Software Testing and Quality Assurance

Lecture 1

Fabian Fagerholm

fabian.fagerholm@aalto.fi

Aalto University
School of Science

CS-E4960 Software Testing and Quality Assurance – Learning Outcomes

- You know and can define the **essential concepts of testing and software quality**.
- You understand the **objectives of software testing** and the **significance of testing and quality assurance as part of software engineering**.
- You know **different ways of organizing testing**, **common testing techniques** as well as other quality practices.
- You can select and apply **appropriate quality practices** in different situations and understand the **strengths and weaknesses of the practices**.
- You understand the purposes that **testing tools and test automation** can be used for and the typical challenges of test automation.

CS-E4960 Software Testing and Quality Assurance – Items from the study guide

Content

- Basics of software testing, concepts, testing techniques and reviews. Test planning, management and tools. The role of testing in quality assurance and quality assurance as part of software process.

Assessment Methods and Criteria

- Lecture attendance, individual assignments, group assignments.

Study Material

- Delivered to the students when the course begins.

Chapters from online books + PDFs (links in MyCourses).

Target group and prerequisites

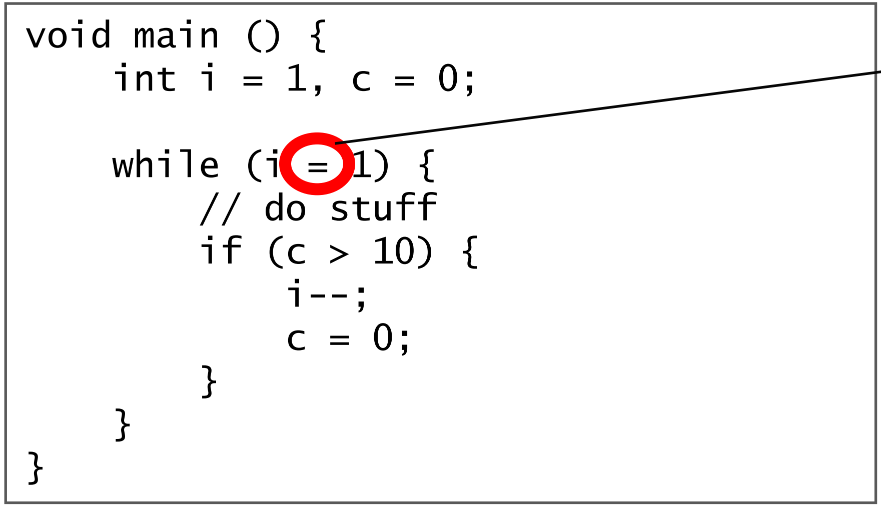
- **Target group**
- Software and Service Engineering major / minor
- Information Networks
- Computer Science
- EIT Digital / ICT Innovation
- Others?

Prerequisites

- Basics of SE (CS-C3150 Software Engineering)
- Some understanding of software modelling (CS-C3180 Software Design and Modelling, UML, or similar)
- For several assignments:
  - Some advanced skills in computer usage: installing and configuring development software, solving technical problems
  - Basic programming knowledge (Python)
  - Basic internet technologies
  - Some experience in software development

What is this course about?

Can you spot the flaw?



```
void main () {
    int i = 1, c = 0;

    while (i = 1) {
        // do stuff
        if (c > 10) {
            i--;
            c = 0;
        }
    }
}
```

In C, the comparison operator is ==. The = operator is the assignment operator.

Consequence:
The loop will never end.

Aalto University
School of Science

Can you spot the flaw?

```
int array[] = new int[5];

for (int i = 0; i <= 5; i++)
   System.out.println(array[i]);
```

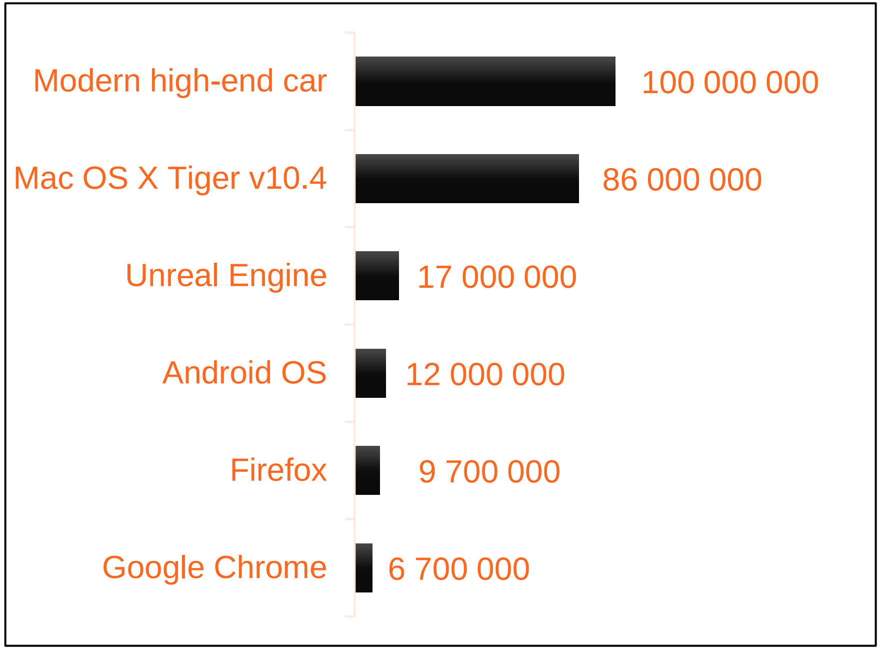
The array has 5 elements, but the counter i runs from 0 to 5 – six positions.

Consequence:
The program will crash, e.g.

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsExc
eption: Index 5 out of bounds for
length 5
```

Aalto University
School of Science

Can you spot the flaw?



Lines of code. (Indicative estimates from various sources.)

- Software systems often consist of millions of lines of code
- An end-user application typically uses a software development framework with millions of lines of code
- How do you know where the fault is once an incident has occurred?
- How can you prevent faults in the first place?

They didn’t spot the flaw…

- First NASA Space Shuttle orbital launch (1981)
  - A software modification caused timing problems between primary and backup systems → safety system stopped the launch
  - The shuttle launch was delayed by two days while software experts were looking for the cause
  - The system was extremely complex with both synchronous and asynchronous real-time modules
  - Lesson learned: quality requires significant and continuous effort because of the realities of software development
  - (Reading: Garman (1981))

They didn’t spot the flaw…

- First Ariane 5 launch (1996)
  - Reused inertial reference system (SRI) software from Ariane 4 system
  - Ariane 4 has a different velocity during liftoff
  - 64-bit floating point value conversion to 16-bit integer value → operand error
  - The SRI turned the rocket nozzles to an extreme position during liftoff → spacecraft disintegrated
  - Causes:
  - Specification and design errors in the SRI software
  - Inadequate analysis and testing of SRI software
  - Lesson learned: you must follow through on quality assurance
  - (Extra reading: ESA (1996))

Unknown Author (CC BY 3.0)

DLR/Thilo Kranz (CC-BY 3.0)

Launch video:
https://www.youtube.com/watch?v=gp_D8r-2hwk

They didn’t spot the flaw…

Nordea bank (2016)

- New reporting system for investment trades taken into use 18 February 2016
- A programming error caused the system to interpret trades as already reported
- ~1.8 million trades were not reported to the Financial Supervisory Authority (~11% of all trades for 6 months)
- The error was not detected during system testing
- Fined 400 000 €

https://www.arvopaperi.fi/uutiset/nordea-maksaa-fivan-400-000-euron-sakon-mukisematta/7a2f0833-6391-33b8-87f0-9aded3eec499 (in Finnish)

Finnish Tax Authority (2019)

- A programming error in external supplier’s mass printing / sending software while sending 27 000 tax reports
- A small portion of the reports were sent to the wrong recipient
- Sensitive information was leaked → data protection breach
- Apparently, the breach was small enough not to not warrant disciplinary action

https://yle.fi/uutiset/3-10915662 (in Finnish)

https://yle.fi/uutiset/3-10918492 (in Finnish)

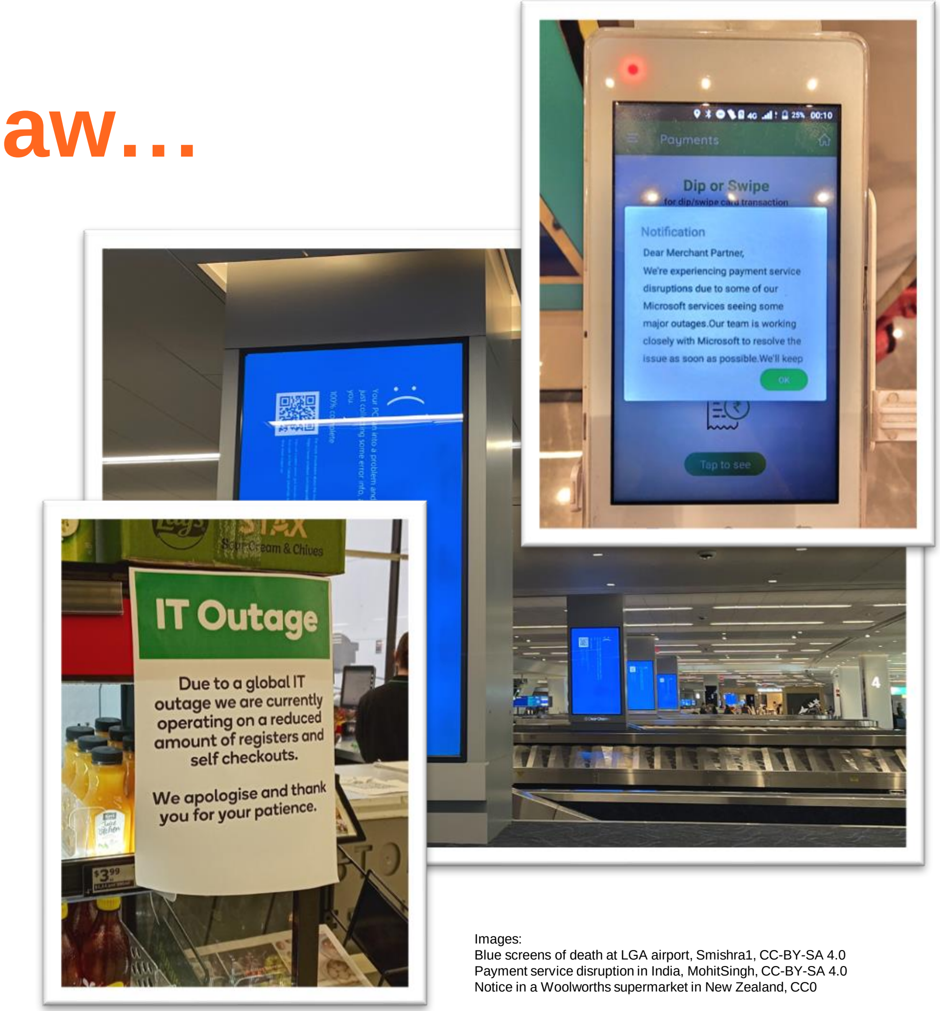
They didn’t spot the flaw…

Crowdstrike (2024)

- Crowdstrike renewed its threat definition channel file format (21 fields vs. old 20 fields)
- A faulty channel file (old format) was deployed automatically, causing Crowdstrike’s operating system kernel module to crash
- Worldwide, millions of Windows-based computers with Crowstrike installed crashed and could not properly restart
- Only the “happy path” of the software was tested
- No regression tests, only valid test data
- Distributed to all customers simultaneously, no staggered rollout
- Estimated financial damage: at least 10 billion USD (10 000 million)

Reading: Crowdstrike (2024)

Aalto University
School of Science



Notification

Dear Merchant Partner,

We're experiencing payment service disruptions due to some of our Microsoft services seeing some major outages.Our team is working closely with Microsoft to resolve the issue as soon as possible.We'll keep

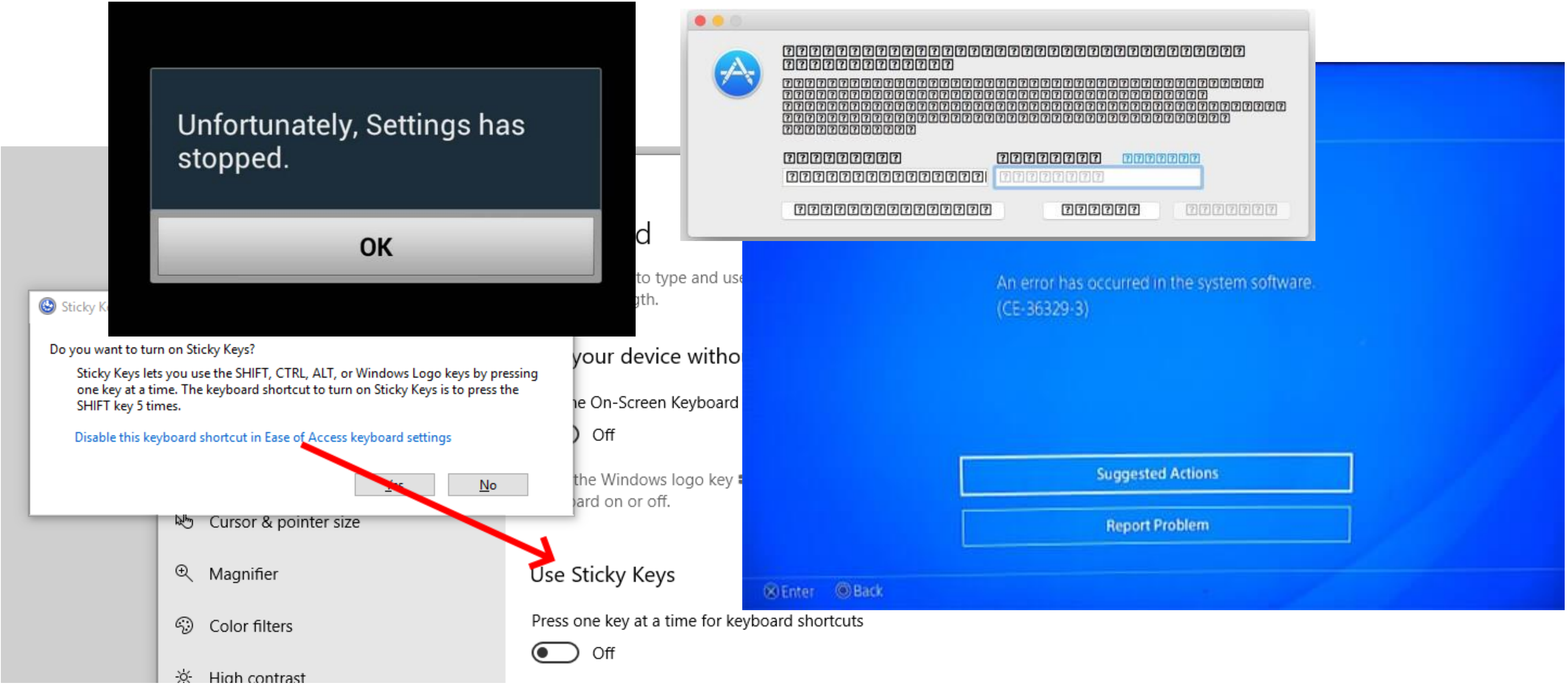
IT Outage

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.

Images:
Blue screens of death at LGA airport, Smishra1, CC-BY-SA 4.0
Payment service disruption in India, MohitSingh, CC-BY-SA 4.0
Notice in a Woolworths supermarket in New Zealand, CC0

Not all flaws are fatal, but still annoying



Unfortunately, Settings has stopped.

OK

Sticky K

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

Yes No

d

to type and use
gth.

your device witho

he On-Screen Keyboard

Off

the Windows logo key
oard on or off.

Cursor & pointer size

Magnifier

Color filters

High contrast

Use Sticky Keys

Press one key at a time for keyboard shortcuts

Off

An error has occurred in the system software.
(CE-36329-3)

Suggested Actions

Report Problem

Enter Back

Discussion

Is defect-free software possible? How?
(Or: Why not?)

(Here, we mean non-trivial, real-world software systems)

Software Testing and Quality Assurance

- Is defect-free software possible? How? (Or: Why not?)
- Eliminating all software defects is usually not feasible
  - The effort (cost and time) required is too high
  - If the system is complex enough, the conditions under which failures occur are not possible to anticipate fully
  - Changed requirements: the software must change before all defects can be found
  - The defects themselves are complex and can occur in many development stages (requirements, specifications, code, etc.)

→ Focus on **quality** and eliminating important defects

- *This course aims to increase your knowledge of software quality and the role of testing in assuring quality*

Preliminaries: Important concepts

Not all terms used on this course are standardized. There can be other definitions. The important thing is to understand the concept.

Important concepts: *Quality*

What is quality?

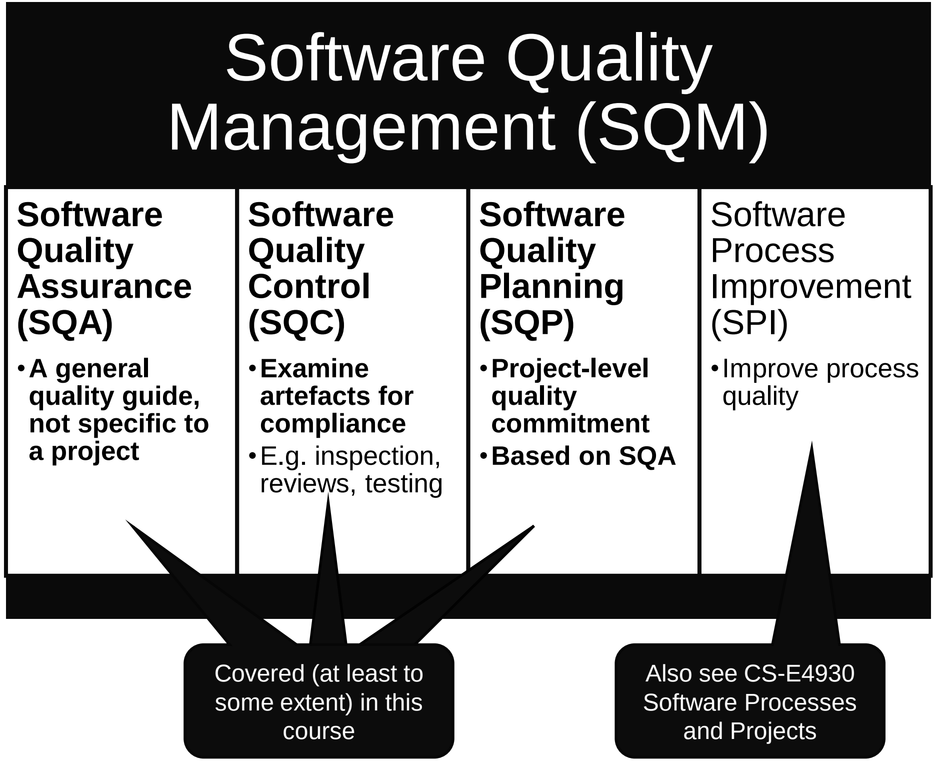
What would a useful definition look like?

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*

Important concepts: *Quality Assurance*

- Providing confidence that quality requirements will be fulfilled
- A quality guide with
  - *standards*, *regulations*, *best practices* and software *tools* to
  - *produce, verify, evaluate and confirm work products* during the software development life cycle.
- Internal purposes: provide confidence for the management
- External purposes: provide confidence to the customers and other external stakeholders.



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

Covered (at least to some extent) in this course

Also see CS-E4930 Software Processes and Projects

Important concepts: Errors to Incidents

Error (mistake)

A human error while making a software artefact (requirement, specification code, …).

Fault (defect)

An incorrectness in a software artefact resulting from an error (also: a bug).

Failure

Manifested inability of system to perform according to specification.

Incident

The symptom associated with a failure. Alerts the user to the occurrence of a failure.

Error: Typing = instead of ==

Fault: An infinite loop in the program

Failure: System freezes on Tuesday at 08:00

Incident: The user notices that the system does not respond on Tuesday at 08:15

Important concepts: Quality control concepts

Test	The act of exercising the system with test cases
Test case	A check of assumptions related to system behaviour
Test suite	A collection of test scripts or test cases
Test script	Instructions for executing a test case
Test log	A record of test execution
Test report	A document describing the conduct and results of testing
Test plan	A document defining: scope of testing, types of testing to perform, test environment, required hardware and software, estimation of effort and resources, risk management, deliverables, key test milestones, and schedule

Discussion

What is your favourite software incident?

What kind of failure is it an example of?

(i.e. in what way did it fail?)

What kind of error led to it?

Could it have been prevented? How?

⚠ **Error** (mistake)

🐞 **Fault** (defect)

× **Failure**

🎆 **Incident**

Practicalities

Learning on this course

Lectures

- Once a week on Tuesdays at 10-12
- Attendance points
- In case of illness, get in touch

Individual assignments

- Individual learning tasks (no cooperation with other students)
- Become available the same day as the lecture
- In MyCourses
- Deadline just before the lecture on the following Tuesday (some tasks have more time)
- Point deduction for late assignments: 25% / day (rounded to nearest 0,5 p)

Group assignments

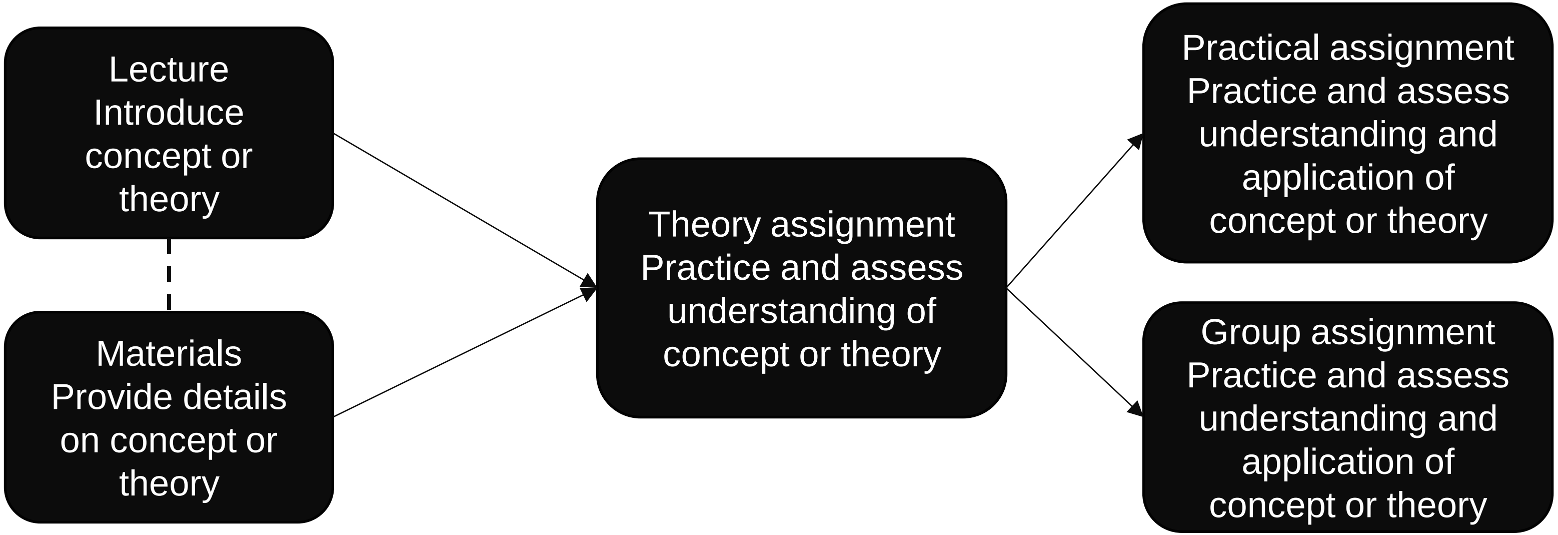
- Cooperation encouraged!
- Support your learning
- Try testing techniques that cannot be done individually
- Some deliverables build up to final assignment
- Details in Lecture 2

Tuesday
Assignment opens

→

Tuesday
+1 week
Assignment deadline

Learning on this course



Lecture
Introduce concept or theory

Materials
Provide details on concept or theory

Theory assignment
Practice and assess understanding of concept or theory

Practical assignment
Practice and assess understanding and application of concept or theory

Group assignment
Practice and assess understanding and application of concept or theory

Aalto University
School of Science

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

See next slide

- 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.

Extra points

- Course feedback, 1p
  - Provide feedback at the end of the course
  - Feedback is anonymous: the system records who has answered separately from the answers
  - Very important – you benefit from past years' feedback!
- Extra points count towards your total points but are not counted as individual or group points

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	Individual assignments 2, Group registration (DL: 20.9.)
39	24.9.2024	Testing techniques: Black-box testing	Individual assignments 3
40	1.10.2024	Testing techniques: Black-box testing	Individual assignments 4, Group assignment 1
41	8.10.2024	Testing techniques: Manual testing	Individual assignments 4, Group assignment 2
42	15.10.2024	(No lecture)	
43	22.10.2024	Testing techniques: White-box testing	
44	29.10.2024	Testing techniques: White-box testing	Individual assignments 5
45	5.11.2024	Testing techniques: Static code analysis and software metrics	Individual assignments 6, Group assignment 3
46	12.11.2024	Continuous Integration and Continuous Delivery/Deployment	Individual assignments 7, Group assignment 4
47	19.11.2024	Guest lecture	Individual assignments 8
48	26.11.2024	Test management	Individual assignments 9
49	3.12.2024	(No lecture)	Individual assignments 10, Group assignment 5

Subject to changes

New assignments

- Optional (no points) reading and discussion
  - Read: Garman 1981, ESA 1996, Crowdstrike 2024
  - Use the course chat to discuss (#famous incidents)
  - What happened in these cases and why?
  - What can we learn about software quality and testing from past incidents?
- **Deadline: before next lecture (Tuesday by 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 ☺

Thank you!
Questions?

Aalto University
School of Science