Using Automated Code Reviews to Achieve "Continuous Quality"

ASQF Agile Night Austria, Oct. 2018

Peter Kofler, 'Code Cop'
@codecopkofler
www.code-cop.org

Peter Kofler

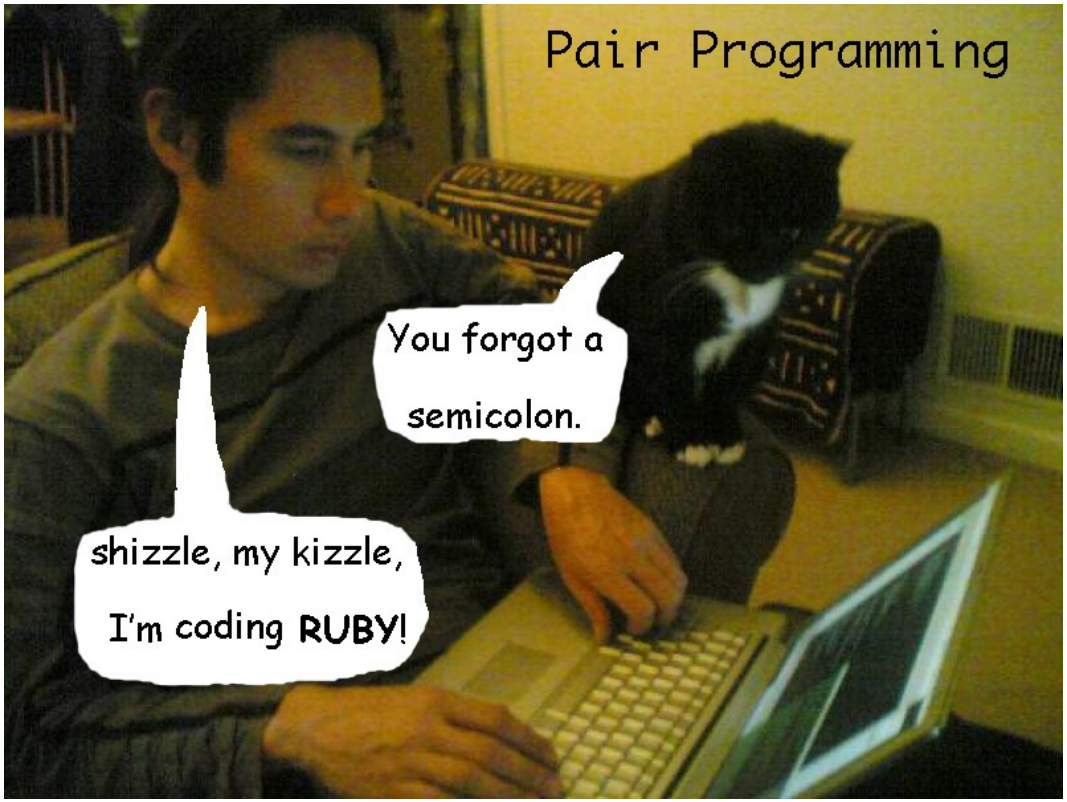
- Ph.D. (Appl. Math.)
- Professional Software Developer for 20 years
- “fanatic about code quality”
- Independent Code Quality Coach

I help development teams with

- Professionalism
- Quality and Productivity
- Continuous Improvement

Mentoring

- Pair Programming
- Programming Workshops
- Deliberate Practice, e.g. Coding Dojos



Developing Quality Software Developers

Agenda

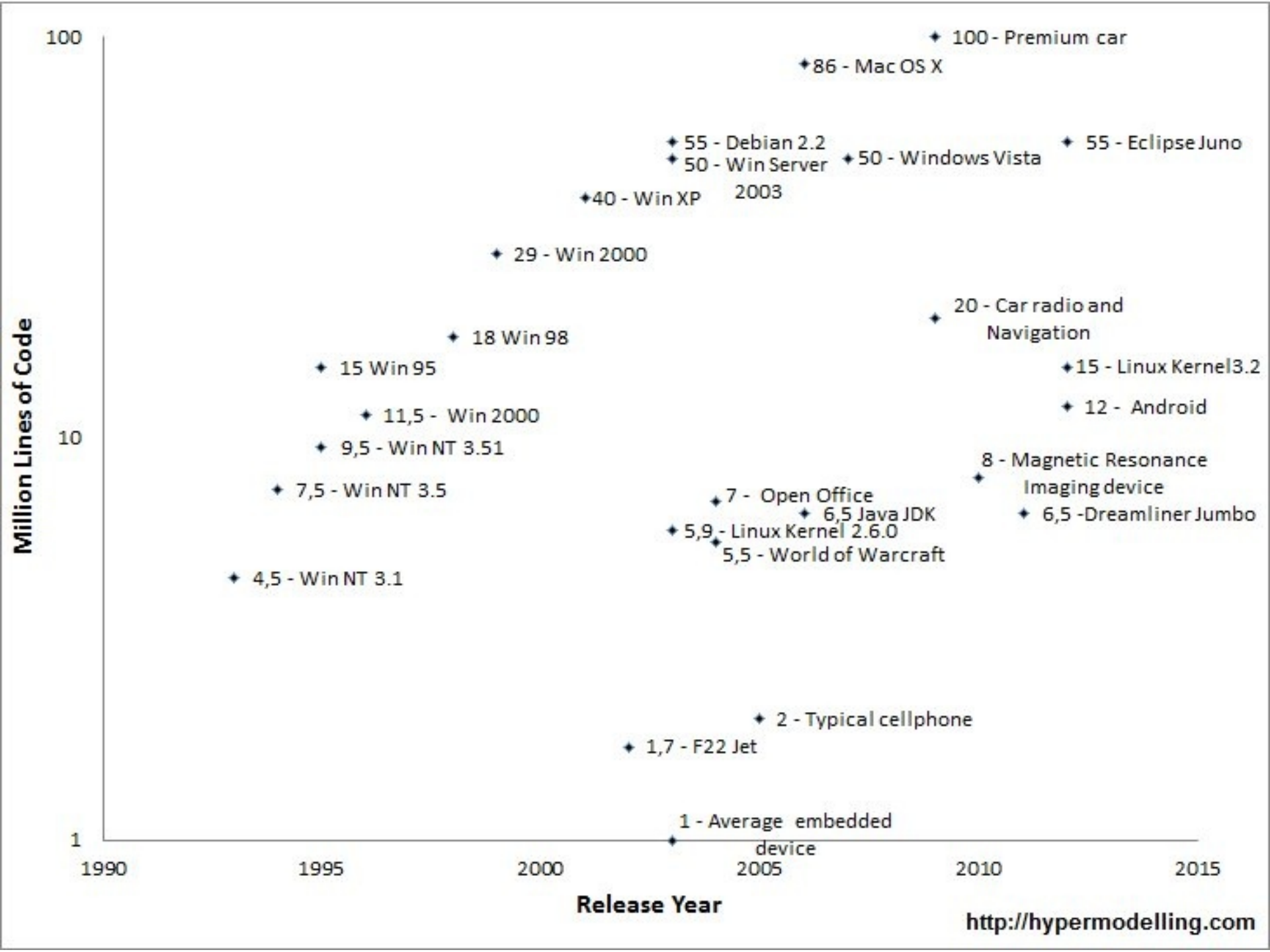
- What's the problem?
- Static Code Analysis
- Examples
- Conclusions

“standing on the shoulder of giants“

- Productive languages
- Complete libraries
- Powerful frameworks
- Automated everything

But

- Requirements and complexity increase
- Technology moves fast
- Knowledge Half-Life of 18 months (2007)
- Abstractions are leaky


100
10
1
Million Lines of Code
100 - Premium car
86 - Mac OS X
55 - Debian 2.2
50 - Win Server 2003
50 - Windows Vista
55 - Eclipse Juno
40 - Win XP
29 - Win 2000
20 - Car radio and Navigation
18 Win 98
15 Win 95
15 - Linux Kernel3.2
11,5 - Win 2000
12 - Android
9,5 - Win NT 3.51
8 - Magnetic Resonance Imaging device
7,5 - Win NT 3.5
7 - Open Office
6,5 Java JDK
6,5 -Dreamliner Jumbo
5,9 - Linux Kernel 2.6.0
5,5 - World of Warcraft
4,5 - Win NT 3.1
2 - Typical cellphone
1,7 - F22 Jet
1 - Average embedded device
1990
1995
2000
2005
2010
2015
Release Year
http://hypermodelling.com

We use Conventions, Guidelines, Best Practices

Bernhard Merkle „Stop the Software Architecture Erosion“

But I cannot remember eveyrthnig.

What are you doing...

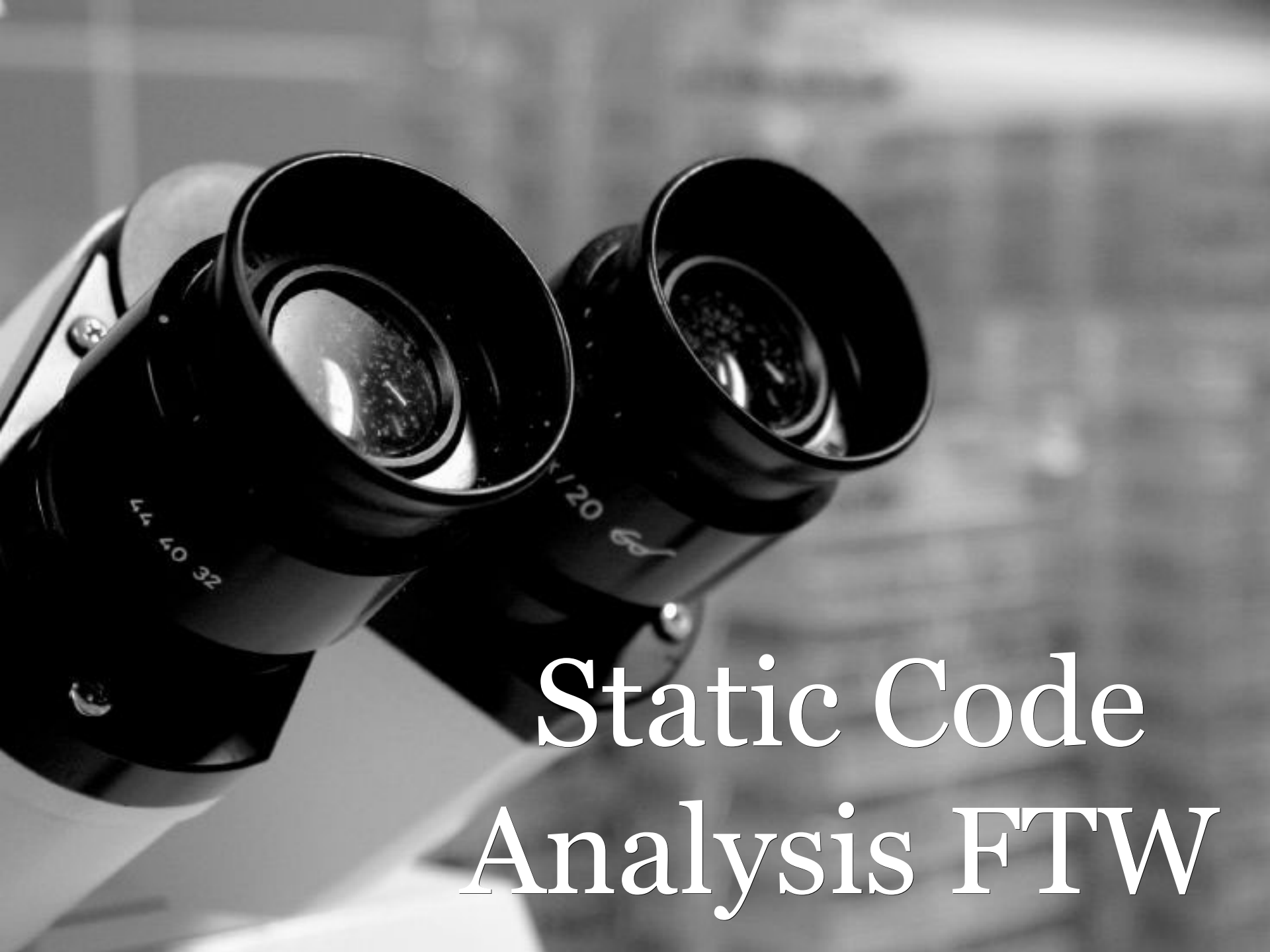
- to ensure external quality? (e.g. “it works”)

- to ensure internal quality? (e.g. “it can be changed”)

One developer is not enough

- Double Program Entry (Punch Cards)
- Code Reviews
- Pair Programming (XP, 1990)
- Pull Requests (GitHub, 2008)
- Mob Programming (2016)
- etc.

Continuous Delivery needs Automated Code Reviews


Static Code
Analysis FTW

Possibilities of Analysis

- Lexical analysis
  - coding conventions, design idioms
- Flow/path analysis
  - null-pointer, dead code
- Dependency analysis
  - architectural/design flaws
- Behavioural Code Analysis/”Commit Mining“
    - social information + time dimension

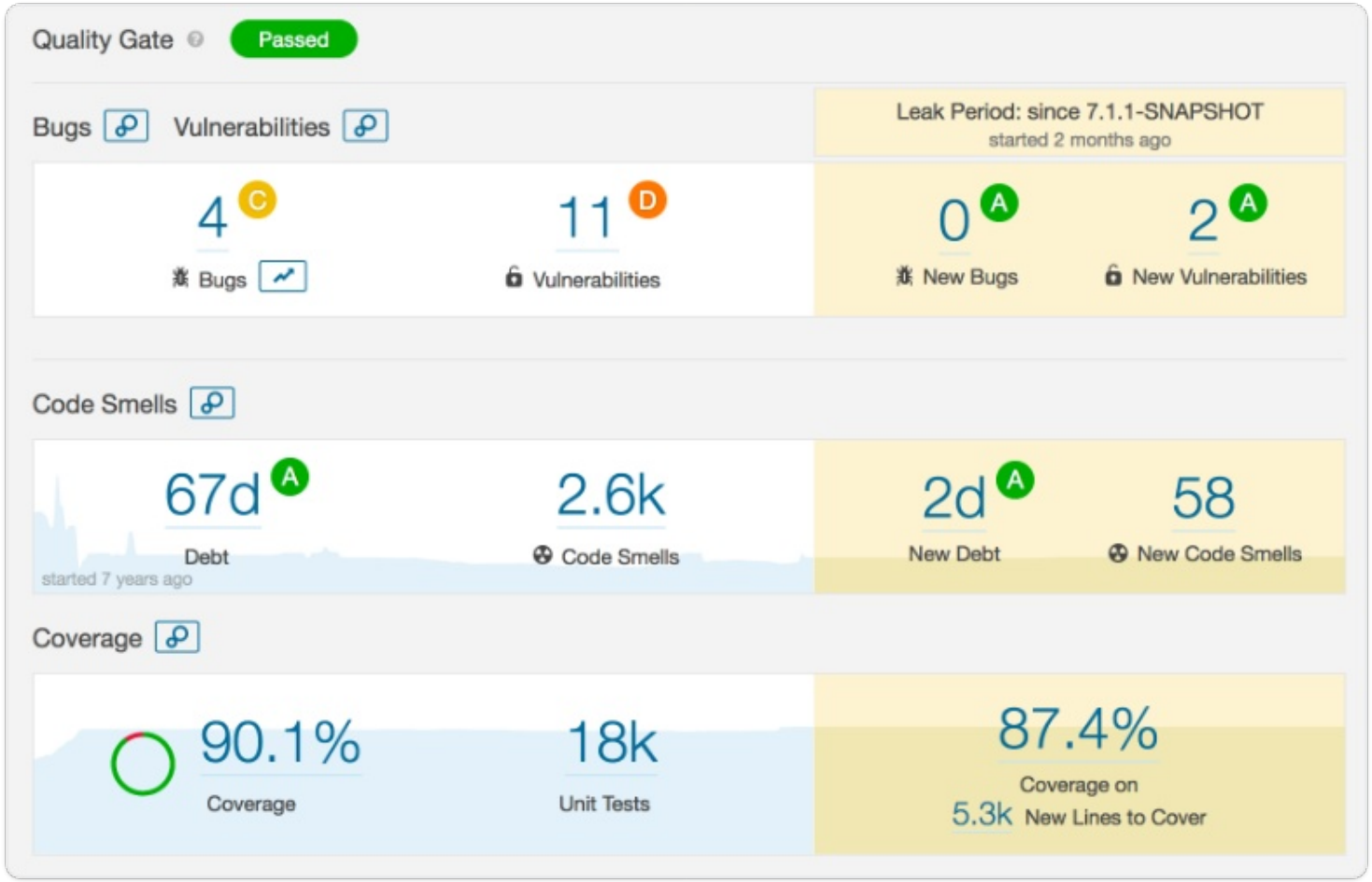
Levels of Analysis

- Micro Level
  - Statements, e.g. =, ==, {}
- Macro Level
  - Class Design, APIs, Error Handling
- Architecture Level
  - Interfaces, Layers, Components

Impact and Cost

- Statements
  - low impact, but can be severe (bug)
  - easy to find, easy to fix
- Class and API (Design)
  - e.g. class coupling → medium impact
  - easy to fix individually
- Architecture
  - e.g. layer violations → high impact
  - hard to fix or easy to fix but in many places

Example: Quality Dashboard



https://www.sonarqube.org/

e.g. SonarQube, CAST

- **Lexical** analysis
- Findings on **Micro** and **Macro** Level

- Works out of the box

- SonarQube is free for many languages
- Commercial extensions for PL/SQL etc.

- History with trend graphs

SonarQube checks

- > 1000 Rules (Java):
- Formatting
- Language Usage
- Redundancies
- (Micro) Performance Improvements
- Maintenance Issues
- (Beginner) Mistakes
- Potential Bugs

Do you care?

- Which issues on Micro and Macro Level are relevant to you?
- Which issues should stop the pipeline?

Problems

- Forces team to follow same rules
    - Rock stars, “Fix it later” ;-)
- Not all violations are equally severe.
- Not all violations are actionable.
- Often too many findings, drowning real issues
- Need to choose subset depending on project

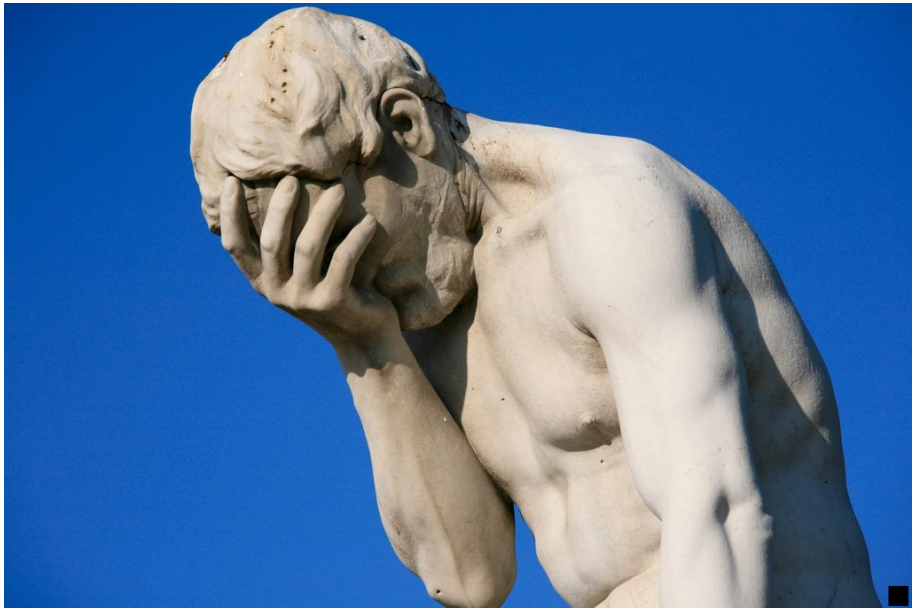
Real World Experience

- Emergency Services
- Evaluation at “Half-Completed” (2015)

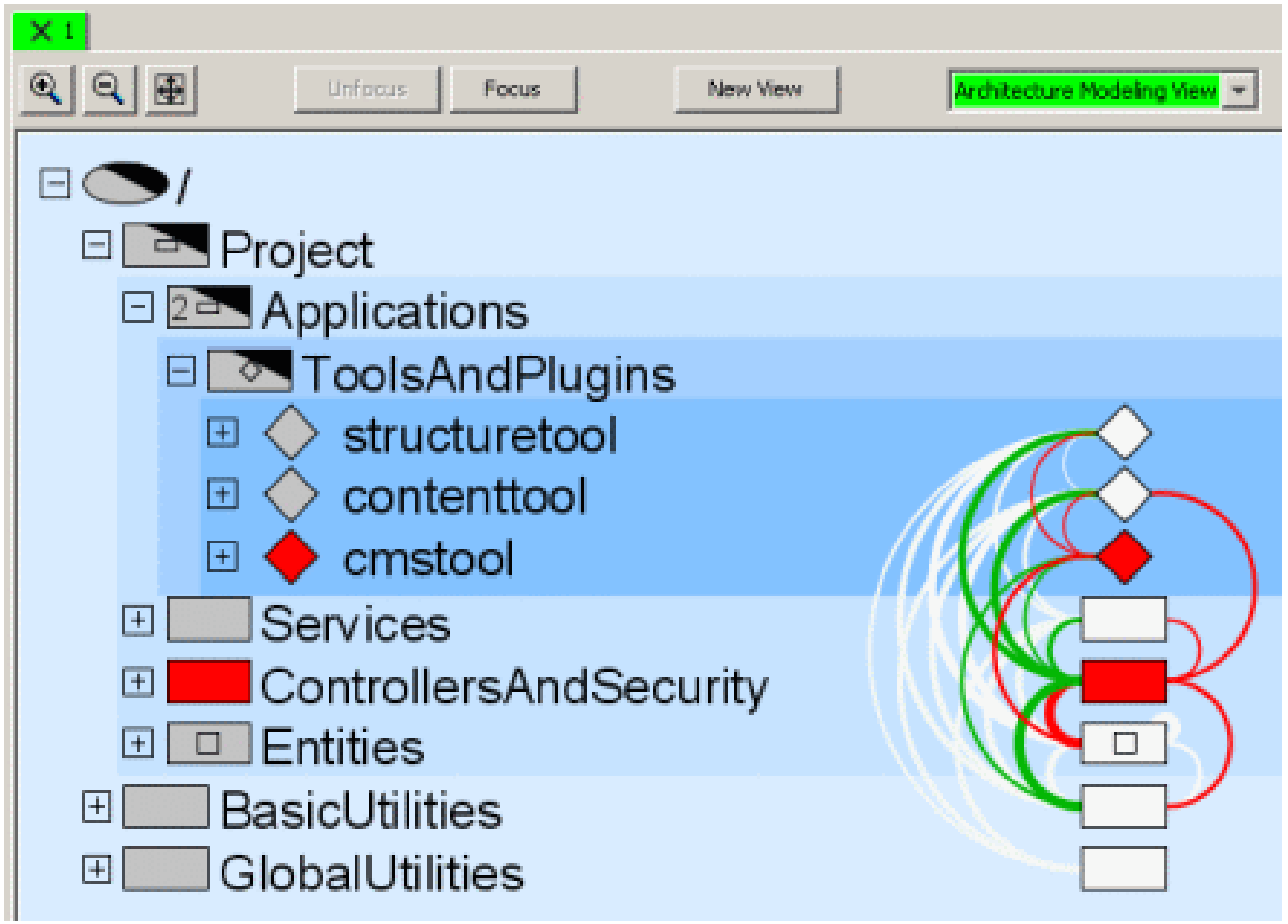
- Excellent team

- Oops, some code excluded from checks

- 9000 open issues
- Will they catch up?



Example: Dependency Analysis



https://www.hello2morrow.com/products/sotograph/sotoarc

e.g. Sotoarc, Structure101

- **Dependency** analysis
- **Macro** and **Architecture** Level

- Needs up front customisation
  - Definition of “the architecture”

- (almost) no free tools

Dependency Analysis finds

- Architecture deviations
- High coupling (Classes and Modules)
- Cycles
- Stability of API
- Bad (Design) Patterns, „Anti-Patterns“
- Usage of internal API
- Usage of forbidden API
- (Probably) Unused classes and methods

Do you care?

- Do you have a defined target structure?
- Which issues should stop the pipeline?
- Why?

Problems

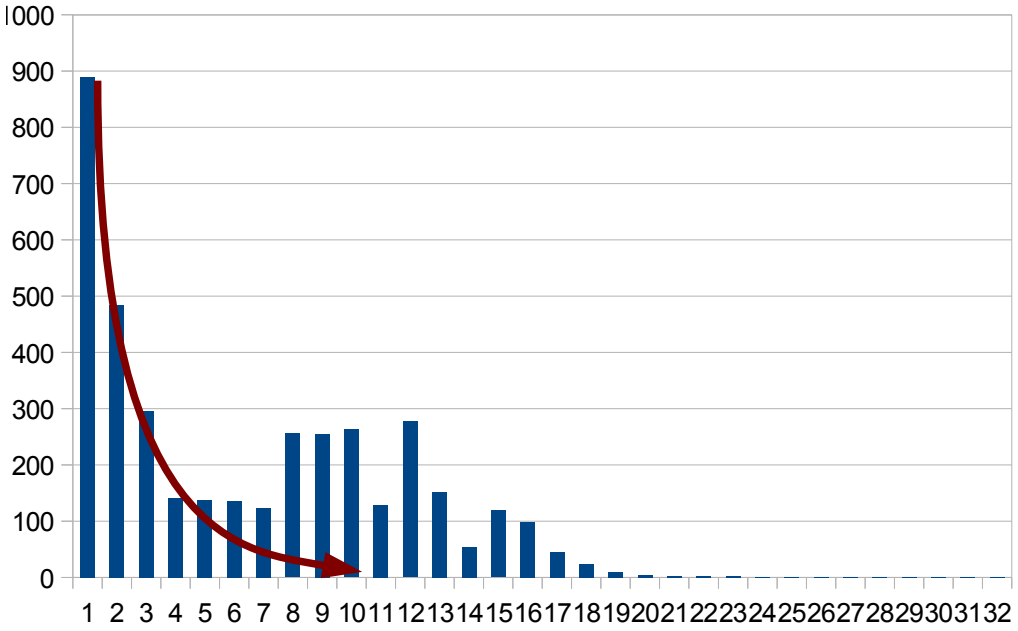
- Usually no detailed architecture definition (or not actionable/measureable)
- Impossible to see problems without tool
- Very abstract, need “senior” level skills
- People often build their own analysers

Real World Experience

- 3rd place Deloitte Technology Fast 500
- “A fresh look at the project” (2016)

- 2 DevOps teams
- High quality

- Layering problem



Continuous Quality?

Death of a Thousand Cuts

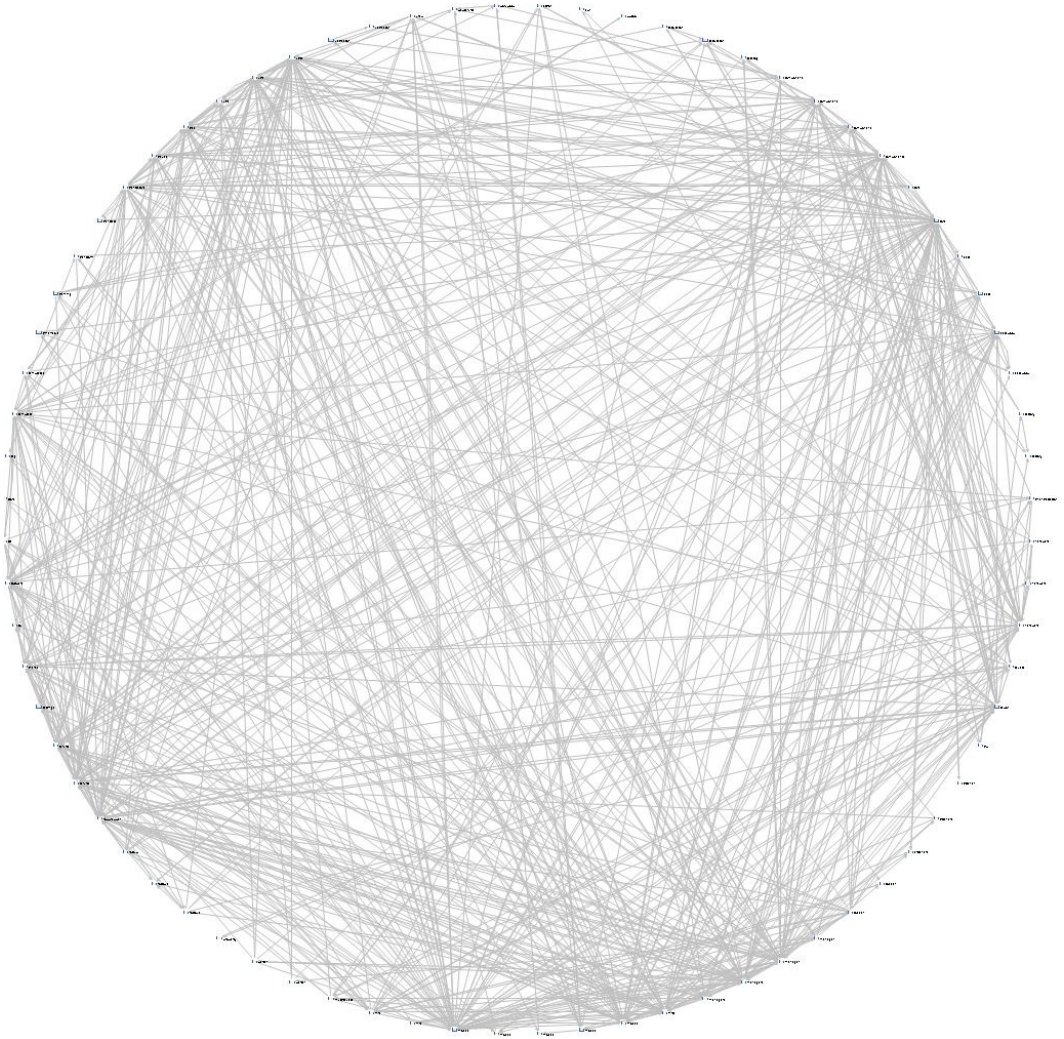
Real World Experience:
Missing Lexical Analysis

- Due Diligence for smaller company (2018)

- 4M lines of code
- 80K warnings
- 190K relevant issues (out of 1.1M)

- **Impossible to fix**
- Impacted offer/price

Real World Experience:
Missing Dependency Analysis



Real World Experience: Missing Dependency Analysis

- Internal product to calculate costs of large outsourcing deals (2013)

- Part of product family
- Several teams working constantly

- “Everything depending on everything”
- **Impossible to fix**
- Impacted stability and regressions

Conclusion

Add Code Analysis to Build Pipeline Today!

Break Pipeline on (selected) Violations

Peter Kofler

@codecopkofler

www.code-cop.org

CC Images

- Bruce http://www.flickr.com/photos/sherpas428/4350620602/
- pairing http://www.flickr.com/photos/dav/94735395/
- agenda http://www.flickr.com/photos/24293932@N00/2752221871/
- Altas https://www.flickr.com/photos/quinnanya/5890297160/
- leaky wall https://www.flickr.com/photos/gammaman/7803857922
- wants you http://www.flickr.com/photos/shutter/105497713/
- automation http://www.flickr.com/photos/aquilaonline/510921786/
- microscope http://www.flickr.com/photos/gonzales2010/8632116/
- head in hands https://www.flickr.com/photos/proimos/4199675334/
- dinosaurs https://www.flickr.com/photos/superamit/3886055392/
- dump http://www.flickr.com/photos/sanmartin/2682745838/
- finish http://www.flickr.com/photos/jayneandd/4450623309/