university of groningen

Why you should care about Technical Debt

Prof. dr.ir. Paris Avgeriou - paris@cs.rug.nl

Software Engineering and Architecture Group

http://www.cs.rug.nl/~paris/

Ireland
Liverpool
Amsterdam
Netherlands
London
Brussels
Cologne
Germany
Belgium
Luxembourg
Frankfurt
Berlin
Poland
Warsaw
Prague
Czechia
Paris
Munich
Vienna
Slovakia
Austria
Budapest
Hungary
France
Switzerland
Slovenia
Zagreb
Romania
Milan
Croatia
Belgrade
Bay of Biscay
Bosnia and Herzegovina
Serbia
Sarajevo
Bucharest
Monaco
Italy
Montenegro
Sofia
Kosovo
Skopje
Bulgaria
Andorra
Rome
Podgorica
Macedonia (FYROM)
Barcelona
Tirana
Albania
Porto
Madrid
Tyrrhenian Sea
Portugal
Spain
Valencia
Greece
Lisbon
Athens
Seville

The Known Universe

university of groningen

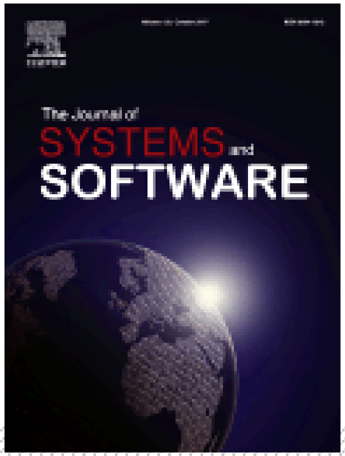
Rankings - Top 100 university

#83	Times Higher Education Worldwide
#59	Academic Ranking of World Universities
#86	U.S. News 'Best Global Universities Ranking'

Founded in 1614

Research Philosophy

- Core business: Software Architecture
- With Dutch & European industry (real problems)
  - Embedded Systems & Enterprise Applications
- Automated Software Engineering
- Evidence-based Software Engineering
  - Evidence matters - empirical research methods



IEEE Software

university of groningen

Outline

› **Introducing the metaphor**
› Emergence of TD
› Concepts of TD and management
› Present and Future

university of groningen

"Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite ... "

"The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise. "

Ward Cunningham, The WyCash portfolio management system, OOPSLA '92

Technical Debt is a collection of design or implementation constructs

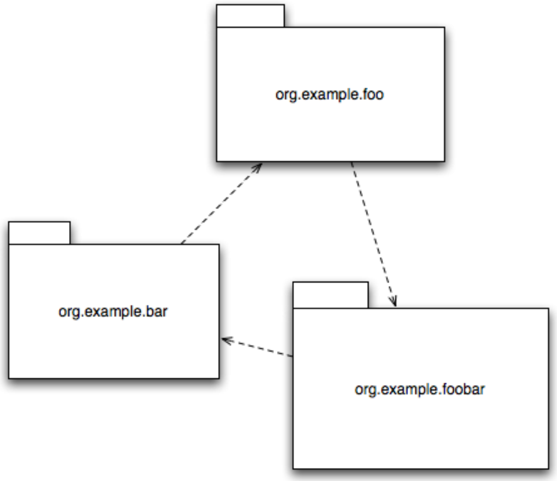
that are expedient in the short term,

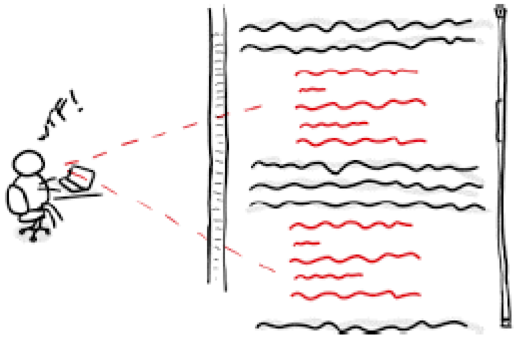
but set up a technical context that

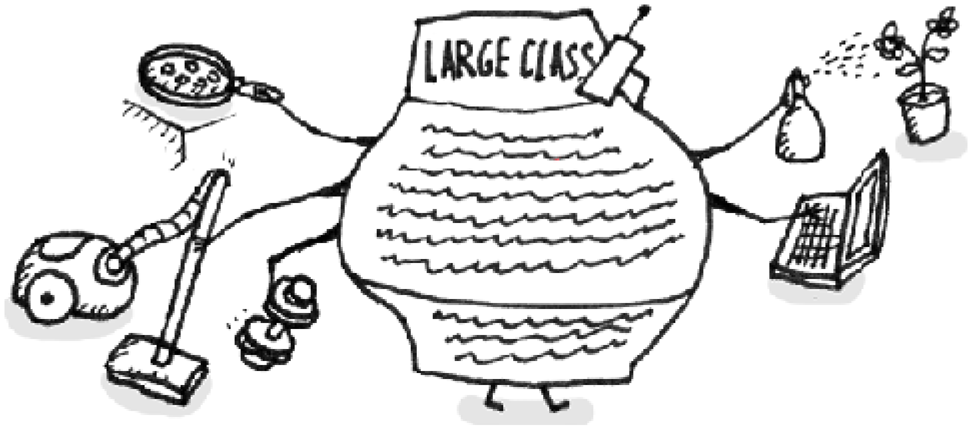
can make future changes more costly or impossible

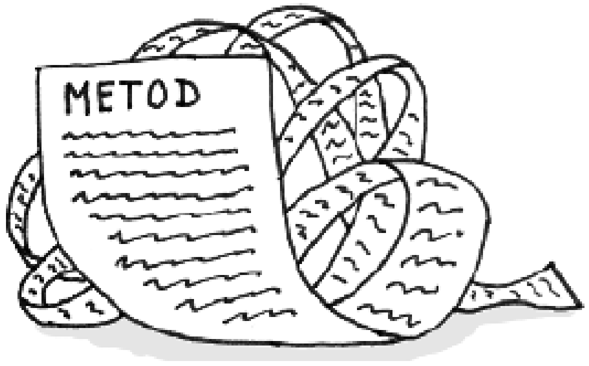
Dagstuhl April 2016

Technical Debt illustrated









Images from https://refactoring.guru/smells

Technical Debt metaphor

- Debt is a necessary tradeoff
  - Loan for investment
  - Quality-- for business value++
- Pay back *principal* (fix TD) + *interest* (maintain SW)
- Debt should be monitored and managed
  - Risk – accumulation may spiral out of control

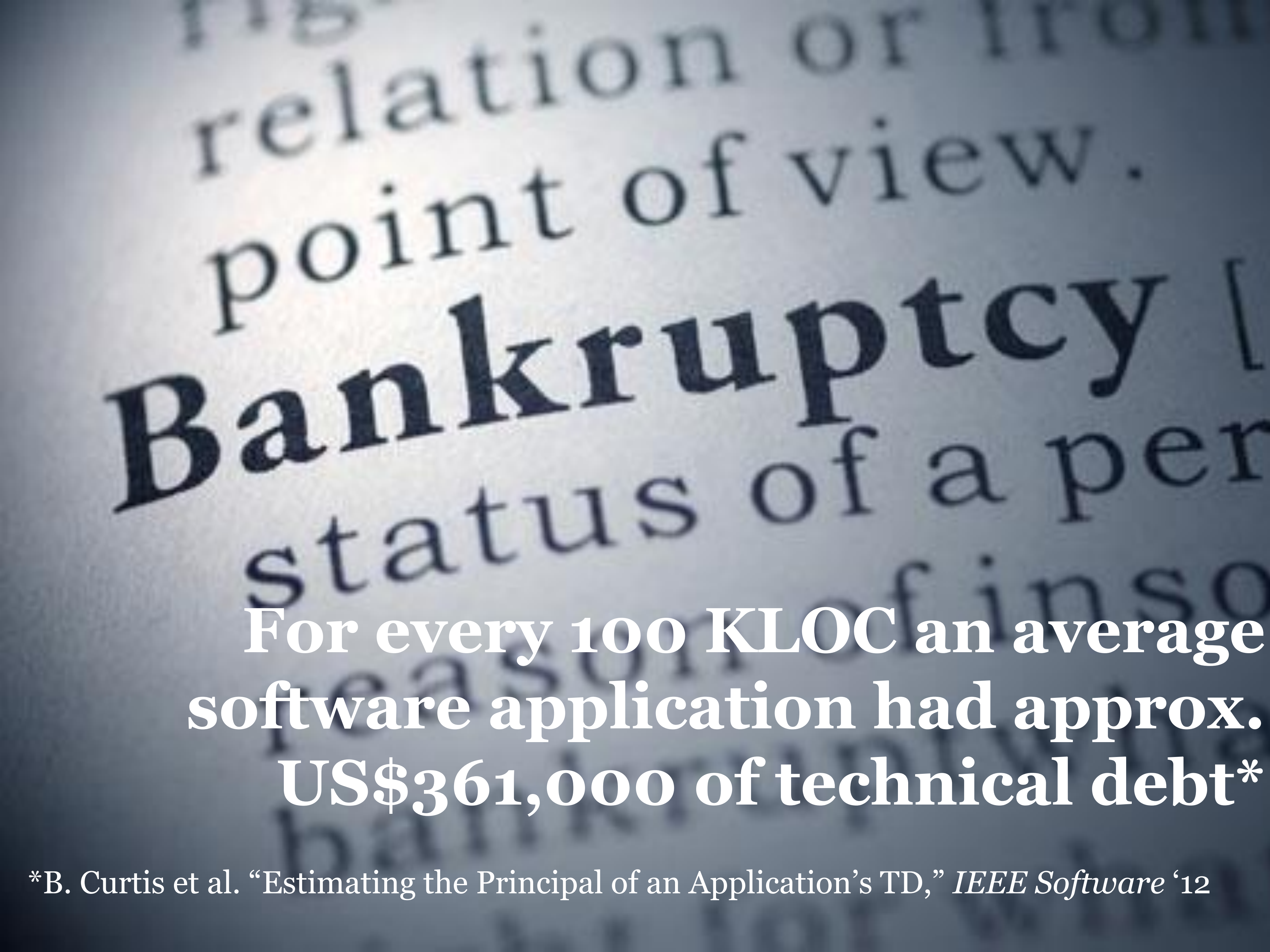
Typical symptoms

› Taking more time to build a feature or fix defects
› Changes ripple through the system
› Rework is often and unexpected
› Deadlines/milestones continuously slipping
› Velocity drops
› Testing becomes very expensive

university of
groningen

Outline

› Introducing the metaphor
› **Emergence of TD**
› Concepts of TD and management
› Present and Future



For every 100 KLOC an average software application had approx. US$361,000 of technical debt*

*B. Curtis et al. “Estimating the Principal of an Application’s TD,” *IEEE Software* ‘12

Is this really new?

Communities

- Maintenance & evolution
- Reengineering / refactoring

Terms

- Aging
- Decay
- Sustainability

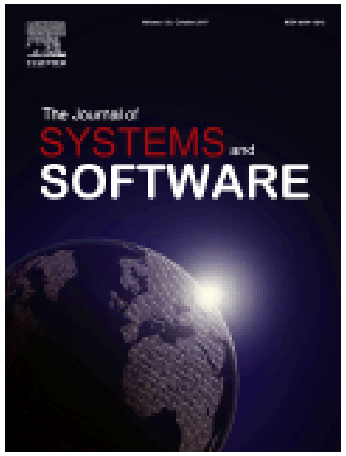
- Little progress
- “Dull” topic

Convergence of SE disciplines

- Program analysis/comprehension
- SW Quality measurement
- Qualitative research methods
- SW risk management

Managing TD>sum of parts!

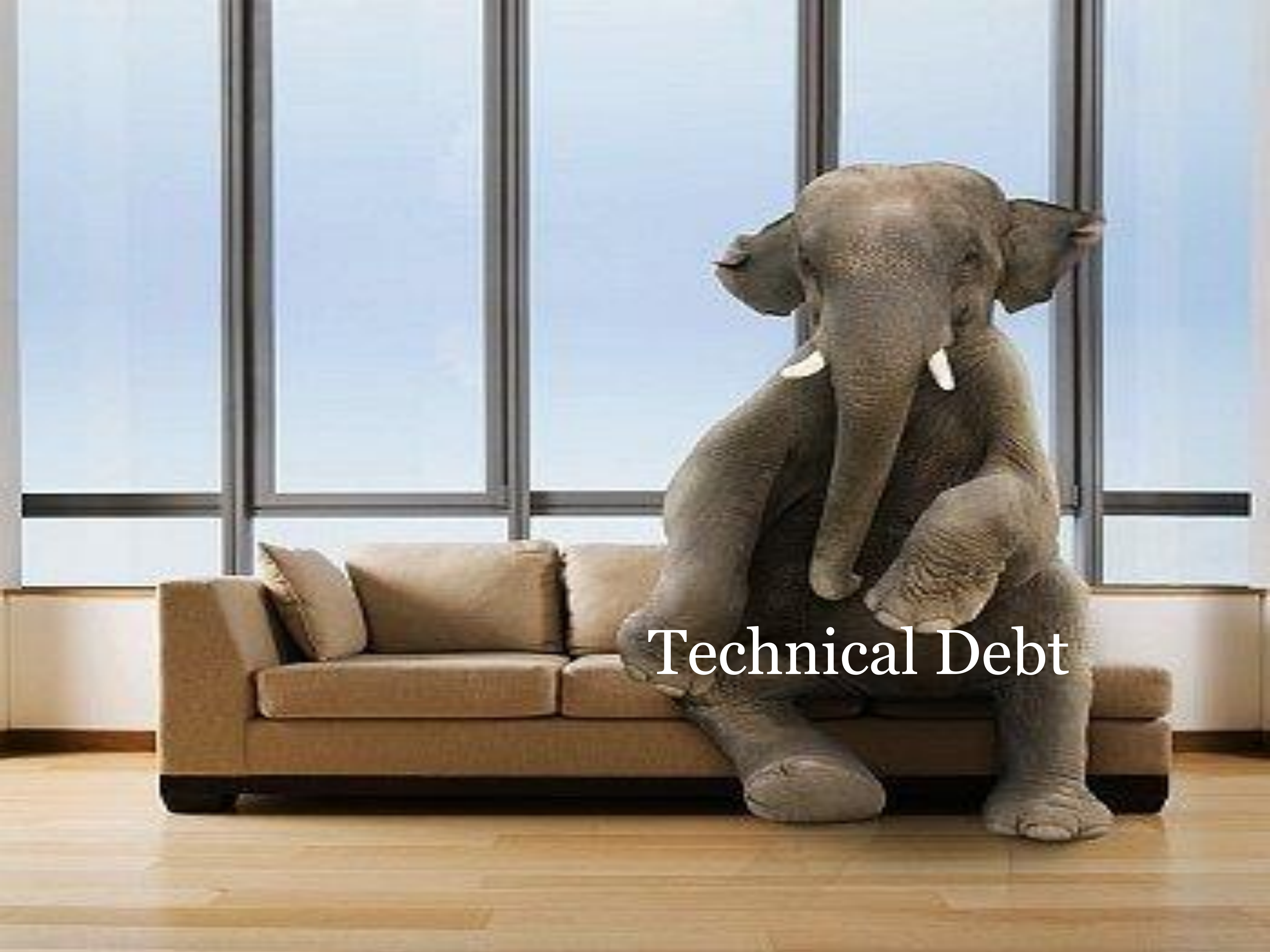
P. Avgeriou et al. *Reducing Friction in Software Development*, IEEE SW '16



IEEE Software

SCHLO
Leibniz-Ze

Z. Li et al., A systematic mapping study on technical debt and its management, JSS 2015



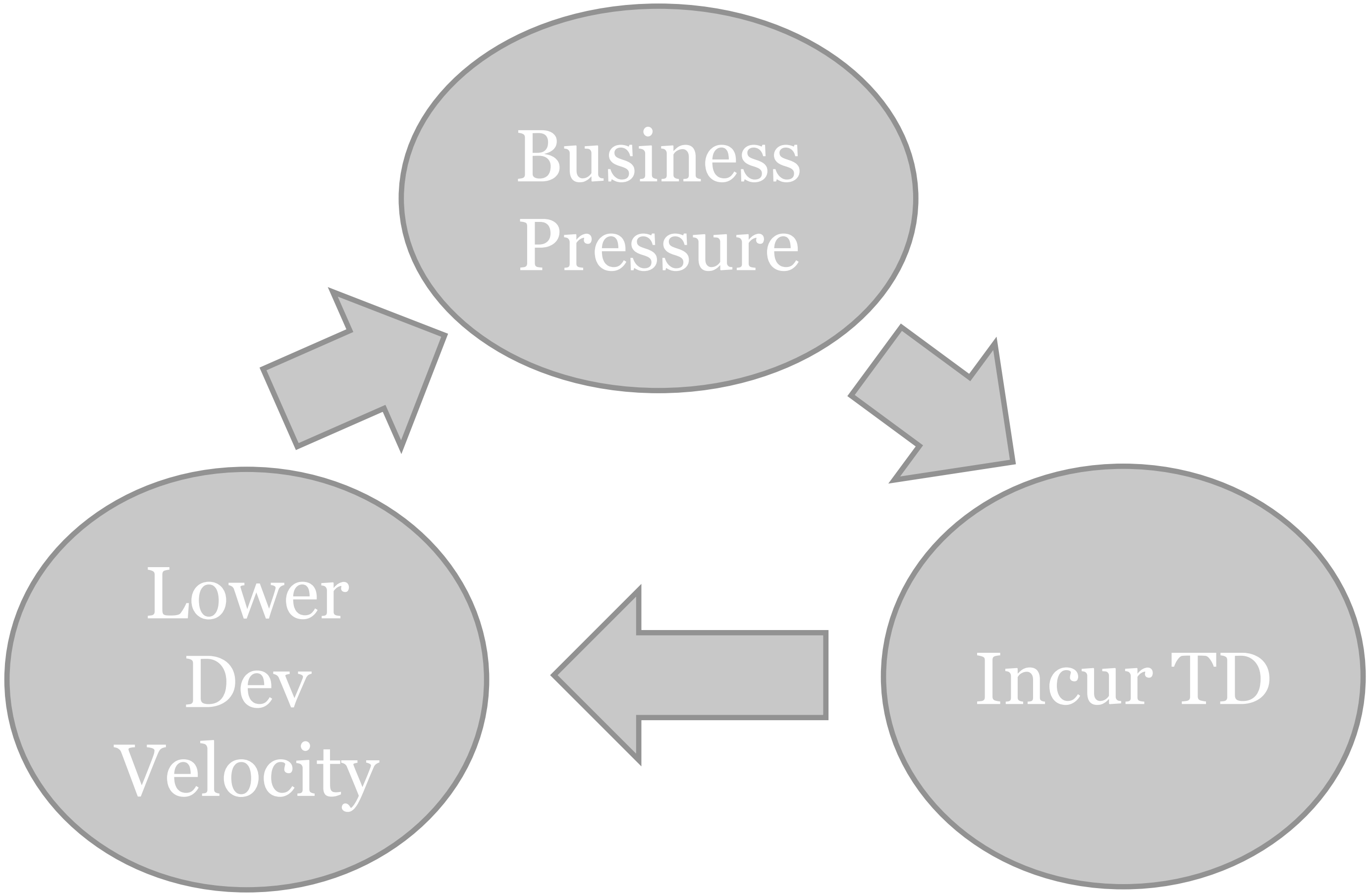
Technical Debt

university of
groningen

Outline

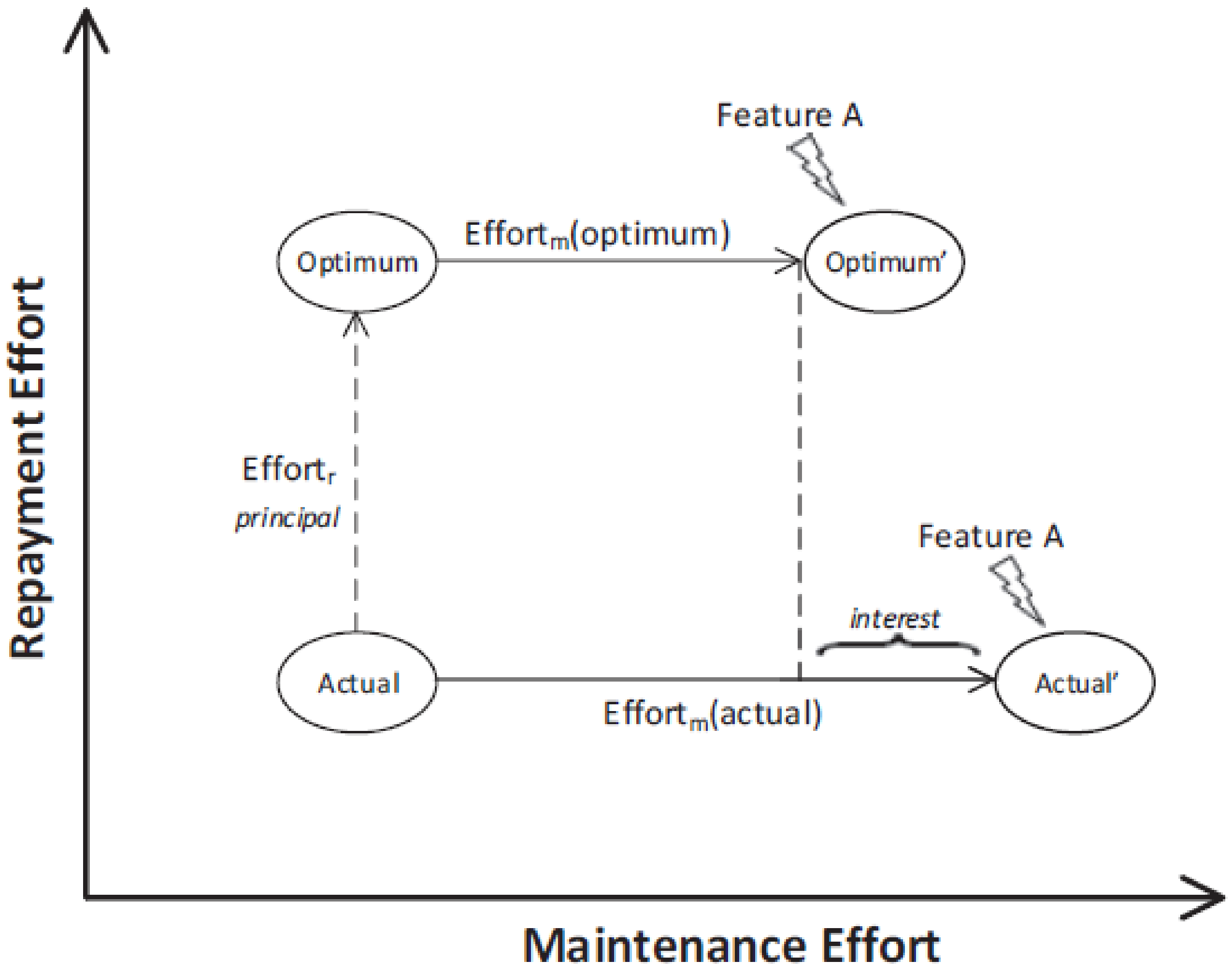
- Introducing the metaphor
- Emergence of TD
- **Concepts of TD and management**
- Present and Future

Vicious circle of technical debt



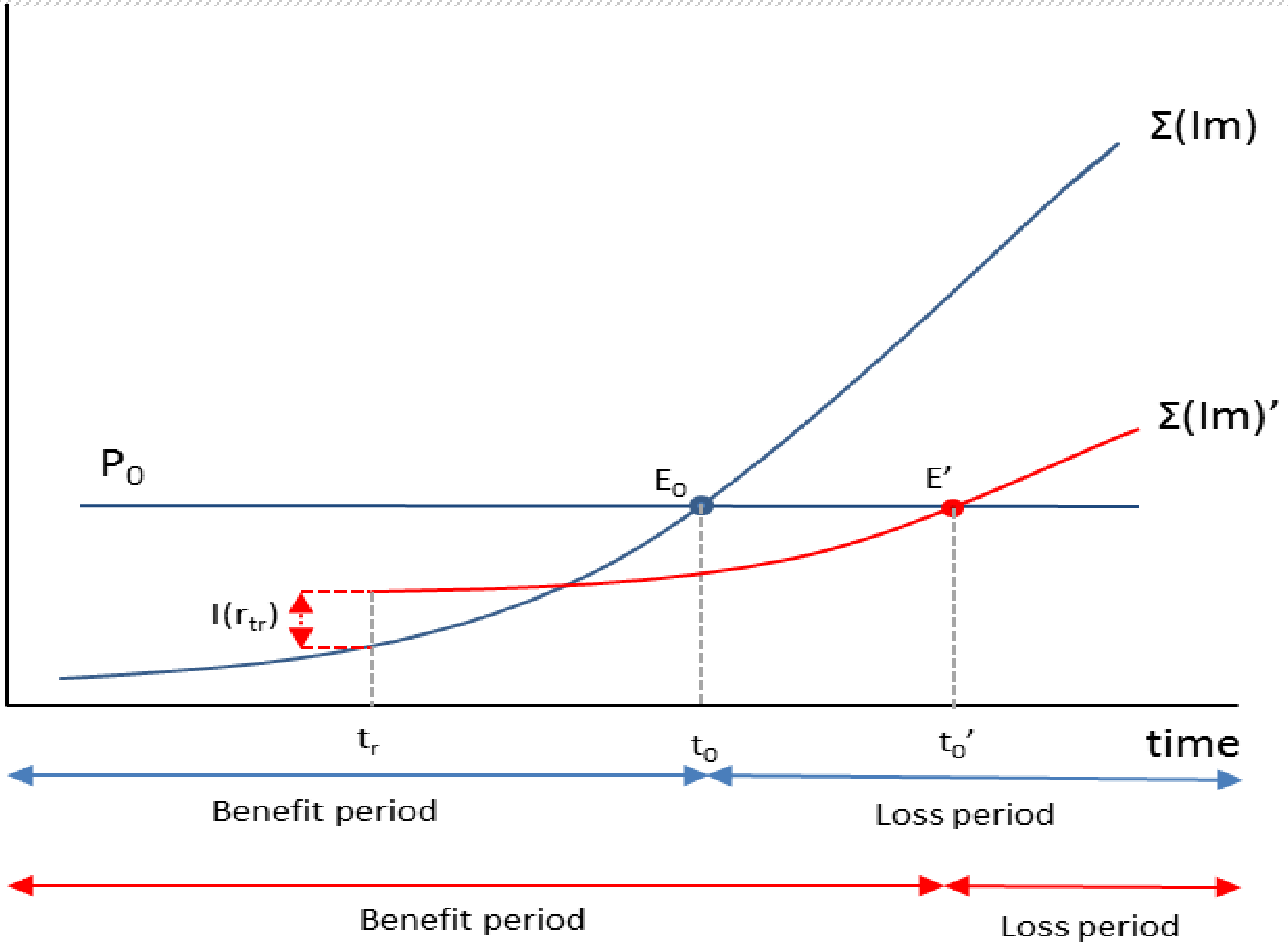
university of groningen

Debt=Principal+Interest



university of groningen

Breaking point: principal vs. interest



university of groningen

Just the code?

Not quite right

- Code
- Requirements
- Architecture
- Design
- Test
- Build
- Documentation
- Infrastructure
- Versioning

…

Technical debt is pervasive

Just the code?

- Code
- Requirements
- Architecture
- Design
- Test
- Build
- Documentation
- Infrastructure
- Versioning

Complex dependencies
Architecture smells
Architecture drift

Just the code?

- Code
- Requirements
- Architecture
- Design
- Test
- Build
- Documentation
- Infrastructure
- Versioning

Low code coverage
Lack of test automation
Residual defects not found

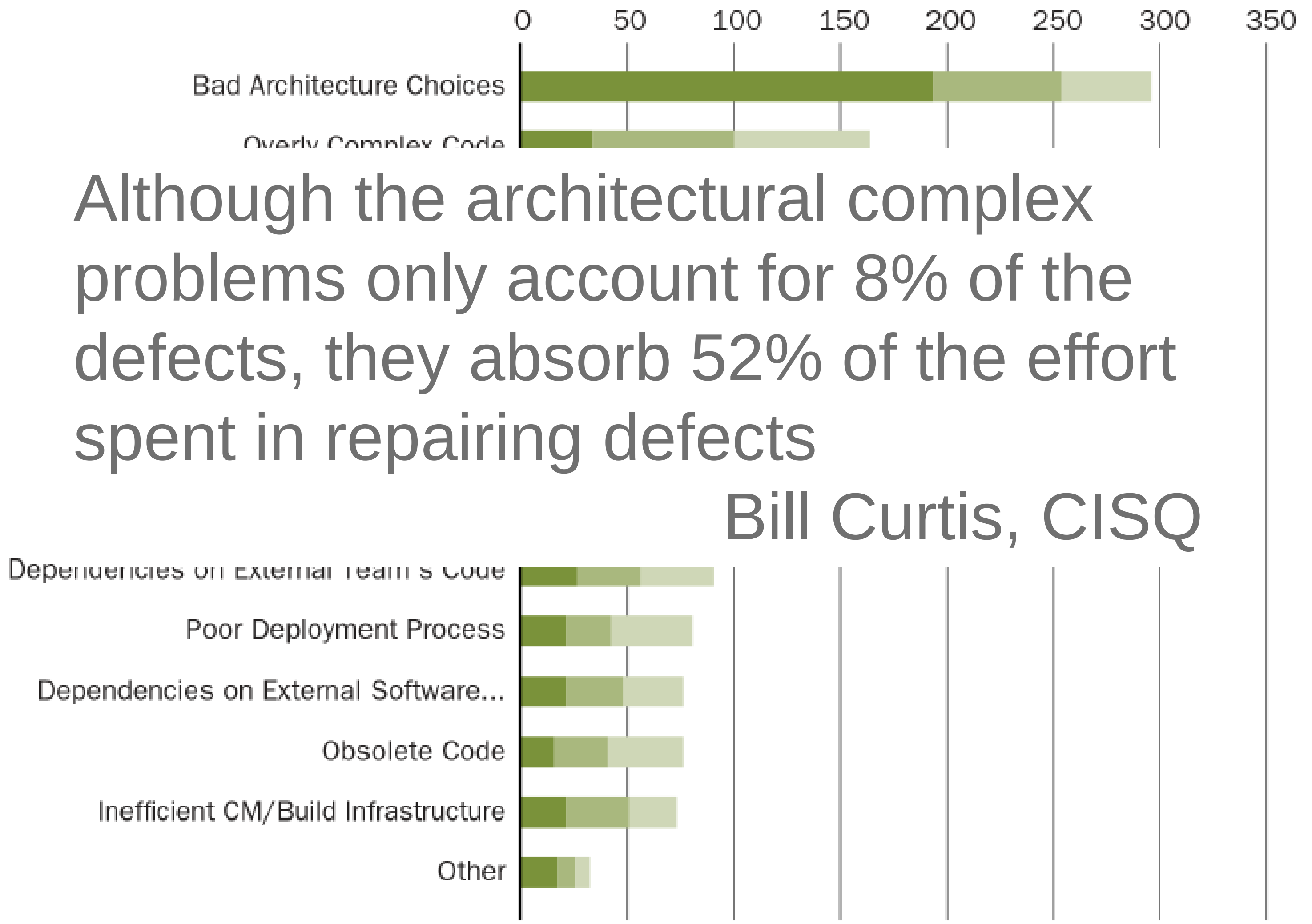
university of groningen

Just the code?

- Code
- Requirements
- Architecture
- Design
- Test
- Build
- Documentation
- Infrastructure
- Versioning

Insufficient/incomplete/out of date
Lack of code comments

Architecture TD is dominant



Although the architectural complex problems only account for 8% of the defects, they absorb 52% of the effort spent in repairing defects

Bill Curtis, CISQ

https://insights.sei.cmu.edu/sei_blog/2015/07/a-field-study-of-technical-debt.html

Debt

Managing TD

- TD prevention
- TD identification
- TD measurement
- TD prioritization
- TD monitoring
- TD repayment
- TD representation/documentation
- TD communication

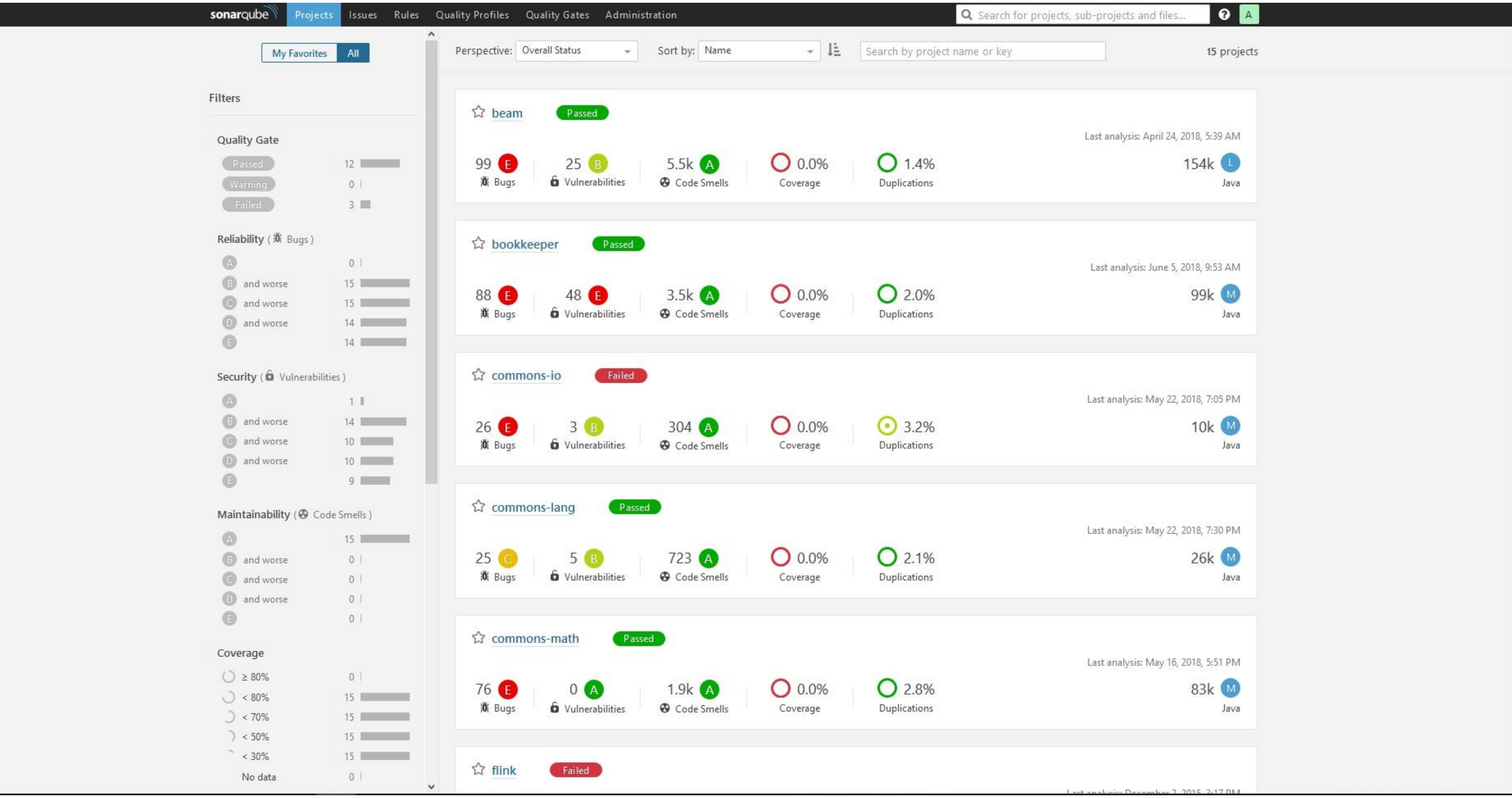
Li et al., Architectural Debt Management in Value-oriented Architecting, Elsevier '14

Managing TD

- TD prevention
- TD identification
- TD measurement
- TD prioritization
- TD monitoring
- TD repayment
- TD representation/documentation
- TD communication

Code analysis
Dependency analysis
Solution comparison
Reverse engineering

TD Identification



sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects, sub-projects and files...

My Favorites | All

Perspective: Overall Status
Sort by: Name
Search by project name or key
15 projects

Filters

Quality Gate

Passed	12
Warning	0
Failed	3

Reliability (Bugs)

A	0
B and worse	15
C and worse	15
D and worse	14
E	14

Security (Vulnerabilities)

A	1
B and worse	14
C and worse	10
D and worse	10
E	9

Maintainability (Code Smells)

A	15
B and worse	0
C and worse	0
D and worse	0
E	0

Coverage

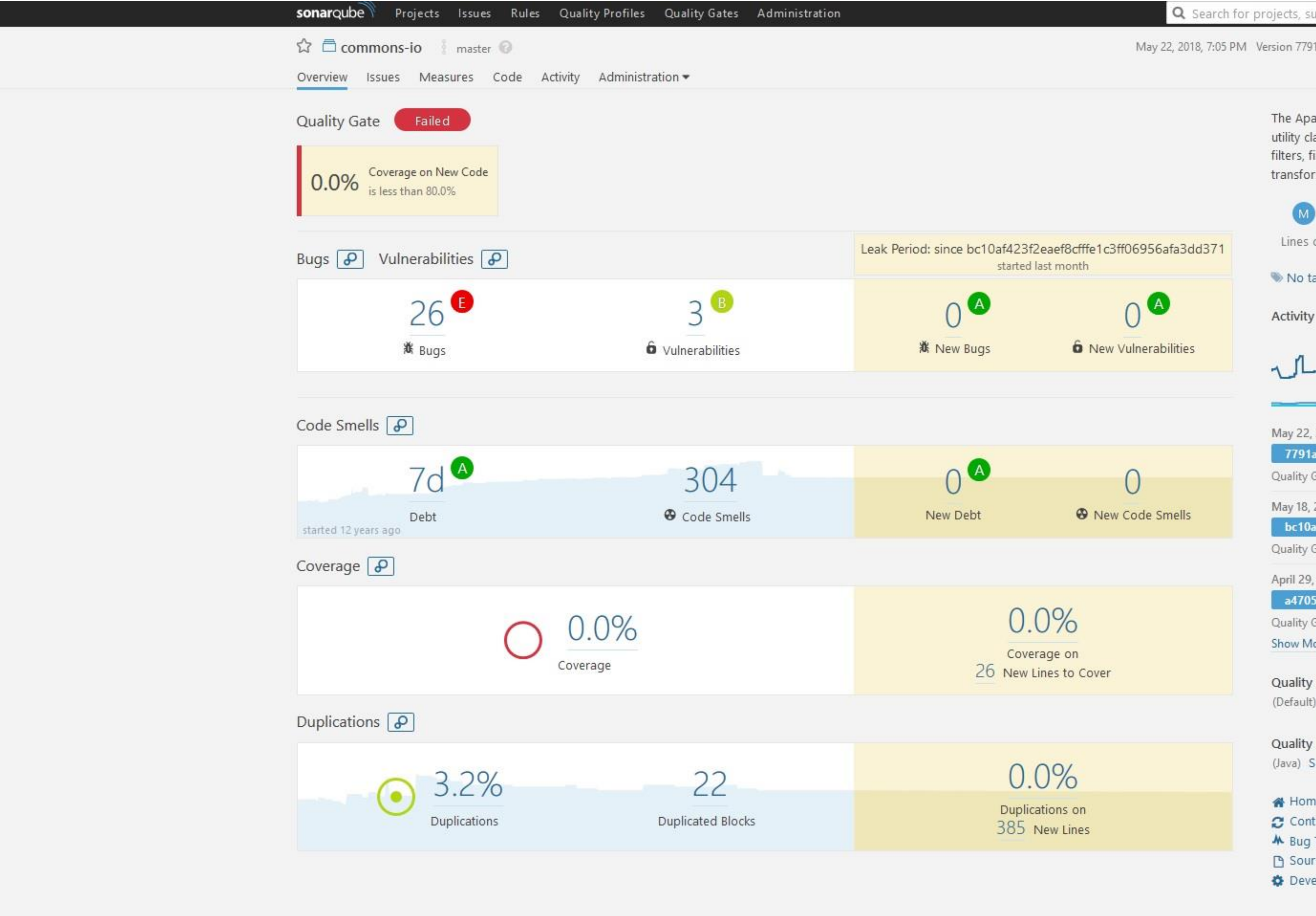
≥ 80%	0
< 80%	15
< 70%	15
< 50%	15
< 30%	15
No data	0

Project	Quality Gate	Last analysis	Bugs	Vulnerabilities	Code Smells	Coverage	Duplications	Size
beam	Passed	April 24, 2018, 5:39 AM	99 E	25 B	5.5k A	0.0%	1.4%	154k L Java
bookkeeper	Passed	June 5, 2018, 9:53 AM	88 E	48 E	3.5k A	0.0%	2.0%	99k M Java
commons-io	Failed	May 22, 2018, 7:05 PM	26 E	3 B	304 A	0.0%	3.2%	10k M Java
commons-lang	Passed	May 22, 2018, 7:30 PM	25 C	5 B	723 A	0.0%	2.1%	26k M Java
commons-math	Passed	May 16, 2018, 5:51 PM	76 E	0 A	1.9k A	0.0%	2.8%	83k M Java
flink	Failed	December 2, 2015, 3:17 PM						

Managing TD

- TD prevention
- TD identification
- TD measurement
- TD prioritization
- TD monitoring
- TD repayment
- TD representation/documentation
- TD communication

Mathematical models
Code metrics
Human estimation



sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

commons-io master

May 22, 2018, 7:05 PM Version 7791

Overview Issues Measures Code Activity Administration

Quality Gate Failed

0.0% Coverage on New Code is less than 80.0%

Bugs Vulnerabilities

Leak Period: since bc10af423f2eaef8cfffe1c3ff06956afa3dd371
started last month

26 E Bugs

3 B Vulnerabilities

0 A New Bugs

0 A New Vulnerabilities

Code Smells

7d A Debt

started 12 years ago

304 Code Smells

0 A New Debt

0 New Code Smells

Coverage

0.0% Coverage

0.0% Coverage on 26 New Lines to Cover

Duplications

3.2% Duplications

22 Duplicated Blocks

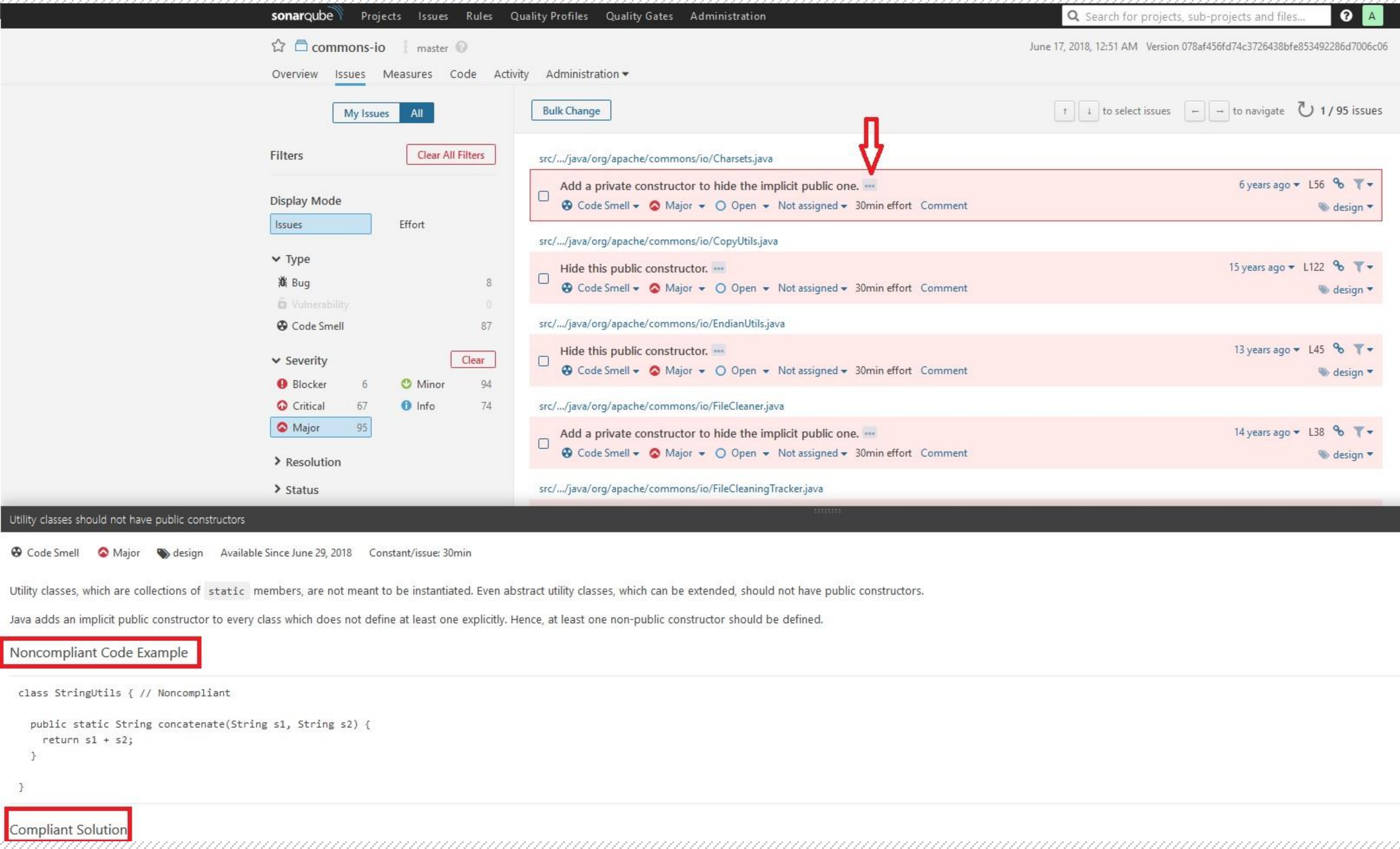
0.0% Duplications on 385 New Lines

Managing TD

- TD prevention
- TD identification
- TD measurement
- TD prioritization
- TD monitoring
- TD repayment
- TD representation/documentation
- TD communication

Refactoring
Automating manual tasks

Refactoring



June 17, 2018, 12:51 AM Version 078af456fd74c3726438bfe853492286d7006c06

1 / 95 issues

Filters

Display Mode: Issues / Effort

Type
- Bug 8
- Vulnerability 0
- Code Smell 87

Severity
- Blocker 6
- Minor 94
- Critical 67
- Info 74
- Major 95

Resolution

Status

src/.../java/org/apache/commons/io/Charsets.java

Add a private constructor to hide the implicit public one. — Code Smell · Major · Open · Not assigned · 30min effort — 6 years ago · L56 · design

src/.../java/org/apache/commons/io/CopyUtils.java

Hide this public constructor. — Code Smell · Major · Open · Not assigned · 30min effort — 15 years ago · L122 · design

src/.../java/org/apache/commons/io/EndianUtils.java

Hide this public constructor. — Code Smell · Major · Open · Not assigned · 30min effort — 13 years ago · L45 · design

src/.../java/org/apache/commons/io/FileCleaner.java

Add a private constructor to hide the implicit public one. — Code Smell · Major · Open · Not assigned · 30min effort — 14 years ago · L38 · design

src/.../java/org/apache/commons/io/FileCleaningTracker.java

Utility classes should not have public constructors

Code Smell · Major · design · Available Since June 29, 2018 · Constant/issue: 30min

Utility classes, which are collections of `static` members, are not meant to be instantiated. Even abstract utility classes, which can be extended, should not have public constructors.

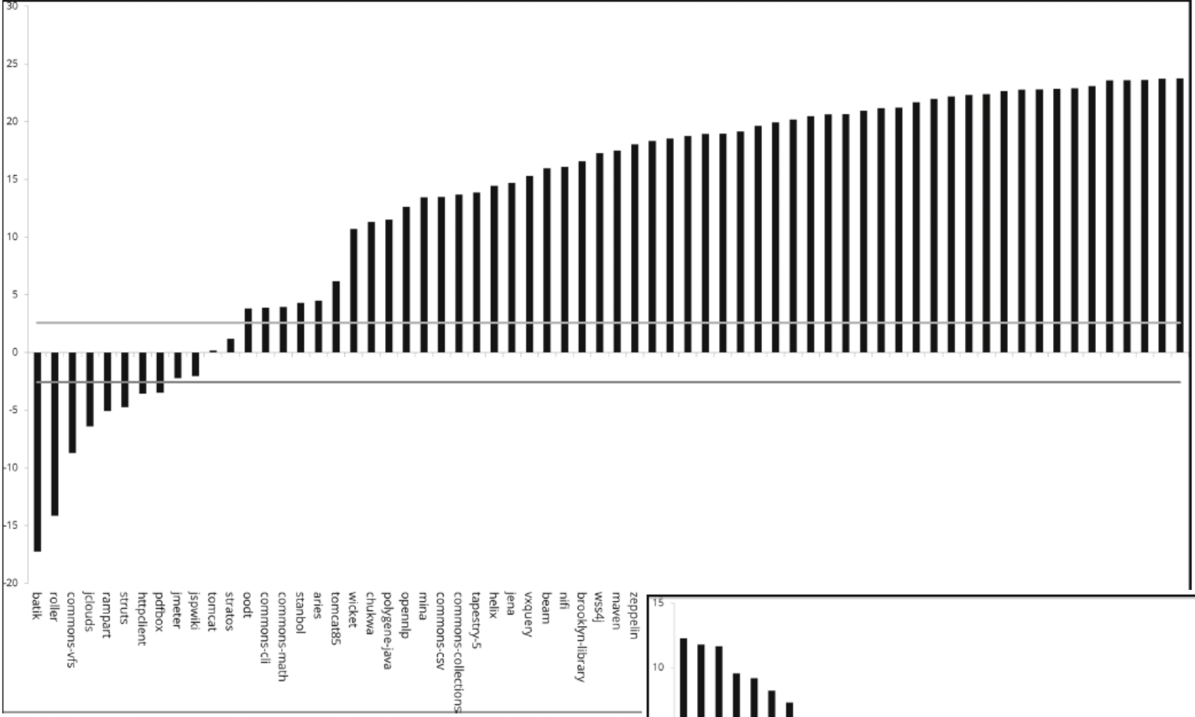
Java adds an implicit public constructor to every class which does not define at least one explicitly. Hence, at least one non-public constructor should be defined.

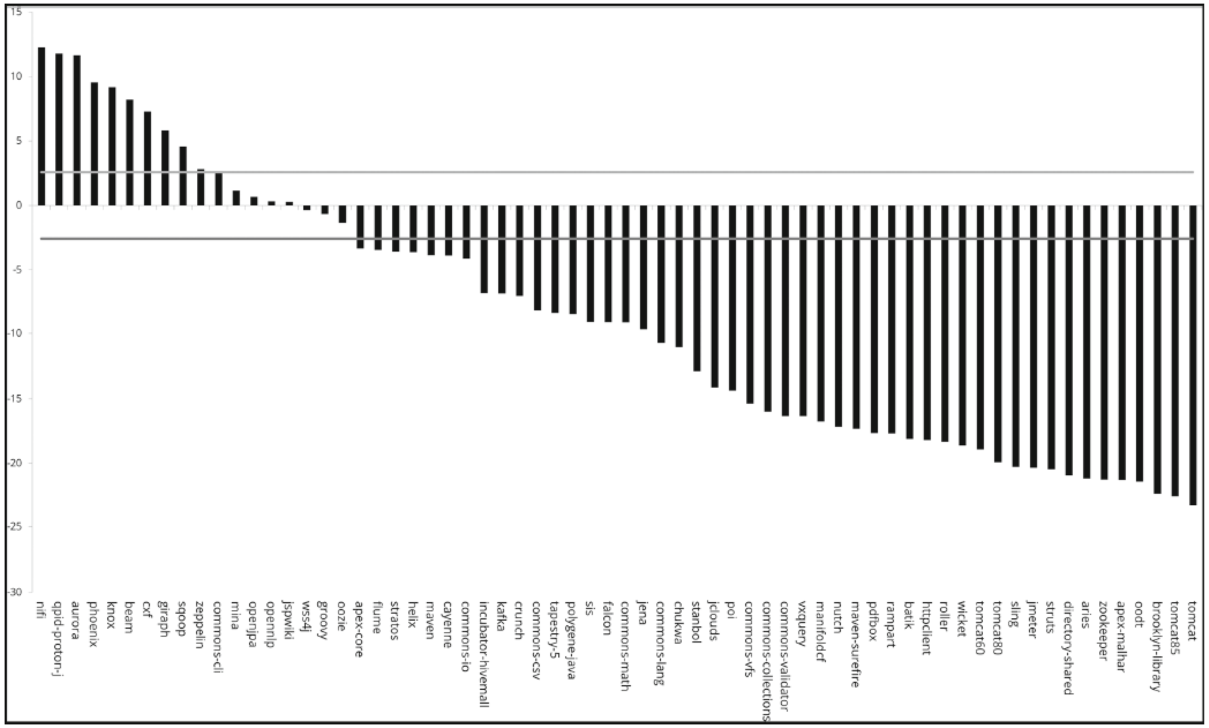
Noncompliant Code Example

```
class StringUtils { // Noncompliant

  public static String concatenate(String s1, String s2) {
    return s1 + s2;
  }

}
```

Compliant Solution

Repaying TD





Digkas et al., The Evolution of TD in the Apache Ecosystem, ECSA ‘17

university of groningen

Repaying TD

- Large variation in survivability of issues
  - 10% fixed within the first month
  - 50% in the first year
  - Some take up to ten years
- Very few issues types with fixing rate >50%
- Duplication and exception handling
  - Frequently encountered
  - Rarely fixed

Digkas et al., How Do Developers Fix Issues and Pay Back TD in the Apache Ecosystem, SANER '18

Outline

- Introducing the metaphor
- Emergence of TD
- Concepts of TD and management
- **Present and Future**

university of groningen

Short deadline
vs.
Long-term sustainability

SW Engineers don’t like TD

Managers don’t mind TD

Communication bridge
Investment opportunity

State of the art

- Whole lifecycle but mostly code and design
- Basic concepts are mature
- Tooling (industrial & prototypes)
- Economic theories

State of practice

SW engineers

- Understand the concept and challenges
- Deal with it during maintenance
- TD management in place but with constraints
  - Resource-intensive
  - Realistically only a portion managed

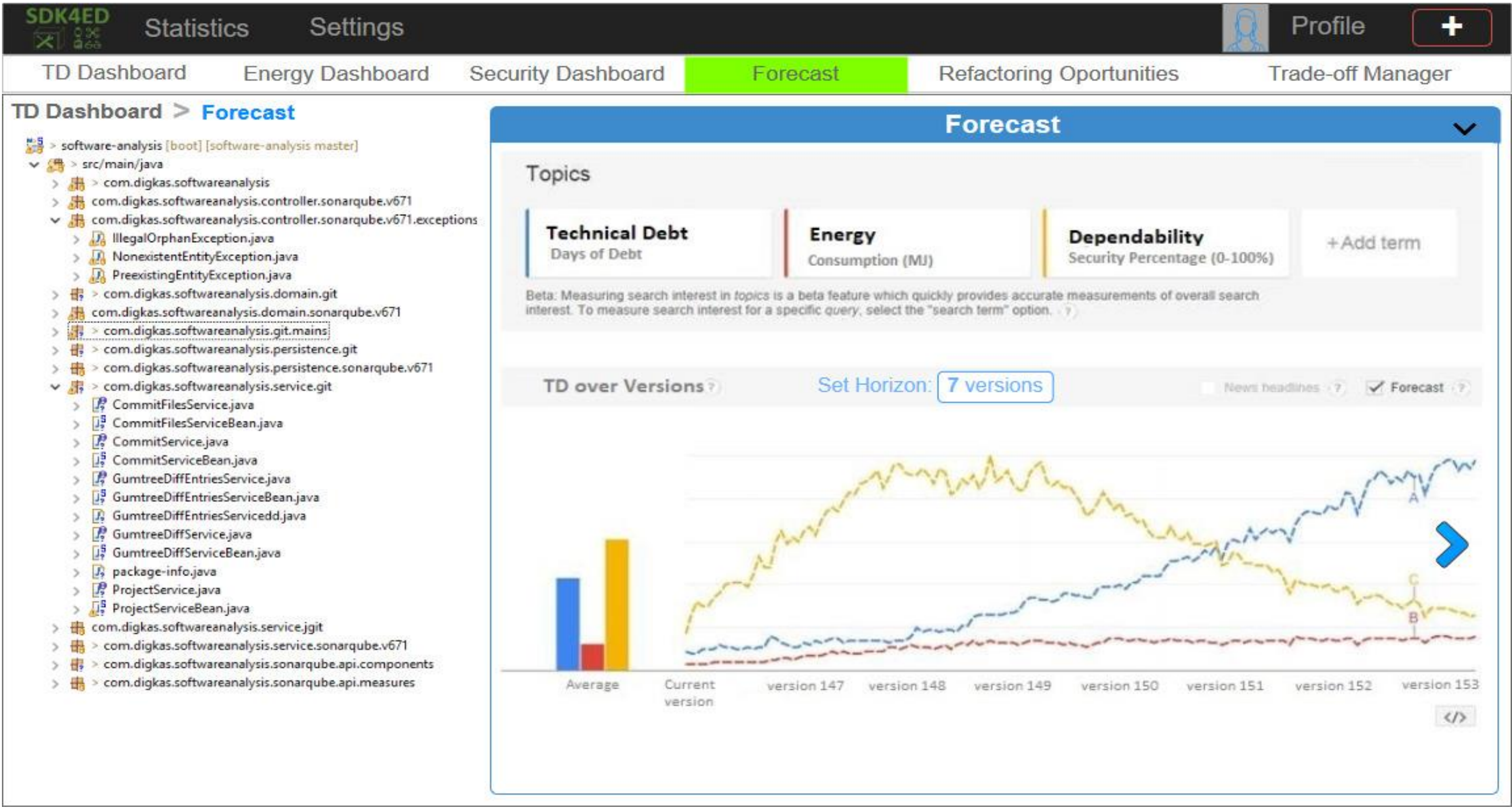
university of groningen

Interplay between qualities

- Theory: Qualities studied as islands
- Practice: Qualities interplay
    - Run-time vs. design time
- Communities needs to interact
- Interoperability
    - Methods and tools

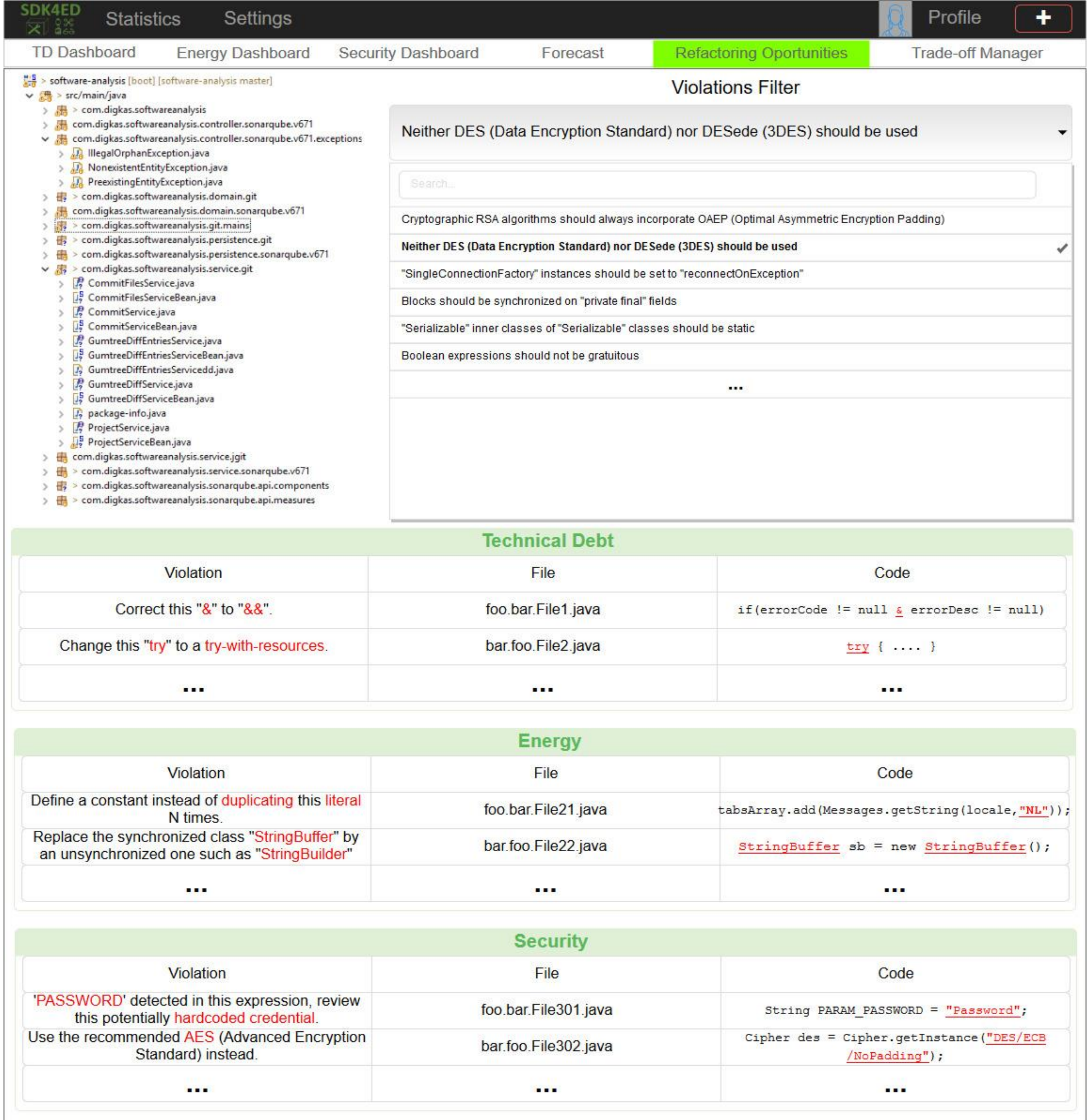
SDK4ED

https://sdk4ed.eu/



```
> software-analysis [boot] [software-analysis master]
  > src/main/java
    > com.digkas.softwareanalysis
    com.digkas.softwareanalysis.controller.sonarqube.v671
    com.digkas.softwareanalysis.controller.sonarqube.v671.exceptions
      IllegalOrphanException.java
      NonexistentEntityException.java
      PreexistingEntityException.java
    > com.digkas.softwareanalysis.domain.git
    com.digkas.softwareanalysis.domain.sonarqube.v671
    > com.digkas.softwareanalysis.git.mains
    > com.digkas.softwareanalysis.persistence.git
    > com.digkas.softwareanalysis.persistence.sonarqube.v671
    > com.digkas.softwareanalysis.service.git
      CommitFilesService.java
      CommitFilesServiceBean.java
      CommitService.java
      CommitServiceBean.java
      GumtreeDiffEntriesService.java
      GumtreeDiffEntriesServiceBean.java
      GumtreeDiffEntriesServicedd.java
      GumtreeDiffService.java
      GumtreeDiffServiceBean.java
      package-info.java
      ProjectService.java
      ProjectServiceBean.java
    com.digkas.softwareanalysis.service.jgit
    > com.digkas.softwareanalysis.service.sonarqube.v671
    > com.digkas.softwareanalysis.sonarqube.api.components
    > com.digkas.softwareanalysis.sonarqube.api.measures
```

Beta: Measuring search interest in *topics* is a beta feature which quickly provides accurate measurements of overall search interest. To measure search interest for a specific *query*, select the "search term" option.



```
software-analysis [boot] [software-analysis master]
  src/main/java
    > com.digkas.softwareanalysis
    com.digkas.softwareanalysis.controller.sonarqube.v671
    com.digkas.softwareanalysis.controller.sonarqube.v671.exceptions
      > IllegalOrphanException.java
      > NonexistentEntityException.java
      > PreexistingEntityException.java
    > com.digkas.softwareanalysis.domain.git
    com.digkas.softwareanalysis.domain.sonarqube.v671
    > com.digkas.softwareanalysis.git.mains
    > com.digkas.softwareanalysis.persistence.git
    > com.digkas.softwareanalysis.persistence.sonarqube.v671
    > com.digkas.softwareanalysis.service.git
      CommitFilesService.java
      CommitFilesServiceBean.java
      CommitService.java
      CommitServiceBean.java
      GumtreeDiffEntriesService.java
      GumtreeDiffEntriesServiceBean.java
      GumtreeDiffEntriesServicedd.java
      GumtreeDiffService.java
      GumtreeDiffServiceBean.java
      package-info.java
      ProjectService.java
      ProjectServiceBean.java
    com.digkas.softwareanalysis.service.jgit
    > com.digkas.softwareanalysis.service.sonarqube.v671
    > com.digkas.softwareanalysis.sonarqube.api.components
    > com.digkas.softwareanalysis.sonarqube.api.measures
```

Violations Filter

Neither DES (Data Encryption Standard) nor DESede (3DES) should be used

- Cryptographic RSA algorithms should always incorporate OAEP (Optimal Asymmetric Encryption Padding)
- **Neither DES (Data Encryption Standard) nor DESede (3DES) should be used** ✓
- "SingleConnectionFactory" instances should be set to "reconnectOnException"
- Blocks should be synchronized on "private final" fields
- "Serializable" inner classes of "Serializable" classes should be static
- Boolean expressions should not be gratuitous
- ...

Technical Debt

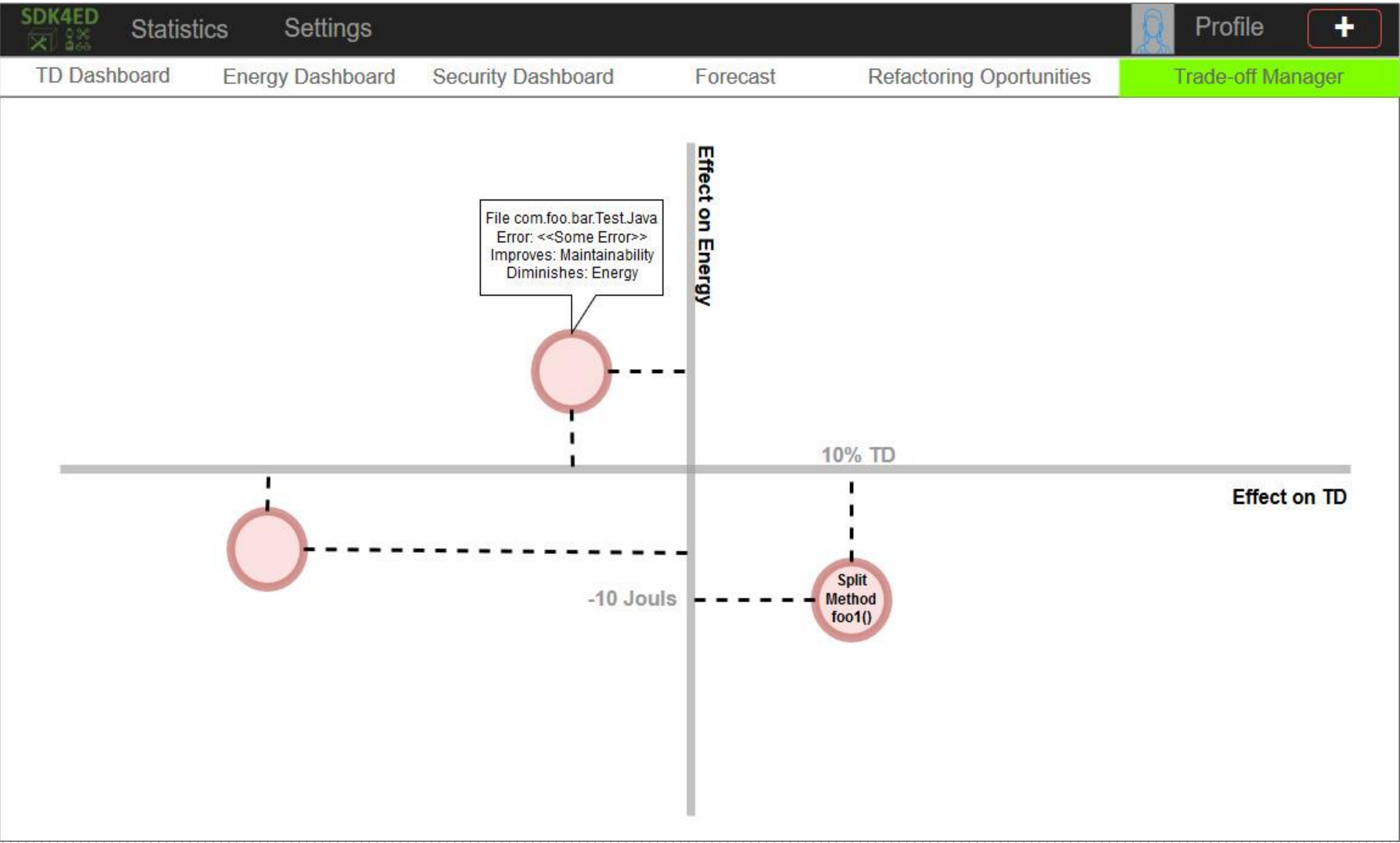
Violation	File	Code
Correct this "&" to "&&".	foo.bar.File1.java	`if(errorCode != null & errorDesc != null)`
Change this "try" to a try-with-resources.	bar.foo.File2.java	`try { .... }`
...	...	...

Energy

Violation	File	Code
Define a constant instead of duplicating this literal N times.	foo.bar.File21.java	`tabsArray.add(Messages.getString(locale,"NL"));`
Replace the synchronized class "StringBuffer" by an unsynchronized one such as "StringBuilder"	bar.foo.File22.java	`StringBuffer sb = new StringBuffer();`
...	...	...

Security

Violation	File	Code
'PASSWORD' detected in this expression, review this potentially hardcoded credential.	foo.bar.File301.java	`String PARAM_PASSWORD = "Password";`
Use the recommended AES (Advanced Encryption Standard) instead.	bar.foo.File302.java	`Cipher des = Cipher.getInstance("DES/ECB/NoPadding");`
...	...	...



File com.foo.bar.Test.Java
Error: <<Some Error>>
Improves: Maintainability
Diminishes: Energy

Join the community

- Bridging the gap between research and practice
- Join efforts

Tech Debt conf @

university of groningen

Thank you

Credits:

Philippe Kruchten

Robert Nord

Ipek Ozkaya

Carolyn Seaman

Zengyang Li

Peng Liang

Areti Ampatzoglou

Apostolos Ampatzoglou

Alexander Chatzigeorgiou