Software Engineering Reviews

TDDC90

autumn 2023

Kristian Sandahl
Department of Computer and Information Science
Linköping University, Sweden
kristian.sandahl@ida.liu.se

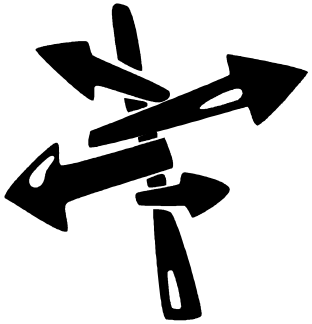
Linköpings universitet

Agenda - Theory

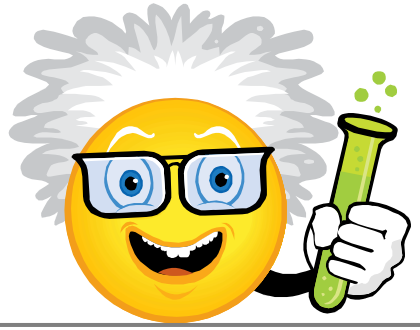
kristian.sandahl
@liu.se

Part I
Inspections

Part II
Other reviews



Part II
Variants and research



Part I
Inspections

Part II
Other reviews

Part III
Variants and research

kristian.sandahl
@liu.se

Part I
Inspections

Systematic inspections

kristian.sandahl
@liu.se

The best way of finding many defects in code and other documents

- Experimentally grounded in replicated studies

Goals:

- Find defects (anomalies)
- Training
- Communications
- Hostage taking

Development over the years

kristian.sandahl
@liu.se

- Fagan publishes results from code and design inspections 1976 in IBM systems journal
- Basili and Selby show the advantage of inspections compared to testing in a tech-report 1985.
- Graham and Gilb publish the book Software inspections 1993. This describes the standard process of today.
- Presentation of the Porter-Votta experiment in Sorrento 1994 starts a boom for replications.
- Sauer et al compare experimental data with behavioural research in a tech-report 1996
- IEEE std 1028 updated 2008

Roles

kristian.sandahl
@liu.se

- Author
- Moderator (aka Inspection leader)
- Reader (if not handled by the Moderator)
- Inspector
- Scribe (aka Recorder)

Process

kristian.sandahl
@liu.se

- Initial:
  - Check criteria
  - Plan
  - Overview
- Individual:
  - Preparation, or
  - Detection
- Group:
  - Detection, or
  - Collection
  - Inspection record
  - Data collection
- Exit:
  - Change
  - Follow-up
  - Document & data handling

Inspection record

kristian.sandahl
@liu.se

- Identification
- Location
- Description
- Decision for entire document:
  - Pass with changes
  - Reinspect

Data collection

kristian.sandahl
@liu.se

- Number of defects
- Classes of defects
- Severity
- Number of inspectors
- Number of hours individually and in meeting
- Defects per inspector
- Defect detection ratio:
  - Time
  - Total defects

Our inspection record

kristian.sandahl
@liu.se

Id	Loc.	Description	Class.
1			
2			
3			
4			
5			
6			
7			
8			

kristian.sandahl
@liu.se

Practical investigation

- 214 code inspections from 4 projects at Ericcson
- Median number of defects = 8
- 90 percentile = 30
- Majority values:
  - up to 3.5 h preparation per document
  - up to 3 h inspection time
  - up to 4000 lines of code
  - 2 to 6 people involved

Inspection rate (IEEE Std 1028-2008)

Requirements or Architecture (2-3 pages per hour)

Source code (100-200 lines per hour)

Regression wrt defect detection ratio

kristian.sandahl
@liu.se

- Preparation time per code line typically 0.005 hours per line (12 minutes per page)
- Size of document have negative effect on DFR, max recommendation 5000 lines
- A certain project is better than two of the others
- 4 inspectors seems best (not significant)
- *Analysis performed by Henrik Berg, LiTH-MAT-Ex-1999-08*

kristian.sandahl
@liu.se

Part II
Other reviews

Other reviews

kristian.sandahl
@liu.se

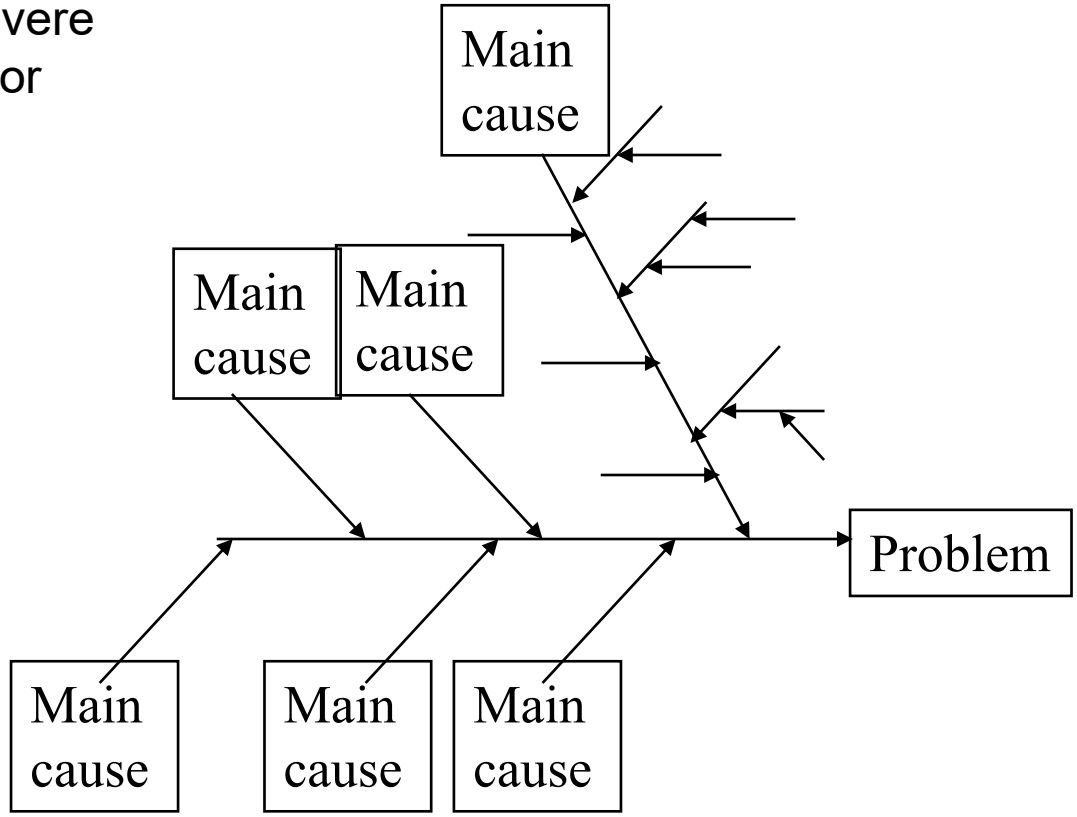
- Management review – check progress
- Technical review – evaluate conformance
- Walk-through – improve product, training
- Audit – 3rd party, independent evaluation

- (Peer) Review
- Buddy-check
- Desk check

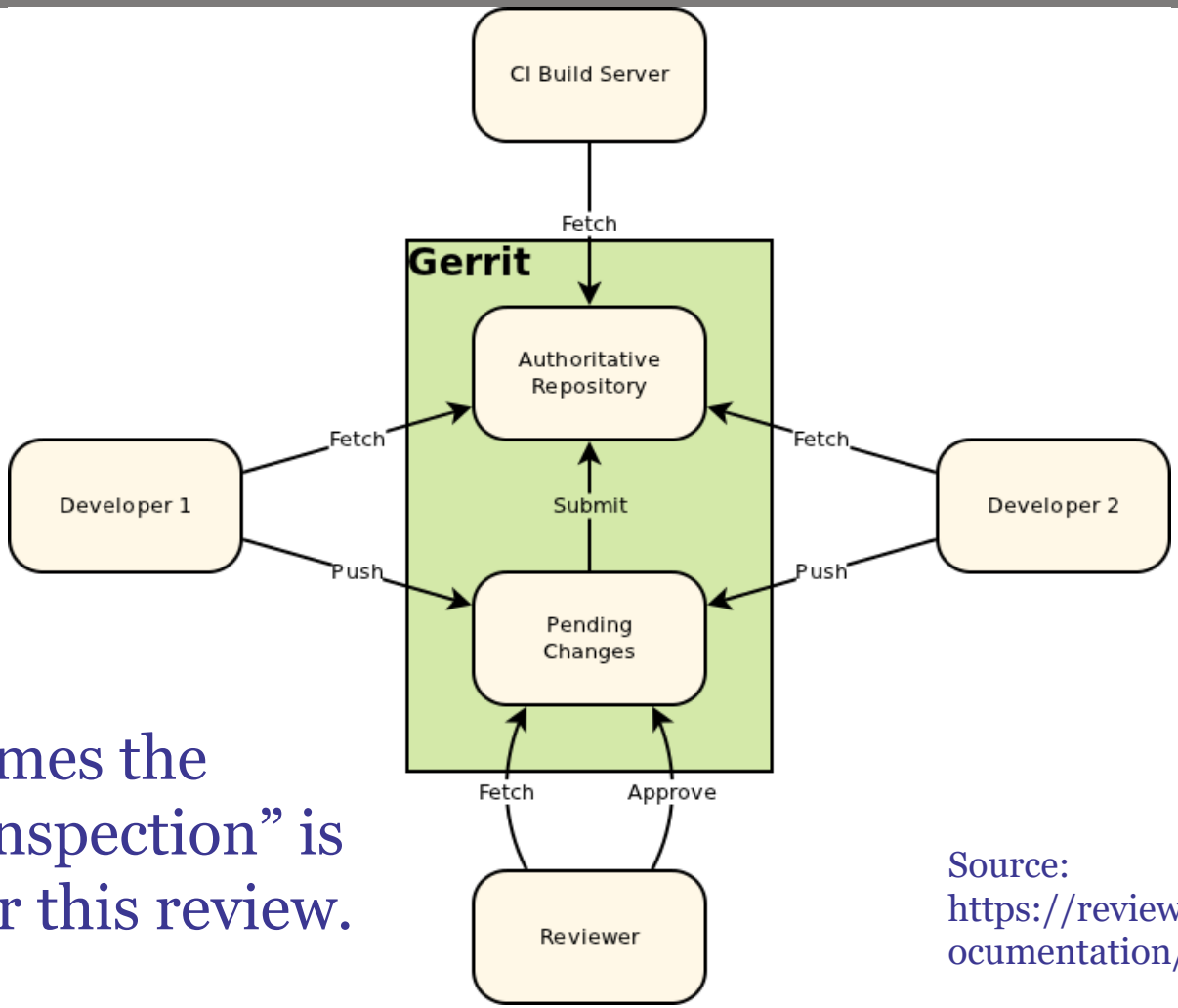
kristian.sandahl
@liu.se

Root-cause analysis

- Performed regularly for severe defects, frequent defects, or random defects
- Popular mind map:
  The Ishikawa diagram
- Parameters:
  - Defect category
  - Visible consequences
  - Did-detect
  - Introduced
  - Should-detect
  - Reason



Tool-based code review in Gerrit



Sometimes the term ”inspection” is used for this review.

Source: https://review.openstack.org/Documentation/intro-quick.html

kristian.sandahl
@liu.se

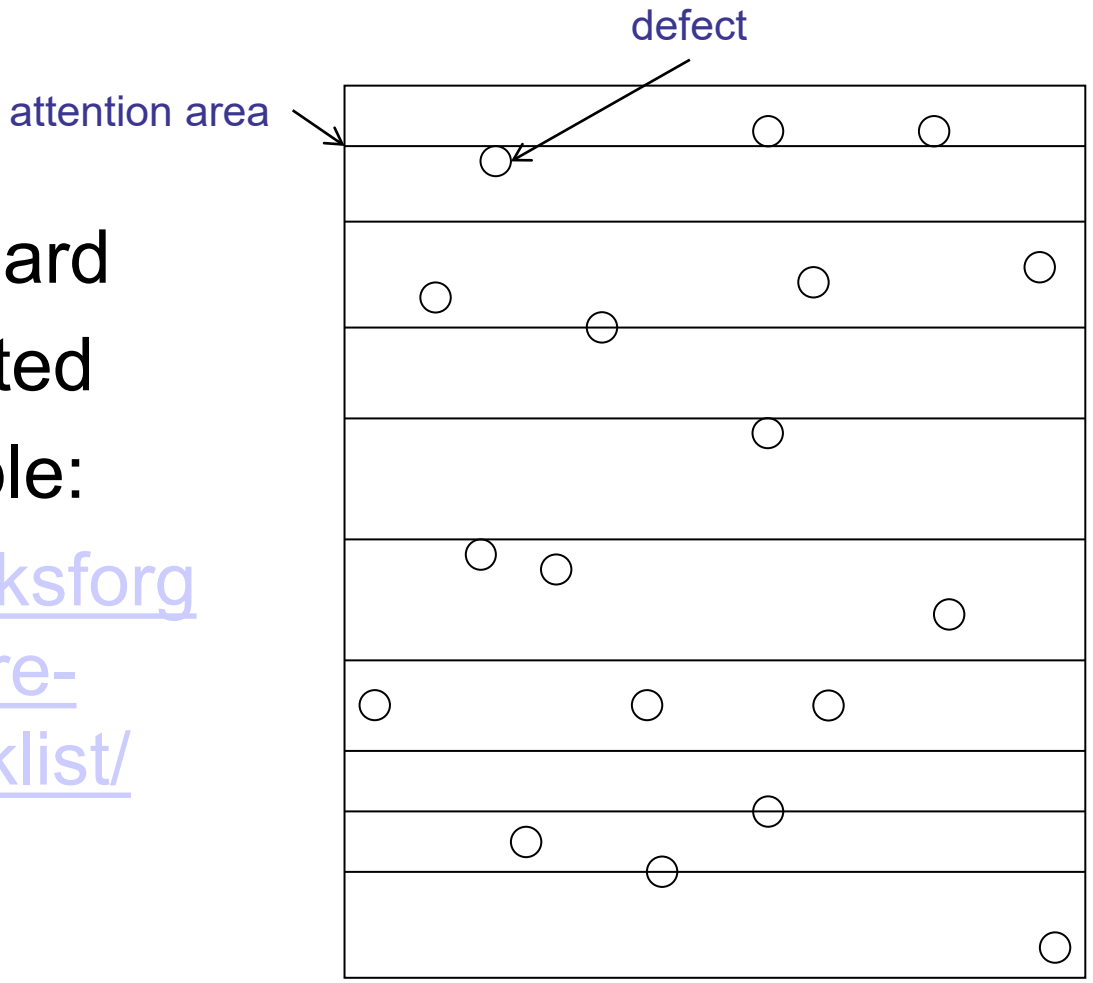
Part II
Variants and research

kristian.sandahl
@liu.se

Reading techniques - checklist

- Checklist
- Industry standard
- Shall be updated
- Simple example:

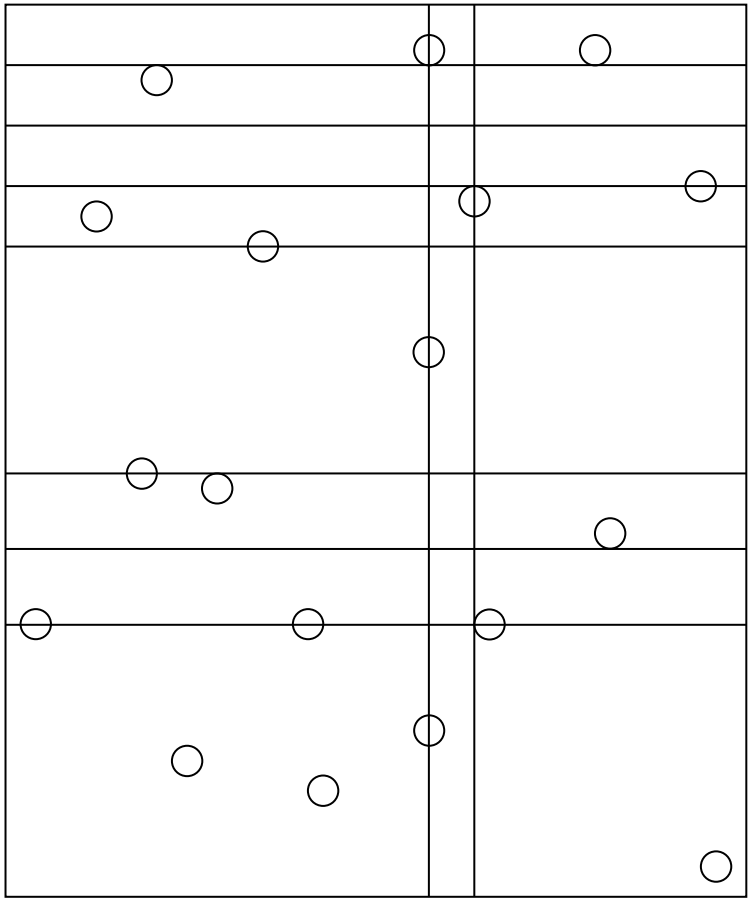
https://www.geeksforgeeks.org/software-inspection-checklist/



kristian.sandahl
@liu.se

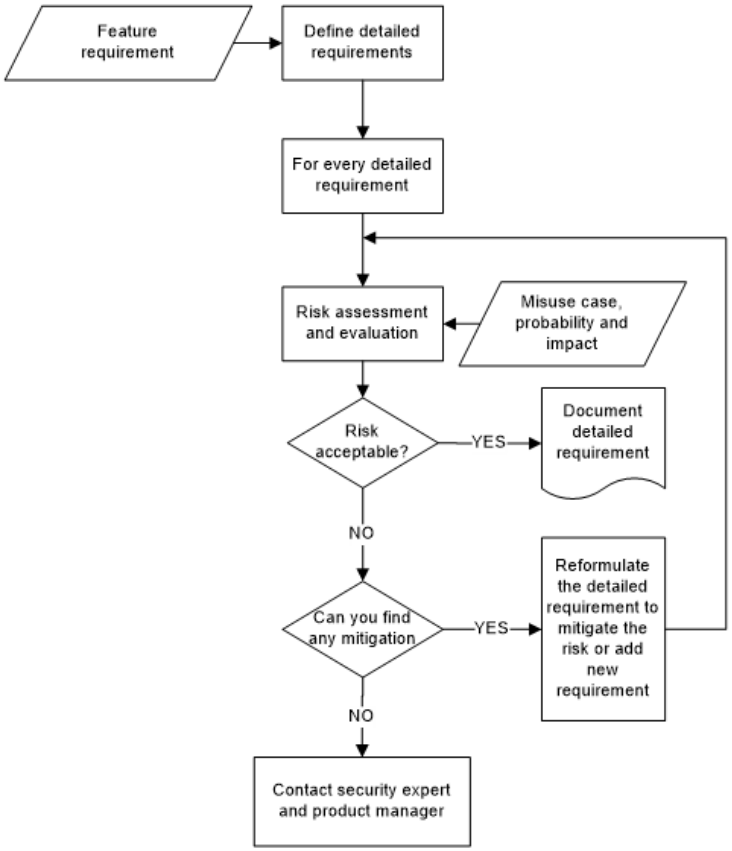
Reading techniques - scenario

- Scenario
- A checklist splitted to different responsibilities
- 30% higher DFR ?



The SRA approach scenario example

- A light-weight security risk assessment method (SRA) to be applied by non-security experts in requirements engineering
  - For every function-level/detailed requirement, perform a risk assessment by answering following questions:
    - What is the asset? What shall be protected?
    - Who has access to asset and how?
    - Can the actor/user, identified above, misuse the asset?
    - What is the probability over certain period and what is the impact of harm?



SRA example

Context: Automated operation and maintenance of handover functions when neighbor nodes provide services jointly.

R2: The node shall collect and log Automatic Neighbor Relationship (ANR) measurement results from the User Equipment (UE) selected for reporting.

SRA example

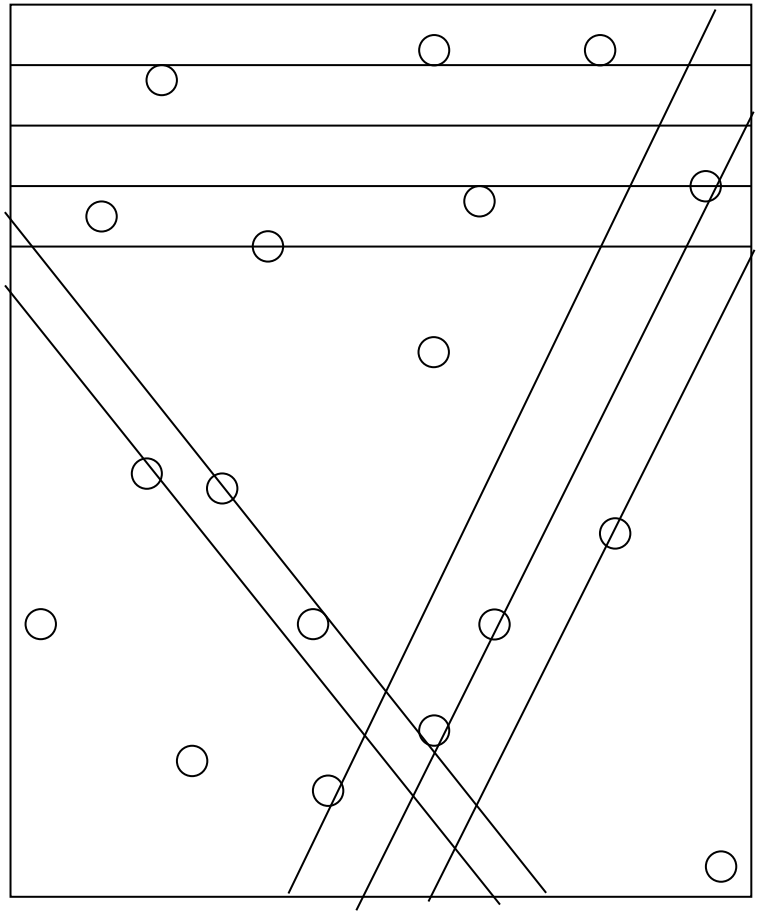
R2: The node shall collect and log Automatic Neighbor Relationship (ANR) measurement results from the User Equipment (UE) selected for reporting.

Asset	Access	Misuse	Probability/ Impact	Risk level
ANR measurement data	End-user of UE	Malicious actor can modify measurement reports	Possible/Serious	Medium

kristian.sandahl
@liu.se

Reading techniques – perspective-based

- Different inspectors repre-sent different roles
- Real or played roles
- 30% higher DFR ?



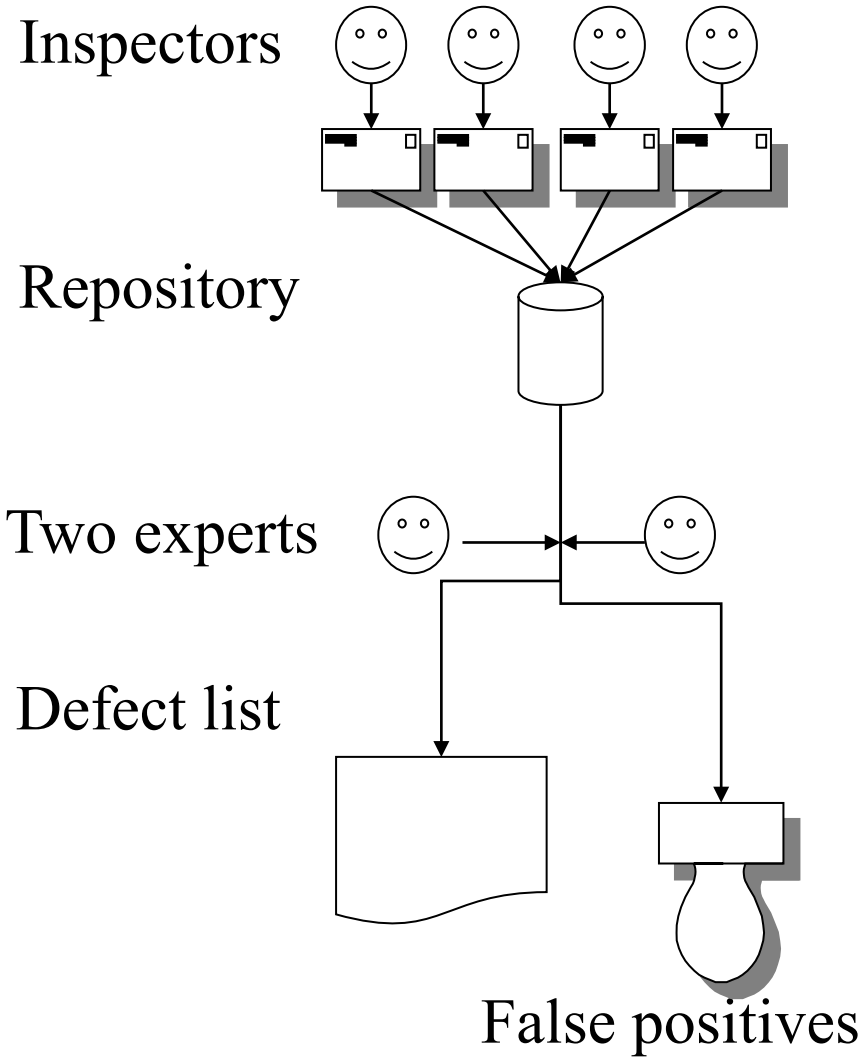
Cost of quality

kristian.sandahl
@liu.se

- Person-hours
- Calender time
- Good reading techniques
- Good data recording

”Optimal” method

kristian.sandahl
@liu.se



Summary - What have we learned today?

kristian.sandahl
@liu.se

- Inspections rule!
- Inspections are expensive