Software Testing and Quality Assurance

Lecture 2

Fabian Fagerholm

fabian.fagerholm@aalto.fi

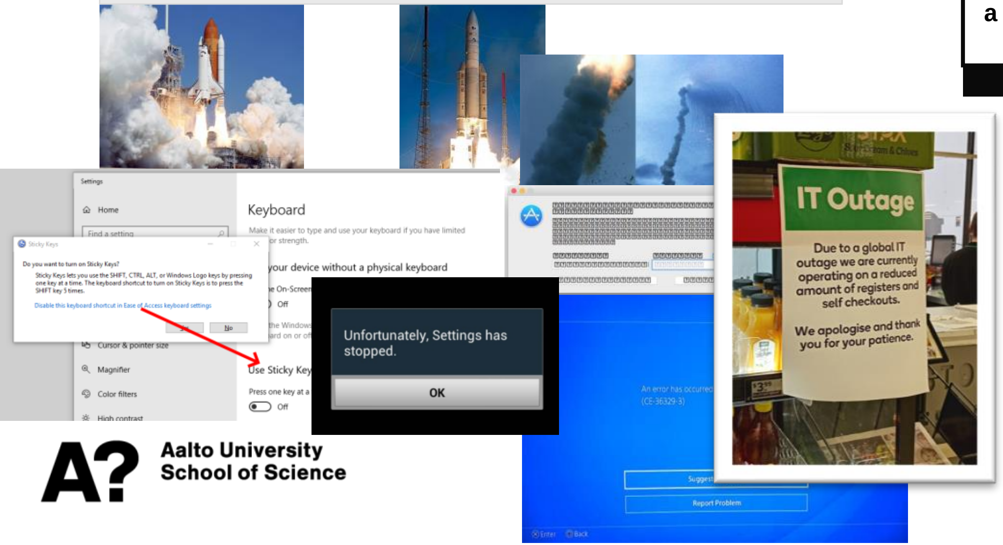
Aalto University
School of Science

Week	Lecture	Topic	Assignment deadlines (Tuesdays 10:00 unless otherwise specified)
36	3.9.2024	Introduction and practicalities	
37	10.9.2024	Software quality	Individual assignments 1
38	17.9.2024	Software testing: levels, test case analysis and design	Group registration (DL: 20.9.)
39	24.9.2024	Testing techniques: Black-box testing	Individual assignments 2, Group assignment 1
40	1.10.2024	Testing techniques: Black-box testing	Individual assignments 3
41	8.10.2024	Testing techniques: Manual testing	Individual assignments 4
42	15.10.2024	(No lecture)	
43	22.10.2024	Testing techniques: White-box testing	Individual assignments 5
44	29.10.2024	Testing techniques: White-box testing	Individual assignments 6, Group assignment 2
45	5.11.2024	Guest lecture	Individual assignments 7
46	12.11.2024	Testing techniques: Static code analysis and software metrics	Individual assignments 8
47	19.11.2024	Continuous Integration and Continuous Delivery/Deployment	Individual assignments 9
48	26.11.2024	Test management	Individual assignments 10, Group assignment 3
49	3.12.2024	(No lecture)	Individual assignments 11, Group assignment 5

Subject to changes

Summary of previous lecture

- Software failures can lead to
  - catastrophic incidents with major losses
  - annoying incidents that contribute to a bad user experience
- Many failures could be avoided by proper attention to testing and quality assurance



Sticky Keys

Do you want to turn on Sticky Keys?

Sticky Keys lets you use the SHIFT, CTRL, ALT, or Windows Logo keys by pressing one key at a time. The keyboard shortcut to turn on Sticky Keys is to press the SHIFT key 5 times.

Disable this keyboard shortcut in Ease of Access keyboard settings

Unfortunately, Settings has stopped.

An error has occurred (CE-36329-3)

IT Outage

Due to a global IT outage we are currently operating on a reduced amount of registers and self checkouts.

We apologise and thank you for your patience.

Aalto University
School of Science

Software Quality Management (SQM)

Software Quality Assurance (SQA)
- **A general quality guide, not specific to a project**

Software Quality Control (SQC)
- **Examine artefacts for compliance**
- E.g. inspection, reviews, testing

Software Quality Planning (SQP)
- **Project-level quality commitment**
- **Based on SQA**

Software Process Improvement (SPI)
- Improve process quality

⚠ **Error** (mistake)

Fault (defect)

✕ **Failure**

Incident

Important concepts: *Quality*

- "Conformance to the requirements" (Crosby)
  - Ignores intrinsic quality differences between products
  - Does not consider whether requirements are appropriate for the product
- "Fitness for use" (Juran)
  - No mechanism to judge better quality when two products are equally fit for use
- ISO/IEC/IEEE 24765:2017 (emphasis added):
  1. degree to which the system *satisfies the stated and implied needs of its various stakeholders*, and thus *provides value*
  2. ability of a product, service, system, component, or process to *meet customer or user needs, expectations, or requirements*
  3. the degree to which a set of inherent characteristics *fulfils requirements*

For testing and quality assurance to work, we need to know what quality means in our specific case

Software Quality

Quality

- Being superior or not inferior
- Being suitable for an intended purpose while satisfying customer expectations
- Perceptual, somewhat subjective: may be understood differently by different people

Consumer focus: specification quality

- Example: How the product compares to competitors

Producer focus: conformance quality

- Example: The degree to which the product/service was produced correctly

Support focus

- Example: The degree to which a product/service is reliable, maintainable, or sustainable

Software Quality

- Software quality is a multifaceted concept
- It is not only a property of the code
- It includes the quality of several artefacts and processes, for example:

Operating procedures

- A software program is part of a larger system and operating environment – it has users that can be other software and humans

Documentation

- Descriptions of the software for development and use

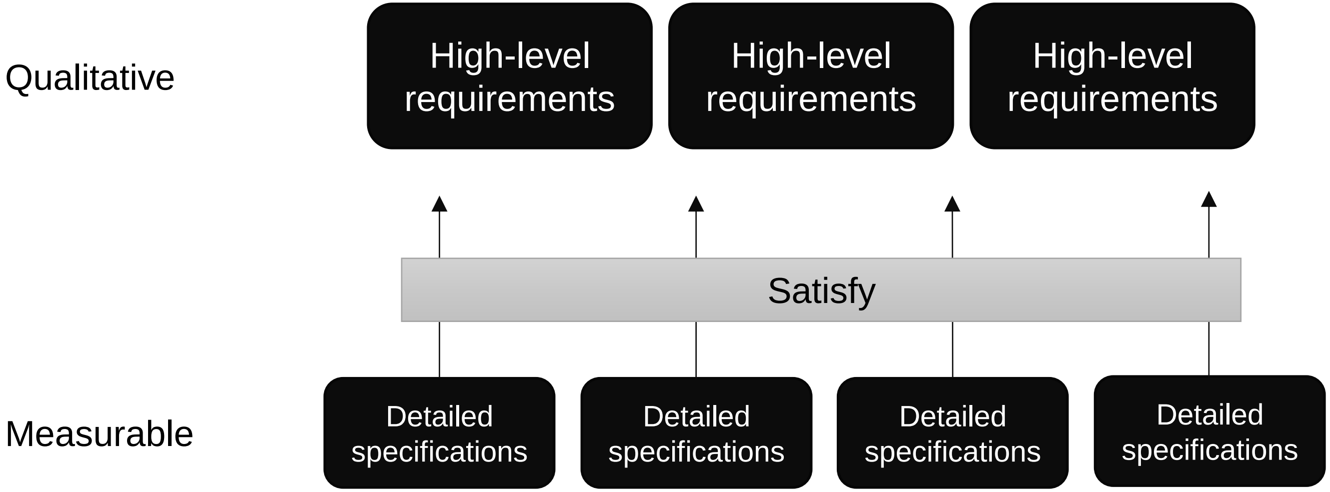
Data

- The data needed in operation

Code

- The software code itself

Decomposing software quality



- High-level requirements are broken down into detailed specifications
- The detailed specifications satisfy the requirements
- The specifications are measurable

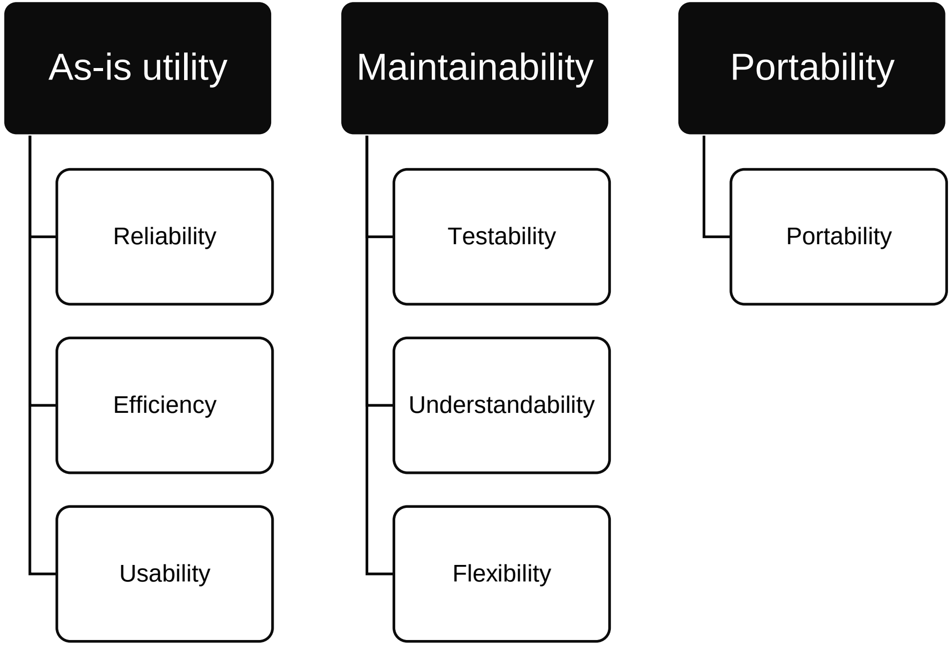
→ A way to test, inspect, review and assure quality

Challenges for software quality:

- What model of quality characteristics (quality requirements) should be used?
- How should the requirements be broken down into measurable, quantitative attributes?

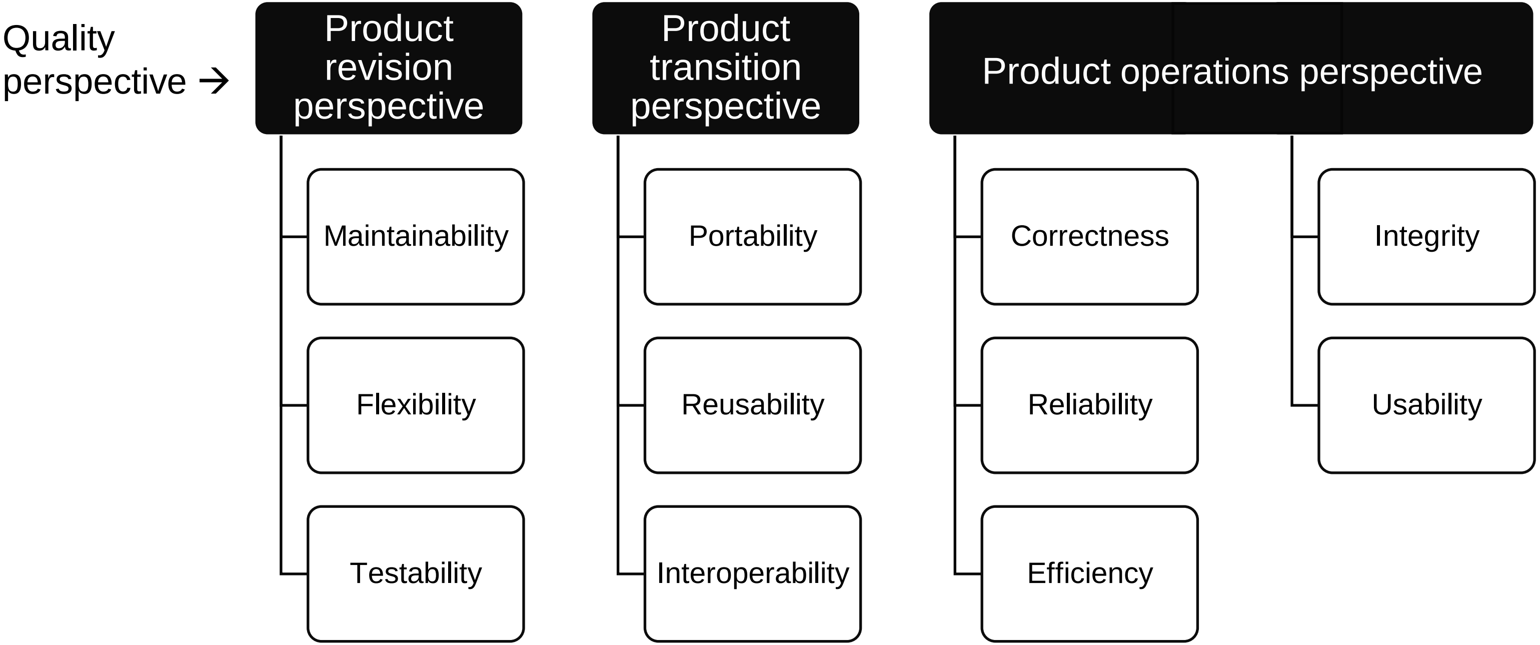
Quality characterisation models

General utility →



Boehm' model

Quality characterisation models

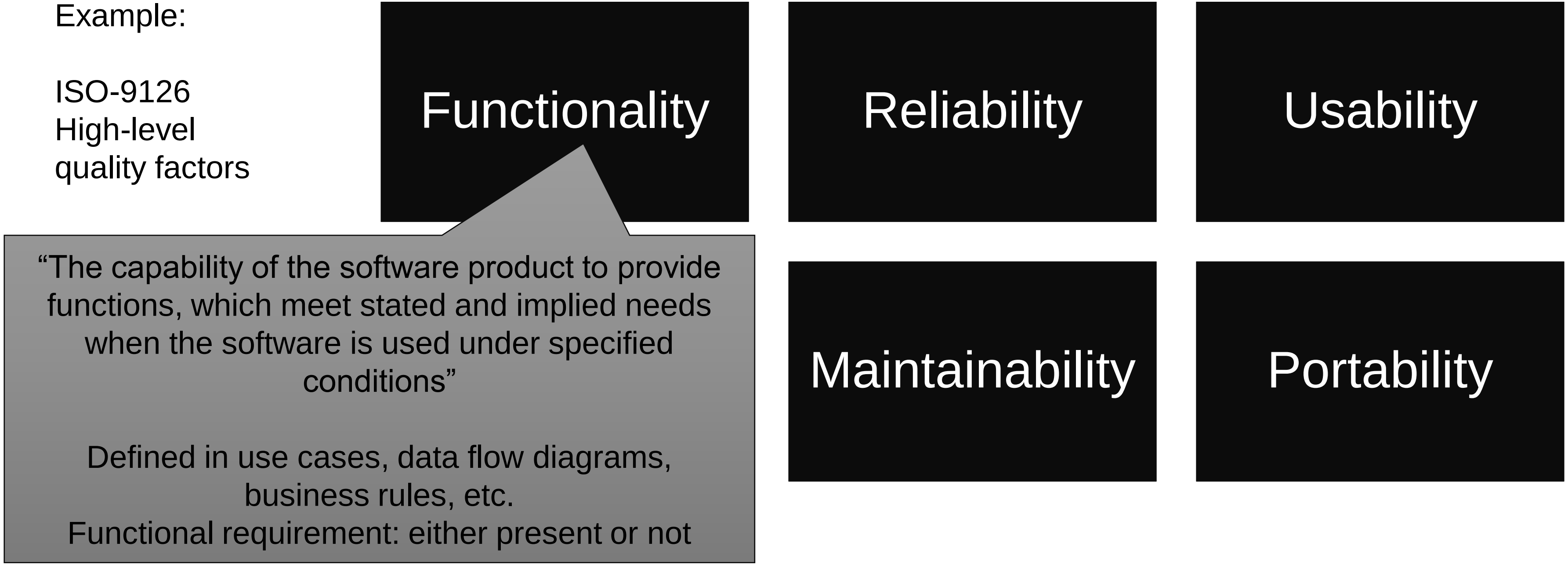


McCall's model

Quality characterisation models

Example:

ISO-9126
High-level
quality factors



Functionality

Reliability

Usability

Maintainability

Portability

“The capability of the software product to provide functions, which meet stated and implied needs when the software is used under specified conditions”

Defined in use cases, data flow diagrams, business rules, etc.
Functional requirement: either present or not

Quality characterisation models

Example:

ISO-9126
High-level
quality factors

Functionality

Reliability

Usability

Efficiency

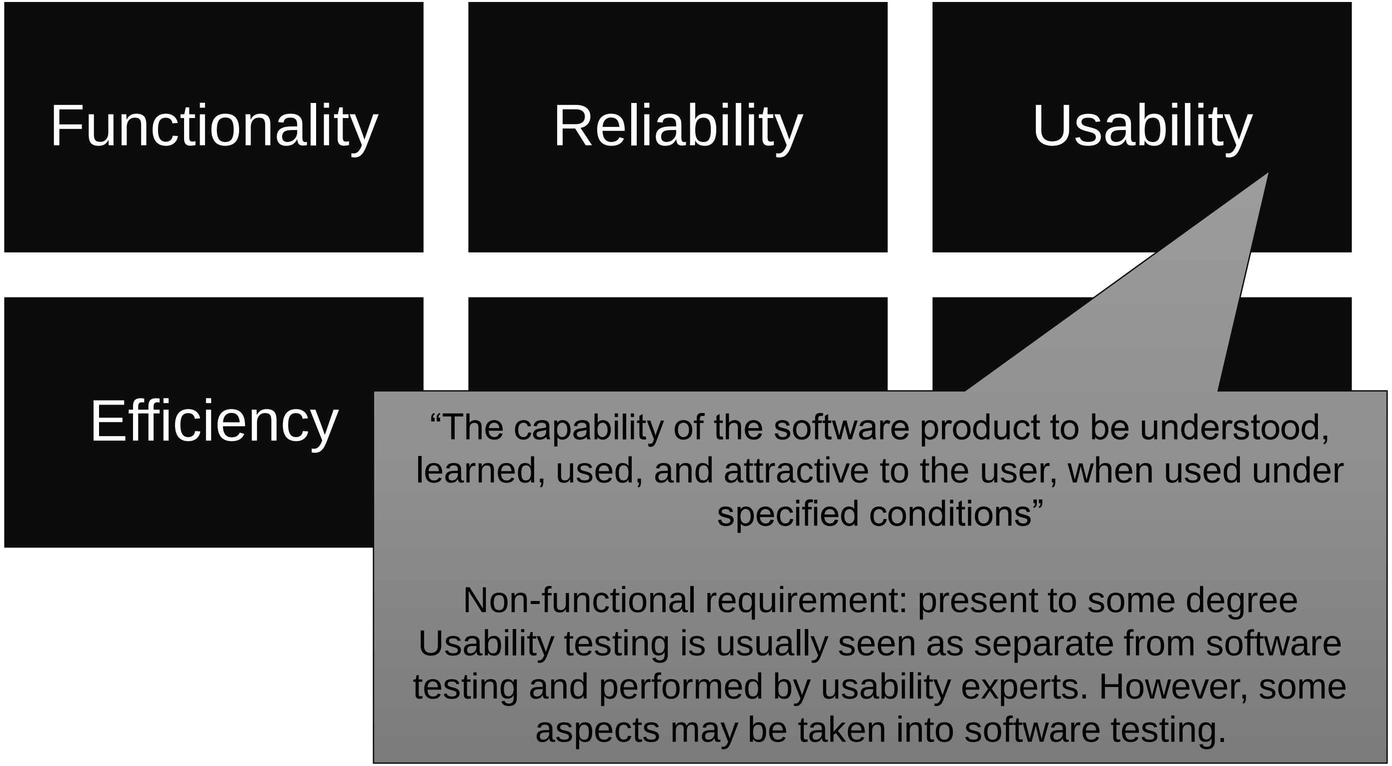
“The capability of the software product to maintain a specified level of performance when used under specified conditions”

Non-functional requirement: present to some degree

Quality characterisation models

Example:

ISO-9126
High-level
quality factors



Functionality

Reliability

Usability

Efficiency

“The capability of the software product to be understood, learned, used, and attractive to the user, when used under specified conditions”

Non-functional requirement: present to some degree
Usability testing is usually seen as separate from software testing and performed by usability experts. However, some aspects may be taken into software testing.

Aalto University
School of Science

Quality characterisation models

"… provide appropriate performance, relative to the amount of resources used, under stated conditions"

Non-functional requirement: present to some degree

"The capability of the software product to be modified" Corrections, improvements, adaptation to changes

Non-functional requirement: present to some degree

"The capability of the software product to be transferred from one environment to another"

Non-functional requirement: present to some degree

nct

ity

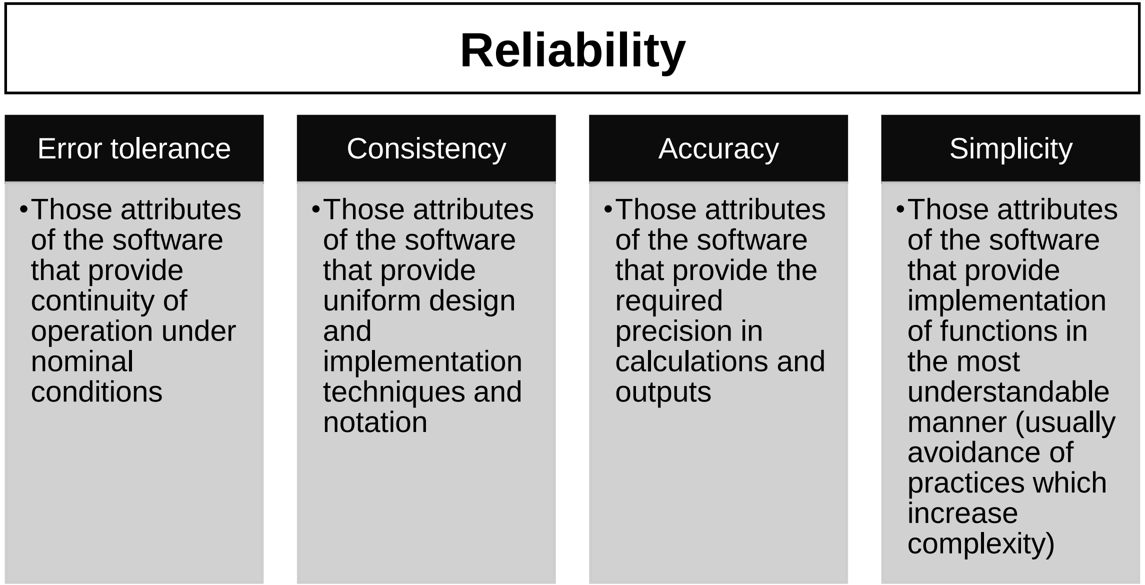
Efficiency

Maintainability

Portability

Decomposing quality criteria: Reliability

- Quality criteria need to be concrete regardless of the quality characterisation model
- Quality criteria are decomposed to a level where one or more *metrics* can be defined to measure each criteria



Reliability

Error tolerance

- Those attributes of the software that provide continuity of operation under nominal conditions

Consistency

- Those attributes of the software that provide uniform design and implementation techniques and notation

Accuracy

- Those attributes of the software that provide the required precision in calculations and outputs

Simplicity

- Those attributes of the software that provide implementation of functions in the most understandable manner (usually avoidance of practices which increase complexity)

McCall’s model

Discussion

Think of examples of quality in real software that you use

What qualities are important?

How would you define them?

How would you assess them?

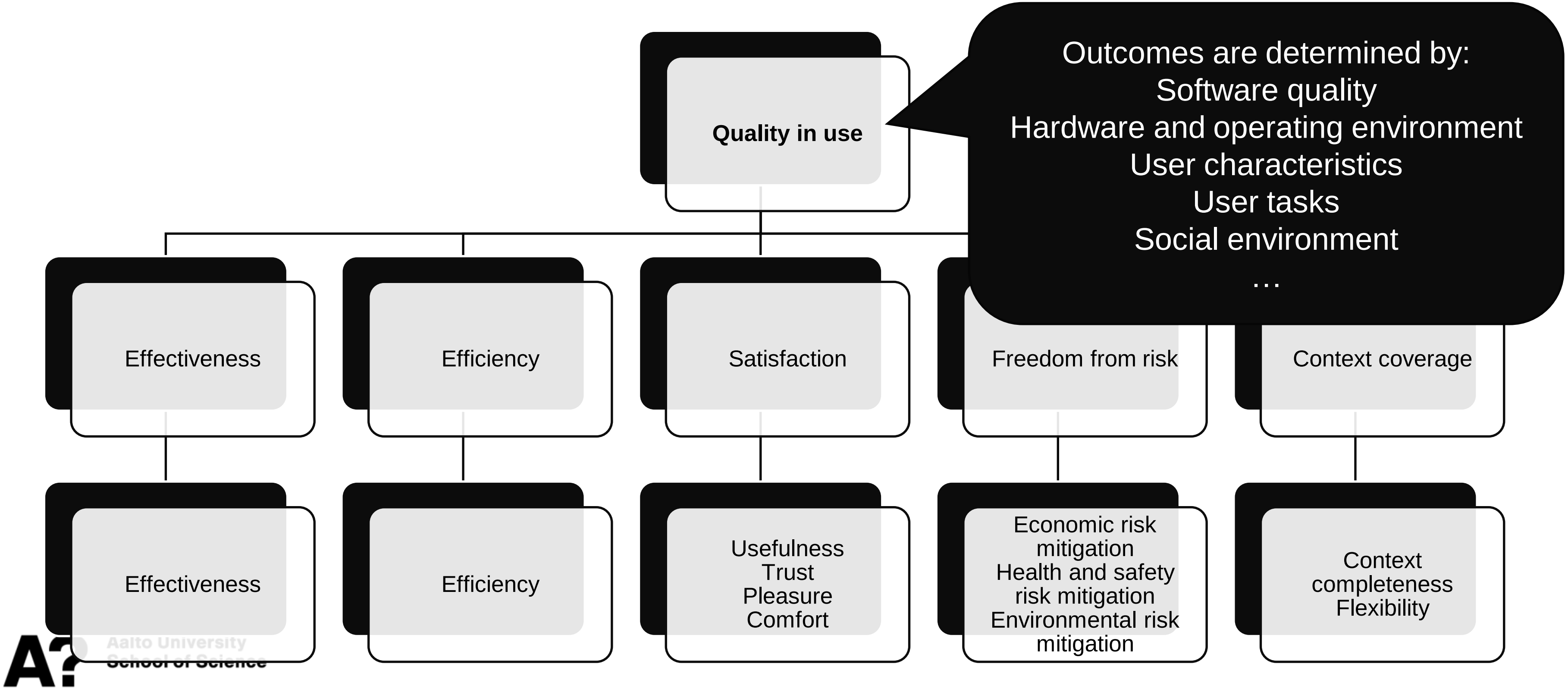
(How do you know if the software meets the quality expectations)

ISO/IEC 25010 (SQuaRE)

Quality in use

Product quality model

ISO/IEC 25010 (SQuaRE): Quality in use

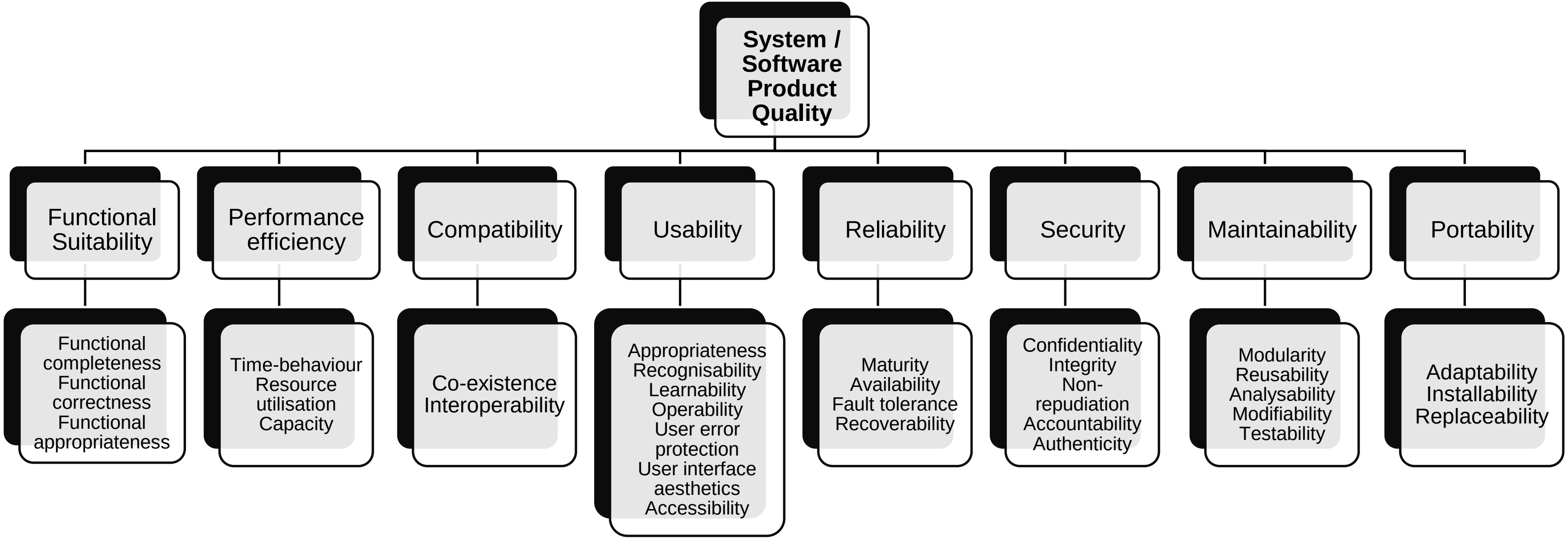


Quality in use

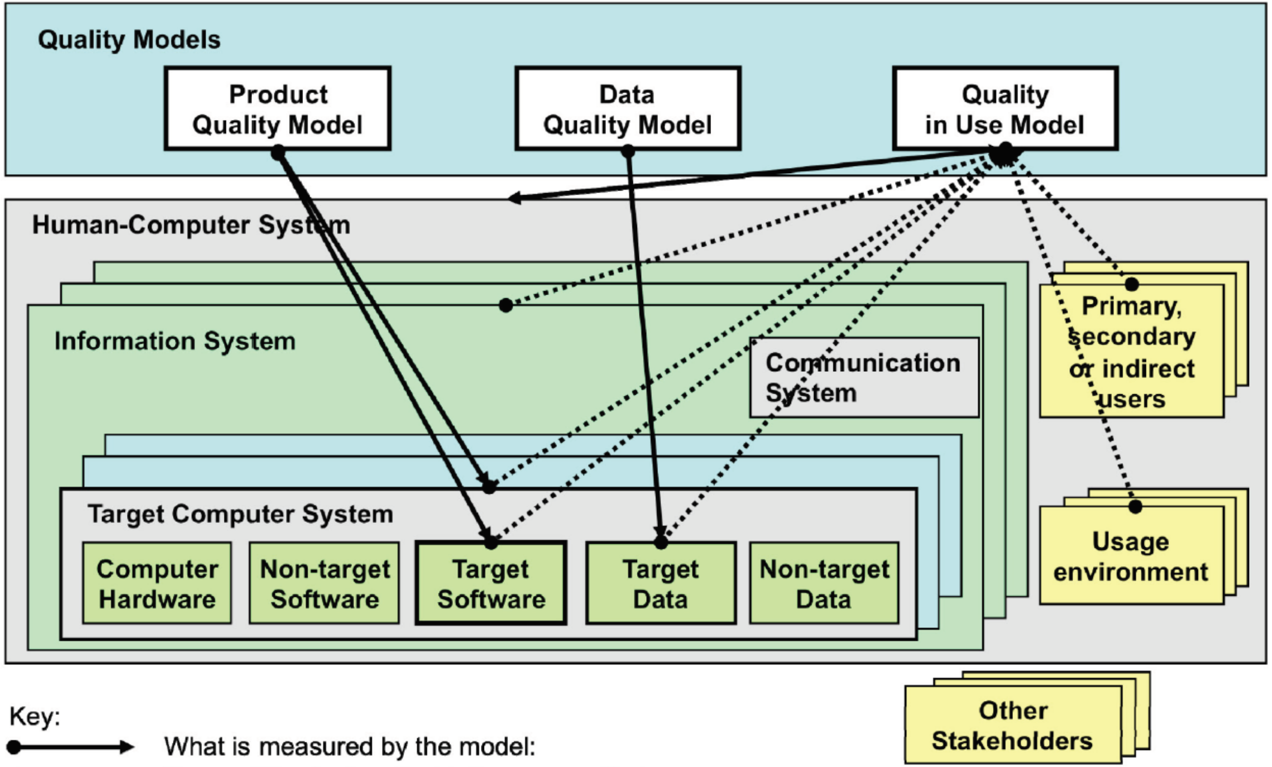
- Effectiveness
  - Effectiveness
- Efficiency
  - Efficiency
- Satisfaction
  - Usefulness
  - Trust
  - Pleasure
  - Comfort
- Freedom from risk
  - Economic risk mitigation
  - Health and safety risk mitigation
  - Environmental risk mitigation
- Context coverage
  - Context completeness
  - Flexibility

Outcomes are determined by:
Software quality
Hardware and operating environment
User characteristics
User tasks
Social environment
…

ISO/IEC 25010 (SQuaRE): Product quality model


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

ISO/IEC 25010 (SQuaRE): Quality model targets



Key:

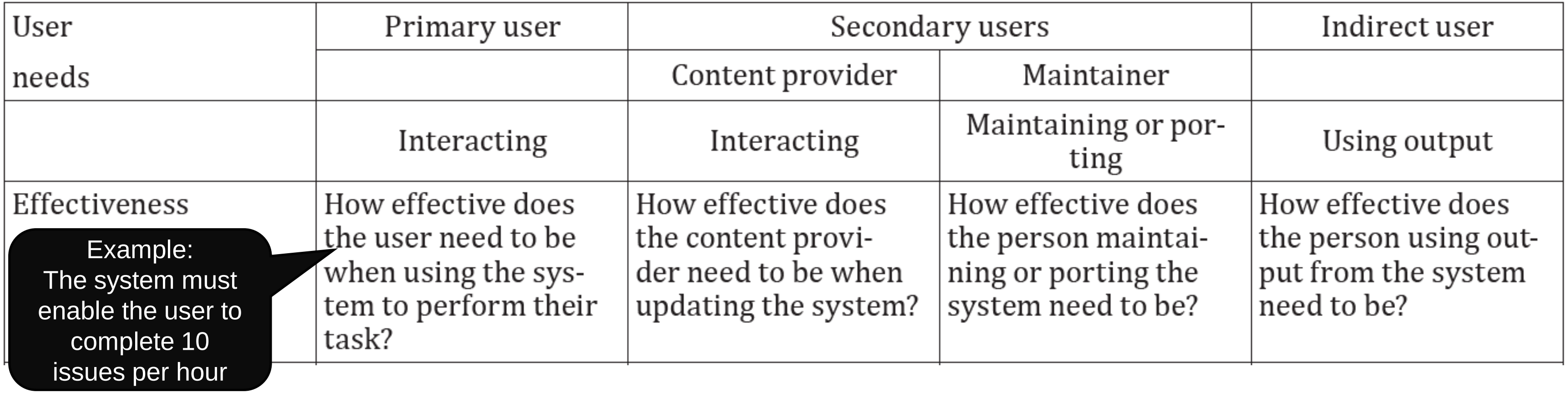
●——▶ What is measured by the model:

◀⋯⋯● Some of the factors that influence quality in use

Aalto University
School of Science

ISO/IEC 25010 (SQuaRE): Using a quality model

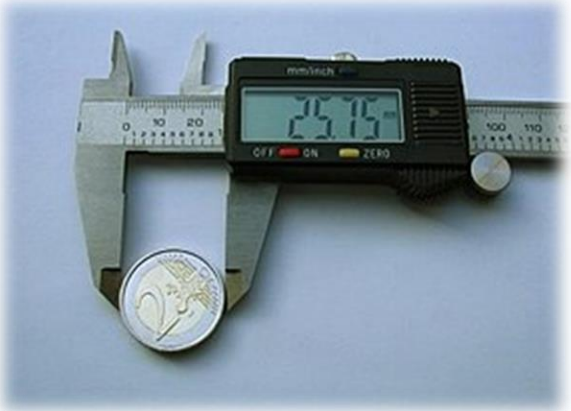
- **Stakeholder perspectives**
  - Primary user: person who interacts with the system to achieve the primary goals
  - Secondary user: provide support
  - Indirect users: person who receives output but does not interact with the system



User needs	Primary user	Secondary users		Indirect user
		Content provider	Maintainer	
	Interacting	Interacting	Maintaining or porting	Using output
Effectiveness	How effective does the user need to be when using the system to perform their task?	How effective does the content provider need to be when updating the system?	How effective does the person maintaining or porting the system need to be?	How effective does the person using output from the system need to be?

Example:
The system must enable the user to complete 10 issues per hour

Software metrics



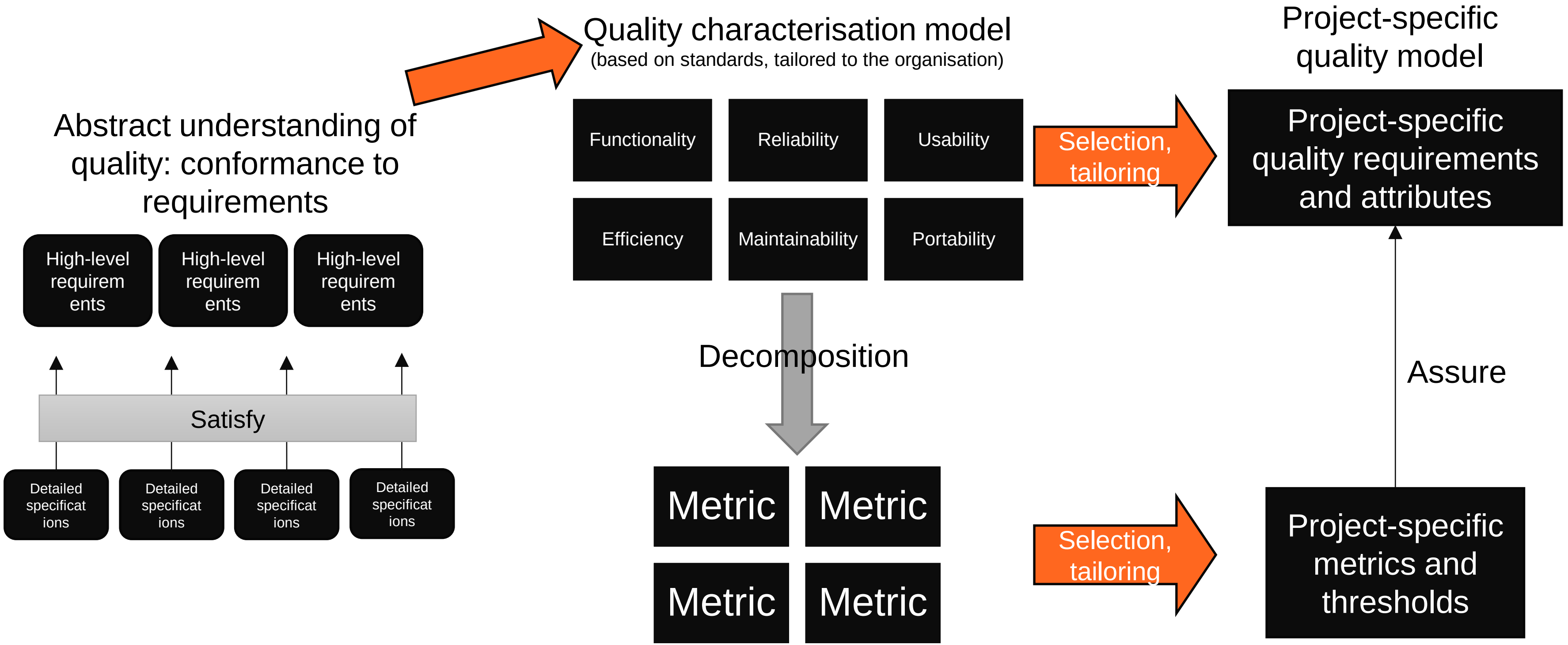
- *Metric:* A quantitative scale or method, which can be used for measurement
- *Measurement:* The process of assigning a number or category to an entity to describe an attribute of that entity
- *Internal metrics*
  - Assessing or predicting quality during production
  - Measures intermediate deliverables
  - E.g. through reviews and inspection
  - Static testing: the software is not run
- *External metrics*
  - Customer metrics
  - Measures the presence of a quality factor in the final product or component
  - Dynamic testing (or in production): the software is run

Example:
Simplicity
→ Is the design organized top down? (yes/no)
→ Are modules independent? (ratio of independent modules)

Example:
Reliability
→ Mean time between failure (MTBF) during operation

Aalto University
School of Science

From abstract to concrete



Quality characterisation model

(based on standards, tailored to the organisation)

Project-specific quality model

Abstract understanding of quality: conformance to requirements

High-level requirements

High-level requirements

High-level requirements

Satisfy

Detailed specifications

Detailed specifications

Detailed specifications

Detailed specifications

Functionality

Reliability

Usability

Efficiency

Maintainability

Portability

Selection, tailoring

Project-specific quality requirements and attributes

Decomposition

Assure

Metric

Metric

Metric

Metric

Selection, tailoring

Project-specific metrics and thresholds

Challenges for software quality:

- What qualitative model of requirements quality characteristics should be used?
- How should the requirements be broken down into measurable, quantitative attributes?

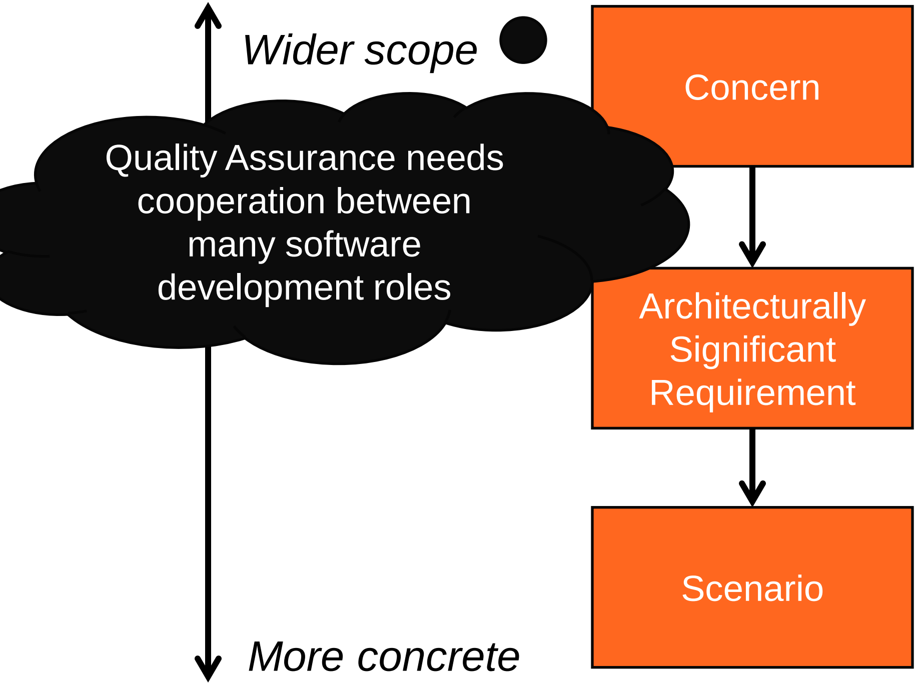
User stories

- Functional descriptions written from a user perspective
- As a <*role*>, I can/want <*capability*> so that <*receive benefit*>
- Can be based on personas (descriptions of persons that represent real-life groups of users)
- User stories do not capture every kind of quality attribute – they most often describe only functionality

The beginnings of acceptance tests

User story	Acceptance criteria
As a *content owner*, I want to *create product content* so that I can *provide information to my customers*	• Log in • Create content page • Edit content page • Save changes • Assign content page to editor for review
An an *editor*, I want to *review content before it is published* so that I can *ensure it is correct and has the right style*	• Log in • View content page • Edit content page • Add comments • Save changes • Re-assign content page to content owner

Linking Quality with Requirements and Software Architecture



Wider scope

Quality Assurance needs cooperation between many software development roles

Concern

Architecturally Significant Requirement

Scenario

More concrete

”The system should be available.”

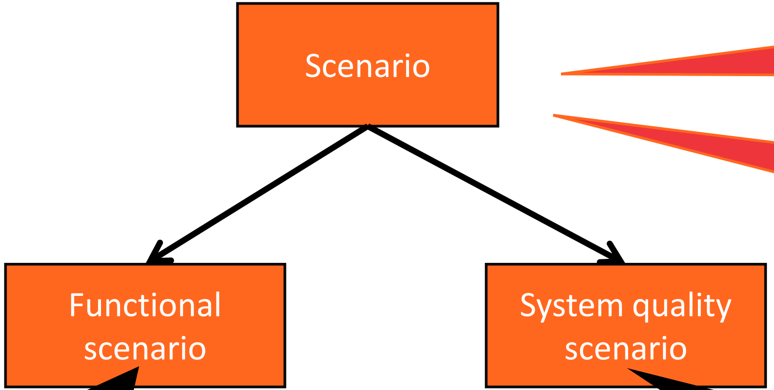
Quality Attribute: Availability (related to Reliability)

”The checkout service should have availability of 99,9%.”

”The connectivity to the payment gateway is lost. The checkout service should notice this, disable the credit card payment option within 5 s, and retry establishing the connectivity.”

The scenario can yield a quality requirement

Scenarios concretise requirements



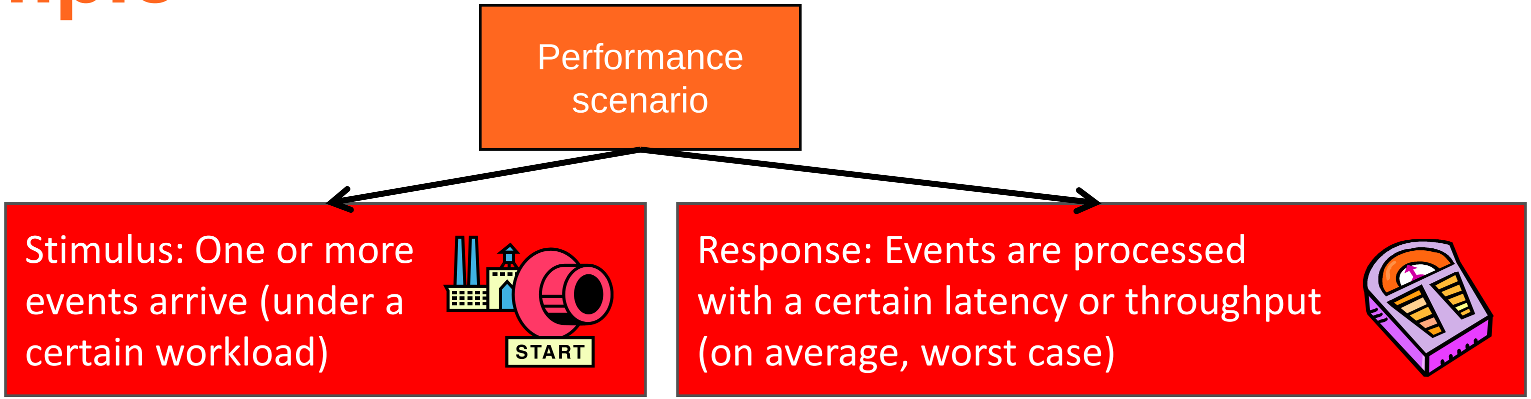
Stimulus:
What triggers the scenario?

Response: What
is the measurable outcome?

At 03:00 CET, on the first day of the month, the system calculates the order summaries per day and country and sends a report to the offices via e-mail.

The connectivity to the payment gateway is lost. The checkout service should notice this, disable the credit card payment option within 5 s, and retry establishing the connectivity.

Example



Performance scenario

Stimulus: One or more events arrive (under a certain workload)

Response: Events are processed with a certain latency or throughput (on average, worst case)

Latency: When the researcher presses *Start*, the simulation world with 100 creatures is initialized and animation begins within an average response time of one second.

Throughput: With 100 interacting creatures, the system can process, store and visualize at least 100 simulation steps per minute.

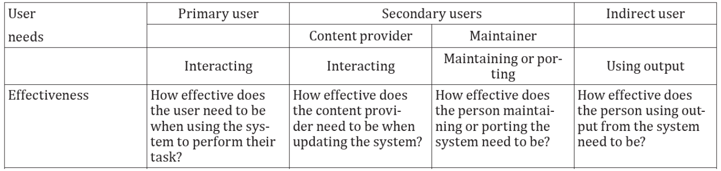
The beginnings of a performance test case

Questions?

Next: Forming groups

Group assignments

- Group assignment 1: Analysing quality using ISO/IEC 25010
- Group assignment 2: Doing testing in a group
- Group assignment 3: Final report
  - Examines a provided software application
  - Creating small quality model
  - Running a few tests
  - Analysing the results of the tests
  - Assessing the quality of the application
- Group assignment 4: Peer review



User	Primary user	Secondary users		Indirect user
needs		Content provider	Maintainer	
	Interacting	Interacting	Maintaining or porting	Using output
Effectiveness	How effective does the user need to be when using the system to perform their task?	How effective does the content provider need to be when updating the system?	How effective does the person maintaining or porting the system need to be?	How effective does the person using output from the system need to be?

How is your learning evaluated?

Component		Max points
Individual	Lecture participation* (10 sessions, 1 point each)	10
	Assignments	70
Group	Assignments	15
	Peer review	5
Total		**100**
Extra points, voluntary	Course feedback	1

- To pass: min 50% of individual points and min 25% of group points.
- Grade: 50p → 1, 61p → 2, 71p → 3, 81p → 4, 91p → 5.
- * Alternative assignment: written task based on lecture slides + materials. Inform course staff by week 2 if you will not attend lectures.

Forming groups for group work

- **Spread around the lecture hall & corridor**
- Approach people you don’t know (that well) from before and find out:
  - *Do you prefer to work online or collocated?*
  - *What are your ambitions in the course?*
- When you find matching people, start gathering more members until you have 4-6 members.
- If you want: Start by picking one friend who you would like to work with in a group
- **If you have formed a group: that’s all for today!**
- After the lecture:
  - Register your group and group name (instructions on MyCourses)
  - Agree on a way to communicate (e.g., the course chat)
  - Set up a (weekly?) meeting schedule for your group
  - Get started on the first group assigment

Week	Lecture	Topic	Assignment deadlines (Tuesdays 10:00 unless otherwise specified)
36	3.9.2024	Introduction and practicalities	
37	10.9.2024	Software quality	Individual assignments 1
38	17.9.2024	Software testing: levels, test case analysis and design	Group registration (DL: 20.9.)
39	24.9.2024	Testing techniques: Black-box testing	Individual assignments 2, Group assignment 1
40	1.10.2024	Testing techniques: Black-box testing	Individual assignments 3
41	8.10.2024	Testing techniques: Manual testing	Individual assignments 4
42	15.10.2024	(No lecture)	
43	22.10.2024	Testing techniques: White-box testing	Individual assignments 5
44	29.10.2024	Testing techniques: White-box testing	Individual assignments 6, Group assignment 2
45	5.11.2024	Guest lecture	Individual assignments 7
46	12.11.2024	Testing techniques: Static code analysis and software metrics	Individual assignments 8
47	19.11.2024	Continuous Integration and Continuous Delivery/Deployment	Individual assignments 9
48	26.11.2024	Test management	Individual assignments 10, Group assignment 3
49	3.12.2024	(No lecture)	Individual assignments 11, Group assignment 5

Subject to changes

Next steps

- Individual assignments in MyCourses
  - Software Quality Basics
  - Test environment setup
- **Deadline: before next lecture (Tuesday by 10:00)**
- Group assignments in MyCourses
  - Group registration and Group name choice – **Deadline: Friday 20.9. at 16:00**
  - Analysing quality using ISO/IEC 25010 – **Deadline: 24.9. at 10:00**
- Getting help
  - Ask in the course chat, see Communication section in MyCourses
  - Personal questions by email

Study smarter, not harder!

- Plan: when and where
- Go deep at your own pace
- Get some rest
- Practice makes perfect
- Enjoy it ☺