Requirements Engineering

Lecture 2

Prof. Dr. Alexander Pretschner

Chair of Software & Systems Engineering
TUM Department of Informatics
Technical University of Munich

Orientation

Recap:

- Core activities in RE
- Context dependency of RE and SE
- RE results as input for whole SE process and thus reason for project failures

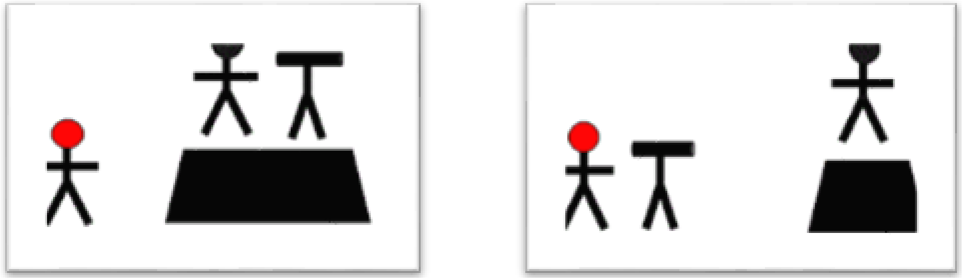
Coming up:

- Quality of requirements
- What makes a requirement a useful requirement?

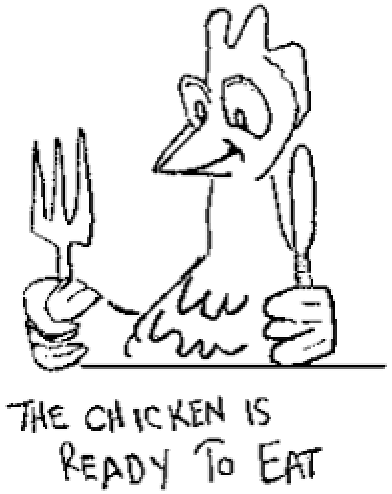
Natural language must be interpreted!

Example: *„I saw a man on the hill with a telescope.“*

=> Who has the telescope?



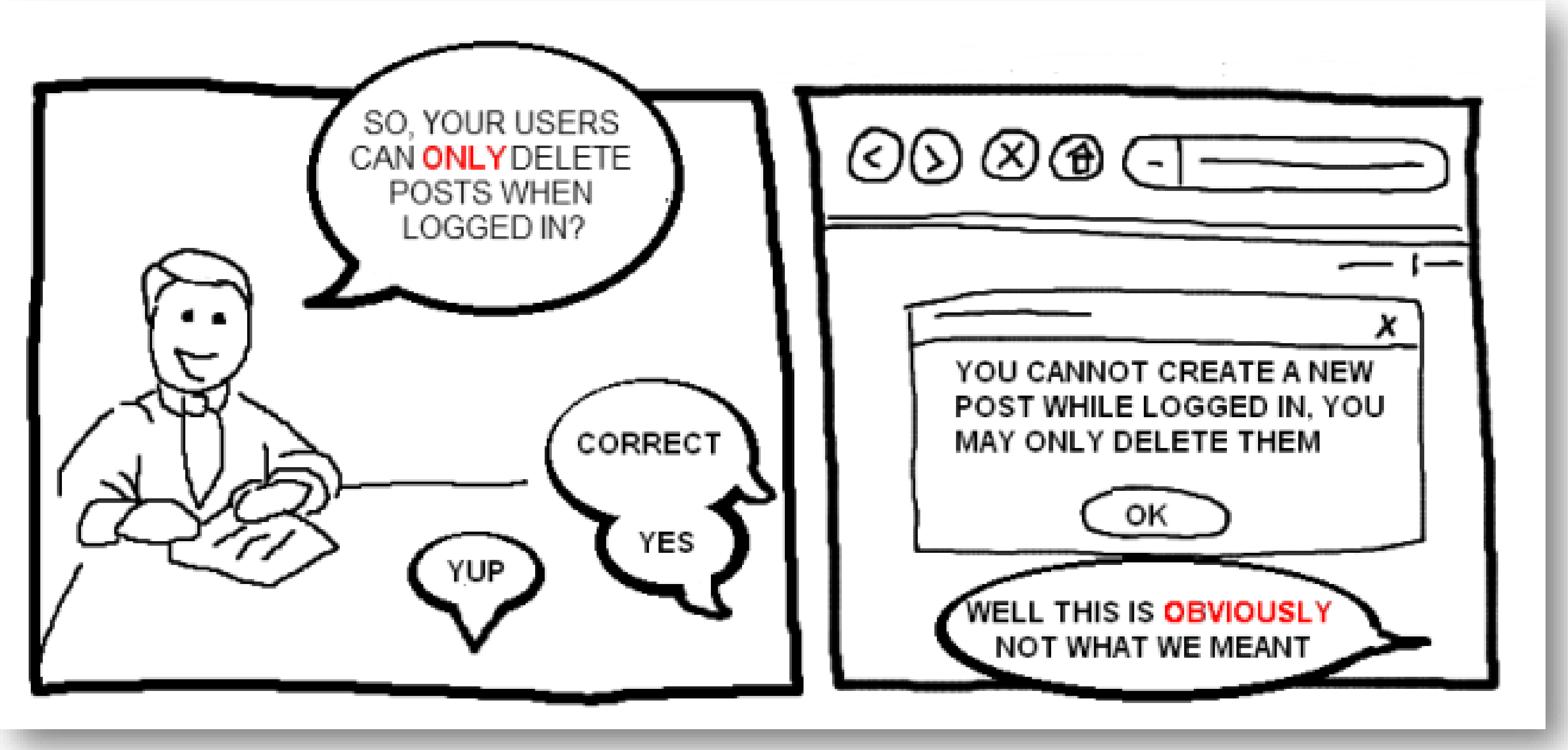
Countless more examples:



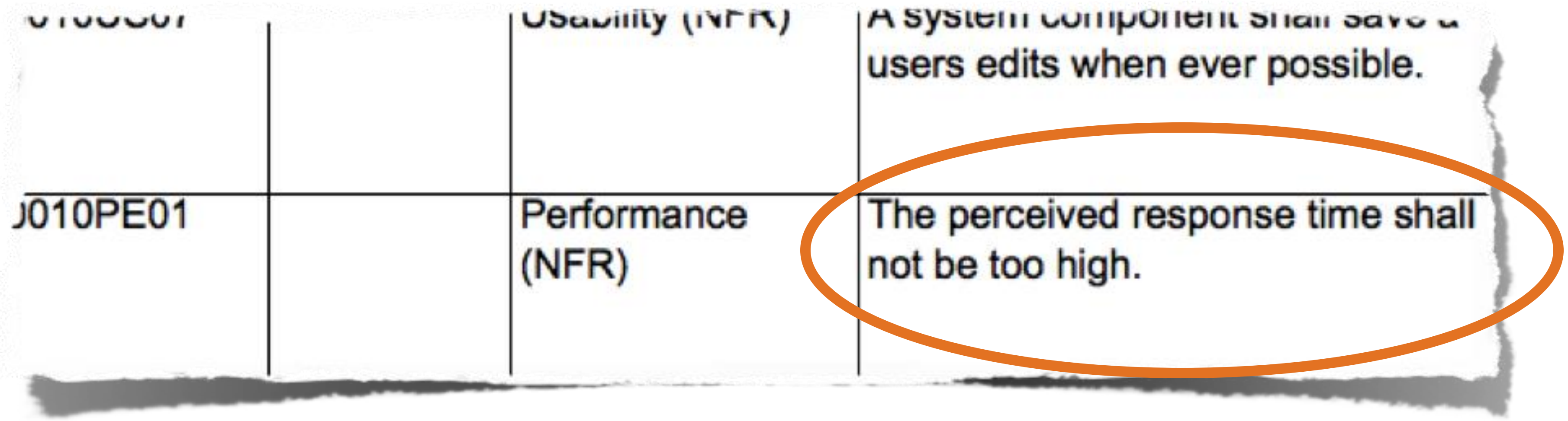
„The father slapped his son because he was drunk“.
Who was drunk?

Image source: https://www.ashishdalela.com/2017/01/15/problem-meaning-artificial-intelligence/

Same holds for Requirements!


SO, YOUR USERS CAN ONLY DELETE POSTS WHEN LOGGED IN?
CORRECT
YES
YUP
X
YOU CANNOT CREATE A NEW POST WHILE LOGGED IN, YOU MAY ONLY DELETE THEM
OK
WELL THIS IS OBVIOUSLY NOT WHAT WE MEANT

Consequences of subpar requirements



010US07		Usability (NFR)	A system component shall save a users edits when ever possible.
010PE01		Performance (NFR)	The perceived response time shall not be too high.

→How can the requirement be clearly implemented?

→How can the requirement be checked and tested?

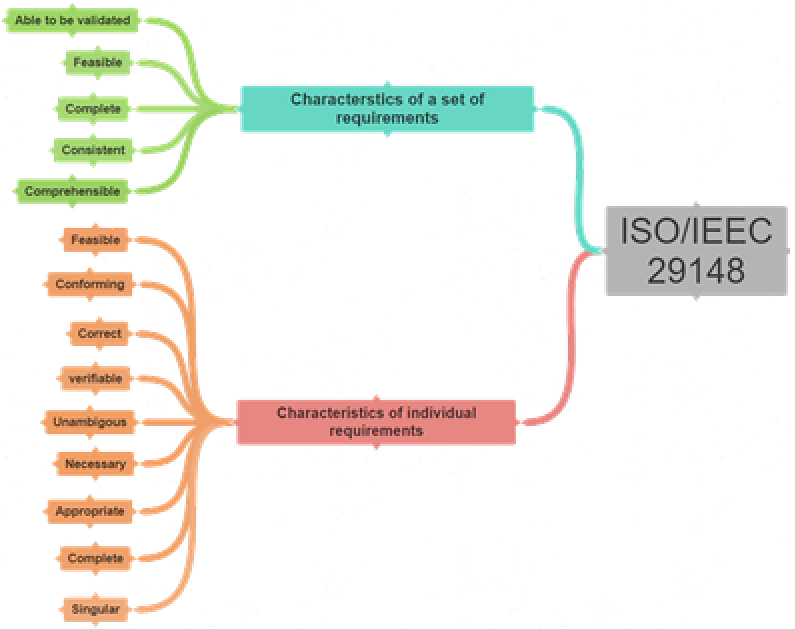
→How can a reasonable cost estimate be made based on this requirement?

→...

Requirements Quality Model versus Quality Requirements

Requirements Quality Model:

describes the quality of requirements



Able to be validated

Feasible

Complete

Consistent

Comprehensible

Characterstics of a set of requirements

Feasible

Conforming

Correct

verifiable

Unambigous

Necessary

Appropriate

Complete

Singular

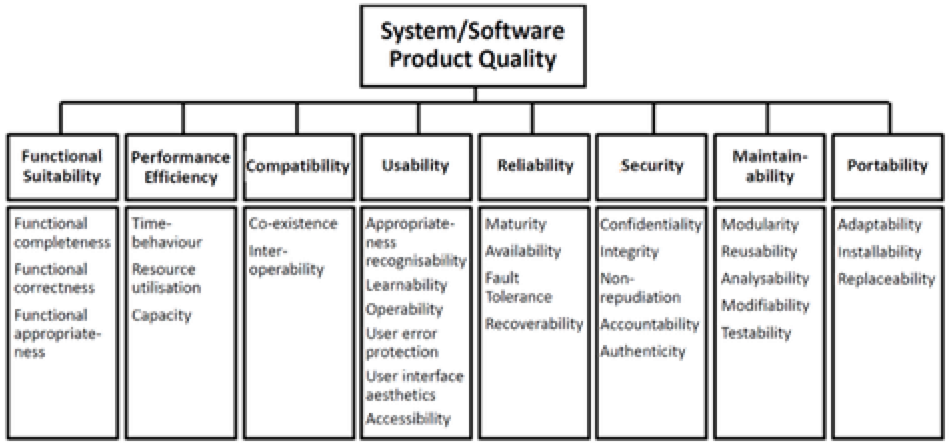
Characteristics of individual requirements

ISO/IEEC 29148

≠

Quality Requirements:

describe the quality of a system



System/Software Product Quality

Functional Suitability	Performance Efficiency	Compatibility	Usability	Reliability	Security	Maintain-ability	Portability
Functional completeness Functional correctness Functional appropriateness	Time-behaviour Resource utilisation Capacity	Co-existence Inter-operability	Appropriateness recognisability Learnability Operability User error protection User interface aesthetics Accessibility	Maturity Availability Fault Tolerance Recoverability	Confidentiality Integrity Non-repudiation Accountability Authenticity	Modularity Reusability Analysability Modifiability Testability	Adaptability Installability Replaceability

Example: ISO/IEC/IEEE Std. 29148

INTERNATIONAL STANDARD

ISO/IEC/IEEE 29148

Second edition
2018-11

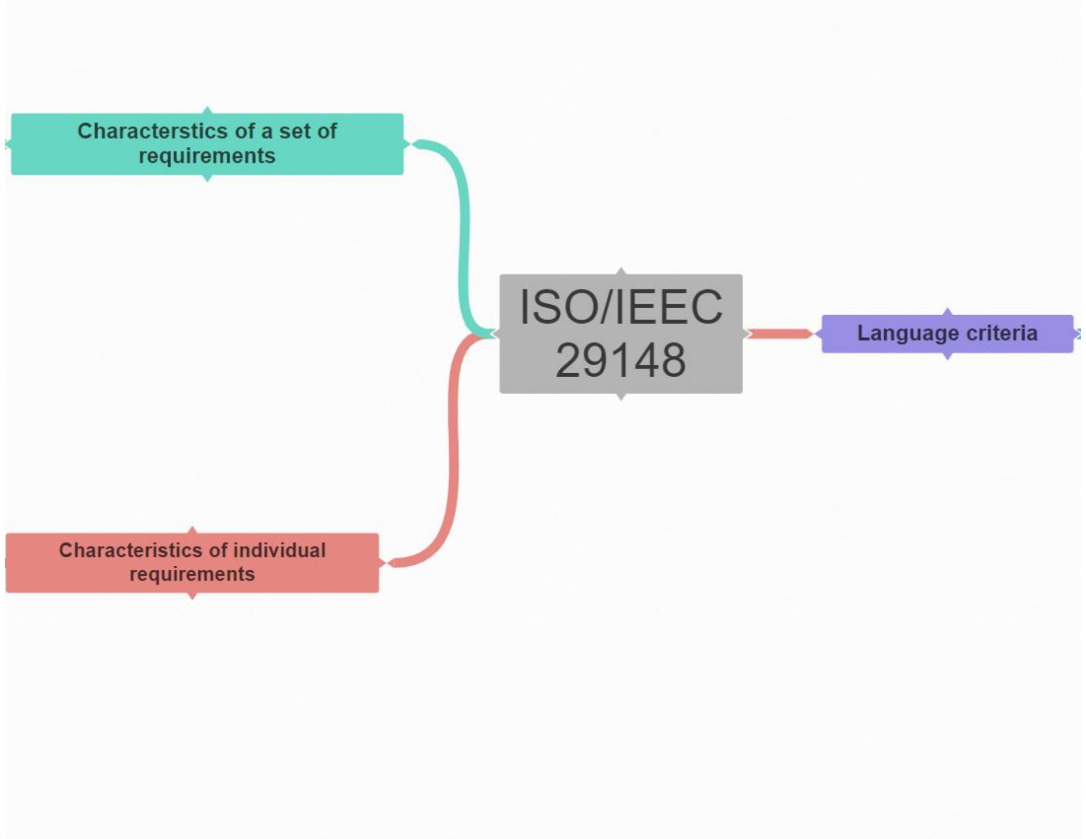
Systems and software engineering — Life cycle processes — Requirements engineering

Ingénierie des systèmes et du logiciel — Processus du cycle de vie — Ingénierie des exigences

ISO IEC
IEEE

Reference number
ISO/IEC/IEEE 29148:2018(E)

© ISO/IEC 2018
© IEEE 2018


Characterstics of a set of requirements
ISO/IEEC 29148
Language criteria
Characteristics of individual requirements

Language criteria will be covered in depth later in this course

Example: ISO/IEC/IEEE Std. 29148

🟩 **Quality criteria for a set of requirements**

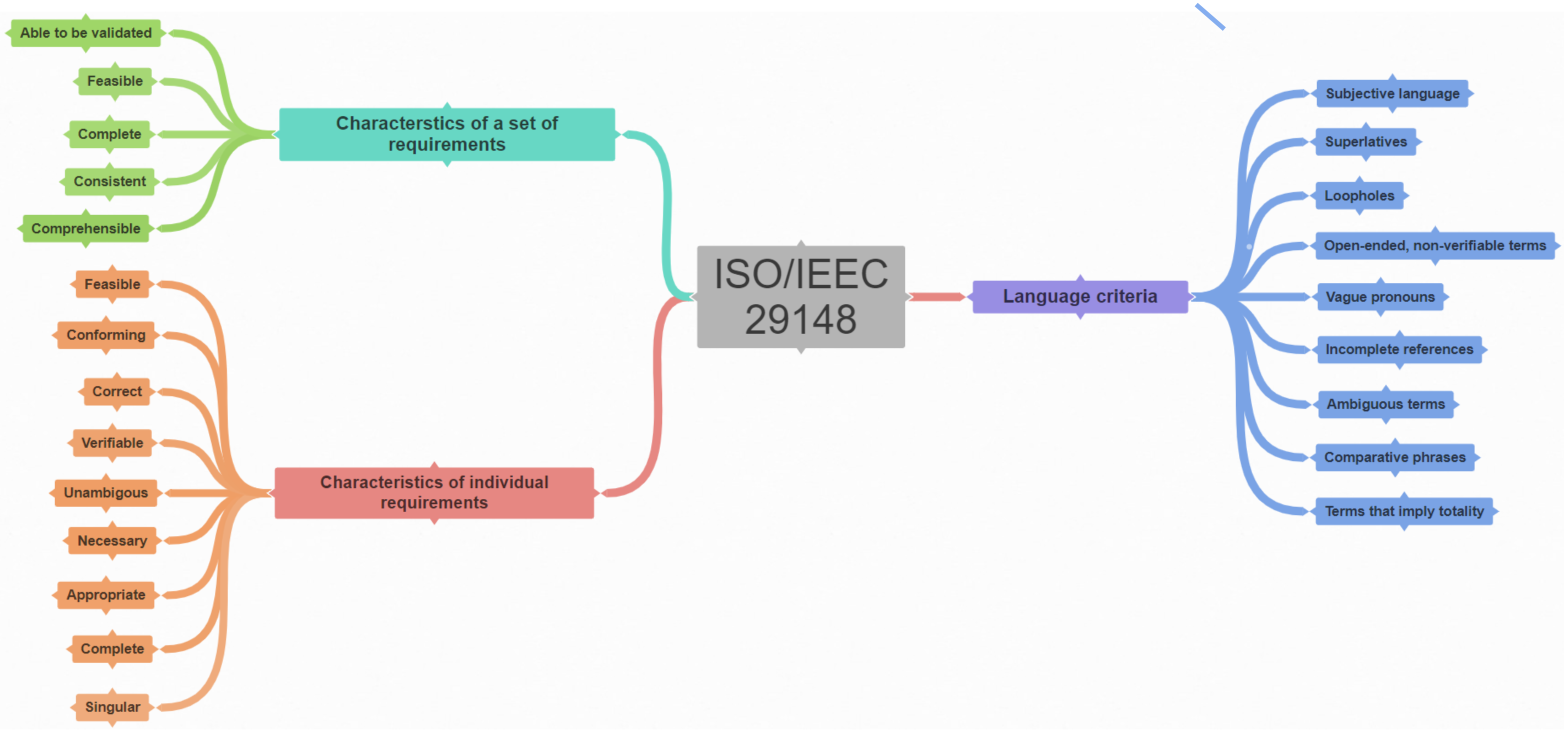
- *Able to be validated*
- *Affordable/Feasible*
- *Complete*
- *Consistent*
- *Comprehensible*

🟧 **Quality criteria for individual requirements**

- *Feasible*
- *Conforming: compliance to template etc*
- *Correct*
- *Verifiable*
- *Unambigous*
- *Necessary*
- *Appropriate level*
- *Complete*
- *Singular*

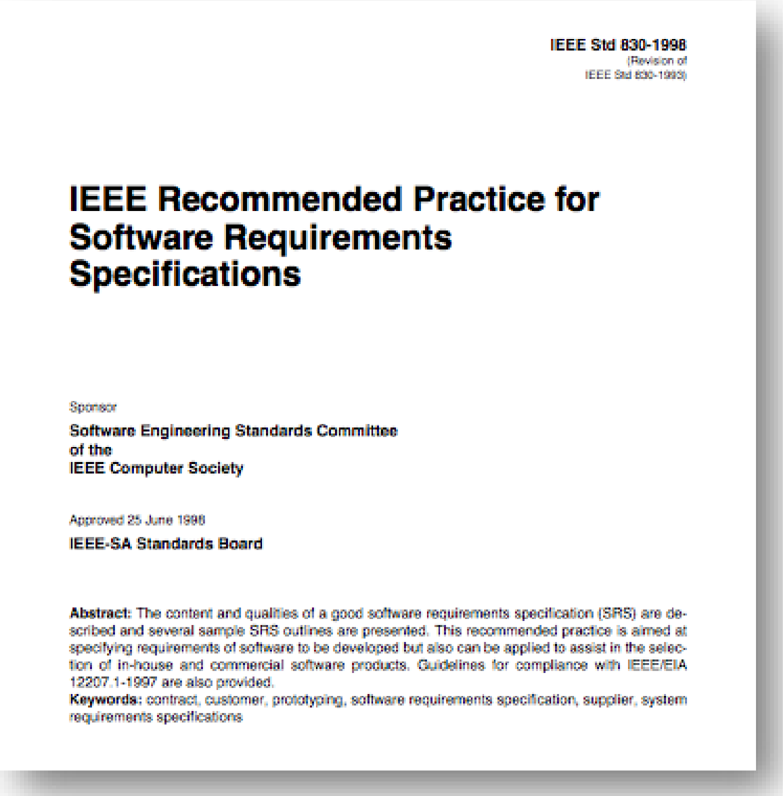
🟦 **Language criteria** (covered later in this course)

- *best practices to achieve these quality criteria*



Second Example: IEEE Std 830-1998 / Part I

- **Consistency**: *Consistency and thus feasibility*
- **Completeness**: *All relevant requirements are recorded*
- **Correctness**: *All recorded requirements represent valid, applicable wishes and specifications of the stakeholders*
- **Clarity**: *Requirements are unambiguous*
- **Verifiability**: *(Fulfillment of the) requirements can be verified beyond any doubt, requirements are measurable if necessary*



IEEE Std 830-1998
(Revision of
IEEE Std 830-1993)

IEEE Recommended Practice for Software Requirements Specifications

Sponsor

**Software Engineering Standards Committee
of the
IEEE Computer Society**

Approved 25 June 1998

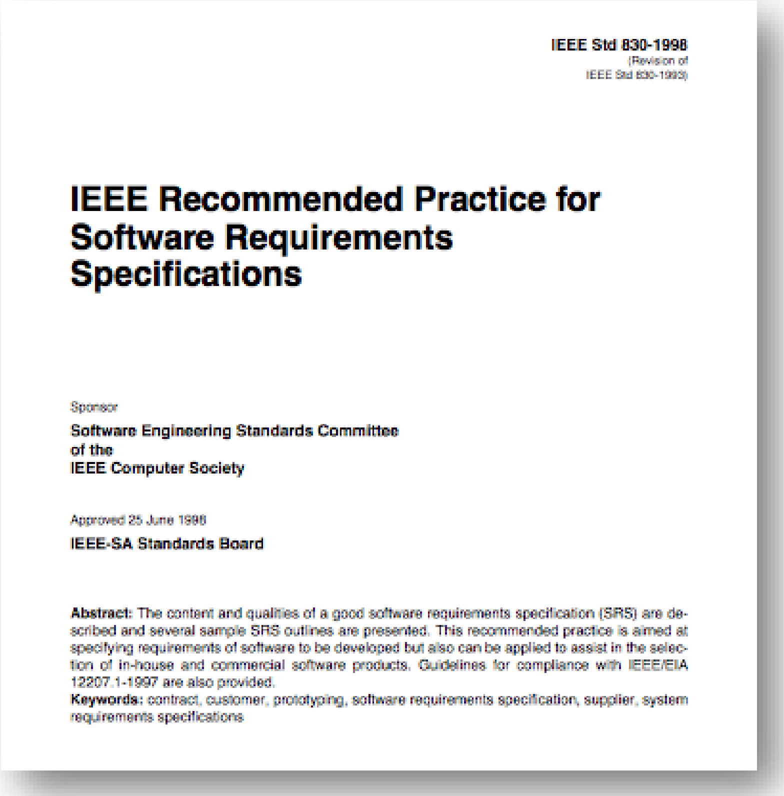
IEEE-SA Standards Board

Abstract: The content and qualities of a good software requirements specification (SRS) are described and several sample SRS outlines are presented. This recommended practice is aimed at specifying requirements of software to be developed but also can be applied to assist in the selection of in-house and commercial software products. Guidelines for compliance with IEEE/EIA 12207.1-1997 are also provided.

Keywords: contract, customer, prototyping, software requirements specification, supplier, system requirements specifications

Second Example: IEEE Std 830-1998 / Part II

- **Modifiability**: *Changes to the requirements can be made in a controlled manner*
- **Traceability**: *The implementation of the requirement can be traced*
- **Prioritization**: *Requirements are ranked according to their importance*
- **Comprehensibility**: *Requirements are easy to understand*



IEEE Std 830-1998
(Revision of
IEEE Std 830-1993)

IEEE Recommended Practice for Software Requirements Specifications

Sponsor

**Software Engineering Standards Committee
of the
IEEE Computer Society**

Approved 25 June 1998

IEEE-SA Standards Board

Abstract: The content and qualities of a good software requirements specification (SRS) are described and several sample SRS outlines are presented. This recommended practice is aimed at specifying requirements of software to be developed but also can be applied to assist in the selection of in-house and commercial software products. Guidelines for compliance with IEEE/EIA 12207.1-1997 are also provided.
Keywords: contract, customer, prototyping, software requirements specification, supplier, system requirements specifications

Terminology in the context of quality assurance (in RE)

Important terms (simplified)

Incorrect (invalid) requirement: Requirement that does **not** reflect the intention of the stakeholders.

Quality deficiency: Requirements that may be valid in terms of content, but have qualitative defects, i.e. violate quality criteria

Validation and Verification (simplified)

Validation: Checking the requirements for validity (right system built)

Verification: Testing the system for compliance with the requirements (correctness: system built right)

Quality assurance (simplified)

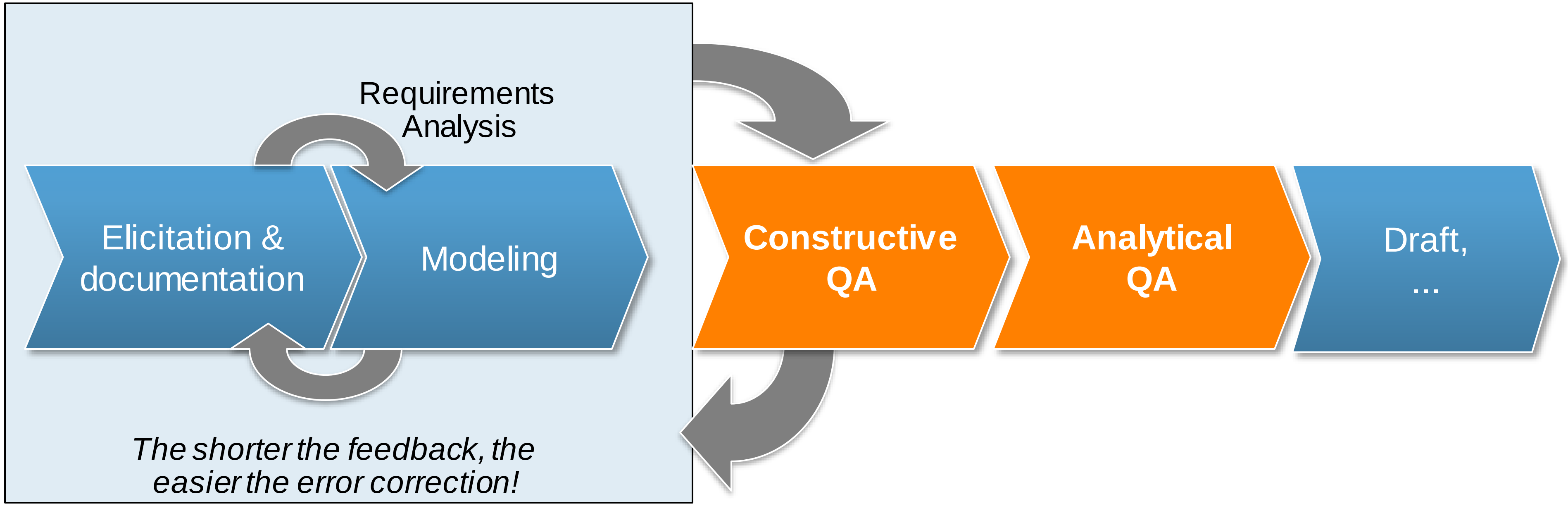
→Systematic measures to identify quality defects

→Attention: Validation and Verification are part of quality assurance

Quality assurance in RE

Constructive quality assurance: Assurance of the quality of the artifacts to be created during the creation process, e.g. by means of model building

Analytical quality assurance: independent and autonomous testing and evaluation of the artifacts created, e.g. within the framework of quality gates



The shorter the feedback, the easier the error correction!

Linguistics in RE

Classification of linguistic quality deficiencies

- ontological/lexical („*The button shall be green“.)*
- syntactic (*"I saw the man on the hill with a telescope."*)
- semantic ("*All persons have a unique national insurance number.*")
- pragmatic ("*The trucks shall treat the roads before they freeze.*")
- weak phrases: („… *as soon as possible.*")
- Eradication/Generalization/Distortion („*The response is never higher than 100ms.“*)

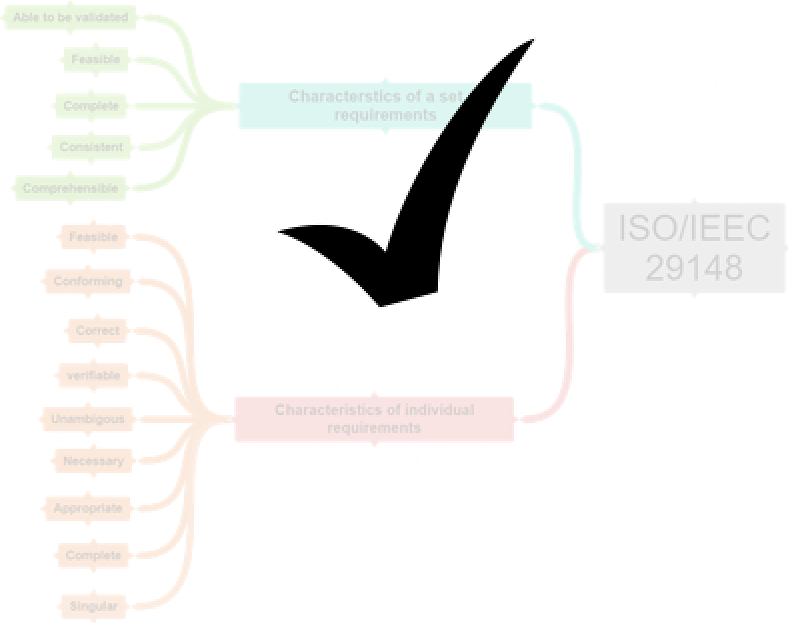
How to avoid this by constructive QA?

More about this later in the course!

Requirements Quality Model versus Quality Requirements

Requirements Quality Model:

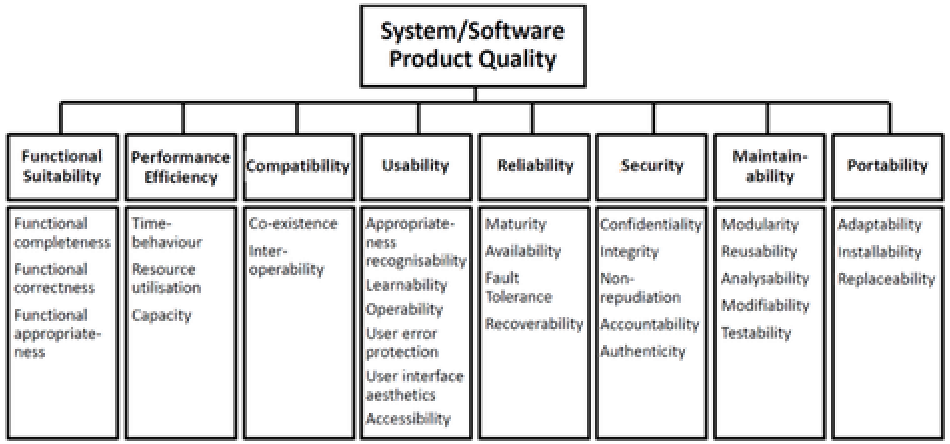
describes the quality of requirements



≠

Quality Requirements:

describe the quality of a system


System/Software Product Quality

Functional Suitability	Performance Efficiency	Compatibility	Usability	Reliability	Security	Maintain-ability	Portability
Functional completeness Functional correctness Functional appropriate-ness	Time-behaviour Resource utilisation Capacity	Co-existence Inter-operability	Appropriate-ness recognisability Learnability Operability User error protection User interface aesthetics Accessibility	Maturity Availability Fault Tolerance Recoverability	Confidentiality Integrity Non-repudiation Accountability Authenticity	Modularity Reusability Analysability Modifiability Testability	Adaptability Installability Replaceability

Software Product Quality: (ISO/IEC/IEEE Std. 25010)

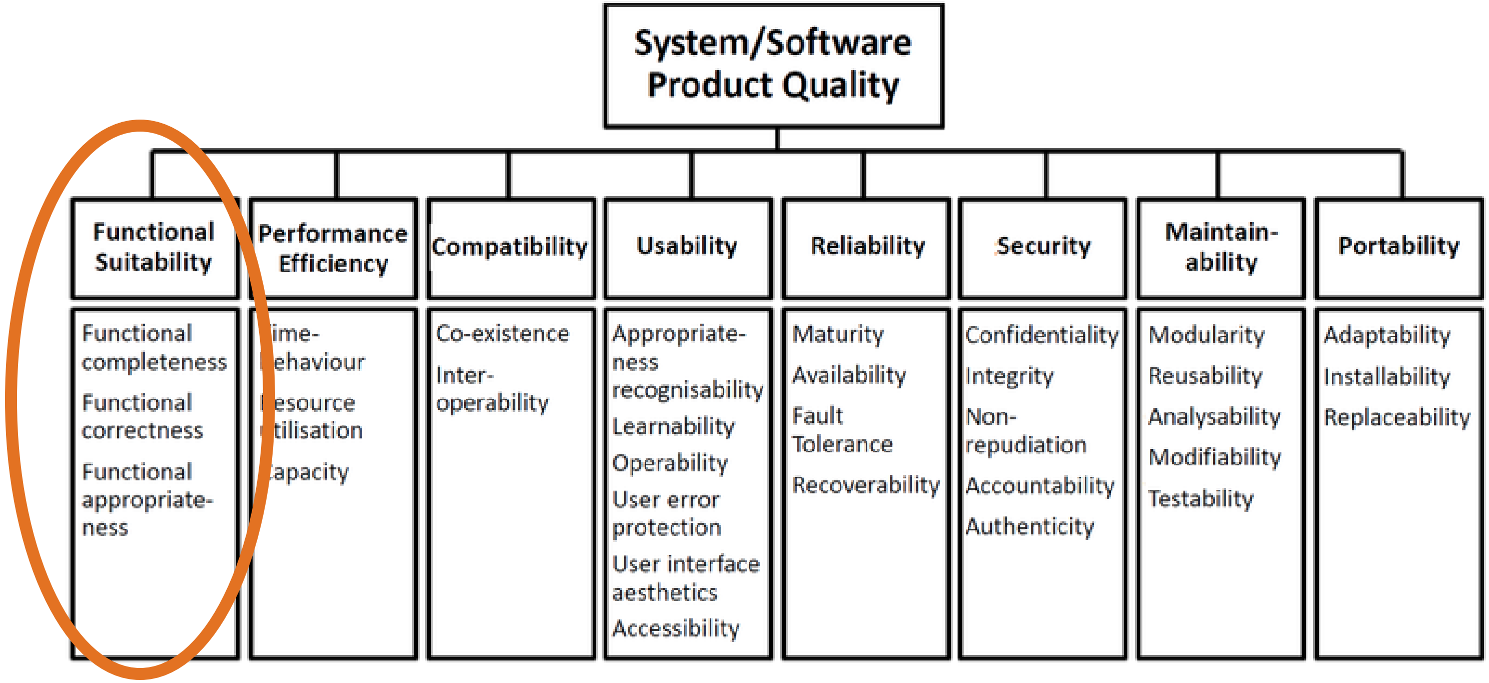
Quality does not only consist of non-functional requirements!

but also: **functional** correctness

internal Quality (properties of code)

vs

external Quality (user-observable)



System/Software Product Quality

Functional Suitability	Performance Efficiency	Compatibility	Usability	Reliability	Security	Maintainability	Portability
Functional completeness Functional correctness Functional appropriateness	Time-behaviour Resource utilisation Capacity	Co-existence Interoperability	Appropriateness recognisability Learnability Operability User error protection User interface aesthetics Accessibility	Maturity Availability Fault Tolerance Recoverability	Confidentiality Integrity Non-repudiation Accountability Authenticity	Modularity Reusability Analysability Modifiability Testability	Adaptability Installability Replaceability

Quality-in-use

Quality-in-use measures to which extent the software meets the **needs** of the user in the working environment (such as productivity, satisfaction and effectiveness)

Differentiation from **Usability**:

- Ease of interaction with the system.

- Part of external quality!

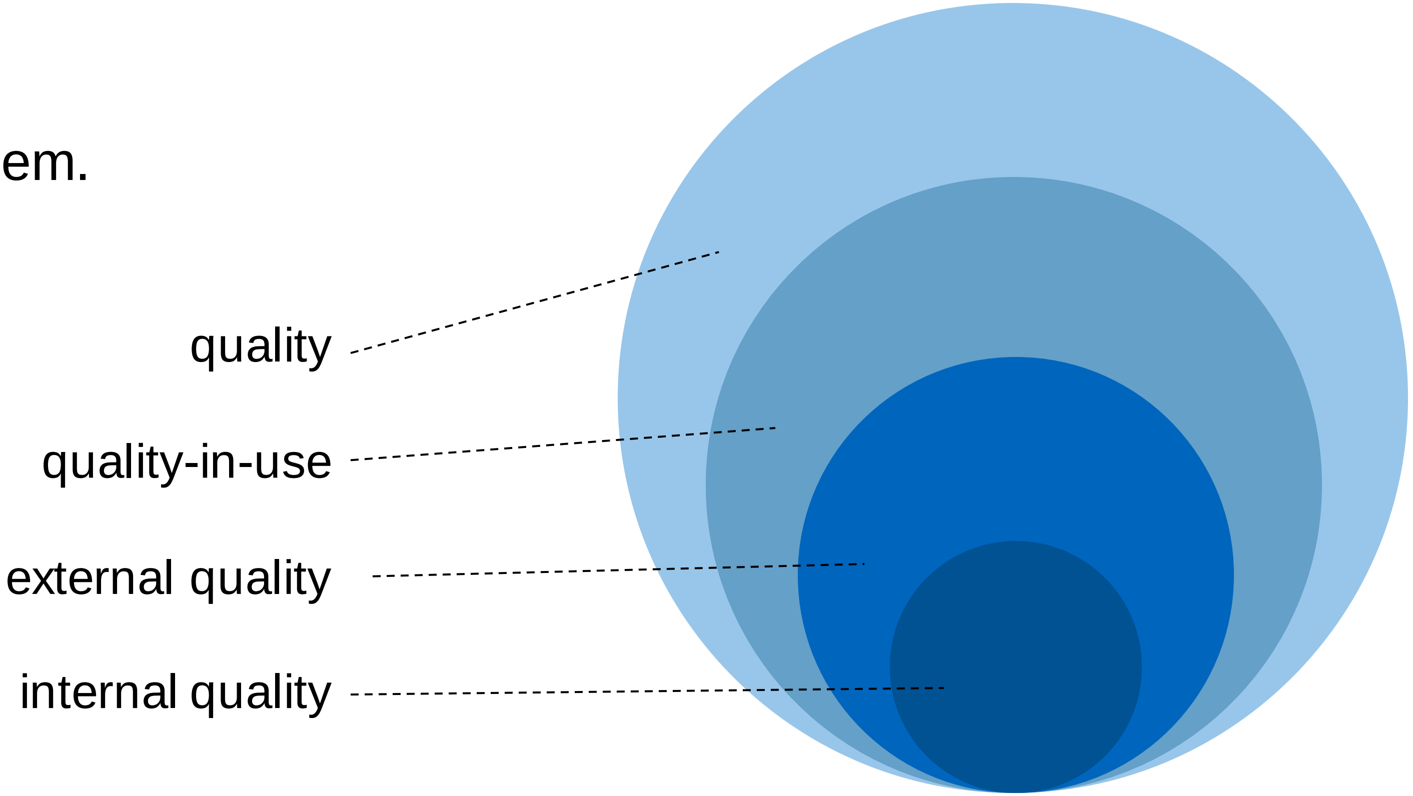


Figure and Definition: Bevan 1999 – Quality in use: Meeting user needs for quality

Stop!

When talking about the "quality of requirements", this wording alone implicitly assumes that these requirements already exist! This may well be the case, in the form of user needs, but knowing the "right" requirements, and writing them down the "right" way, precedes the notion of quality (unless "validity" is considered a quality).

This problem of validity is a fundamental problem, giving rise, among other things, to design thinking.

What are the requirements for rent-a-scooter?

Measuring System Quality

Functional Quality:

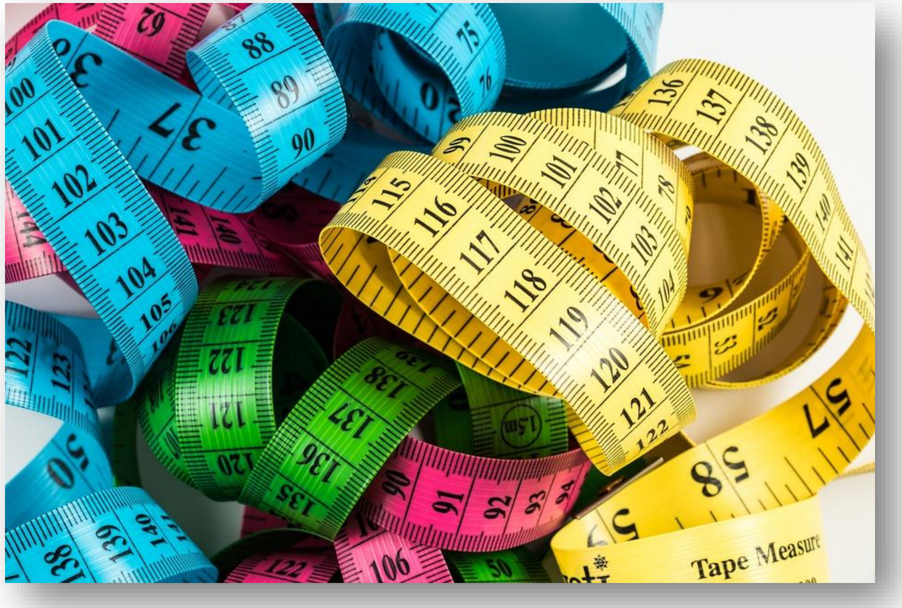
Is the system doing as intended? **=> Testing!**

Without verifiable requirements, testing is impossible!

Non-functional Quality:

Is the system "well" programmed? Quality is subjective!

- How to verify non-functional requirements?
- How to formulate non-functional requirements?



More details in Lecture 6

Outline and Outlook

Terms and Definitions *(Lecture 1)*

Core activities *(Lecture 1)*

Quality models for Requirements

Natural language as one source of possible quality defects

Engineering models in RE

Stakeholder and Requirements Elicitation

Goals and Goal-oriented RE

Non-functional requirements

Functional requirements

Formalization

Agile Processes

Requirements Management and Quality Assurance

Trends in Research