Quality Monitoring

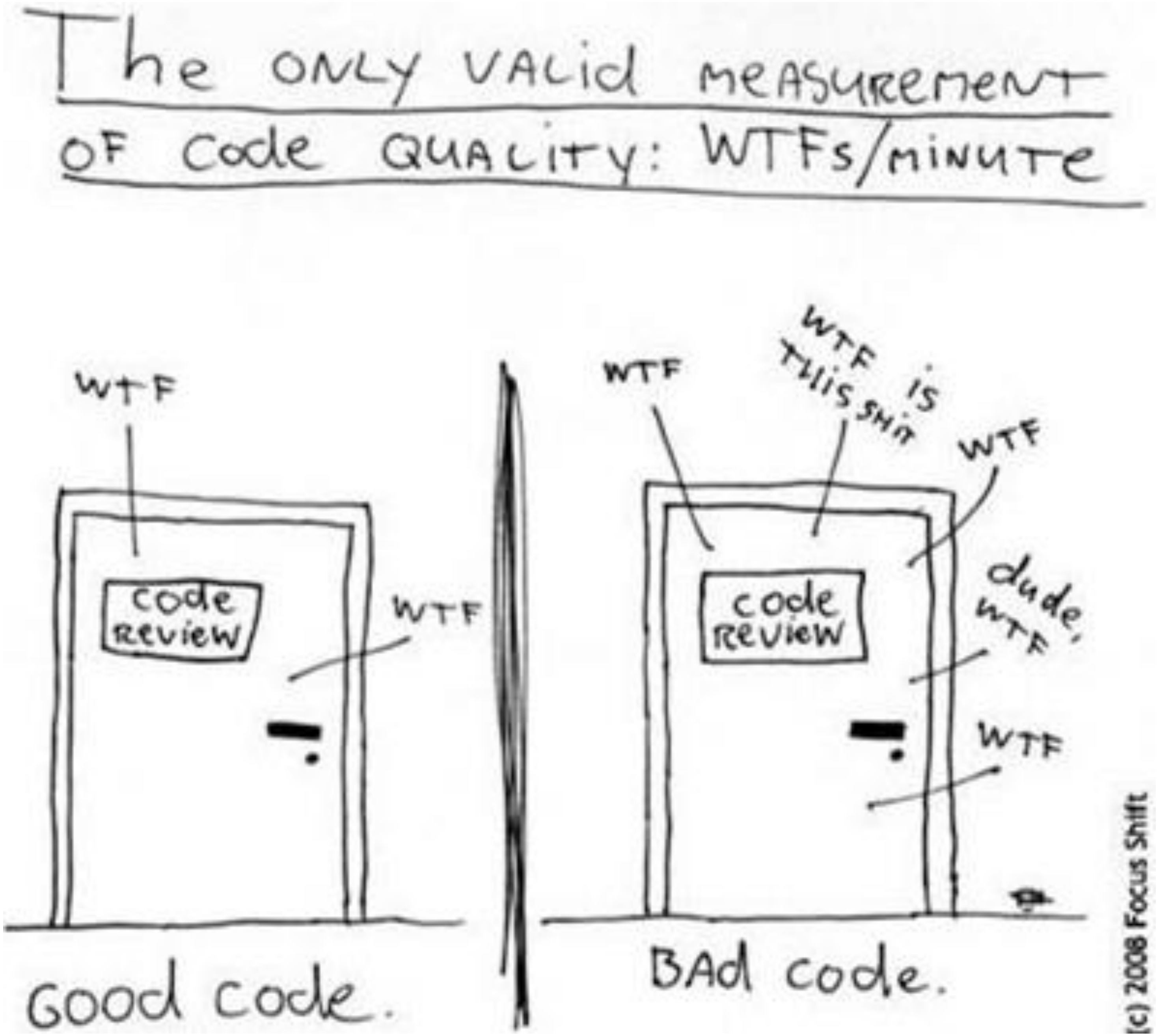
Kamil Król

Quality?

“the standard of something as measured against other things of a similar kind; the degree of excellence of something”

Monitor? Monitoring?

"observe and check the progress or quality of (something) over a period of time; keep under systematic review"


The only valid measurement of code quality: WTFs/minute
WTF
code review
WTF
Good code.
WTF
WTF is this shit
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift

How do we check the quality?

- **Manual**
  - Manual testing
  - Conventions within the team
  - Definition of done
  - Pair programming
  - Code reviews
- **Automatized**
  - Good coverage of good tests
  - Static code analysis

Manual testing

- Tests which are carried manually by a person or the testing team

👍

- Easy to define

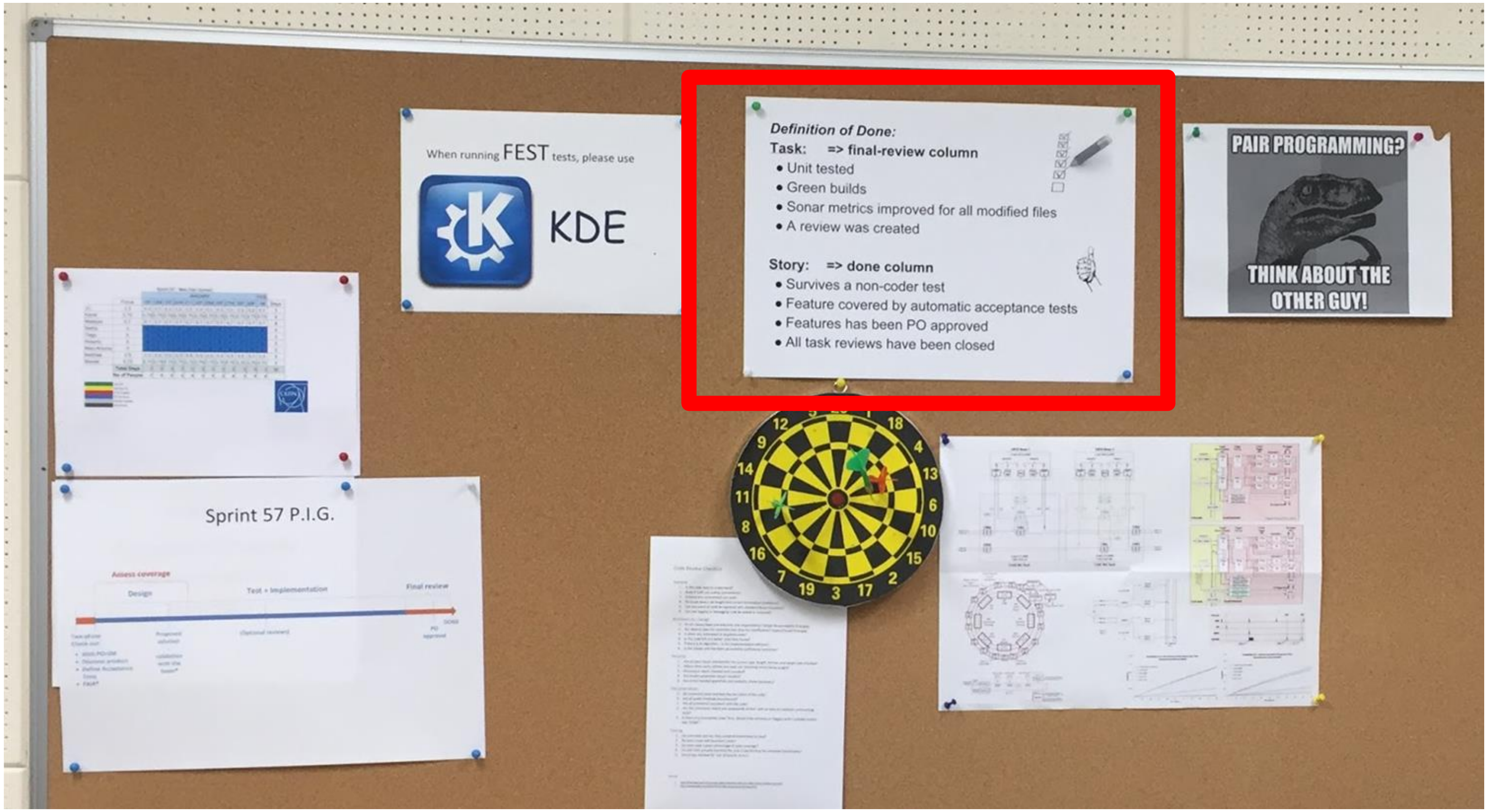
👎

- Expensive to run on daily basis
- Prone to human errors
- Boring ...

Code conventions

- Common way of producing the code
- Shared within the team / within the organisation
- Should be as close as possible to industry standards

Definition of done



Definition of Done:

Task: => final-review column

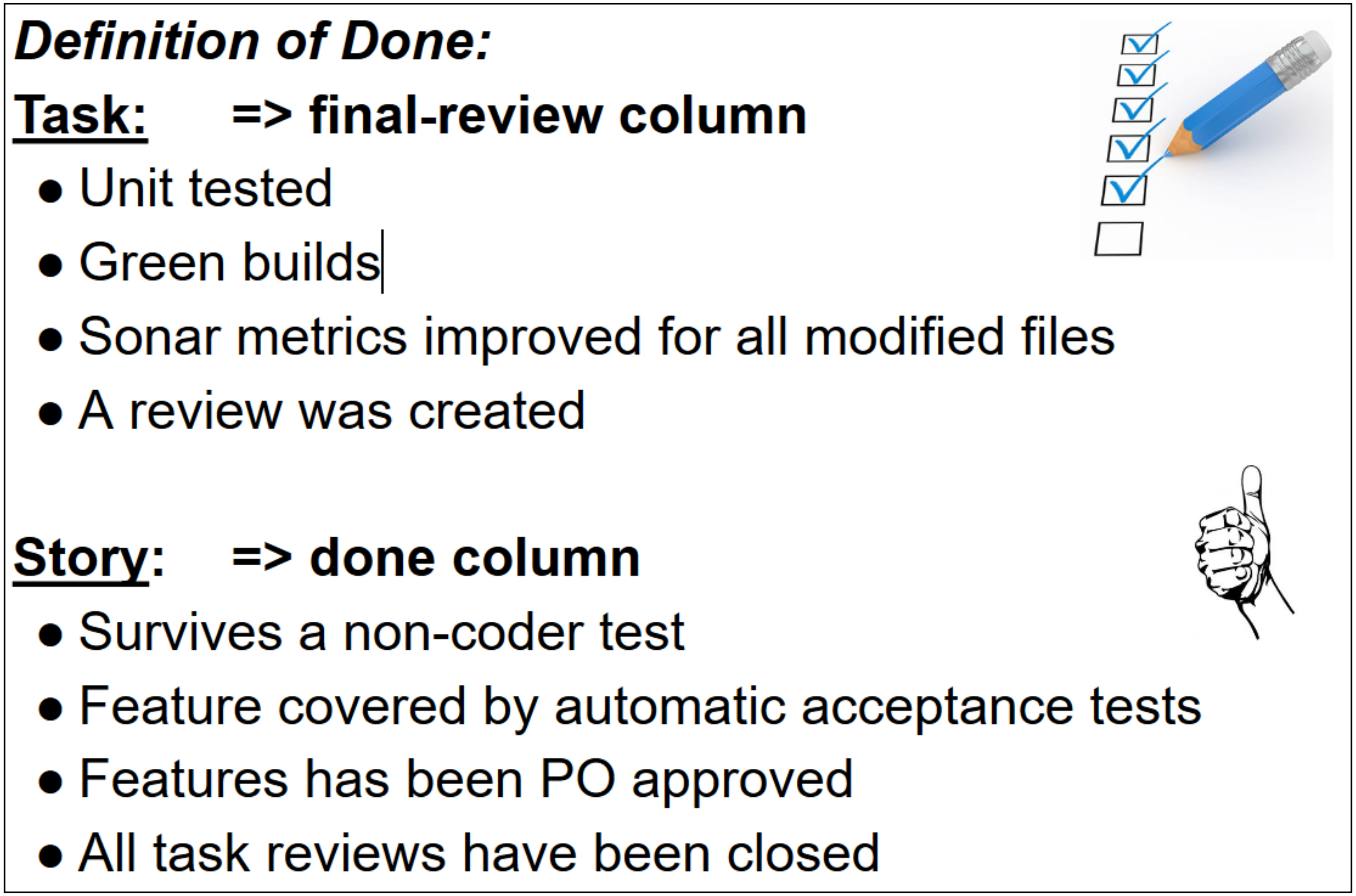
- Unit tested
- Green builds
- Sonar metrics improved for all modified files
- A review was created

Story: => done column

- Survives a non-coder test
- Feature covered by automatic acceptance tests
- Features has been PO approved
- All task reviews have been closed

List of conditions which need to be completed before the functionality is considered as DONE.

Definition of done



Definition of Done:

Task: **=> final-review column**

- Unit tested
- Green builds
- Sonar metrics improved for all modified files
- A review was created

Story: **=> done column**

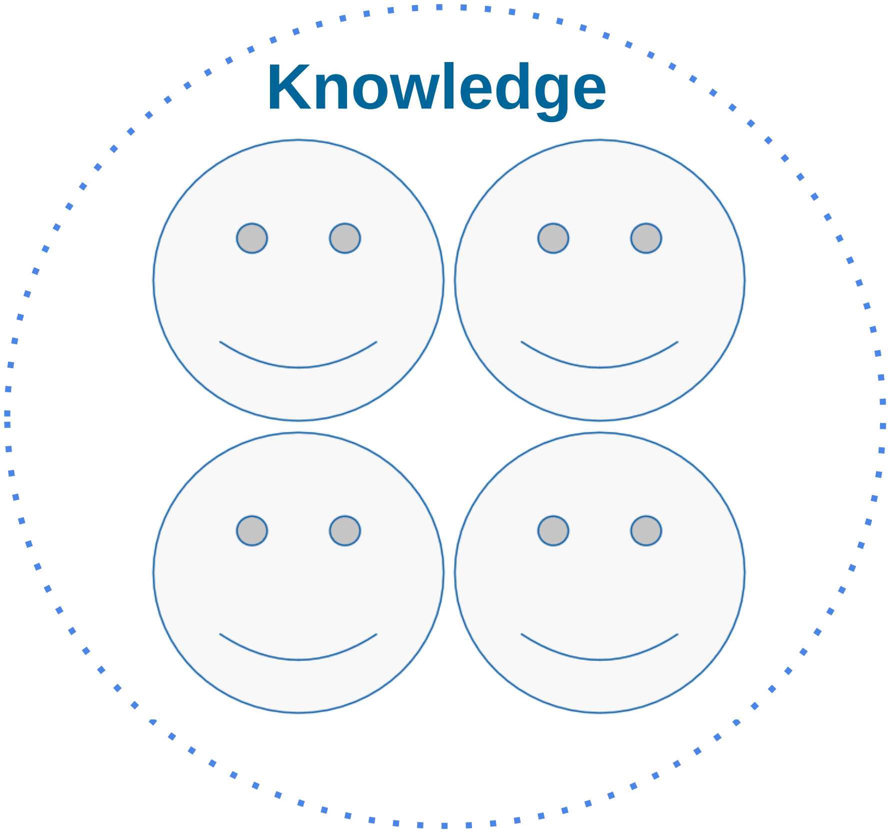
- Survives a non-coder test
- Feature covered by automatic acceptance tests
- Features has been PO approved
- All task reviews have been closed

Pair programming

- 2 people - 2 different opinions
- Driver + Navigator
- Often change roles

- Immediate feedback
- On-spot bug fixes
- **Synergy effect**

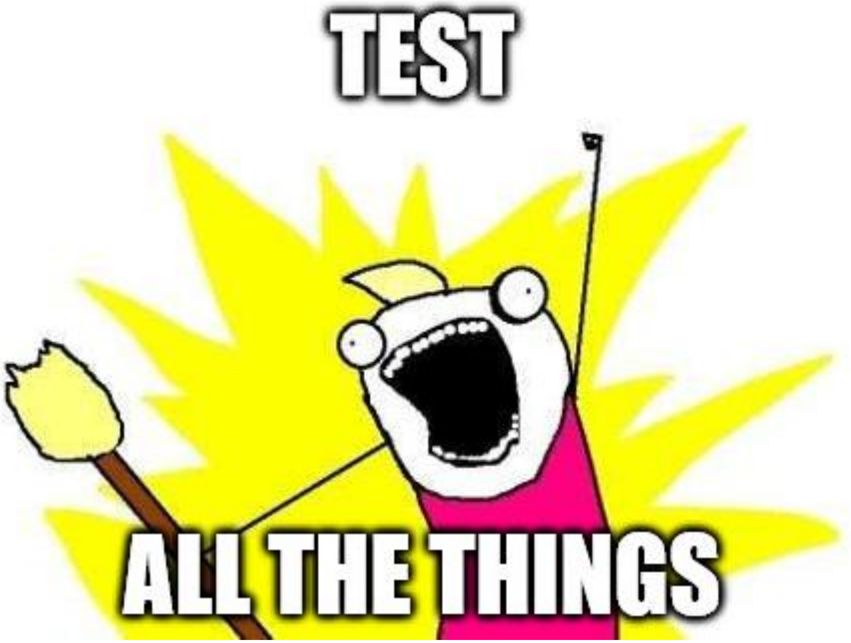
Code reviews



- Systematic review with
  - Co-worker
  - Team
  - External people
- Face 2 Face v. ‘Online’

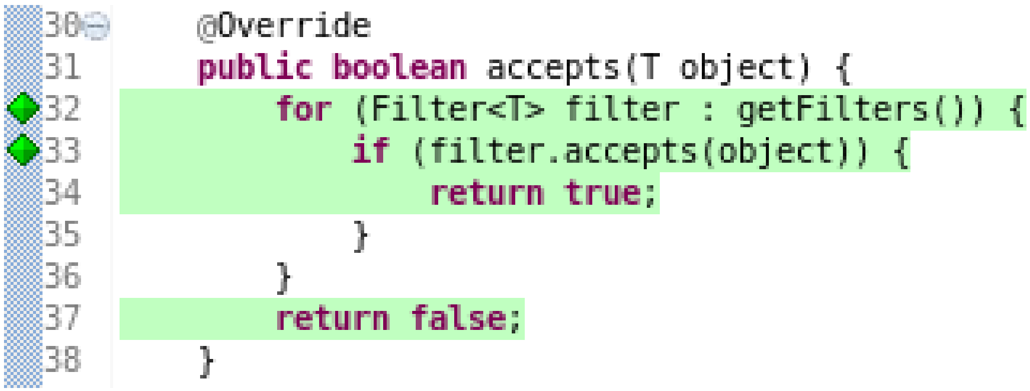
- Fresh mind reviewers
- Improvement ideas
- Great way of sharing knowledge

Good coverage of good tests

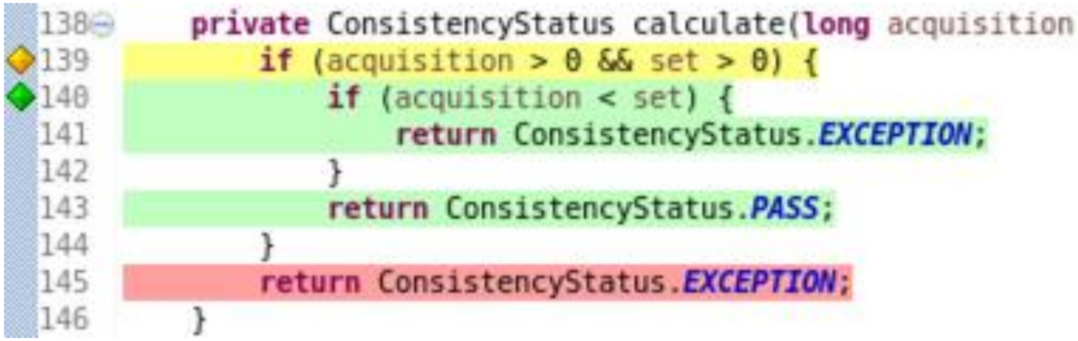


- **We can (should) test at different levels**
  - Unit testing
  - Integration testing
  - Acceptance testing
  - System testing
- **Automatic tests**
  - Quick
  - Repeatable
  - Easy to maintain
  - Need to be executed continuously

Good coverage of good tests



```
    @Override
    public boolean accepts(T object) {
        for (Filter<T> filter : getFilters()) {
            if (filter.accepts(object)) {
                return true;
            }
        }
        return false;
    }
```



```
    private ConsistencyStatus calculate(long acquisition
        if (acquisition > 0 && set > 0) {
            if (acquisition < set) {
                return ConsistencyStatus.EXCEPTION;
            }
            return ConsistencyStatus.PASS;
        }
        return ConsistencyStatus.EXCEPTION;
    }
```

- How much do we cover with tests?
- Test coverage

Legend:

- instruction covered
- instruction partially covered
- instruction uncovered

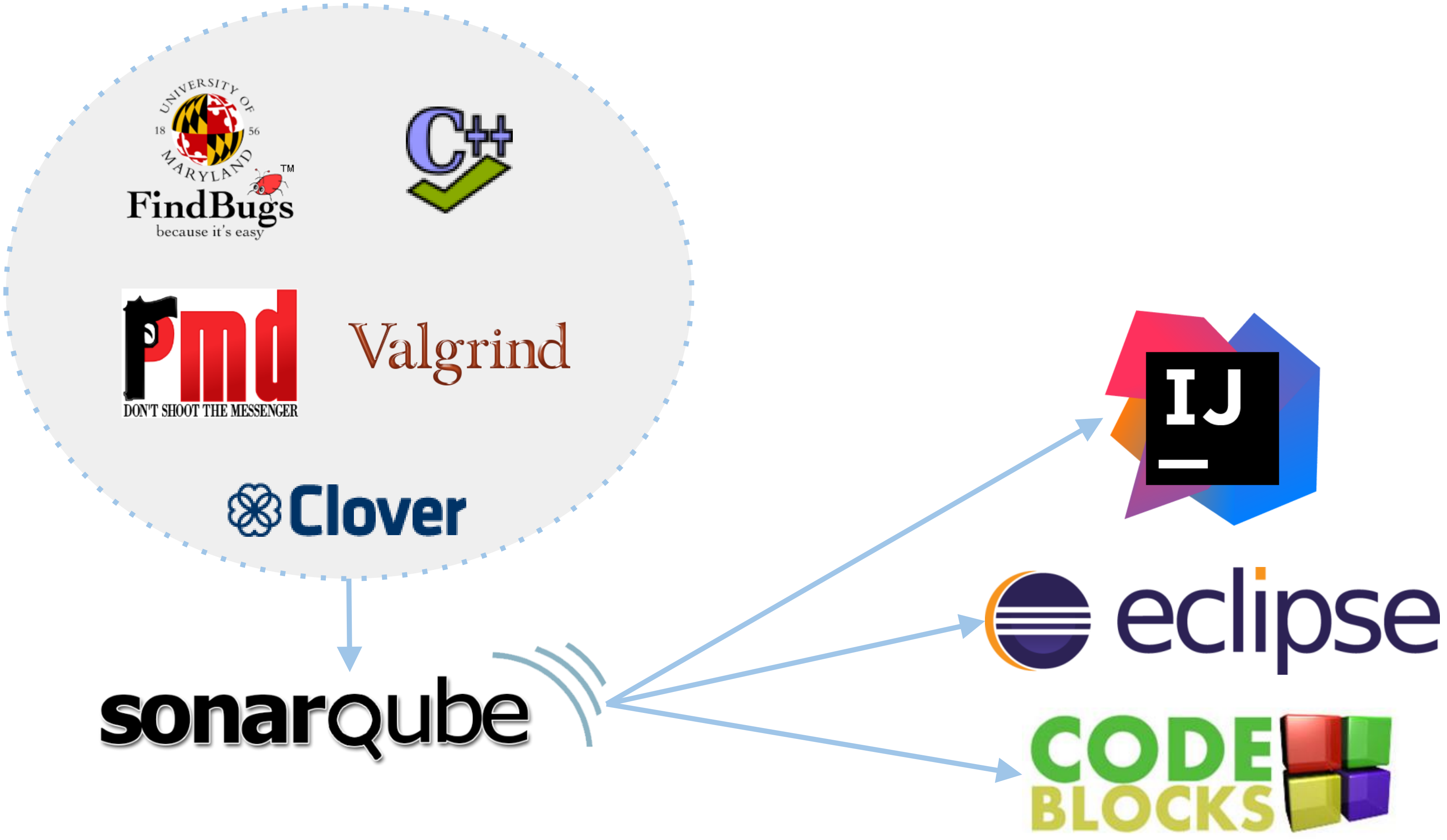
Static code analysis

Tools which help you to find bugs

Are there any?

- Static = without running software
- Can (or even should!) be automatized
- Find errors which are hard to spot by a human
- Easy shortcut to the better quality

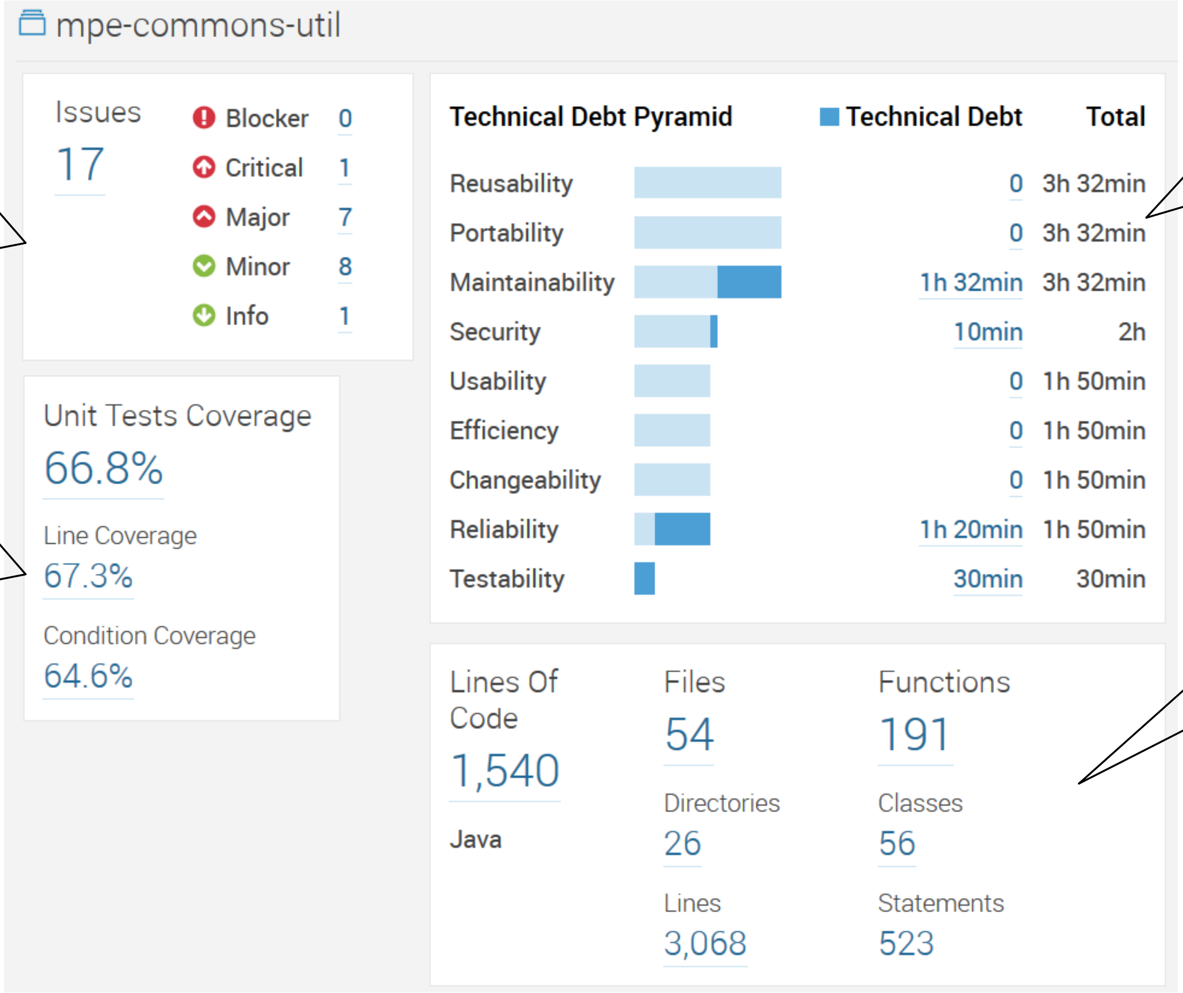
Static code analysis



Static code analysis

Code quality issues

Test coverage



mpe-commons-util

Issues

17

Severity	Count
Blocker	0
Critical	1
Major	7
Minor	8
Info	1

Technical Debt Pyramid	Technical Debt	Total
Reusability	0	3h 32min
Portability	0	3h 32min
Maintainability	1h 32min	3h 32min
Security	10min	2h
Usability	0	1h 50min
Efficiency	0	1h 50min
Changeability	0	1h 50min
Reliability	1h 20min	1h 50min
Testability	30min	30min

Unit Tests Coverage

66.8%

Line Coverage

67.3%

Condition Coverage

64.6%

Lines Of Code

1,540

Java

Files

54

Directories

26

Lines

3,068

Functions

191

Classes

56

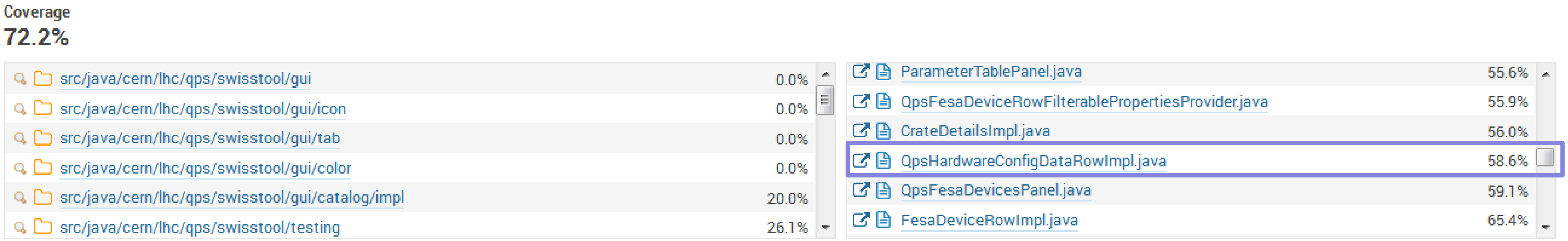
Statements

523

Complexity metrics

Project statistics

Static code analysis

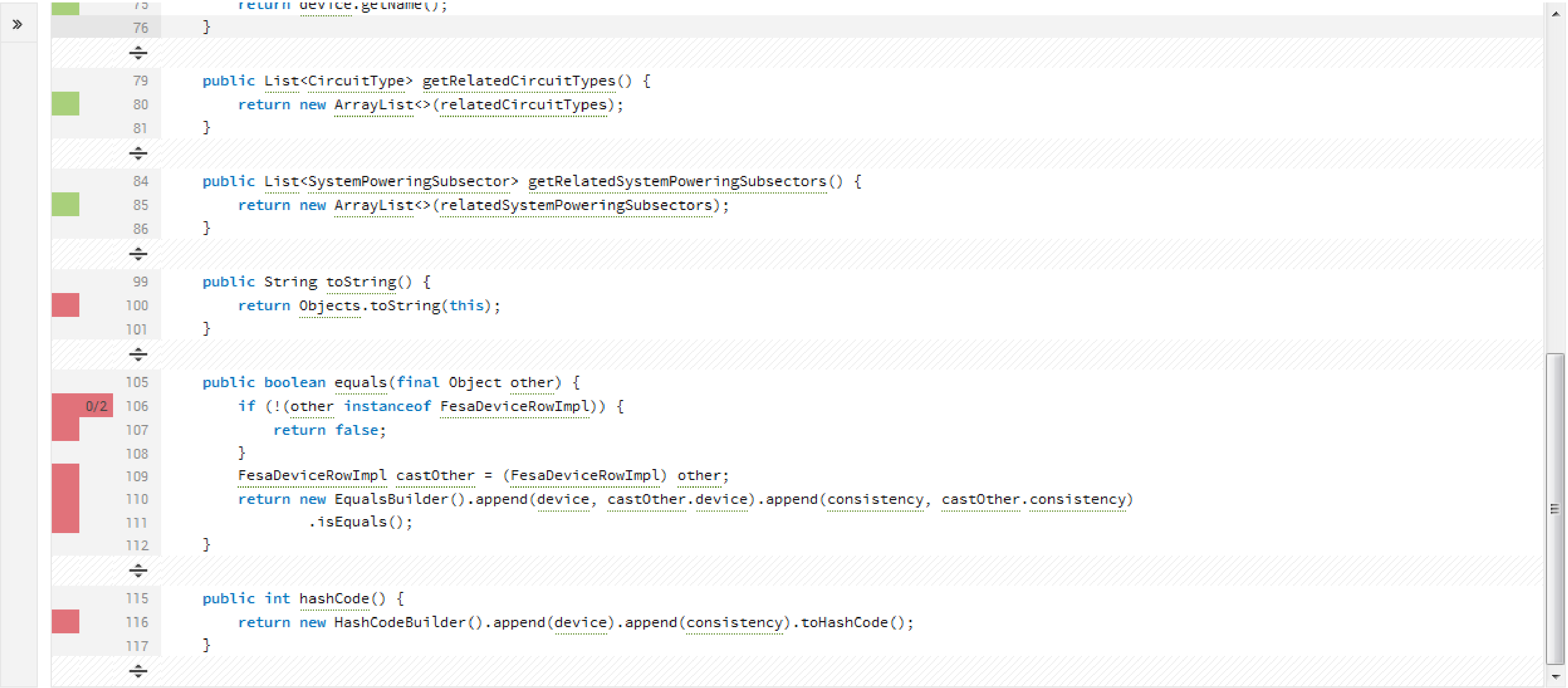


Coverage
72.2%

src/java/cern/lhc/qps/swisstool/gui	0.0%
src/java/cern/lhc/qps/swisstool/gui/icon	0.0%
src/java/cern/lhc/qps/swisstool/gui/tab	0.0%
src/java/cern/lhc/qps/swisstool/gui/color	0.0%
src/java/cern/lhc/qps/swisstool/gui/catalog/impl	20.0%
src/java/cern/lhc/qps/swisstool/testing	26.1%

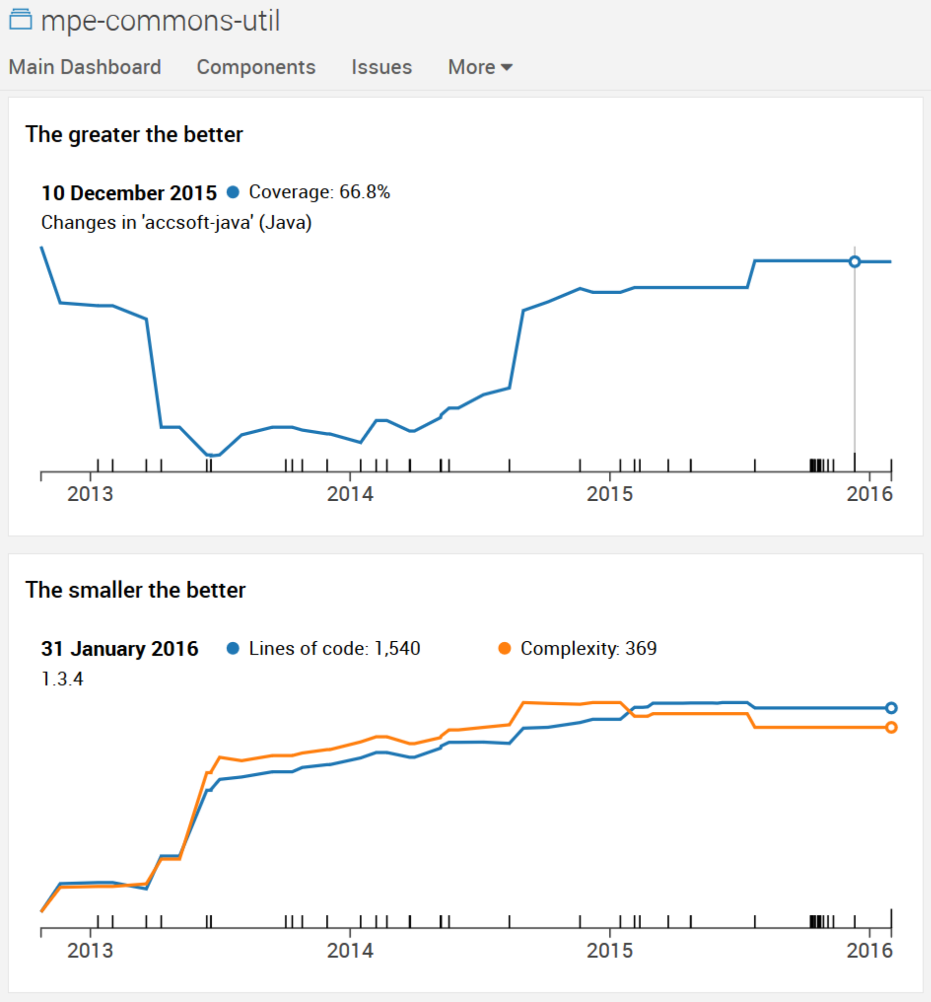
ParameterTablePanel.java	55.6%
QpsFesaDeviceRowFilterablePropertiesProvider.java	55.9%
CrateDetailsImpl.java	56.0%
QpsHardwareConfigDataRowImpl.java	58.6%
QpsFesaDevicesPanel.java	59.1%
FesaDeviceRowImpl.java	65.4%

Static code analysis



```
        return device.getName();
    }

    public List<CircuitType> getRelatedCircuitTypes() {
        return new ArrayList<>(relatedCircuitTypes);
    }

    public List<SystemPoweringSubsector> getRelatedSystemPoweringSubsectors() {
        return new ArrayList<>(relatedSystemPoweringSubsectors);
    }

    public String toString() {
        return Objects.toString(this);
    }

    public boolean equals(final Object other) {
        if (!(other instanceof FesaDeviceRowImpl)) {
            return false;
        }
        FesaDeviceRowImpl castOther = (FesaDeviceRowImpl) other;
        return new EqualsBuilder().append(device, castOther.device).append(consistency, castOther.consistency)
                .isEquals();
    }

    public int hashCode() {
        return new HashCodeBuilder().append(device).append(consistency).toHashCode();
    }
```

Static code analysis



The greater the better

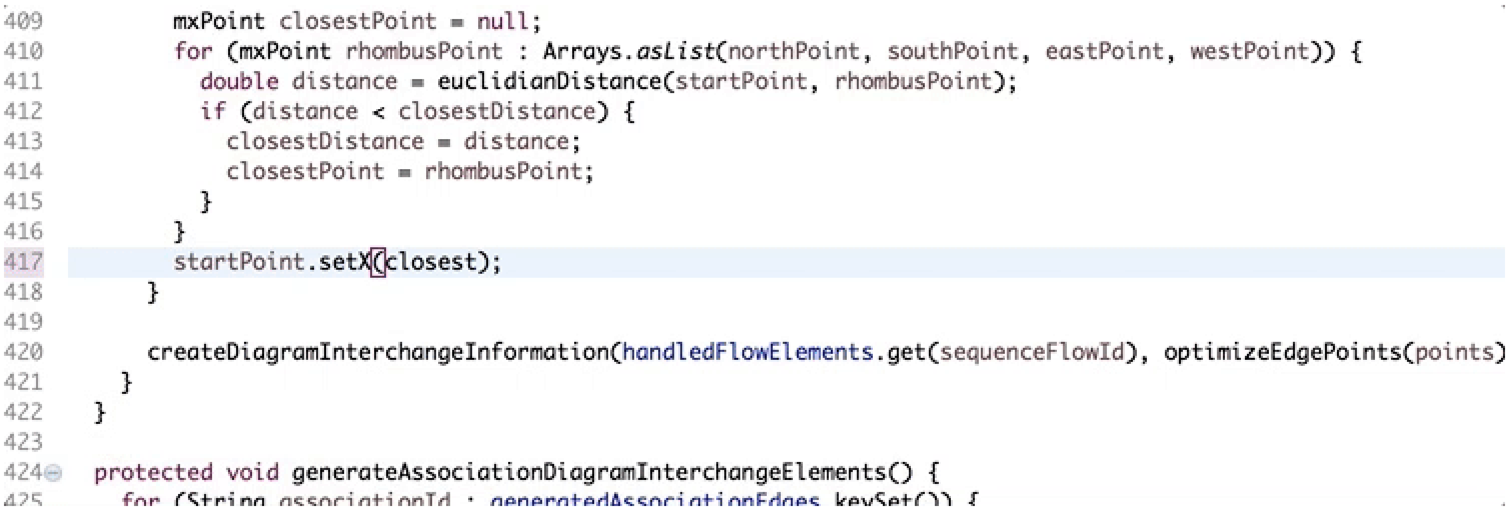
10 December 2015 ● Coverage: 66.8%
Changes in 'accsoft-java' (Java)

The smaller the better

31 January 2016 ● Lines of code: 1,540 ● Complexity: 369
1.3.4

Project statistics (quality metrics) over the time

Static code analysis



```
        mxPoint closestPoint = null;
        for (mxPoint rhombusPoint : Arrays.asList(northPoint, southPoint, eastPoint, westPoint)) {
          double distance = euclidianDistance(startPoint, rhombusPoint);
          if (distance < closestDistance) {
            closestDistance = distance;
            closestPoint = rhombusPoint;
          }
        }
        startPoint.setX(closest);
      }

      createDiagramInterchangeInformation(handledFlowElements.get(sequenceFlowId), optimizeEdgePoints(points)
    }
  }

  protected void generateAssociationDiagramInterchangeElements() {
    for (String associationId : generatedAssociationEdges.keySet()) {
```

Source: SonarLint

Static code analysis

Life demo time!

You can play on your own at:
http://sonar.cern.ch

Quality Monitoring - Summary

- **Code is read much more often than it is written!**
- Keep an eye on the code quality - continuous monitoring can help you a lot.
- Variety of tools available - choose your own set and take advantage of them.

Thanks a lot!

Any questions?