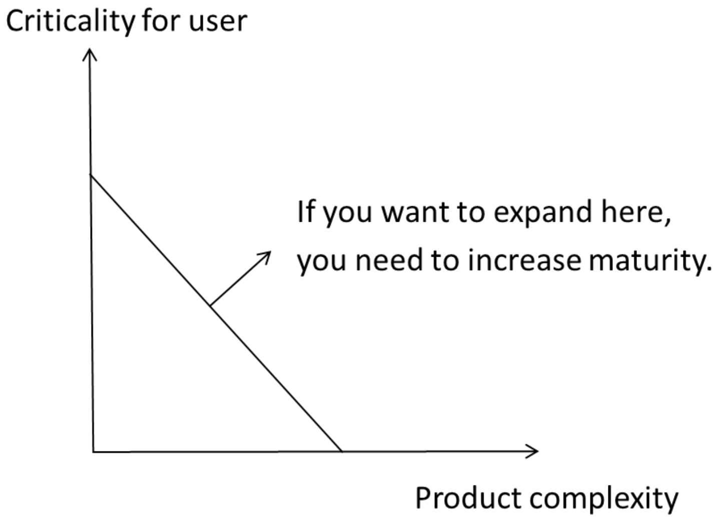
Course Introdction TDDE46

Kristian Sandahl, Azeem Ahmad

Linköping University, Sweden

Quality work in a software company is done for different reasons



Criticality for user

If you want to expand here, you need to increase maturity.

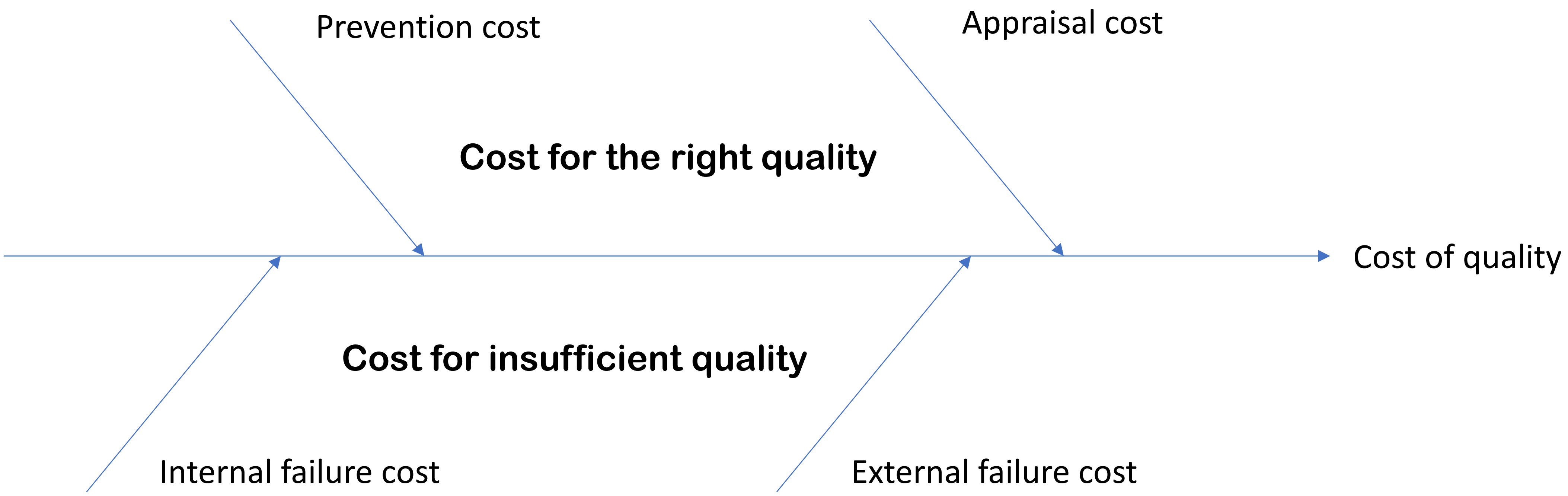
Product complexity

Take new challenges

Grow your assets

Avoid complaints

Quality is free?



Prevention cost

Appraisal cost

Cost for the right quality

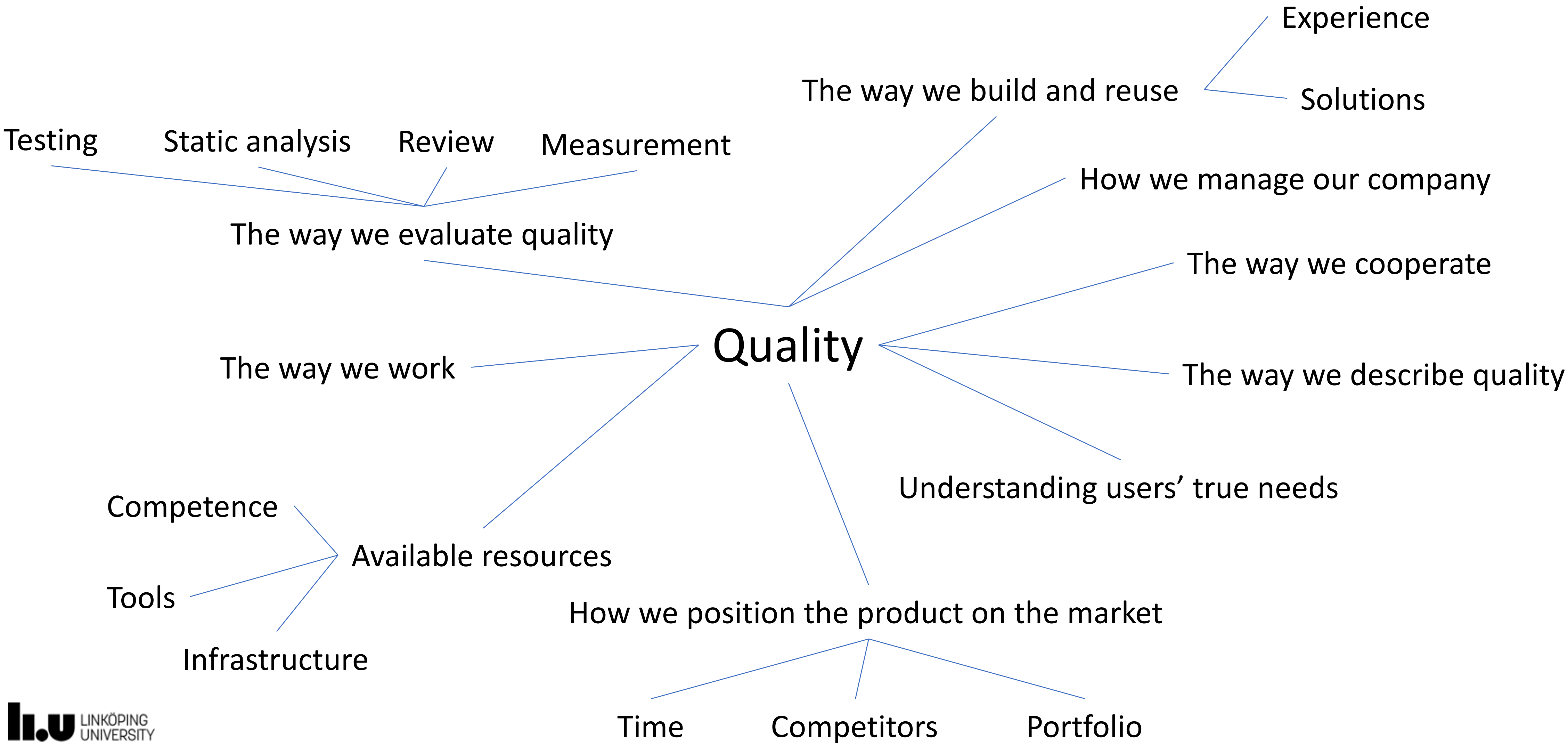
Cost of quality

Cost for insufficient quality

Internal failure cost

External failure cost

Drivers for quality



Quality

The way we evaluate quality

Testing

Static analysis

Review

Measurement

The way we build and reuse

Experience

Solutions

How we manage our company

The way we cooperate

The way we describe quality

Understanding users’ true needs

How we position the product on the market

Time

Competitors

Portfolio

The way we work

Available resources

Competence

Tools

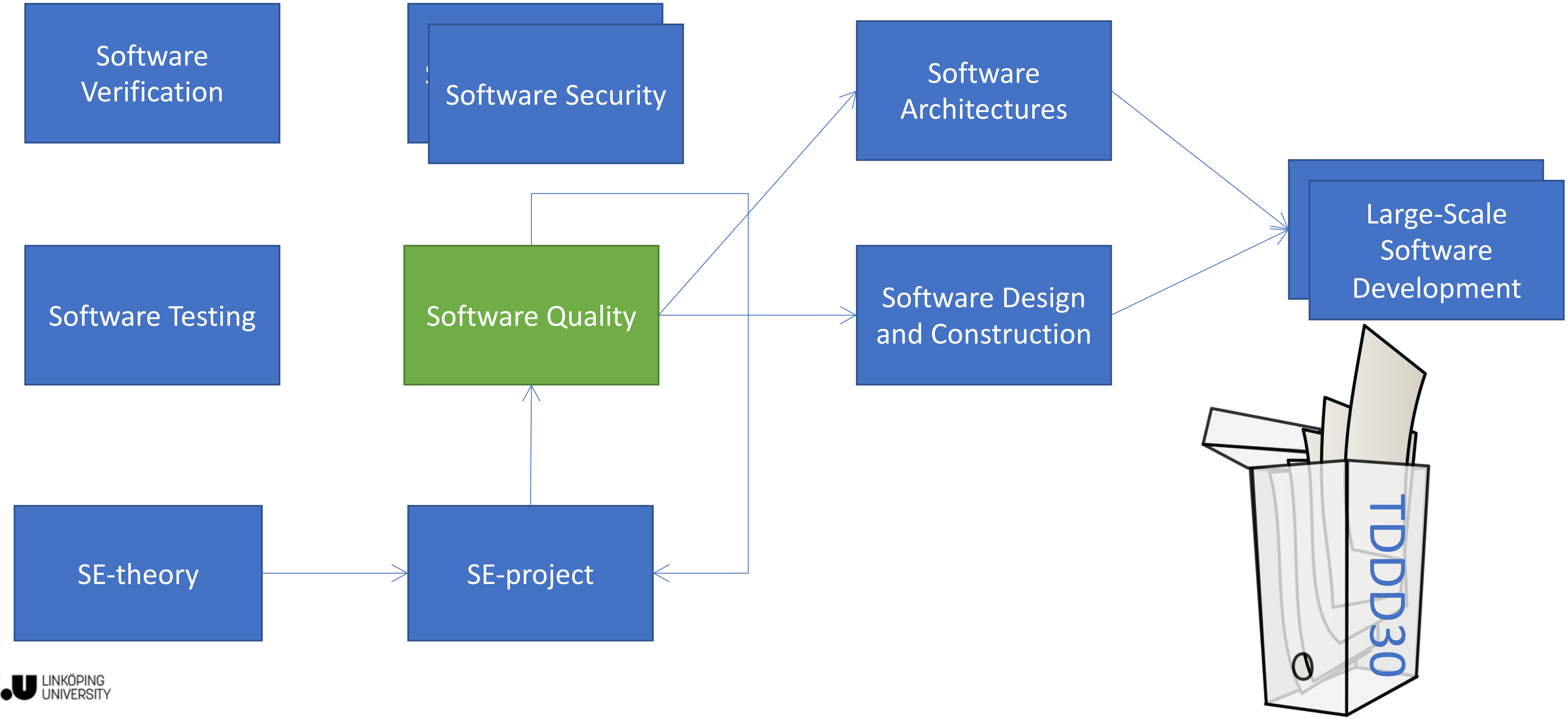
Infrastructure

Course content

- Measuring software products, processes and resources
- Quality planning
- Describing processes and practices with SEMAT Essence
- Process evaluation and improvement
- Software quality management
- Information quality and tools
- Coaching bachelor projects
- Evaluating bachelor projects

- Taught in other courses:
  - Software reviews
  - Formal verification
  - Software testing
  - Design for quality factors
- Taught at IEI:
  - Quality management
  - Statistical process control

The position in the course programme



Software Verification

Software Security

Software Architectures

Large-Scale Software Development

Software Testing

Software Quality

Software Design and Construction

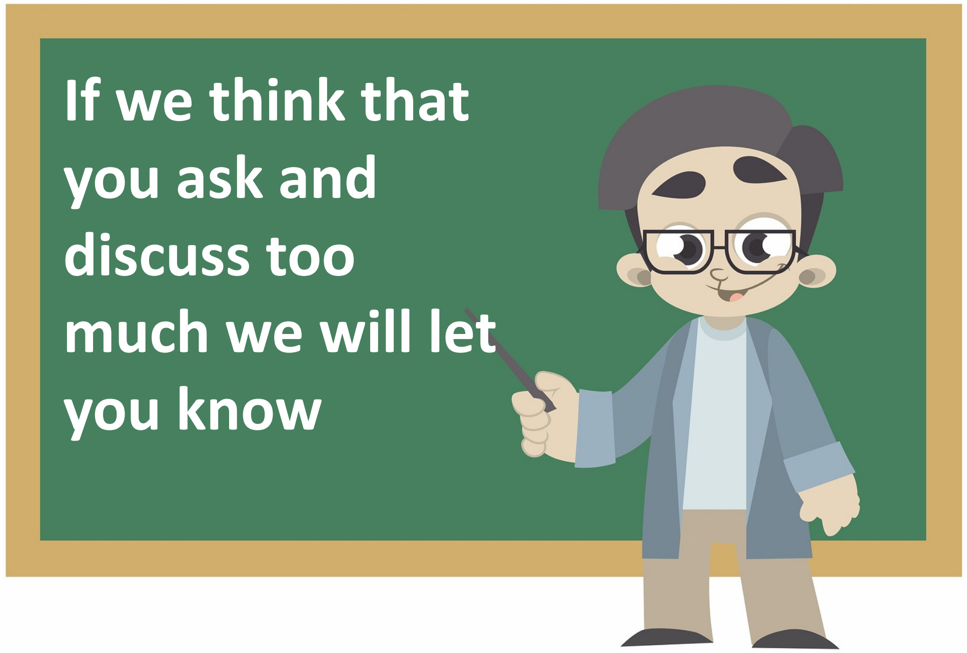
SE-theory

SE-project

Lectures give the most common theoretical aspects of software quality

- Well-known concepts
- Introduction to scientific literature and standards
- Practical hints

- Do not replace own studies



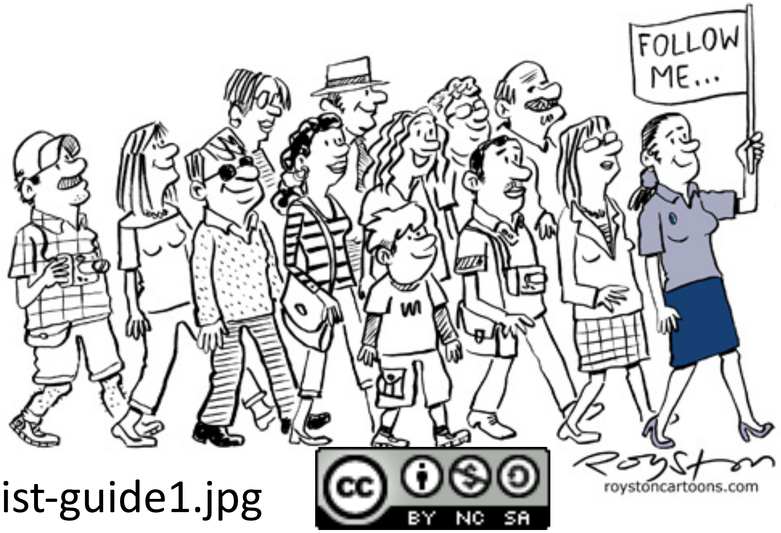
If we think that you ask and discuss too much we will let you know

Labs are there to give you hands on and reflection

- Tools for measurement
- Process modeling and improvement
- Verify requirements for an API
- Visualizing metrics with a dashboard
- Tools for robustness and availability.

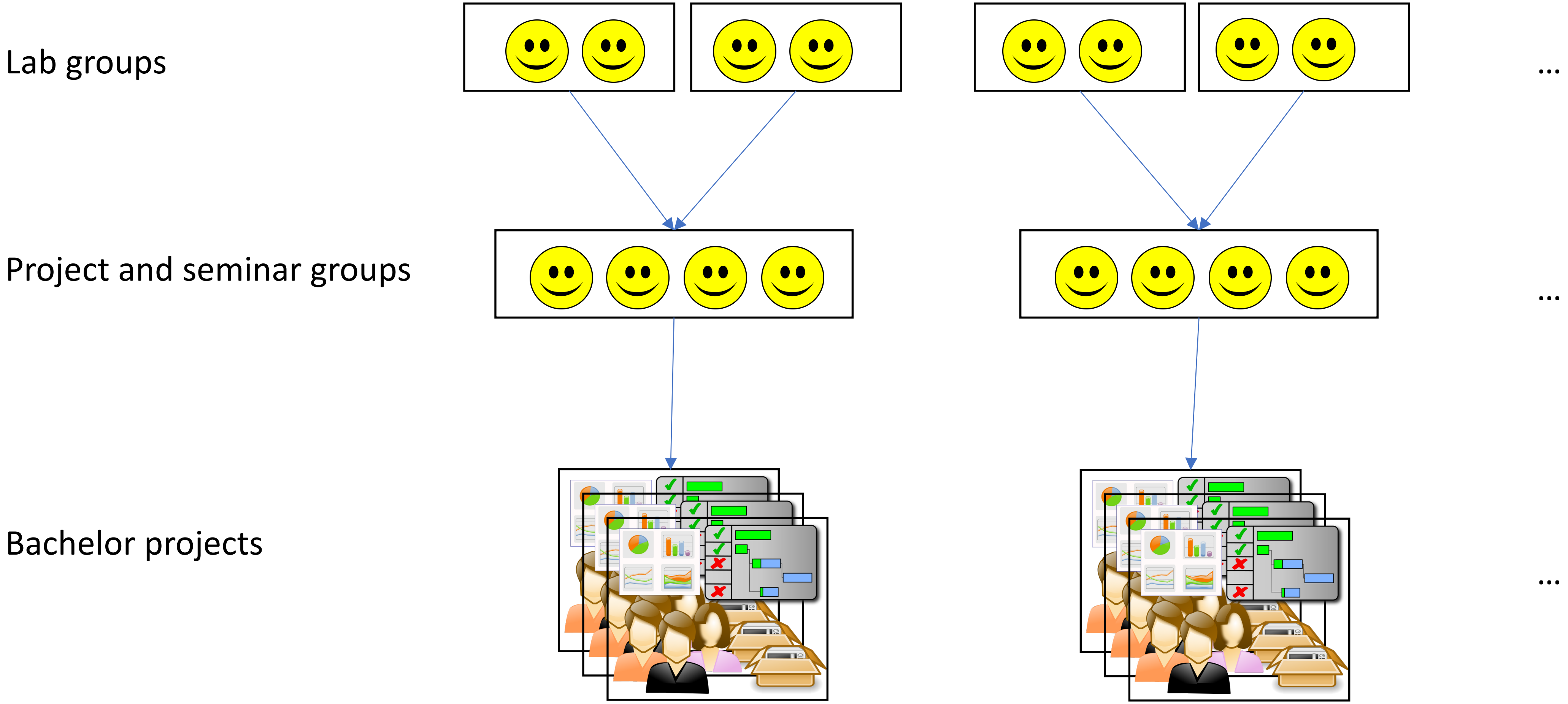
You will coach the bachelor thesis projects to get the process and people experience

- This is a true need
- You will help them to achieve:
  - Measurable quality requirements
  - A useful quality plan
- You will practice:
  - Identify, suggest and evaluate a process improvement
  - Evaluate a product



http://www.culturaltravelguide.com/wp-content/uploads/2012/05/plan-a-trip-tourist-guide1.jpg

Organisation



Lab groups

Project and seminar groups

Bachelor projects

…

…

…

In seminars we discuss articles and account for coaching activities

- Metrics and estimation
- Processes and practices
- Software quality management
- Automatization

- Prepare by reading
- Write a short individual report on major take-aways

- Each group will also give a presentation on a selected topic
- 20 min discussion of challenges in the coaching project

Finally, we write a report of the coaching project and individual contributions

- Common part about the coaching project
- Individual appendices:
  - Deepened follow up from seminars
  - Deepened follow up from 1-2 labs
  - Deepened follow up from the coaching project
  - Independent essay within the scope of the course

Grading rewards your ambition and achievements

- Lab series (G/U) – accomplish the lab goals, 2 hp
- Project work (U/3/4/5):
  - Seminars – **presence** and active participation, plus reflection report
  - Well-performed coaching
  - Common written report on the project

- Individual appendix:
- 10-line abstracts with major results
- 2-3 good references
- Understandable English
- About 4 pages excluding pictures and references

= grade 3

LINKÖPING UNIVERSITY

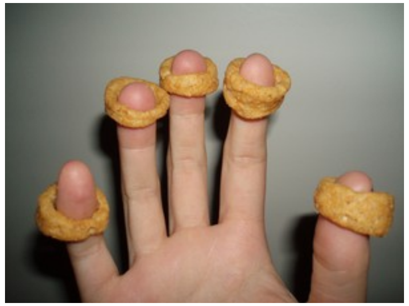
Grading rewards your ambition and achievements

- Individual appendix:
- 10-line abstracts with major results
- 10 good references
- Good, fault-free English
- 4-8 pages excluding pictures and references

= grade 4

- Same as grade 4 and:
- Clear comparison between 2-3 distinct findings in the coaching projects and the literature

= grade 5



Improvements from last year

- Clearer instruction on the coaching project
- New structure of seminars

Objectives for next 45 minutes

- What is software quality?
- Different views on quality.
- What are software quality factors?
- ISO 25010 standard in detail.

What does ‘Quality’ mean to you?

- You are a project manager of the BlaBla communication AB in Sweden. Your company develops networking hardware such as routers and switches. Your company also develops software for routers and switches to make it easy to use for the customers. You are competing with other 5 companies to get the contract for 1 million devices for the third world countries as part of United Nation campaign. This win this contract is very critical for your financial planning otherwise you can go bankrupt. Let’s suppose, the contract is to submit the following documentation of one line by just filling the blank. Your answer (i.e. for the blank) must be between 1-7 words.

Your hardware and/or software has a good quality because it ______________!

Go to www.menti.com and use the code 53 10 76

WE OFFER 3 KINDS OF SERVICES

GOOD · CHEAP · FAST

BUT YOU CAN PICK ONLY TWO

GOOD & CHEAP WON'T BE FAST

FAST & GOOD WON'T BE CHEAP

CHEAP & FAST WON'T BE GOOD

©OPEN ROAD BRANDS, LLC

https://us.amazon.com/Open-Road-Brands-Services-Metal/dp/B073HK22KK

What is Software Quality?

- Quality is a complex and multifaceted concept (David Garvin)
- Quality = Fitness for Purpose (Juran)
- Quality = Zero Defect (Crosby)

Different Views of Quality, David Garvin:[1]

- **Transcendental view**: that quality is something that you immediately recognize, but cannot explicitly define.
- **User view**: sees quality in terms of an end user’s specific goals. If a product meets those goals, it exhibits quality.
    - A product is of good quality, if it satisfy a large number of users
- **Manufacturer view**: defines quality in terms of the original specification of the product. If the product conforms to the specification, it exhibits quality.
    - Does the product satisfy the specification/requirements?
- **Product view**: suggests that quality can be tied to inherent characteristics (e.g., module approach and feature's re-usability) of a product.
- **Value-based view**: measures quality based on how much a customer is willing to pay for a product.
    - How much a customer is willing to pay for a certain level of quality?

[1] Garvin, D., “What Does ‘Product Quality’ Really Mean?” Sloan Management Review, Fall 1984, pp. 25–45

Software Quality Factors

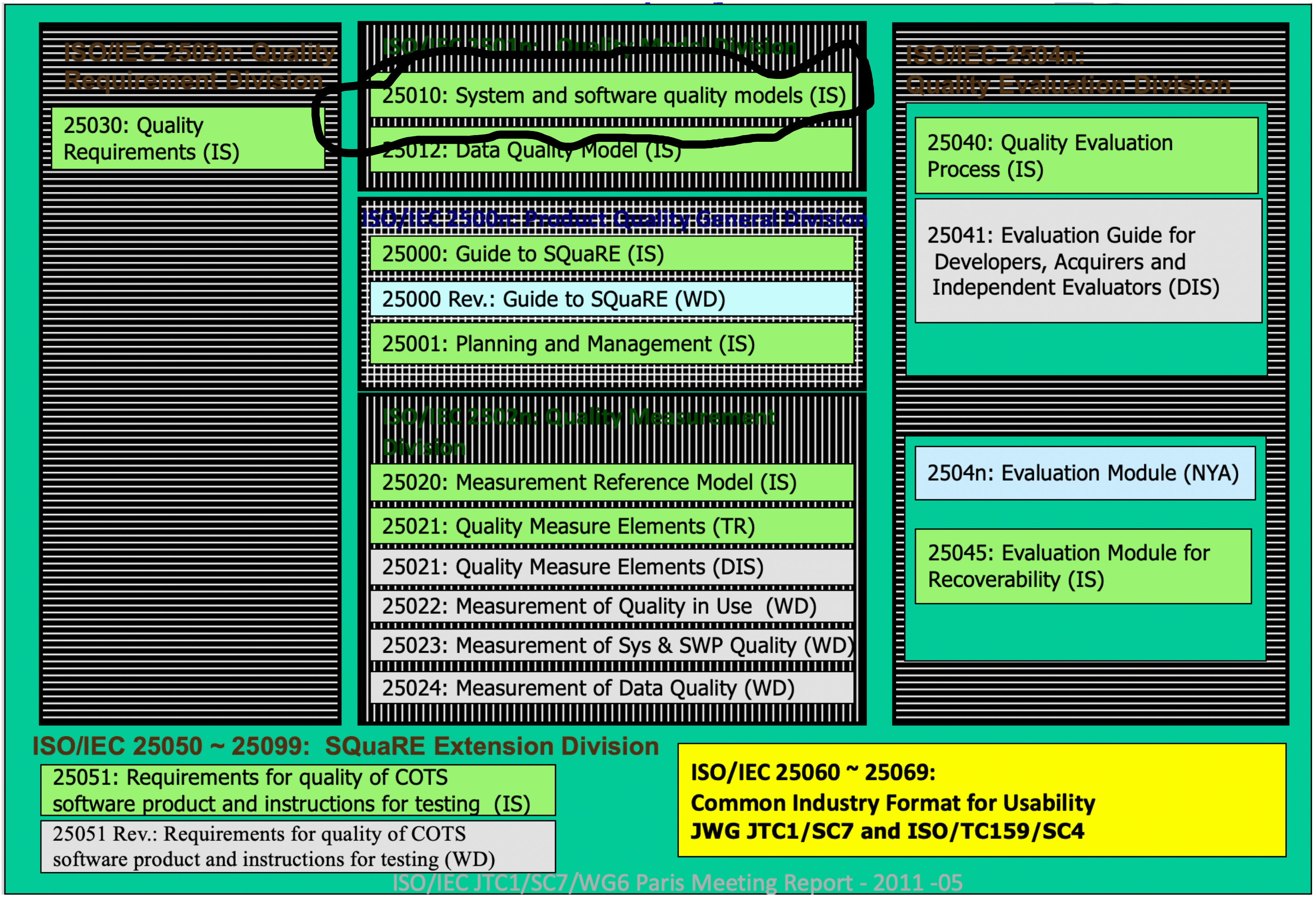
- To know that quality has improved, it would be helpful to be able to measure quality in term of factors.
- Quality Factors (Availability, Reliability, Maintainability, Security …..):

Software Quality Standards & Models

- Standards (i.e. international, regional, national or organizational):
  - Increase customer's confidence in the products
  - Help to achieve effective quality management
  - Avoid repetitions of past mistakes
    - Developed by:
      - IEEE, ISO, DOD (US Department of Defence), ANSI (American National Standard Organization), EIA (Electronic Industries Association) and others
- Models:
  - McCall -> 11 quality factors (1977)
  - Deutsch and Willis -> 12 quality factors (1988)
  - Evans and Marciniak-> 15 quality factors (1987)
  - Many others (Seminar 1 is about quality models)

Software Quality Standards – ISO 250XX

Software Product Quality Requirements and Evaluation (SQuaRE)



ISO/IEC 2503n: Quality Requirement Division

25030: Quality Requirements (IS)

ISO/IEC 2501n: Quality Model Division

25010: System and software quality models (IS)

25012: Data Quality Model (IS)

ISO/IEC 2500n: Product Quality General Division

25000: Guide to SQuaRE (IS)

25000 Rev.: Guide to SQuaRE (WD)

25001: Planning and Management (IS)

ISO/IEC 2502n: Quality Measurement Division

25020: Measurement Reference Model (IS)

25021: Quality Measure Elements (TR)

25021: Quality Measure Elements (DIS)

25022: Measurement of Quality in Use (WD)

25023: Measurement of Sys & SWP Quality (WD)

25024: Measurement of Data Quality (WD)

ISO/IEC 2504n: Quality Evaluation Division

25040: Quality Evaluation Process (IS)

25041: Evaluation Guide for Developers, Acquirers and Independent Evaluators (DIS)

2504n: Evaluation Module (NYA)

25045: Evaluation Module for Recoverability (IS)

ISO/IEC 25050 ~ 25099: SQuaRE Extension Division

25051: Requirements for quality of COTS software product and instructions for testing (IS)

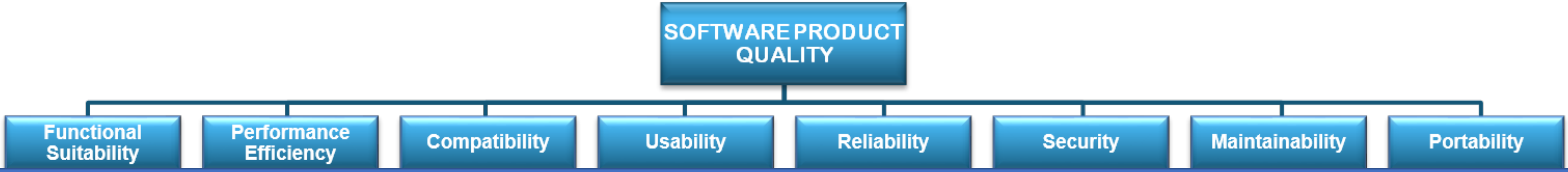
25051 Rev.: Requirements for quality of COTS software product and instructions for testing (WD)

ISO/IEC 25060 ~ 25069:
Common Industry Format for Usability
JWG JTC1/SC7 and ISO/TC159/SC4

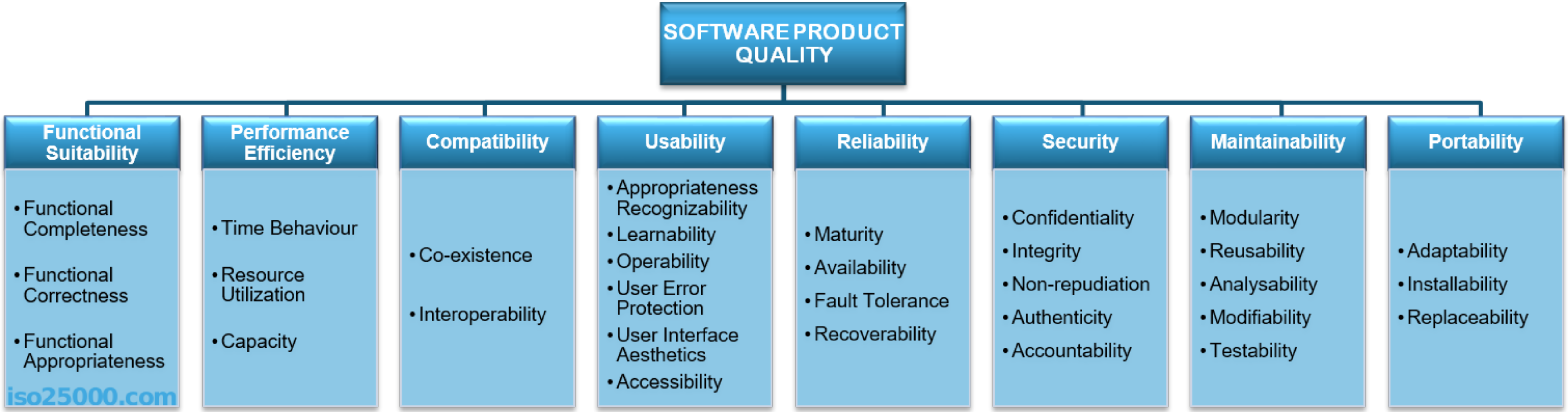
ISO/IEC JTC1/SC7/WG6 Paris Meeting Report - 2011 -05

ISO/IEC 25010

“The quality of a system is the degree to which the system **satisfies the stated and implied needs of its various stakeholders**, *and thus provides value. Those stakeholders' needs (functionality, performance, security, maintainability, etc.) are precisely what is represented in the quality model [……]”* (www.iso2500.com)



ISO/IEC 25010



SOFTWARE PRODUCT QUALITY

Functional Suitability
- Functional Completeness
- Functional Correctness
- Functional Appropriateness

Performance Efficiency
- Time Behaviour
- Resource Utilization
- Capacity

Compatibility
- Co-existence
- Interoperability

Usability
- Appropriateness Recognizability
- Learnability
- Operability
- User Error Protection
- User Interface Aesthetics
- Accessibility

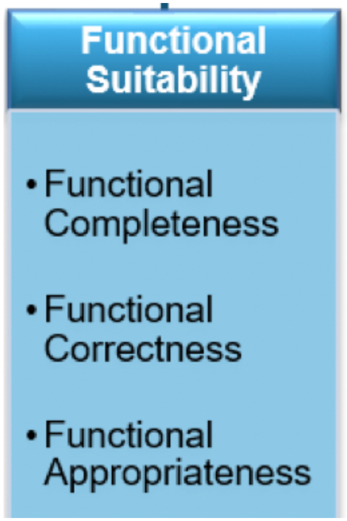
Reliability
- Maturity
- Availability
- Fault Tolerance
- Recoverability

Security
- Confidentiality
- Integrity
- Non-repudiation
- Authenticity
- Accountability

Maintainability
- Modularity
- Reusability
- Analysability
- Modifiability
- Testability

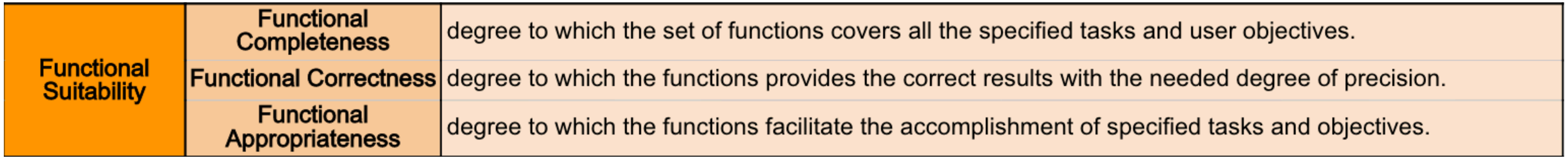
Portability
- Adaptability
- Installability
- Replaceability

ISO/IEC 25010



Functional Suitability

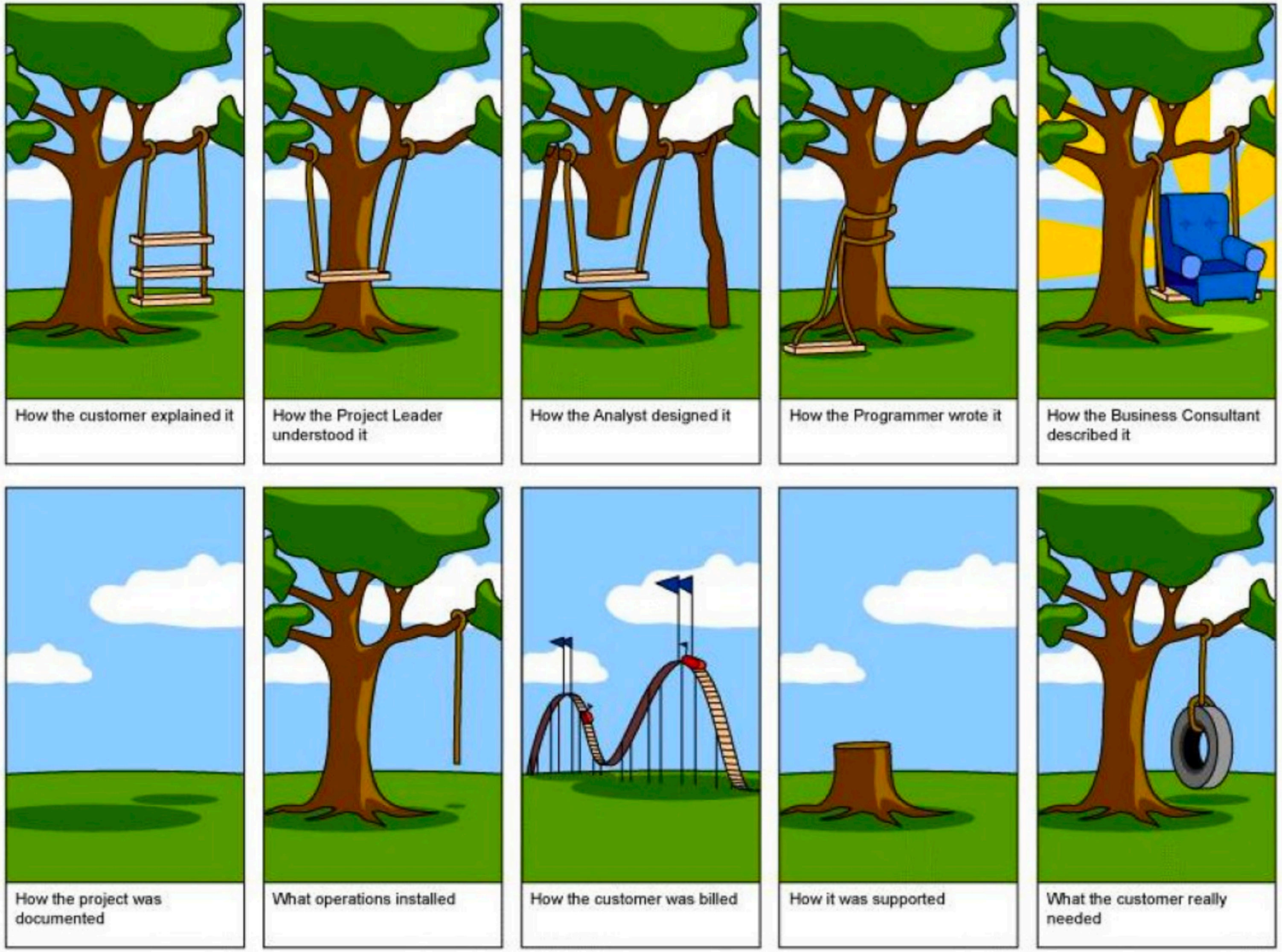
- Functional Completeness
- Functional Correctness
- Functional Appropriateness



Functional Suitability	Functional Completeness	degree to which the set of functions covers all the specified tasks and user objectives.
	Functional Correctness	degree to which the functions provides the correct results with the needed degree of precision.
	Functional Appropriateness	degree to which the functions facilitate the accomplishment of specified tasks and objectives.

- Some lessons to take away; what should you do?
  - Check if all use cases have been written.
  - Use traceability matrices to make sure that software matched with all business requirements **completely and correctly.**

  - *Everyone in Sweden should pay tax -> complete?*
  - *Car rental prices shall show all applicable taxes (including 6% of kummun tax) ->correct?*



How the customer explained it

How the Project Leader understood it

How the Analyst designed it

How the Programmer wrote it

How the Business Consultant described it

How the project was documented

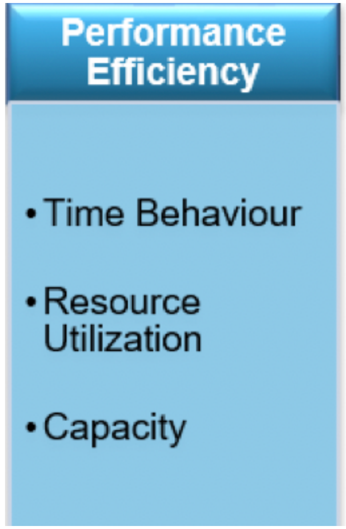
What operations installed

How the customer was billed

How it was supported

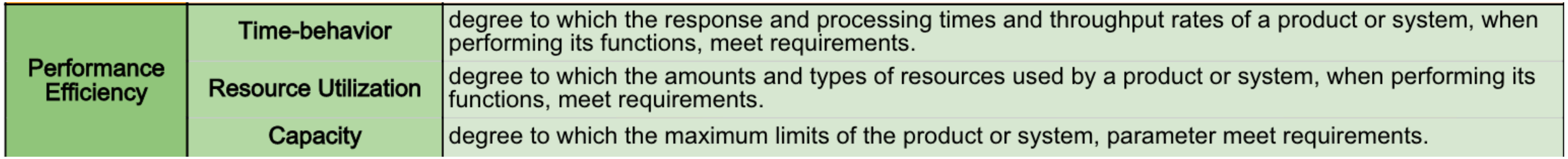
What the customer really needed

ISO/IEC 25010



Performance Efficiency

- Time Behaviour
- Resource Utilization
- Capacity



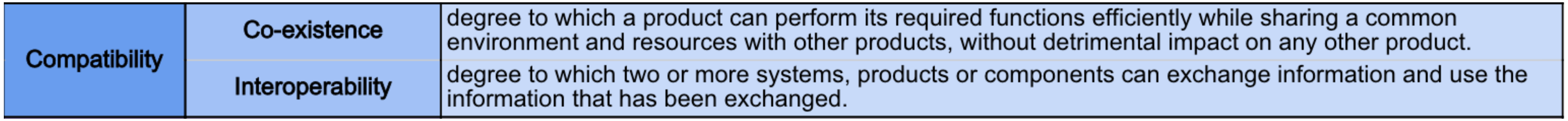
Performance Efficiency	**Time-behavior**	degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
	Resource Utilization	degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
	Capacity	degree to which the maximum limits of the product or system, parameter meet requirements.

- Some lessons to take away; what should you do?
  - Select appropriate data structure when writing code
  - Select appropriate algorithms
  - Consider data cashing, memory leakage, multi-threading, locking and other issues

ISO/IEC 25010

Compatibility

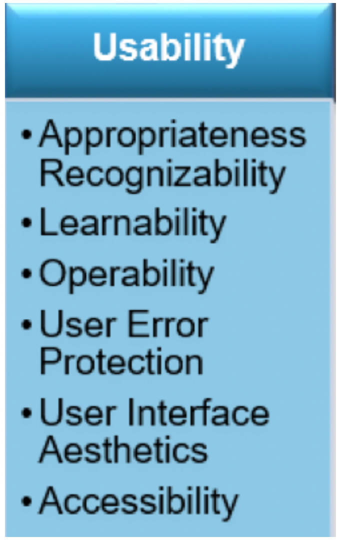
- Co-existence
- Interoperability



Compatibility	Co-existence	degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
	Interoperability	degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

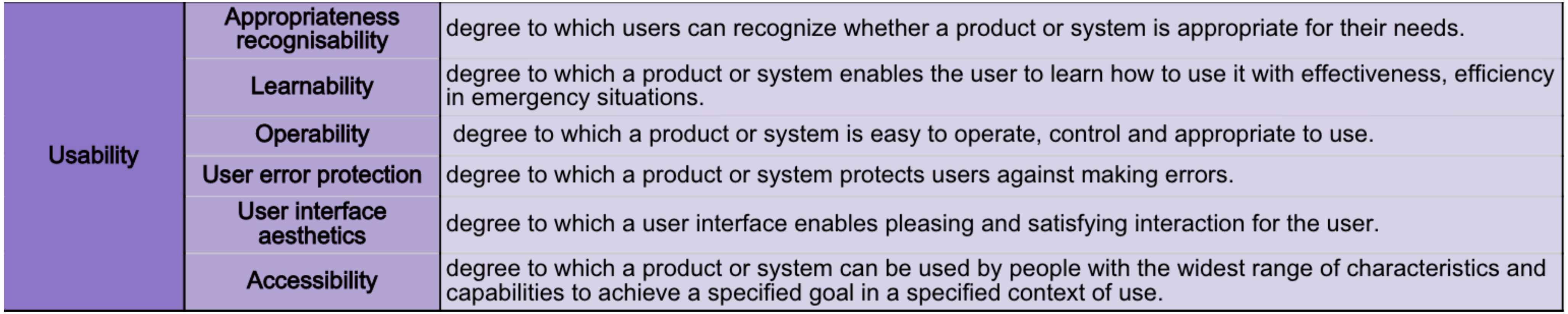
- Some lessons to take away; what should you do?
  - Write code that can communicate with different platform such as Windows, Mac, Linux, Ubuntu
  - Ability to use different functions (open-source, COTS, etc.) to make one complete program

ISO/IEC 25010



Usability

- Appropriateness Recognizability
- Learnability
- Operability
- User Error Protection
- User Interface Aesthetics
- Accessibility



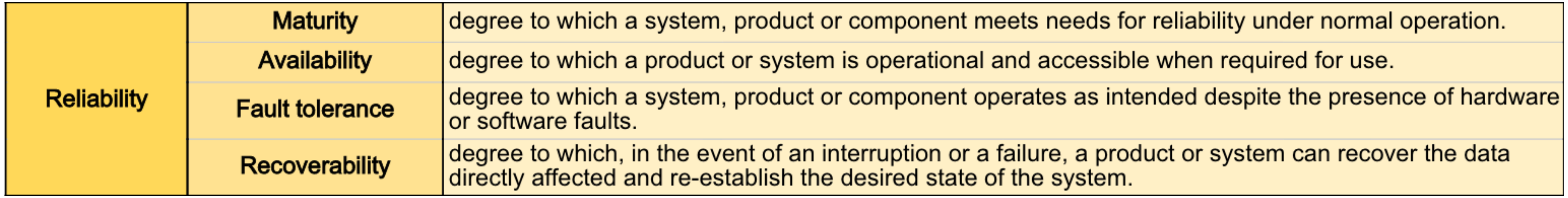
Usability	**Appropriateness recognisability**	degree to which users can recognize whether a product or system is appropriate for their needs.
	Learnability	degree to which a product or system enables the user to learn how to use it with effectiveness, efficiency in emergency situations.
	Operability	degree to which a product or system is easy to operate, control and appropriate to use.
	User error protection	degree to which a product or system protects users against making errors.
	User interface aesthetics	degree to which a user interface enables pleasing and satisfying interaction for the user.
	Accessibility	degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

- Keep it simple
- Keep it consistent
- Include a search bar

ISO/IEC 25010

Reliability

- Maturity
- Availability
- Fault Tolerance
- Recoverability



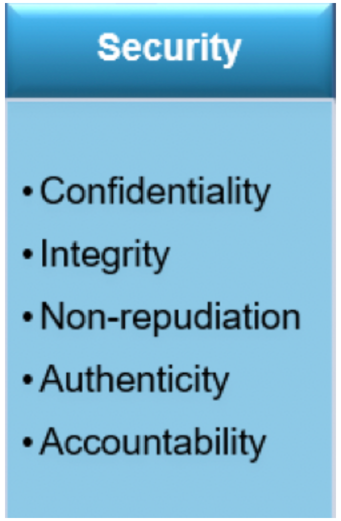
Reliability	Maturity	degree to which a system, product or component meets needs for reliability under normal operation.
	Availability	degree to which a product or system is operational and accessible when required for use.
	Fault tolerance	degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
	Recoverability	degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

- Some lessons to take away; what should you do?
  - Pay attention to exception handling
  - Keep the state save, when crashing and load program back to that state – crashing gracefully

99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.

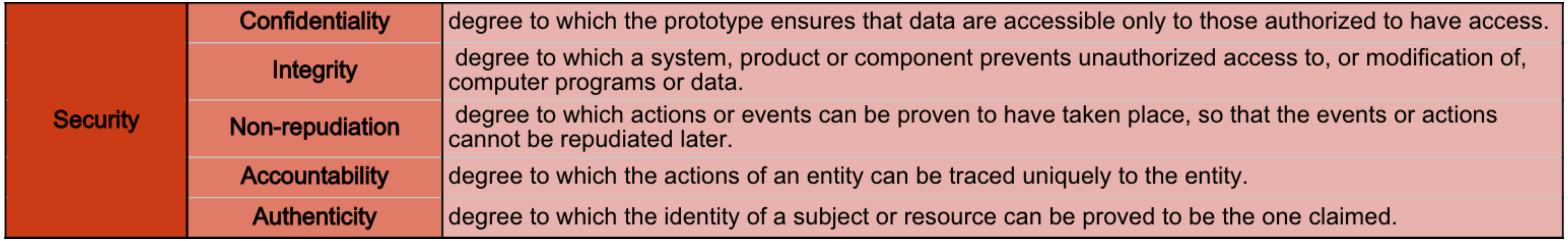
127 little bugs in the code...

ISO/IEC 25010



Security

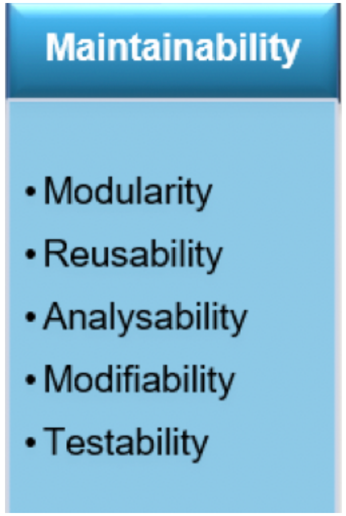
- Confidentiality
- Integrity
- Non-repudiation
- Authenticity
- Accountability



Security	**Confidentiality**	degree to which the prototype ensures that data are accessible only to those authorized to have access.
	Integrity	degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.
	Non-repudiation	degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.
	Accountability	degree to which the actions of an entity can be traced uniquely to the entity.
	Authenticity	degree to which the identity of a subject or resource can be proved to be the one claimed.

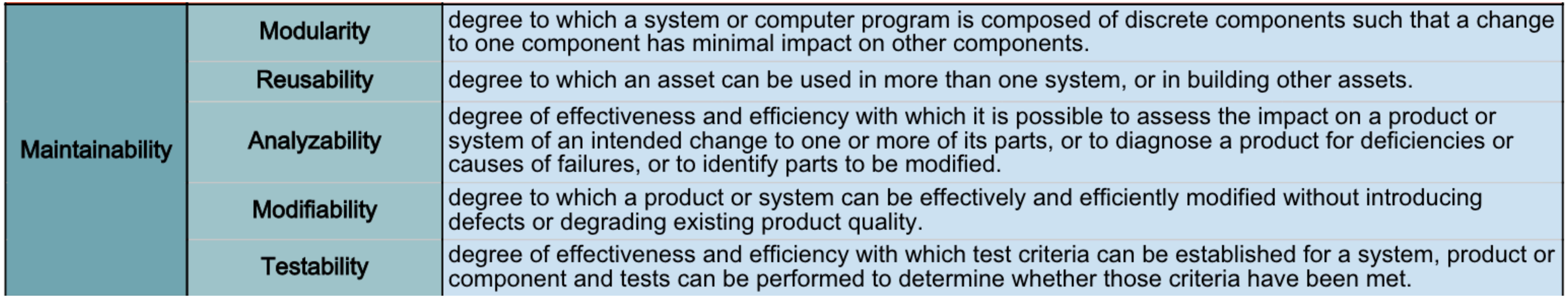
- Some lessons to take away; what should you do?
  - Pay attention to three AAAs (Authentication, Authorization & Access Control) for taking care of CIA (Confidentiality, Integrity and Availability) attributes
  - Pay attention to OWASP vulnerabilities' recommendation and focus on doing input validation to take care of issues related with CIA
  - Make sure that there are test cases (security testing) to test the above aspects

ISO/IEC 25010



Maintainability

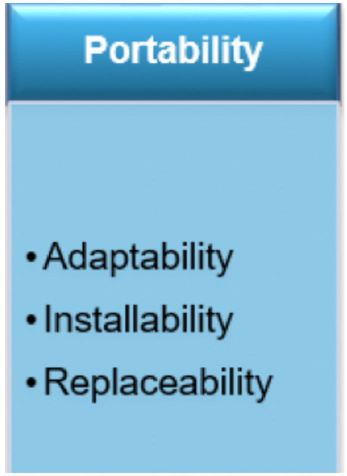
- Modularity
- Reusability
- Analysability
- Modifiability
- Testability



Maintainability	**Modularity**	degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
	Reusability	degree to which an asset can be used in more than one system, or in building other assets.
	Analyzability	degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
	Modifiability	degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
	Testability	degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

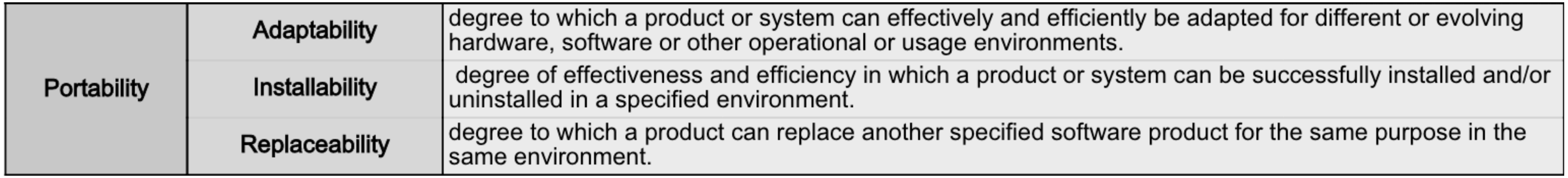
- Some lessons to take away; what should you do?
  - Apply object-oriented principles to take care of concerns such as reusability & modularity while doing both, design and coding.
  - Use static code analysis tools

ISO/IEC 25010



Portability

- Adaptability
- Installability
- Replaceability



Portability	Adaptability	degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
	Installability	degree of effectiveness and efficiency in which a product or system can be successfully installed and/or uninstalled in a specified environment.
	Replaceability	degree to which a product can replace another specified software product for the same purpose in the same environment.

- Some lessons to take away; what should you do?
  - Either write N software version for N platform or use JVM (Java Virtual Machine)
  - Either the executable code is portable or the source code must be re-compiled for each platform

Key Points

- Consider activities related to software quality as early as possible in SDLC
- Software quality depends in the eyes of behold
- Certainly the nature of the software dictates more concern on some of these factors than others (safety, for example)
- There are no universally applicable model or standard for software quality. Each organization can choose a model or standard according to their need
- Standards and Models are a good start.