ThoughtWorks®

FINDING TOXIC CODE

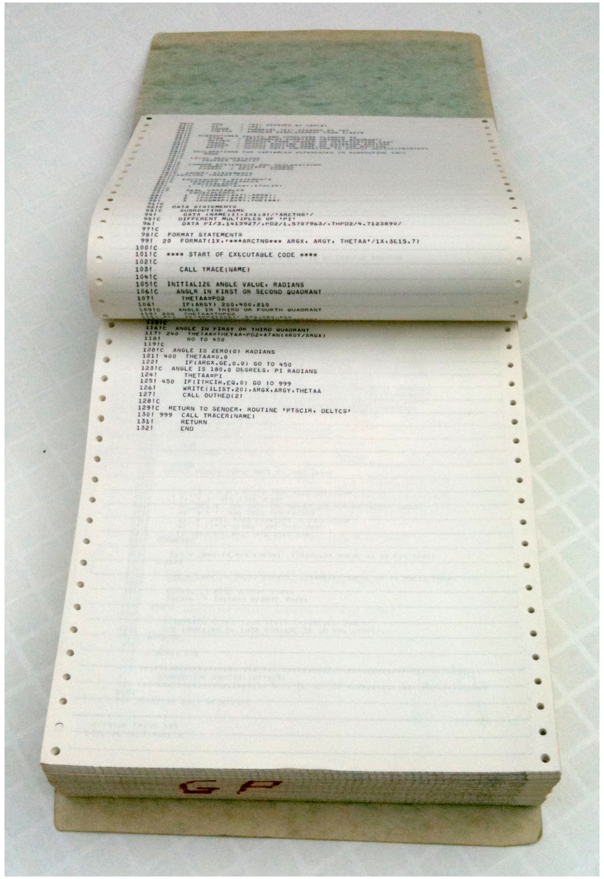
Experiences and techniques for finding dangerous code in large multi-language codebases

Kornelis (Korny) Sietsma - @kornys on Twitter

WHO AM I?

```
80 PRINT""
LIST
10 PRINT "VINTAGE COMPUTER FESTI
VAL"
20 PRINT ""
30 PRINT "     E U R O P E"
40 PRINT ""
50 PRINT "    2012 EDITION"
60 PRINT ""
70 PRINT "COMPUTER MUSEUM ITALY"

80 PRINT ""
 14010
RUN
Error 40 on line 9999 : END
PRINT 'CIAO ALBERTO'_
```



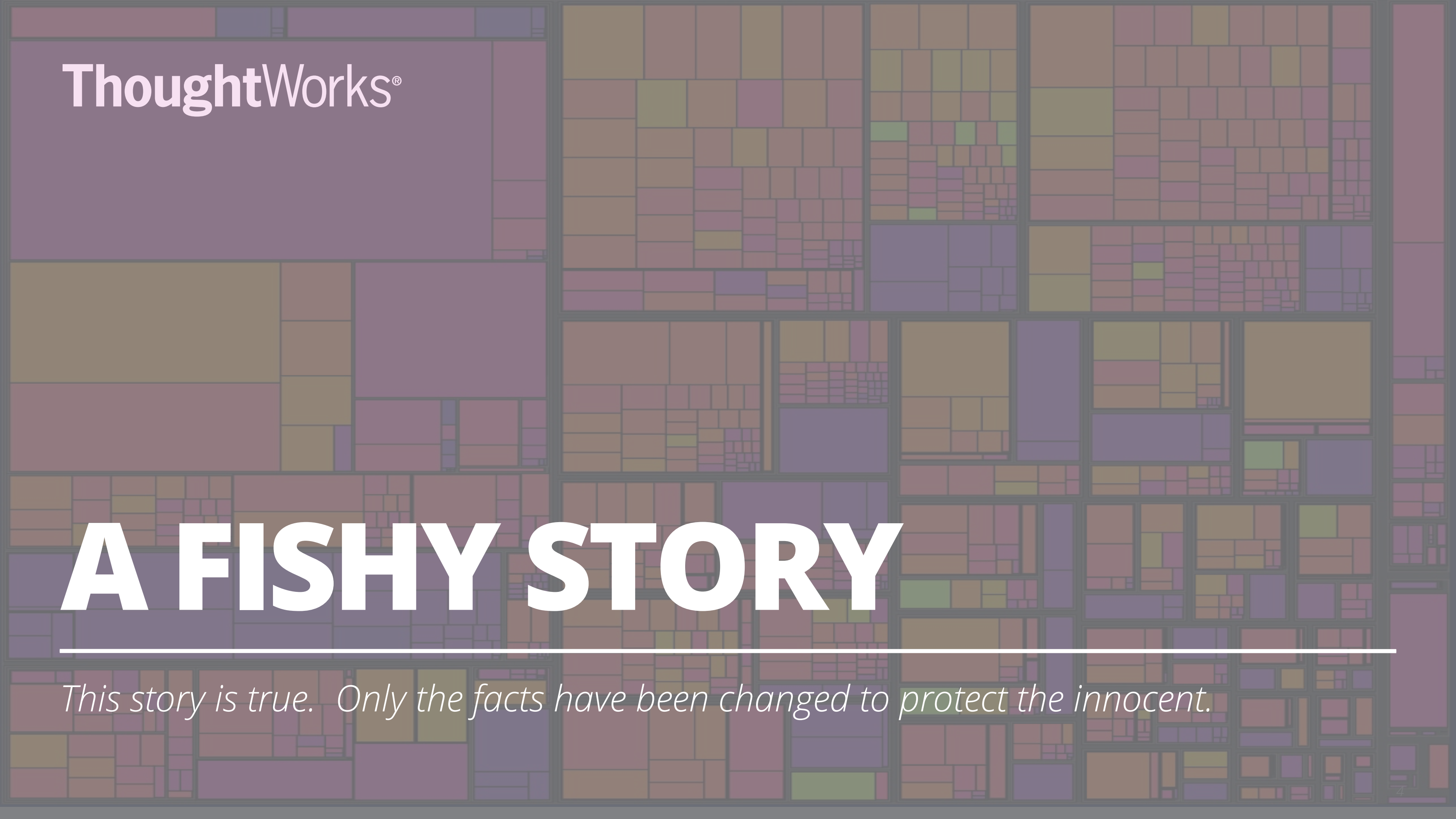
```
C  DATA STATEMENTS
C    SUBROUTINE NAME
       DATA (NAME(I),I=1,3)/'ARCTNG'/
C    DIFFERENT MULTIPLES OF 'PI'
       DATA PI/3.1415927/,PD2/1.5707963/,THPD2/4.7123890/
C
C  FORMAT STATEMENTS
   20  FORMAT(1X,'***ARCTNG*** ARGX, ARGY, THETAA'/1X,3E15.7)
C
C  **** START OF EXECUTABLE CODE ****
C
       CALL TRACE(NAME)
C
C  INITIALIZE ANGLE VALUE, RADIANS
C    ANGLR IN FIRST OR SECOND QUADRANT
       THETAA=PD2
       IF(ARGY) 200,400,210
C    ANGLE IN THIRD OR FOURTH QUADRANT
  200  THETAA=THPD2
C
C  ANGLE IN FIRST OR THIRD QUADRANT
  240  THETAA=THETAA-PD2+ATAN(ARGY/ARGX)
       GO TO 450
C
C  ANGLE IS ZERO(0) RADIANS
  400  THETAA=0.0
       IF(ARGX.GE.0.0) GO TO 450
C  ANGLE IS 180.0 DEGREES, PI RADIANS
       THETAA=PI
  450  IF(ITRCIR.EQ.0) GO TO 999
       WRITE(ILIST,20),ARGX,ARGY,THETAA
       CALL OUTHED(2)
C
C  RETURN TO SENDER, ROUTINE 'PTSCIR, DELTCS'
  999  CALL TRACER(NAME)
       RETURN
       END
```

WHAT DO I DO NOW?

ThoughtWorks®

Consulting, Delivery, Agile, Technical excellence

And the occasional “Help us work out what is going wrong” project.



ThoughtWorks®

A FISHY STORY

This story is true. Only the facts have been changed to protect the innocent.

FISHCORP HAD A PROBLEM

Old “FishNet” system – ugly and hard to change.

New dev manager – Mr Squid;

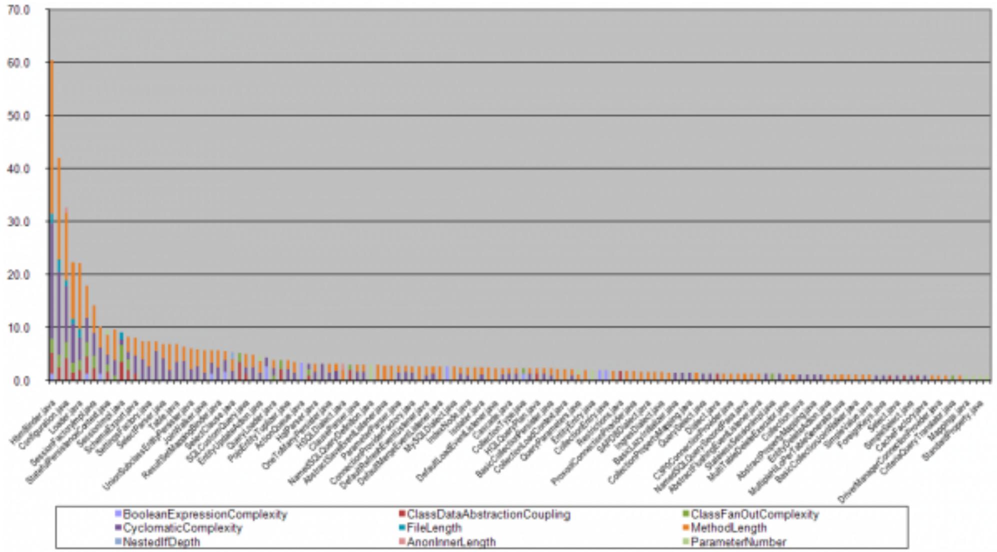
New system: “SquidNet” – very pretty, but very very buggy, late to ship, and getting later.

“Can you help us work out what is going wrong?”

YOU HAVE TWO WEEKS

- Workshops
- Whiteboard sessions
- Process mapping
- 1 million lines of code!
  - How do we quickly review 1 million lines of code?
  - C++, C#, JS, SQL stored procedures...

"TOXIC" CODE - ERIK DÖRNENBURG BLOG 2008

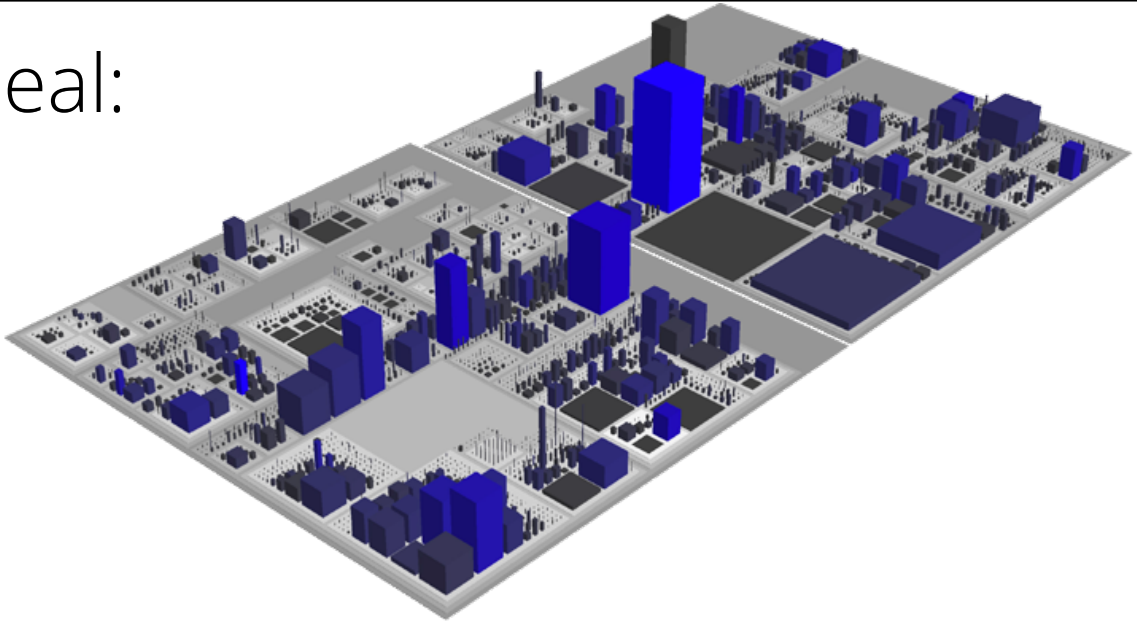


Metric	Level	Threshold
File Length	file	500
Class Fan-Out Complexity	class	30
Class Data Abstraction Coupling	class	10
Anon Inner Length	inner class	35
Method Length	method	30
Parameter Number	method	6
Cyclomatic Complexity	method	10
Nested If Depth	statement	3
Nested Try Depth	statement	2
Boolean Expression Complexity	statement	3
Missing Switch Default	statement	1

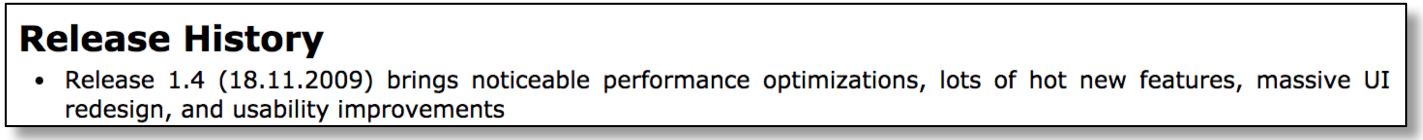
https://erik.doernenburg.com/2008/11/how-toxic-is-your-code/

CODECITY

Looks ideal:



But…

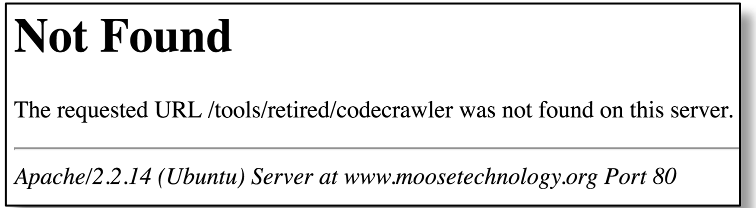


Release History

- Release 1.4 (18.11.2009) brings noticeable performance optimizations, lots of hot new features, massive UI redesign, and usability improvements

GRAVEYARD OF TOOLS

CodeCrawler:

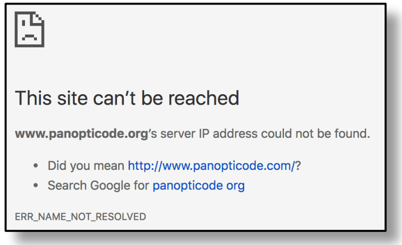


Not Found

The requested URL /tools/retired/codecrawler was not found on this server.

Apache/2.2.14 (Ubuntu) Server at www.moosetechnology.org Port 80

Panopticode:



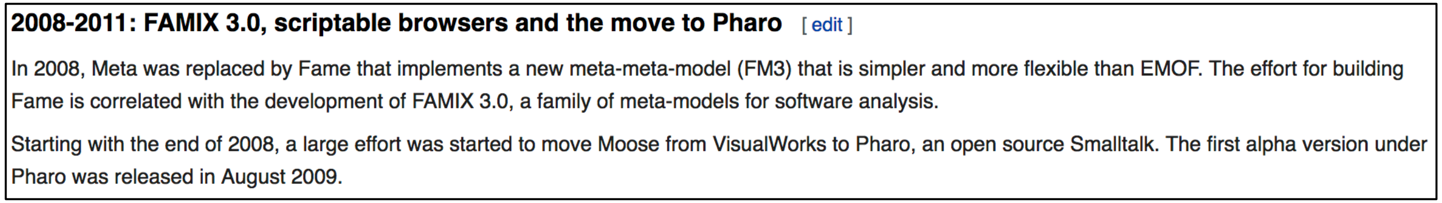
This site can't be reached

www.panopticode.org's server IP address could not be found.

- Did you mean http://www.panopticode.com/?
- Search Google for panopticode org

ERR_NAME_NOT_RESOLVED

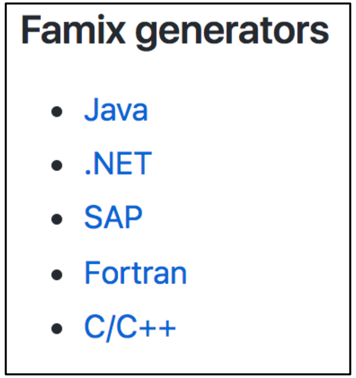
Moose Technology:



2008-2011: FAMIX 3.0, scriptable browsers and the move to Pharo [edit]

In 2008, Meta was replaced by Fame that implements a new meta-meta-model (FM3) that is simpler and more flexible than EMOF. The effort for building Fame is correlated with the development of FAMIX 3.0, a family of meta-models for software analysis.

Starting with the end of 2008, a large effort was started to move Moose from VisualWorks to Pharo, an open source Smalltalk. The first alpha version under Pharo was released in August 2009.



Famix generators

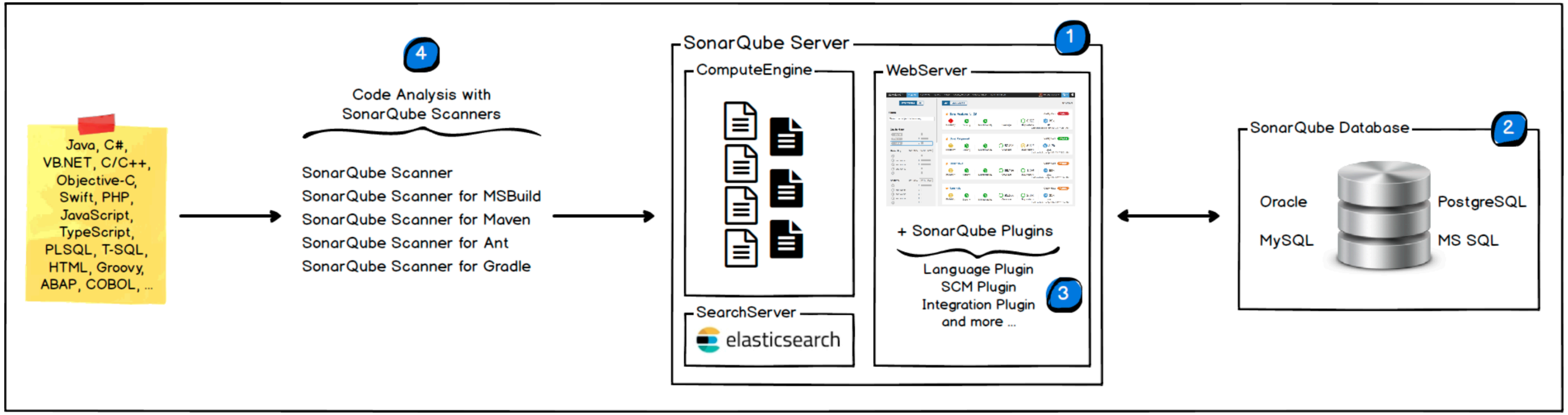
- Java
- .NET
- SAP
- Fortran
- C/C++

WHAT ABOUT SONARQUBE?

Architecture

The SonarQube Platform is made of 4 components:

1. One **SonarQube Server** starting 3 main processes:
    a. a **Web Server** for developers, managers to browse quality snapshots and configure the SonarQube instance
    b. a **Search Server** based on Elasticsearch to back searches from the UI
    c. a **Compute Engine Server** in charge of processing code analysis reports and saving them in the SonarQube Database
2. One **SonarQube Database** to store:
    - the configuration of the SonarQube instance (security, plugins settings, etc.)
    - the quality snapshots of projects, views, etc.
3. Multiple **SonarQube Plugins** installed on the server, possibly including language, SCM, integration, authentication, and governance plugins
4. One or more **SonarQube Scanners** running on your Build / Continuous Integration Servers to analyze projects

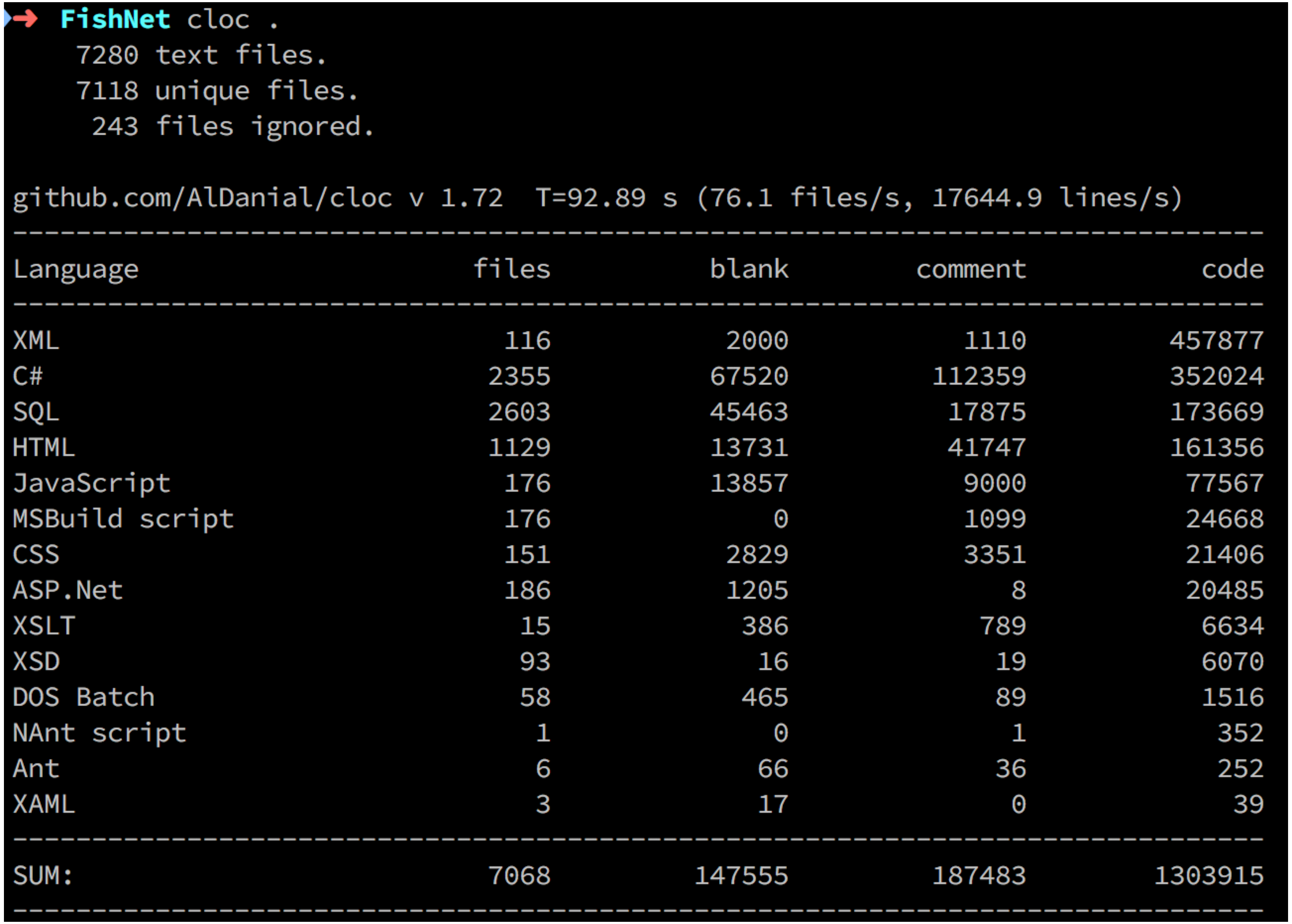


Java, C#, VB.NET, C/C++, Objective-C, Swift, PHP, JavaScript, TypeScript, PLSQL, T-SQL, HTML, Groovy, ABAP, COBOL, ...
4
Code Analysis with SonarQube Scanners
SonarQube Scanner
SonarQube Scanner for MSBuild
SonarQube Scanner for Maven
SonarQube Scanner for Ant
SonarQube Scanner for Gradle
SonarQube Server
1
ComputeEngine
WebServer
+ SonarQube Plugins
Language Plugin
SCM Plugin
Integration Plugin
and more ...
3
SearchServer
elasticsearch
SonarQube Database
2
Oracle
MySQL
PostgreSQL
MS SQL

HOW ABOUT REALLY LIGHTWEIGHT TOOLS?

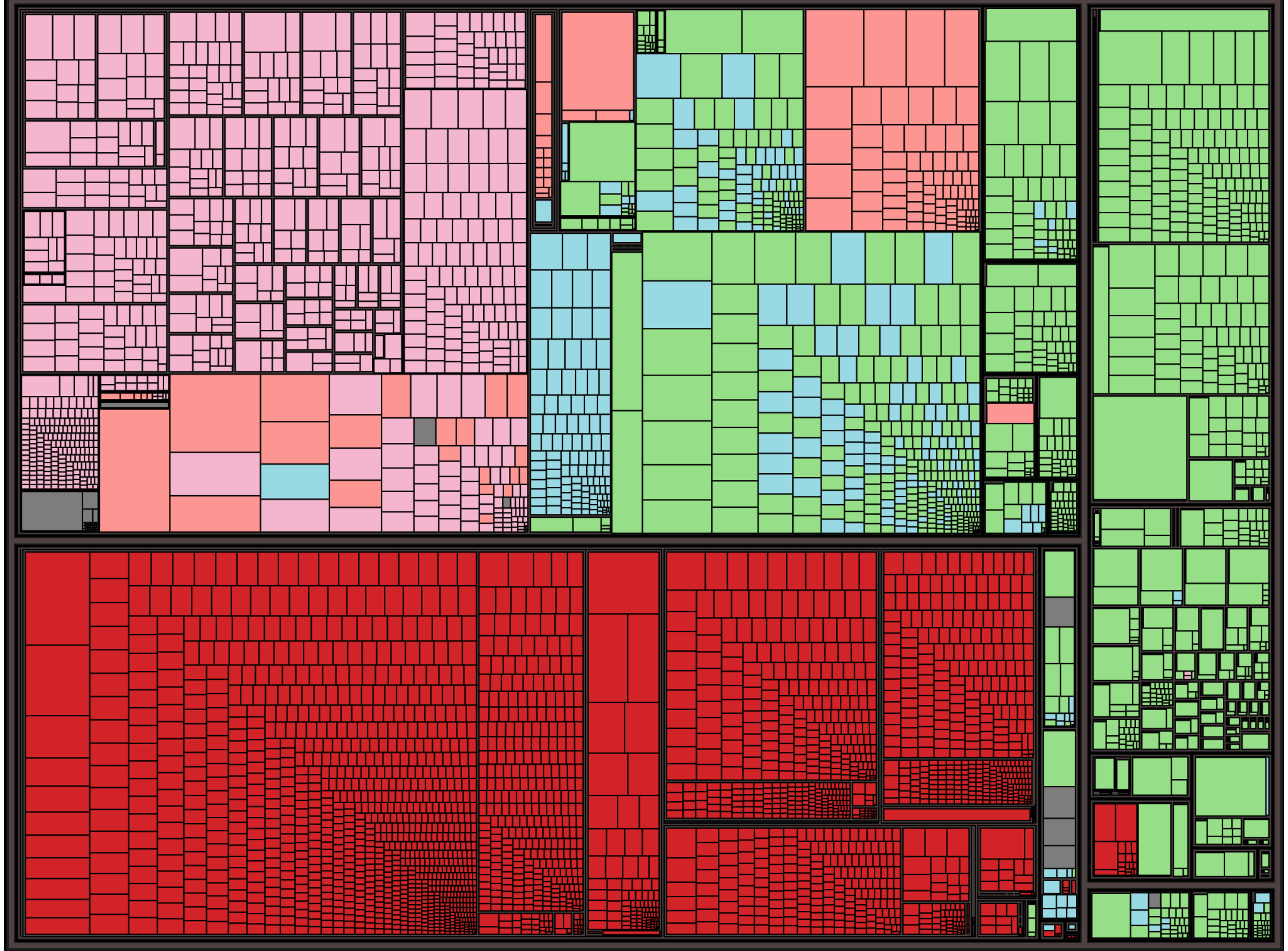
Something quick, simple, cross-language, works with just source code.

What about `CLOC`?

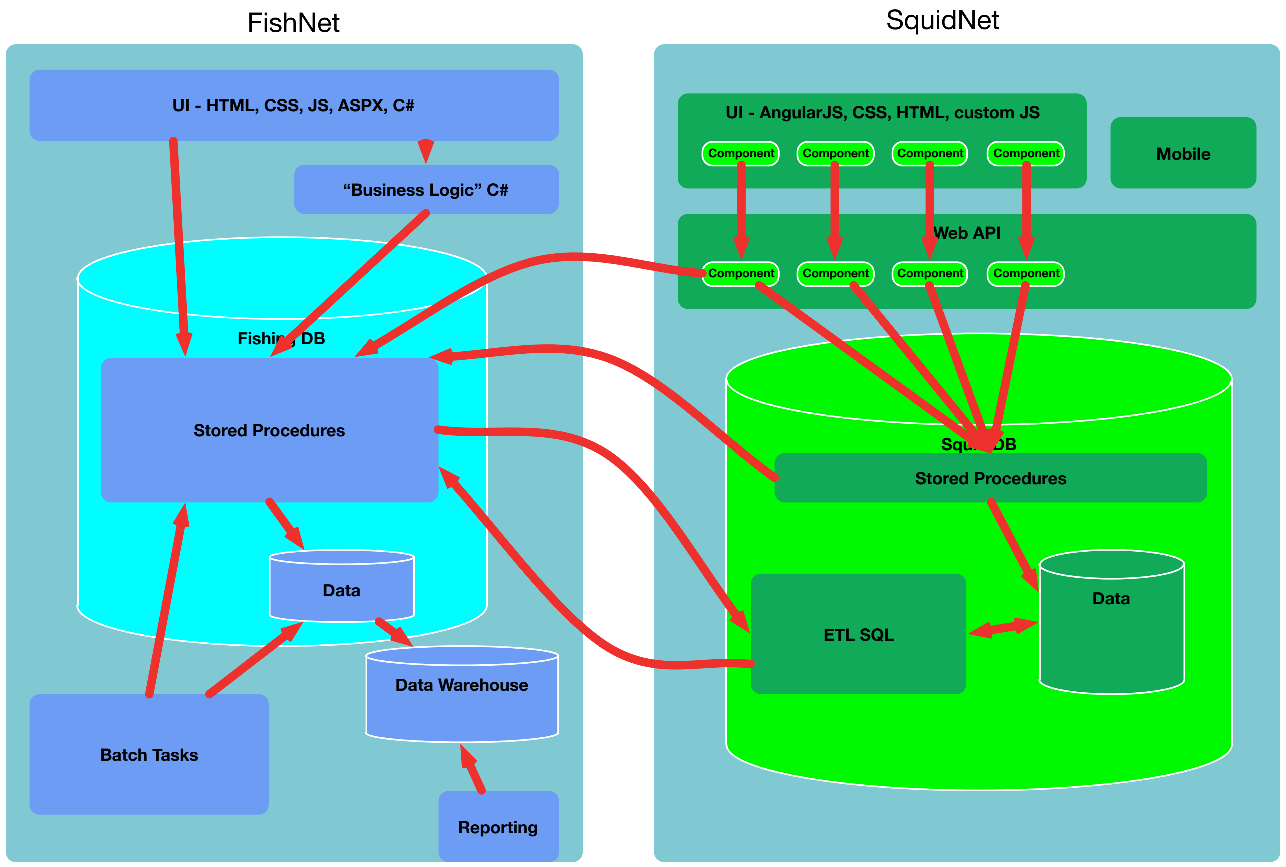


```
➜ FishNet cloc .
    7280 text files.
    7118 unique files.
     243 files ignored.

github.com/AlDanial/cloc v 1.72  T=92.89 s (76.1 files/s, 17644.9 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
XML                            116           2000           1110         457877
C#                            2355          67520         112359         352024
SQL                           2603          45463          17875         173669
HTML                          1129          13731          41747         161356
JavaScript                     176          13857           9000          77567
MSBuild script                 176              0           1099          24668
CSS                            151           2829           3351          21406
ASP.Net                        186           1205              8          20485
XSLT                            15            386            789           6634
XSD                             93             16             19           6070
DOS Batch                       58            465             89           1516
NAnt script                      1              0              1            352
Ant                              6             66             36            252
XAML                             3             17              0             39
-------------------------------------------------------------------------------
SUM:                          7068         147555         187483        1303915
-------------------------------------------------------------------------------
```



ARCHITECTURE



HOW BIG IS TOO BIG?

"Simply stated, an object should be no bigger than the size of my head when pressed up against the monitor – basically a screenful of code."

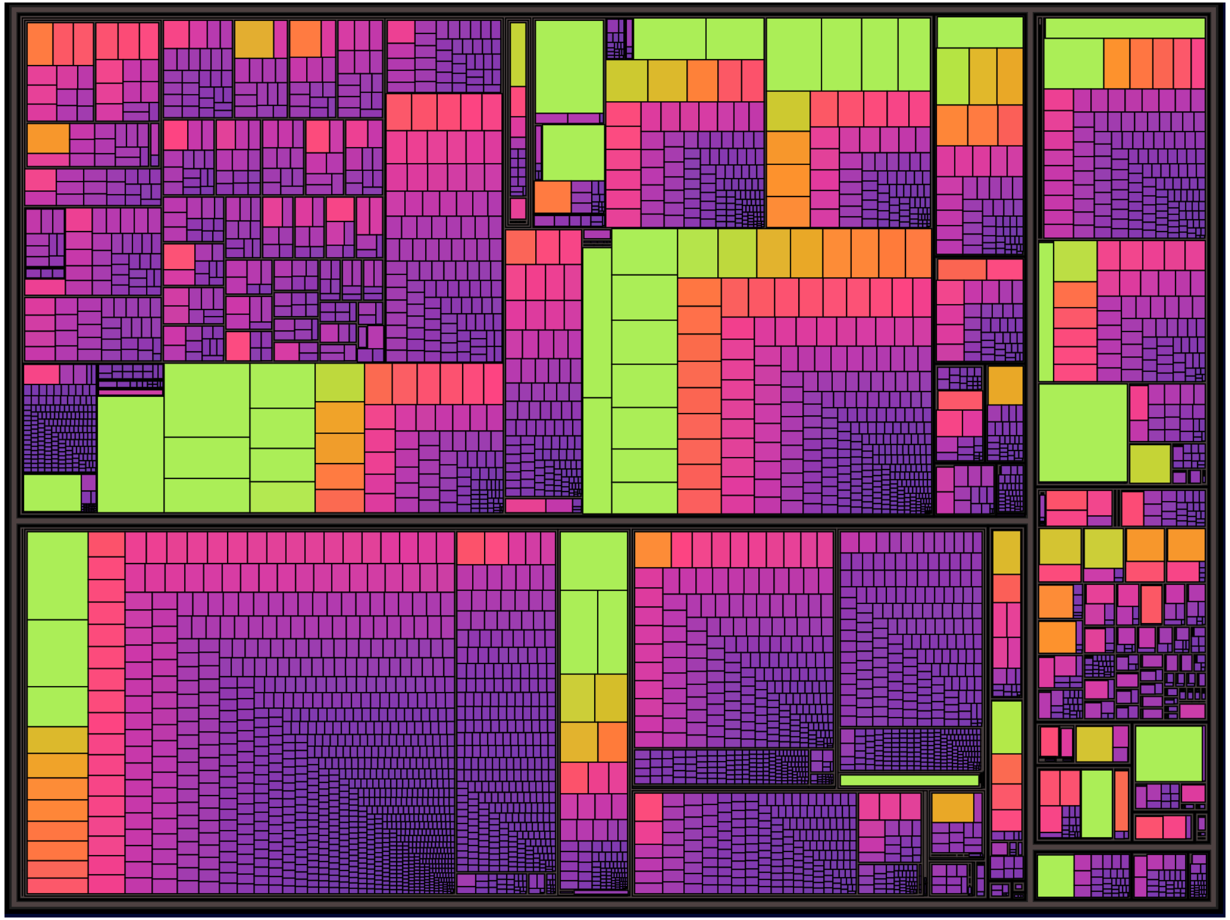
- James Lewis (@boicy)

http://bovon.org/archives/350

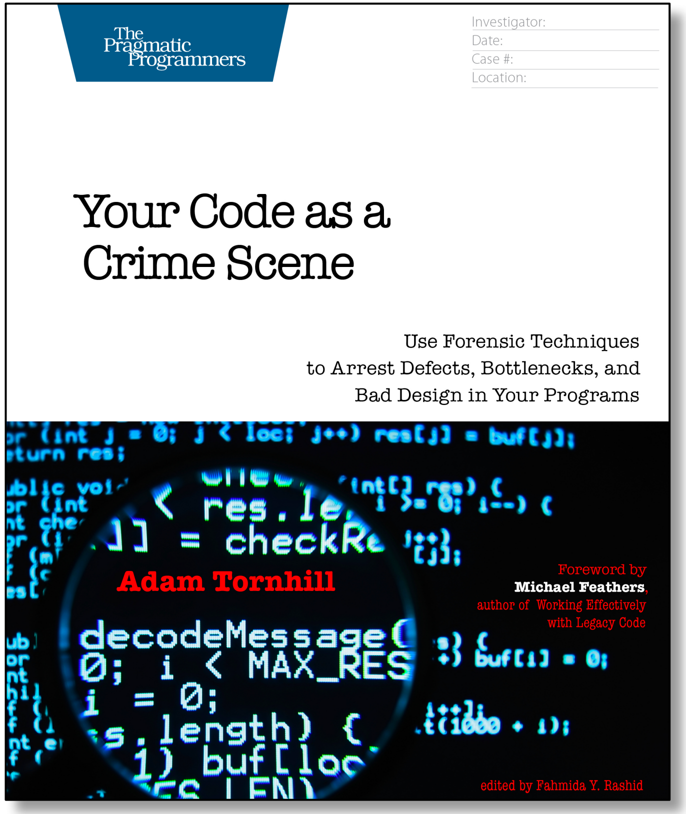
Simple cross-language code smell 1:

Too many lines of code

(LINES OF CODE - BETTER VIEWED IN THE APP!)



CODE-MAAT – SCM-BASED INFORMATION


The Pragmatic Programmers

Investigator:
Date:
Case #:
Location:

Your Code as a Crime Scene

Use Forensic Techniques to Arrest Defects, Bottlenecks, and Bad Design in Your Programs

Adam Tornhill

Foreword by **Michael Feathers**, author of Working Effectively with Legacy Code

edited by Fahmida Y. Rashid

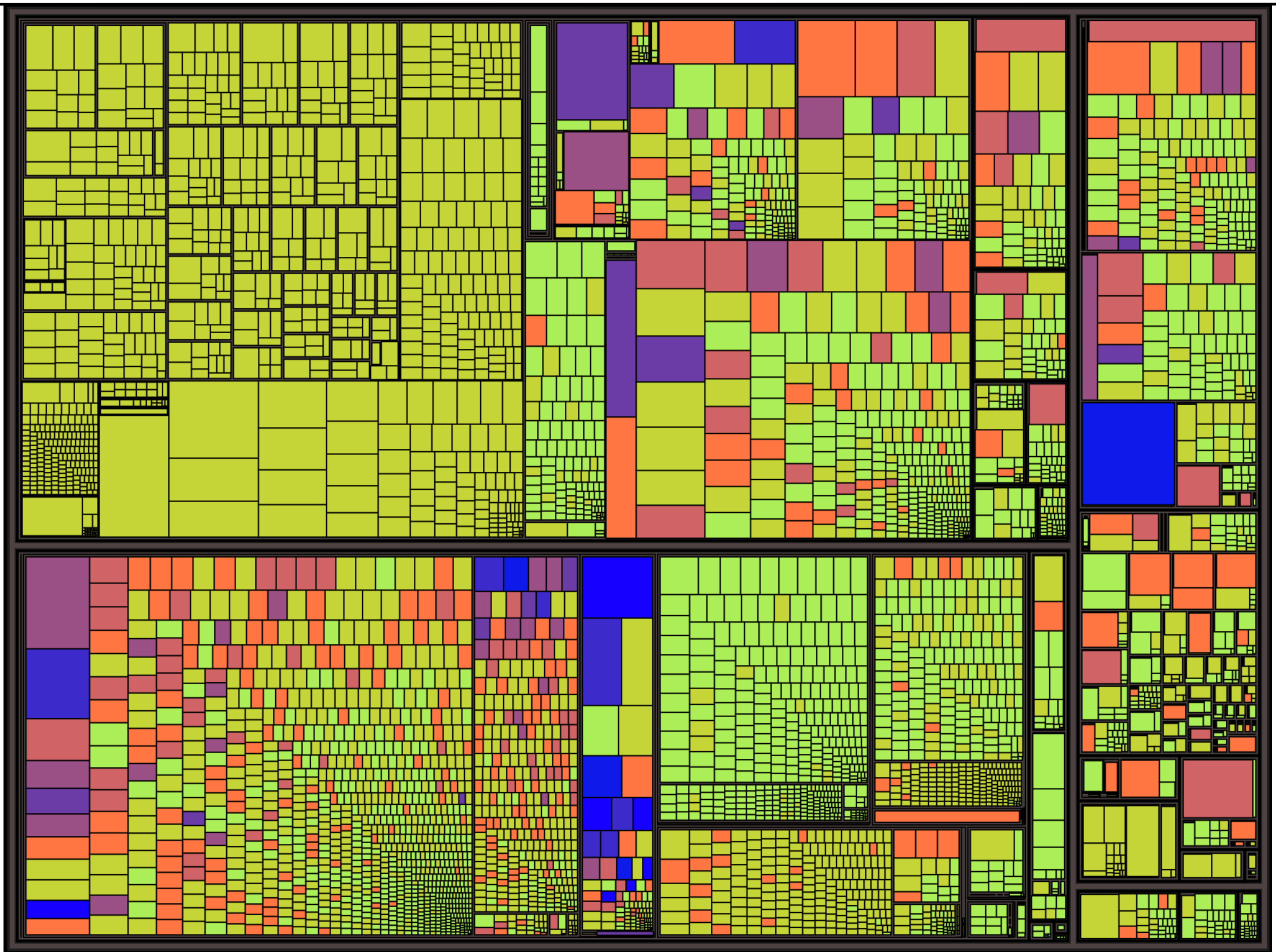
- Ownership and Authors
- Code age
- (Logical coupling)
- (Code churn)
- … and more

https://github.com/adamtornhill/code-maat

Simple cross-language code smell 2:

Too few authors

AUTHORS – BETTER VIEWED IN THE APP



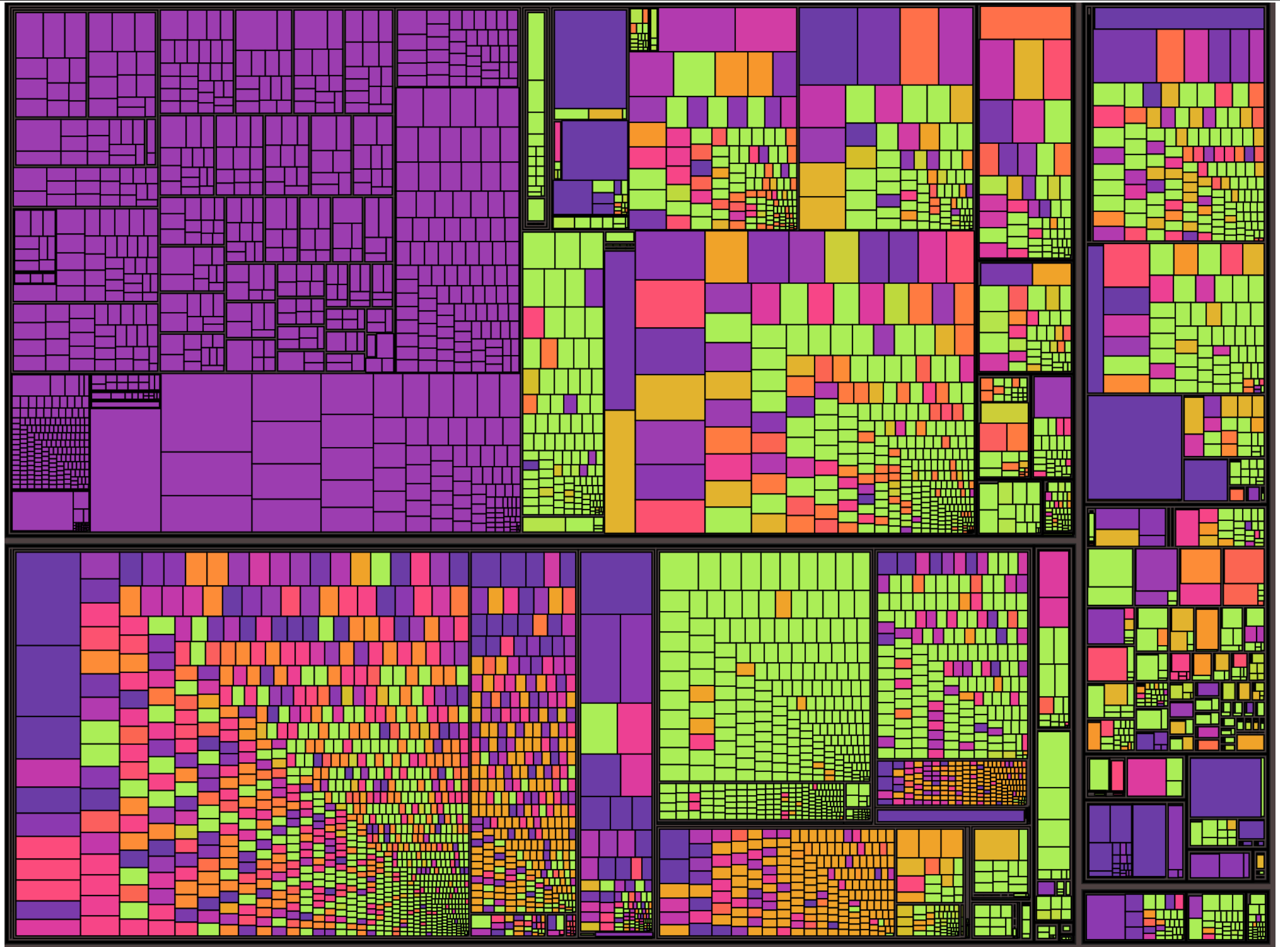
Simple cross-language code smell 3:

Too little change

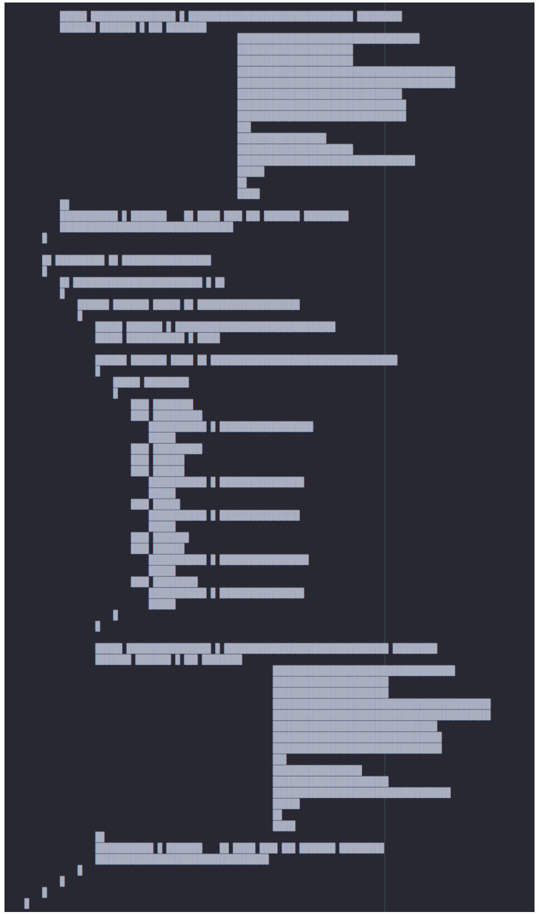
OPINIONS MAY DIFFER!

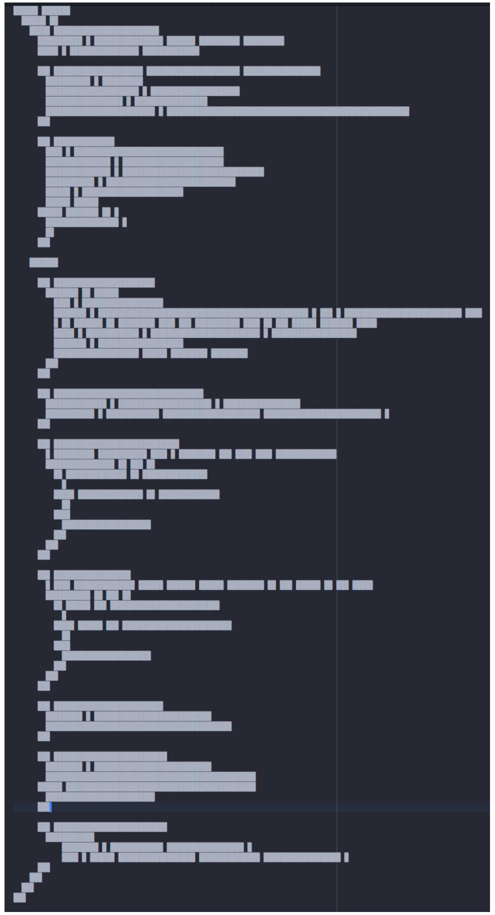
- Living code tends to change – people use it, they find refactorings, they make changes.
- Static unchanging code might be perfect – or it might contain lurking undiscovered bugs. Either way, over time, collective knowledge drops to zero.
- If it is static because it is perfect, it should be extracted out into a standalone library, with a *lot* of automated tests.

AGE – BETTER SHOWN IN THE APP



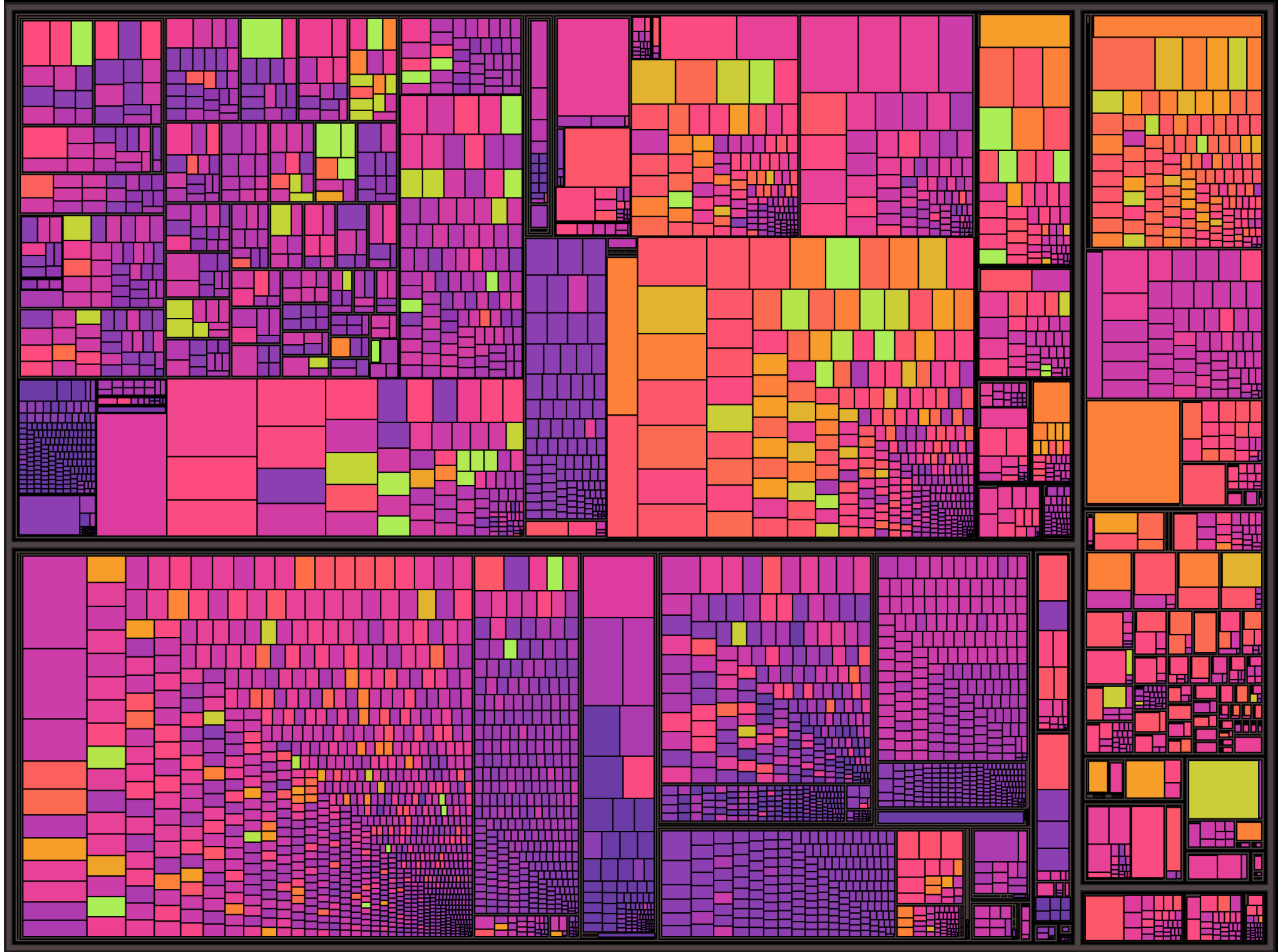
HOW ABOUT IDENTIFYING COMPLEXITY?





Simple cross-language code smell 4:

Using code indentation as a proxy for complexity



HOW DO OTHER PROJECTS LOOK?

GOV.UK VERIFY

Verify– microservices in Java, Ruby, Python

Linux – large C codebase

Kubernetes – mostly Go

MongoDB – C++, C, Go, JavaScript

VSCode – TypeScript, JavaScript

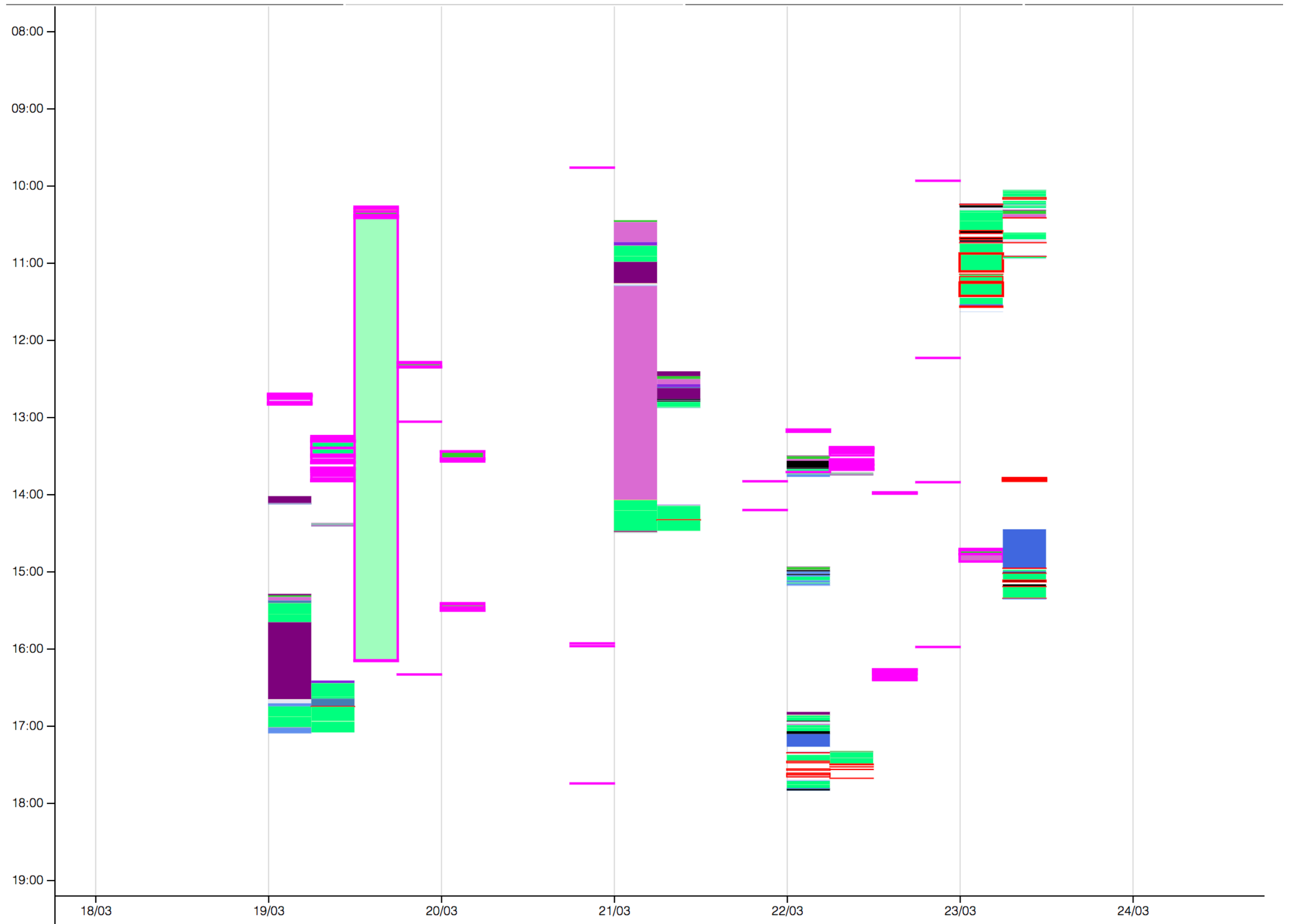
OTHER AREAS TO EXPLORE

Test quality – “temporal coupling” can detect it, but hard to use reliably. Also bad tests can look better than good tests.

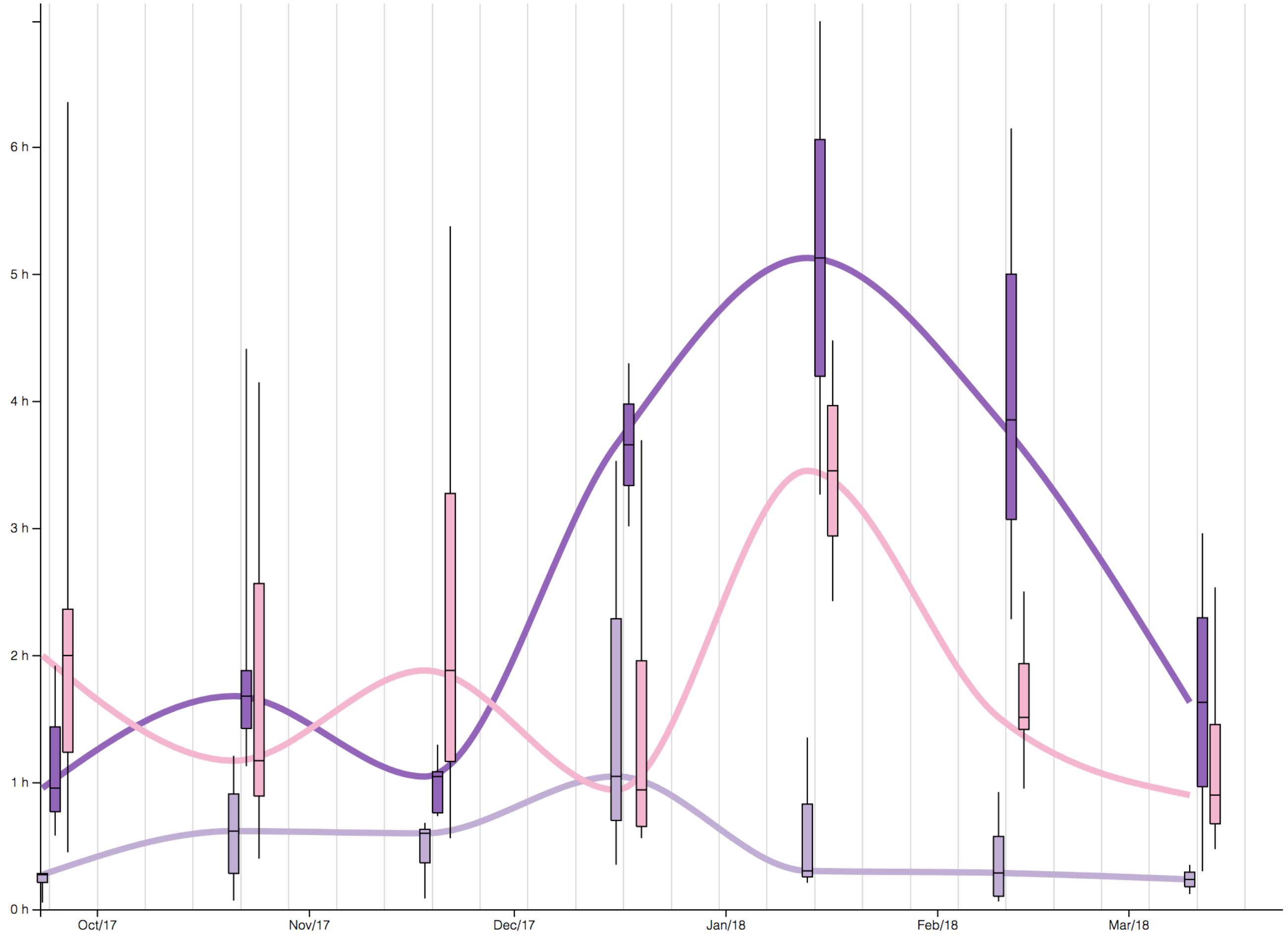
Duplication! Can be spotted by hand, but tooling would be nice.

Deployment data – e.g. release timings, time between development and production.

RELEASE TIME - DETAILS



RELEASE TIME – LONG TERM TRENDS



WHAT DID WE TELL FISHCORP?

- Your old code is complex, badly tested, mostly only understood by 1 or 2 people
- Your new code is even worse - complex, full of duplication, badly tested, and still tightly coupled to your old code.
- You need to move away from giant databases and ETL jobs
- You need to build something new.

THANK YOU!

QUESTIONS?

Simple code smell summary:

- *Classes/Files too large*
- *Too few authors*
- *Too little change*
- *Too much complexity (via indentation)*

Code will (eventually) be at github.com/kornysietsma   Twitter: @kornys   Email: korny@thoughtworks.com

ThoughtWorks®

IMAGE CREDITS

Fanfold Paper – Arnold Reinhold (via WikiMedia)

HP-85 computer – Wolfgang Stief (via WikiMedia)

James Lewis' Head - @boicy on Twitter

Your Code as a Crime Scene cover – Pragmatic Programmers