal	Maintenance	Estimates	Risk	Design Ideas
Containing Business Stakeholder Classes																			
Client																			
	Investment Manager																		
	Product Manager																		
	Strategic Program Manager																		
	Chief Executive																		
Functional Beneficiary																			
	Manager of Operators																		
	Business Decision Maker																		
	Relative of Operator																		
	Product Manager																		

Stakeholder Class (Class of stakeholders who share a particular stakeholding in the project)	Stakeholder Role (The job title, department, or organization that might indicate a role for this class of stakeholder)	Stakeholder Name (Or their representative) Stakeholder Rationale (Consider benefits and impacts) Necessary Involvement (When and how much time)	Goals	Business Constraints	Technical Constraints	Functionality	Look and Feel	Usability	Peformance	Safety	Operational Environment	Portability	Security	Cultural Acceptance	Legal	Maintenance	Estimates	Risk	Design Ideas
Internal Consultant *(continued)*																			
	Marketing Specialist																		
	Sales Specialist																		
	Technology Expert																		
	Standards Specialist																		
	Testing Specialist																		
	Organizational Architect																		
	Aesthetics Specialist																		
	Graphics Specialist																		
Sponsor																			
	Sponsor																		
	Project Champion																		
Wider Environment Stakeholder Classes																			
Customer																			
	Department Manager																		
	Another Organization																		
	Member of the Public																		
Interfacing Technology																			
	External Meta Manager																		
External Consultants																			
	Auditors																		
	Focus Group																		

Stakeholder Class (Class of stakeholders who share a particular stakeholding in the project)	Stakeholder Role (The job title, department, or organization that might indicate a role for this class of stakeholder)	Stakeholder Name (Or their representative) Stakeholder Rationale (Consider benefits and impacts) Necessary Involvement (When and how much time)	Goals	Business Constraints	Technical Constraints	Functionality	Look and Feel	Usability	Peformance	Safety	Operational Environment	Portability	Security	Cultural Acceptance	Legal	Maintenance	Estimates	Risk	Design Ideas
External Consultants *(continued)*																			
	Security Specialist																		
	Environmental Specialist																		
	Safety Specialist																		
	Outsource																		
	Cultural Specialist																		
	Legal Specialist																		
	Packaging Designer																		
	Manufacturer																		
	Negotiator																		
	Public Opinion																		
	COTS Supplier																		
	Inspector																		
Negative Stakeholders																			
	Competitor																		
	Hacker																		
	Political Party																		
	Pressure Group																		
	Public Opinion																		
Core Project Team Stakeholder Classes																			
Core Team Members																			
	Project Manager																		
	Business Analyst																		

Stakeholder Class (Class of stakeholders who share a particular stakeholding in the project)	Stakeholder Role (The job title, department, or organization that might indicate a role for this class of stakeholder)	Stakeholder Name (Or their representative) Stakeholder Rationale (Consider benefits and impacts) Necessary Involvement (When and how much time)	Goals	Business Constraints	Technical Constraints	Functionality	Look and Feel	Usability	Peformance	Safety	Operational Environment	Portability	Security	Cultural Acceptance	Legal	Maintenance	Estimates	Risk	Design Ideas
Core Team Members *(continued)*																			
	Requirements Analyst																		
	System Analyst																		
	Tester																		
	Technical Writer																		
	Systems Architect																		
	Systems Designer																		

More on SRS templates

2014 9th International Conference on the Quality of Information and Communications Technology

Towards a System Requirements Specification Template that Minimizes Combinatorial Effects

Alberto Rodrigues da Silva[1], Jan Verelst[2], Herwig Mannaert[2], David Almeida Ferreira[1], Philip Huysmans[2]
{alberto.silva, david.ferreira}@inesc-id.pt, {jan.verelst, herwig.mannaert, philip.huysmans}@ua.ac.be

[1]IST/Universidade de Lisboa & INESC-ID
Lisbon, Portugal

[2]University of Antwerp & Normalized Systems Institute
Antwerp, Belgium

***Abstract*—This paper introduces the problem of combinatorial effects based on the evidence of many dependencies that explicitly or implicitly exist among the elements commonly used on system requirements specification (SRS). We start from the analysis and comparison of three popular SRS templates (namely IEEE 830-1998, RUP and Withall templates), mainly from the perspective of the constructs and models involved. Then we propose and discuss a set of practical recommendations to help defining a SRS template that may better prevent (to some extent) the referred problem.**

***Keywords*—System Requirements Specification (SRS); SRS templates; Combinatorial Effect (CE)**

artifacts, which are important to minimize or prevent (to some extent).

In a recent work we discussed the result of our experiences in looking for CE at different levels in the RE-process [1]. The examples covered CE at the RE level based on the adoption of notations and techniques such as UML (classes and use case diagrams), DEMO/EO, and BPMN models. Those examples are relatively straightforward, but enough to show the omnipresence of such instabilities in the RE level. As a result, we described the need for a research agenda focusing on the systematic research into CE and related issues at the RE domain in order to build enterprises and their information

See: Alberto Rodrigues da Silva, Jan Verelst, Herwig Mannaert, David Ferreira, Philip Huysmans, Towards a System Requirements Specification Template that Minimizes Combinatorial Effects, in Proceedings of QUATIC'2014 Conference, 2014 , IEEE Computer Society. http://isg.inesc-id.pt/alb/static/papers/2014/c123-as-Quatic-SRS-Template-2014.pdf

More on SRS templates

TABLE I. RELATING CONSTRUCTS AND MODELS WITH SRS TEMPLATES

			Templates		
Level	**Models**	**Constructs**	**IEEE 830**	**RUP**	**Withall**
Business	Glossary	Terms, definitions, acronyms	1.3 Definitions, acronyms, and abbreviations	1.3 Definitions, acronyms, and abbreviations	1.4 Glossary
	Stakeholders model	Stakeholders	-	-	-
	Goals model	Goals	1.2 Scope	1.2 Scope	1.1 Objective of the system
System	Context model	Systems, sub-systems, components, nodes	2.1 Product perspective	-	2.1 Scope
	Domain model	Entities, classes, objects	-	-	2.4 Main business entities
	Functional model	Actors, users	2.3 User characteristics	2.1 Use-case model survey	2.4 Main business entities
		Use-cases, user-stories	2.2 Product functions 3.2 Functional requirements	2.1 Use-case model 3.1 Use-Case reports	3. Functional Areas
	Quality attributes model	Qualities, metrics, utility values	3. Specific Requirements (all except 3.2)	3.2 Supplementary requirements	4. Non-Functional Requirements

See: Alberto Rodrigues da Silva, Jan Verelst, Herwig Mannaert, David Ferreira, Philip Huysmans, Towards a System Requirements Specification Template that Minimizes Combinatorial Effects, in Proceedings of QUATIC'2014 Conference, 2014 , IEEE Computer Society. http://isg.inesc-id.pt/alb/static/papers/2014/c123-as-Quatic-SRS-Template-2014.pdf

More on SRS templates

TABLE II. PROPOSED SRS TEMPLATE

Level	Template Structure	Comments
Business	1. Introduction	*Chapter 1 presents the Business View*
	1.1 Purpose of the document	
	1.2 Glossary	***Glossary model**, with lists of domain-specific and system-specific terms*
	1.3 Stakeholders	***Stakeholders model**, with generic stakeholders and concrete stakeholders*
	1.4 Goals	***Goals model**, e.g., using visual notations such as i* or KAOS*
	1.7 References	
	1.8 Organization of the Document	
System	2. System overview	*Chapter 2 presents the high-level System View*
	2.1 Context	***Context model** defines how the system is organized in subsystems and how it interoperates with other external systems*
	2.3 Key Assumptions	
	2.4 Technical Constraints	*e.g., contraints related the use of some tool, database server, or development process*
	2.4 Main Exclusions	
	3. Subsystem A	*Chapters 3, 4, 5, .. present the Subsystem Views*
	3.1 Informational Entities	***Domain model**, e.g., using visual notations such as UML class diagrams or ER*
	3.2 Actors	*List of actors*
	3.2 Functional Requirements	*List of functional requirements*
	3.3 Use cases	***Use case model**, e.g., using visual notations such as UML use cases and textual narratives*
	3.4 NFRs (at subsystem level)	***NFR Models**, using e.g. QA Scenarios for describing subsystem-specific NFRs*
	...	
	N. High-Level Non-Functional Requirements	*Chapter N presents **NFR Models**, using e.g. QA Scenarios, for describing NFRs defined generically at (sub)system levels*
	N.1 NFR 1	
	N.1 NFR 2	
	...	
	N+1. Concerns Integration	*Chapter N+1 presents the integration and composition of concerns, based on some CORE or AORE approach*
-	Appendixes	*Complementary data to the SRS, for example: Versions of the Document; Traceability Matrixes*

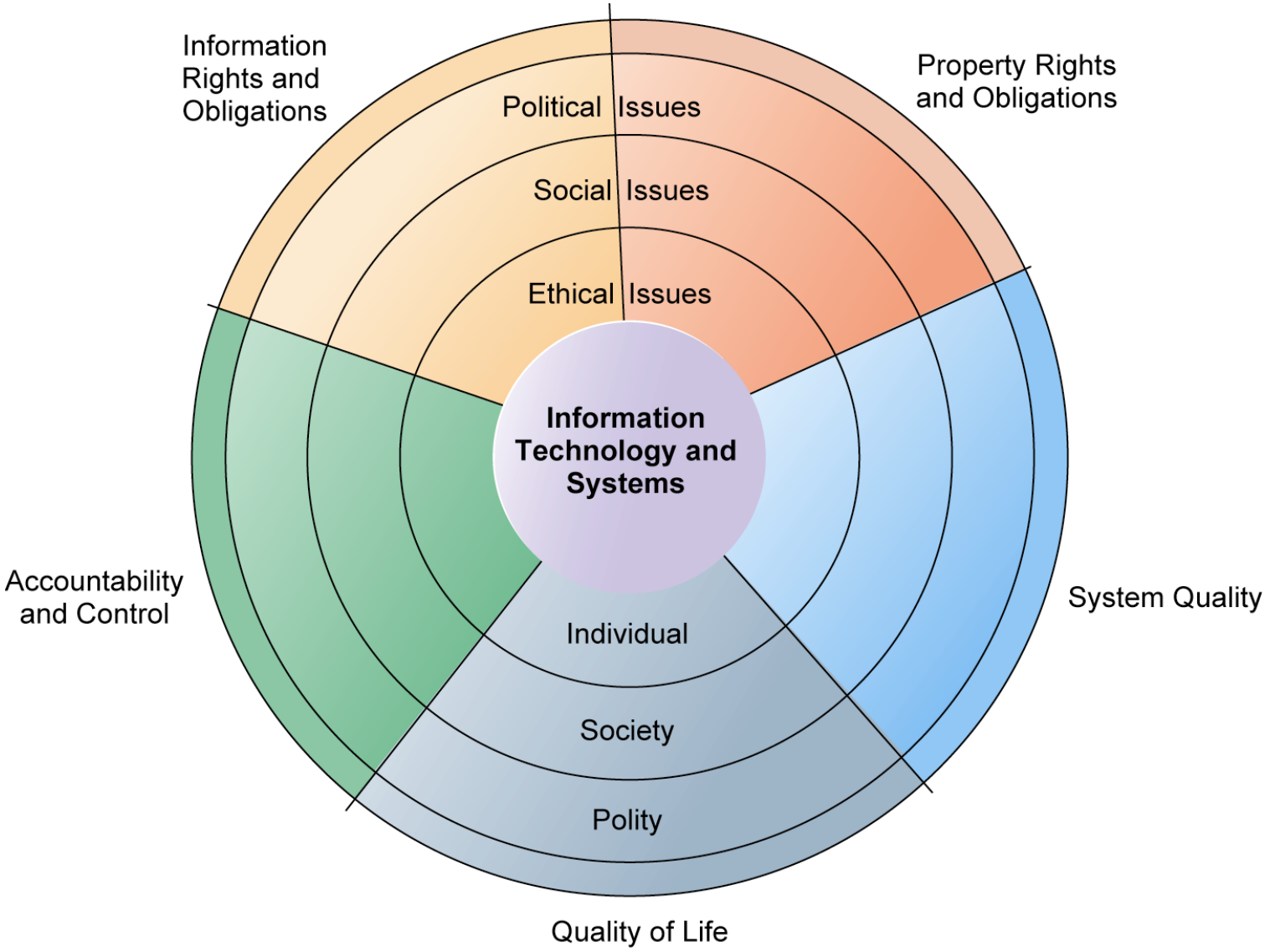
See: Alberto Rodrigues da Silva, Jan Verelst, Herwig Mannaert, David Ferreira, Philip Huysmans, Towards a System Requirements Specification Template that Minimizes Combinatorial Effects, in Proceedings of QUATIC'2014 Conference, 2014 , IEEE Computer Society. http://isg.inesc-id.pt/alb/static/papers/2014/c123-as-Quatic-SRS-Template-2014.pdf

AMS

Requirements Engineering Fundamentals…

…other requirements from… the context:

Ethical, Social, and Political Issues in an Information Society



The Moral Dimensions of Information Systems

- **Accountability, liability, and control**
  - **Computer-related liability problems**
- **System quality: Data quality and system errors**
- **Quality of life: Equity, access, and boundaries**
  - **Balancing power: Center versus periphery**
  - **Rapidity of change: Reduced response time to competition**
  - **Maintaining boundaries: Family, work, and leisure**
  - **Dependence and vulnerability**
- **Quality of life: Equity, access, and boundaries**
  - **Computer crime and abuse**
  - **Employment: Trickle-down technology and reengineering job loss**
  - **Equity and access: Increasing racial and social class cleavages**
  - **Health risks: RSI, CVS, and Technostress**

Ethics in our profession…

www.acm.org/about/code-of-ethics/

Home > About ACM > ACM Code Of Ethics And Professional Conduct

ACM Code of Ethics and Professional Conduct

Adopted by ACM Council 10/16/92.

Preamble

Contents & Guidelines

Preamble

Commitment to ethical professional conduct is expected of every member (voting members, associate members, and student members) of the Association for Computing Machinery (ACM).

This Code, consisting of 24 imperatives formulated as statements of personal responsibility, identifies the elements of such a commitment. It contains many, but not all, issues professionals are likely to face. Section 1 outlines fundamental ethical considerations, while Section 2 addresses additional, more specific considerations of professional conduct. Statements in Section 3 individuals who have a leadership role, whether in the workplace or in a organizations like ACM. Principles involving compliance with this Code ar

The Code shall be supplemented by a set of Guidelines, which provide e dealing with the various issues contained in the Code. It is expected th changed more frequently than the Code.

The Code and its supplemented Guidelines are intended to serve as a b in the conduct of professional work. Secondarily, they may serve as a ba formal complaint pertaining to violation of professional ethical standards

It should be noted that although computing is not mentioned in the imp is concerned with how these fundamental imperatives apply to one's co professional. These imperatives are expressed in a general form to emp which apply to computer ethics are derived from more general ethical pr

It is understood that some words and phrases in a code of ethics are su and that any ethical principle may conflict with other ethical principles in related to ethical conflicts can best be answered by thoughtful consider rather than reliance on detailed regulations.

[Back to the Top]

Contents & Guidelines

1. General Moral Imperatives.
2. More Specific Professional Responsibilities.
3. Organizational Leadership Imperatives.
4. Compliance with the Code.

[Back to the Top]

ACM Case Studies

Written by leading domain experts for software engineers, ACM Case Studies

1. GENERAL MORAL IMPERATIVES.

As an ACM member I will ....

- **1.1 Contribute to society and human well-being.**
- **1.2 Avoid harm to others.**
- **1.3 Be honest and trustworthy.**
- **1.4 Be fair and take action not to discriminate.**
- **1.5 Honor property rights including copyrights and patent.**
- **1.6 Give proper credit for intellectual property.**
- **1.7 Respect the privacy of others.**
- **1.8 Honor confidentiality.**

2. MORE SPECIFIC PROFESSIONAL RESPONSIBILITIES.

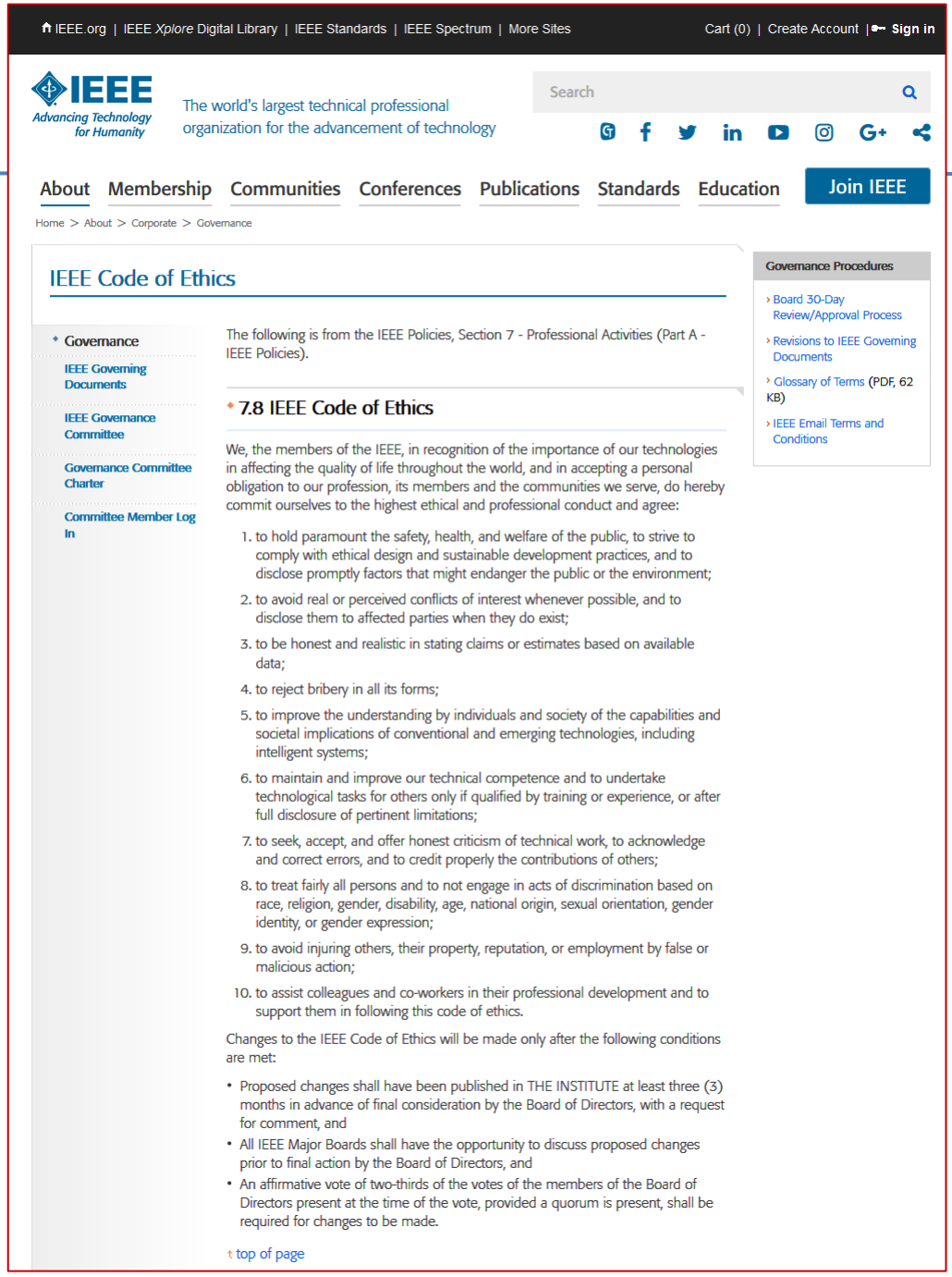
…

3. ORGANIZATIONAL LEADERSHIP IMPERATIVES.

…

4. COMPLIANCE WITH THE CODE.

Ethics in our profession…



IEEE.org | IEEE *Xplore* Digital Library | IEEE Standards | IEEE Spectrum | More Sites

Cart (0) | Create Account | Sign in

IEEE
Advancing Technology for Humanity

The world's largest technical professional organization for the advancement of technology

Search

About | Membership | Communities | Conferences | Publications | Standards | Education | Join IEEE

Home > About > Corporate > Governance

IEEE Code of Ethics

- Governance
  - IEEE Governing Documents
  - IEEE Governance Committee
  - Governance Committee Charter
  - Committee Member Log In

The following is from the IEEE Policies, Section 7 - Professional Activities (Part A - IEEE Policies).

7.8 IEEE Code of Ethics

We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members and the communities we serve, do hereby commit ourselves to the highest ethical and professional conduct and agree:

1. to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment;
2. to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;
3. to be honest and realistic in stating claims or estimates based on available data;
4. to reject bribery in all its forms;
5. to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems;
6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;
8. to treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;
9. to avoid injuring others, their property, reputation, or employment by false or malicious action;
10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

Changes to the IEEE Code of Ethics will be made only after the following conditions are met:

- Proposed changes shall have been published in THE INSTITUTE at least three (3) months in advance of final consideration by the Board of Directors, with a request for comment, and
- All IEEE Major Boards shall have the opportunity to discuss proposed changes prior to final action by the Board of Directors, and
- An affirmative vote of two-thirds of the votes of the members of the Board of Directors present at the time of the vote, provided a quorum is present, shall be required for changes to be made.

t top of page

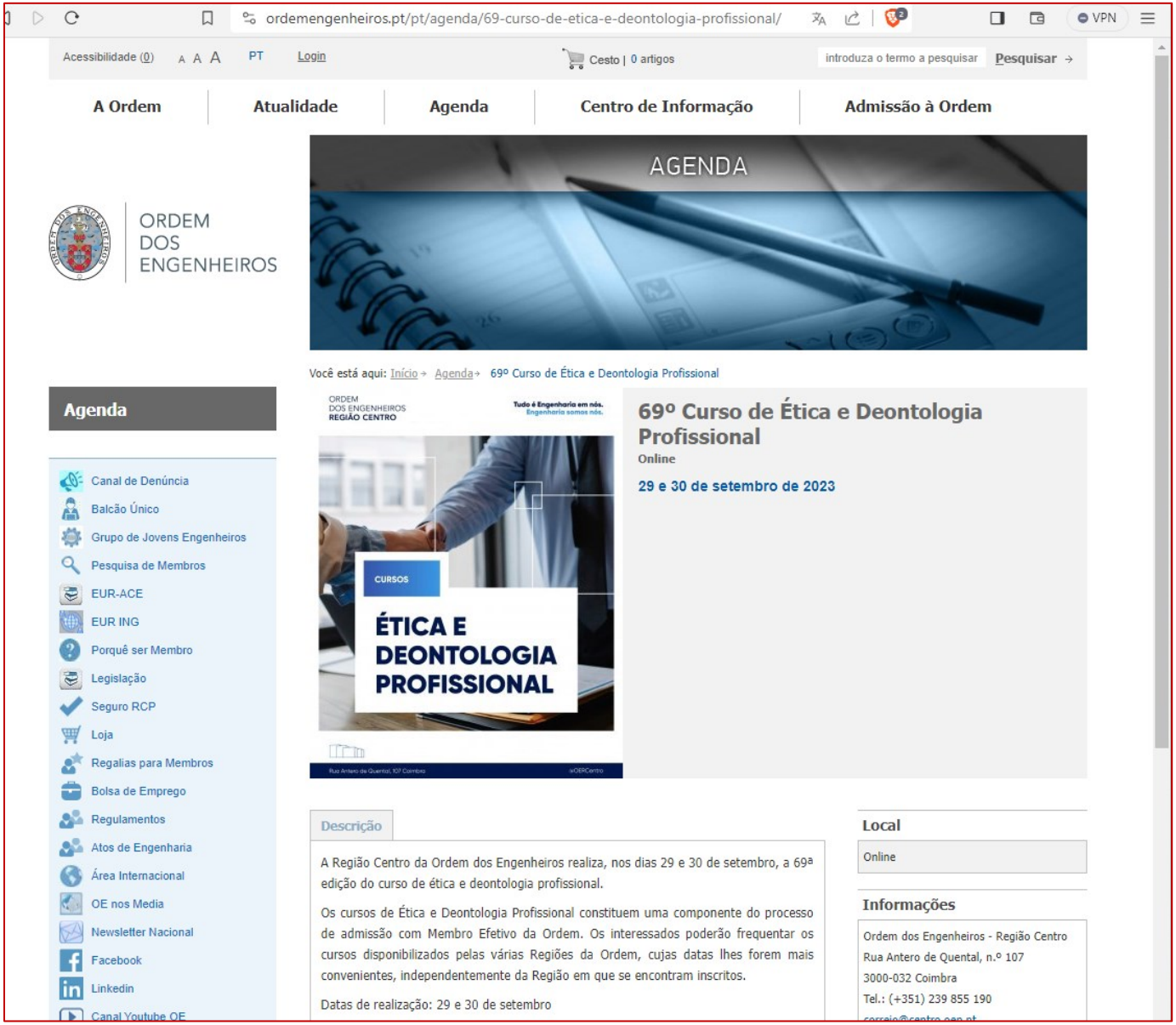
Governance Procedures

- Board 30-Day Review/Approval Process
- Revisions to IEEE Governing Documents
- Glossary of Terms (PDF, 62 KB)
- IEEE Email Terms and Conditions

https://www.ieee.org/about/corporate/governance/p7-8.html

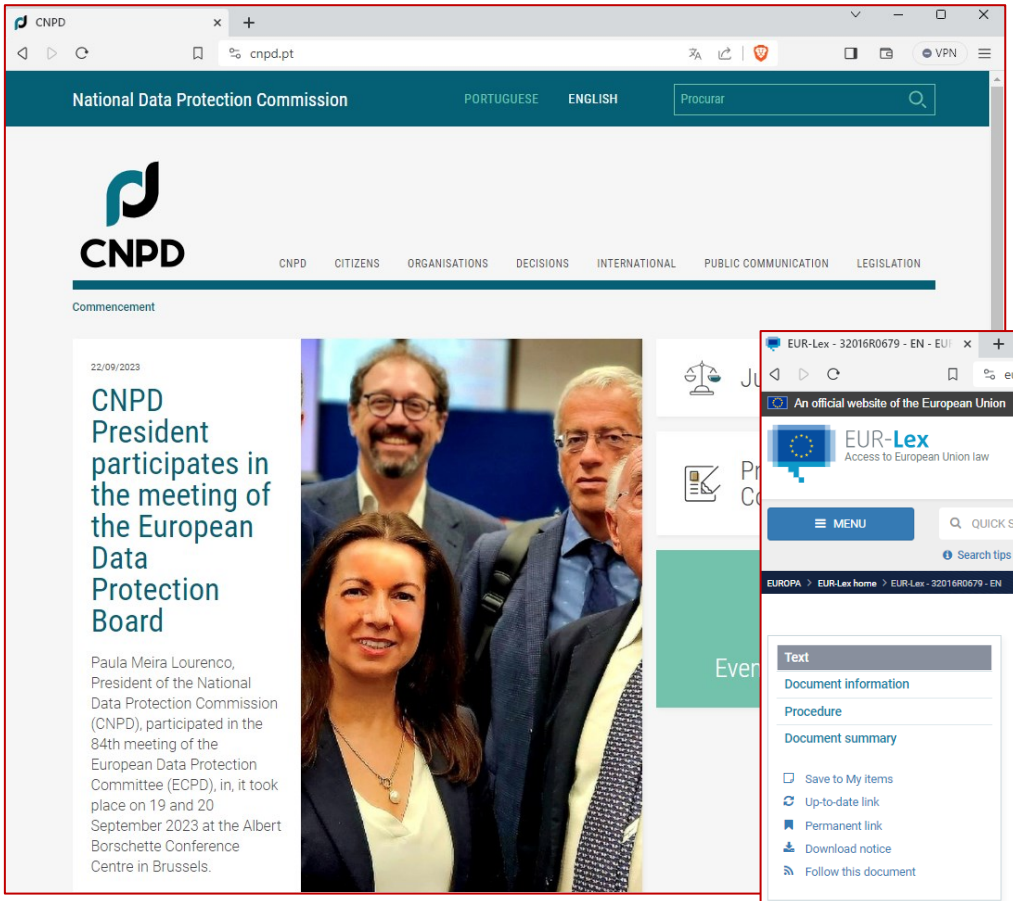
Ethics in our profession…

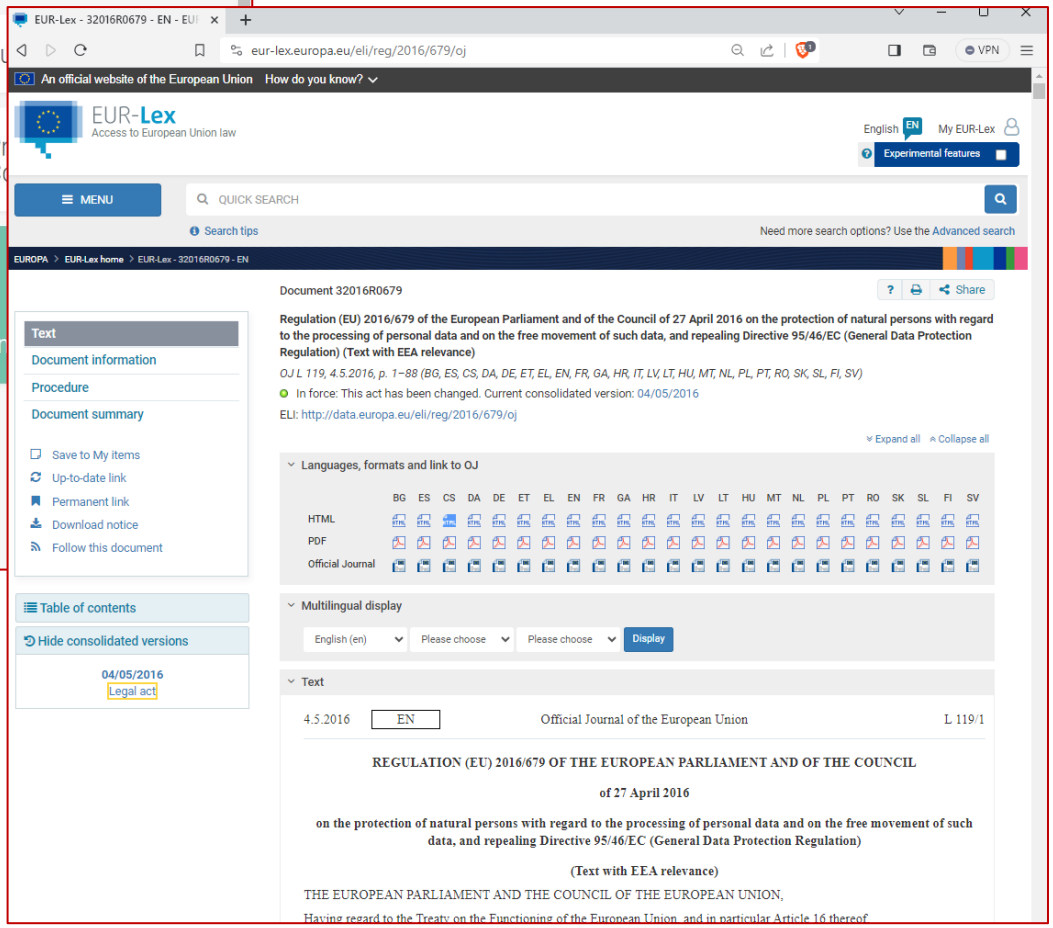
http://www.ordemengenheiros.pt



“Regulatory Compliance” are requirements imposed by the context of the business!!!

Example:
GDPR Data Privacy Requirements





Please learn about “Compliance…”