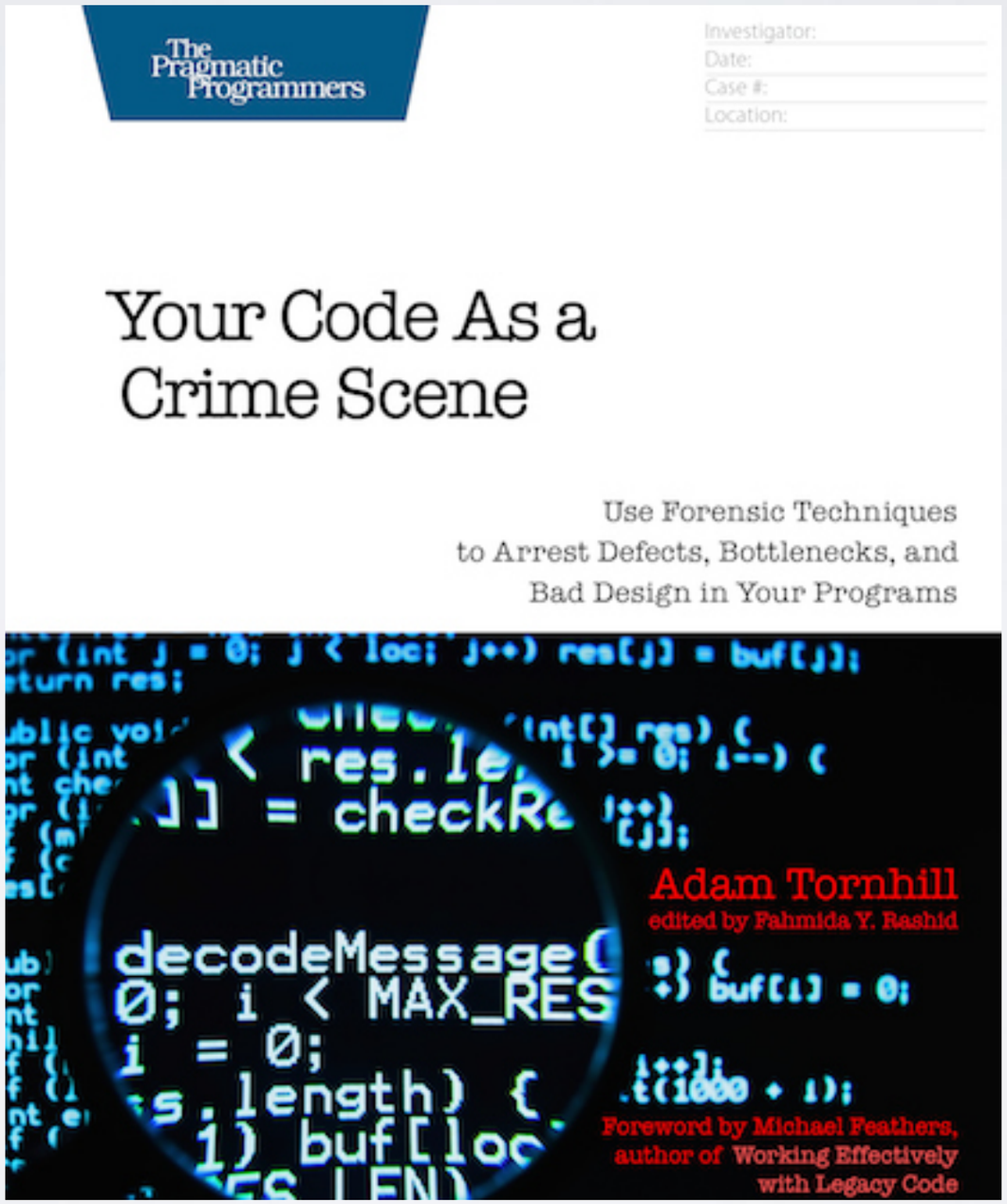
TREASURE ISLAND VERSION CONTROL

@StefanSeegers

TREASURE ISLAND?

- Hotspots
- Architectural Decay
- Commit Messages
- Knowledge Map
- Tools

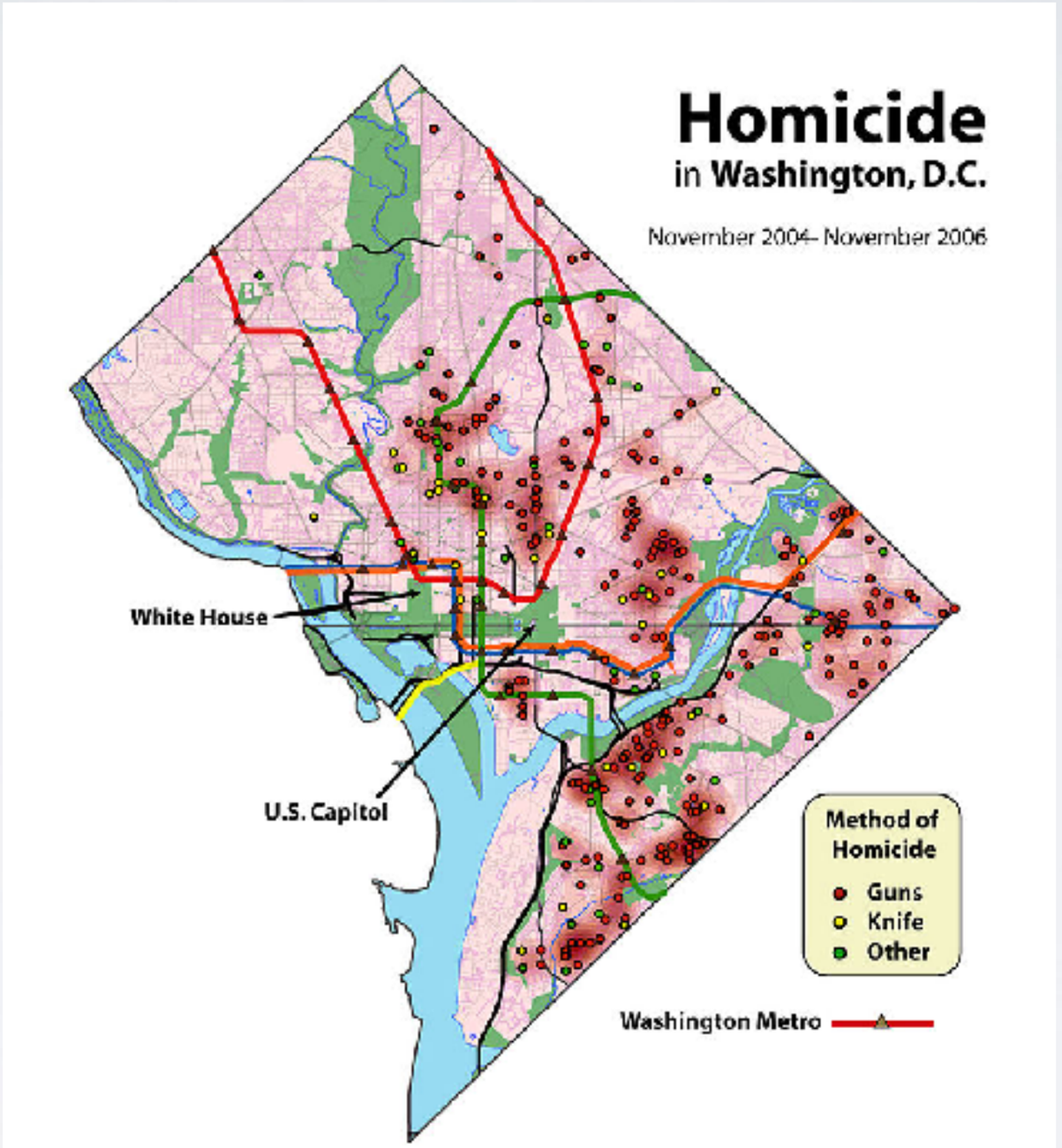
WHERE DID MY JOURNEY STARTED?



CRIMES?

- Crimes have offenders
- To find offenders, forensic psychology could be used
- Code can also be an offender to a system
- To find offenders, we're investigating our VCS
- Crimes and code have several things in common!

GEOGRAPHICAL OFFENDER PROFILING



https://commons.wikimedia.org/wiki/File:DChomicides.jpg

CRIMES IN CODE?

- Bugs
- Changing code takes too long
- Changing code causes new Bugs
- Code is hard to understand or evolve
- ...

HOW TO DETECT OR EVEN PREDICT CRIMES?

- Find Hotspots!
- Hotspots are our Guide to Improvements and Refactorings

HOW TO FIND HOTSPOTS?

- Static code analysis?
  - SonarQube
  - Structure 101
  - Codecity

CODECITY

package ~ district

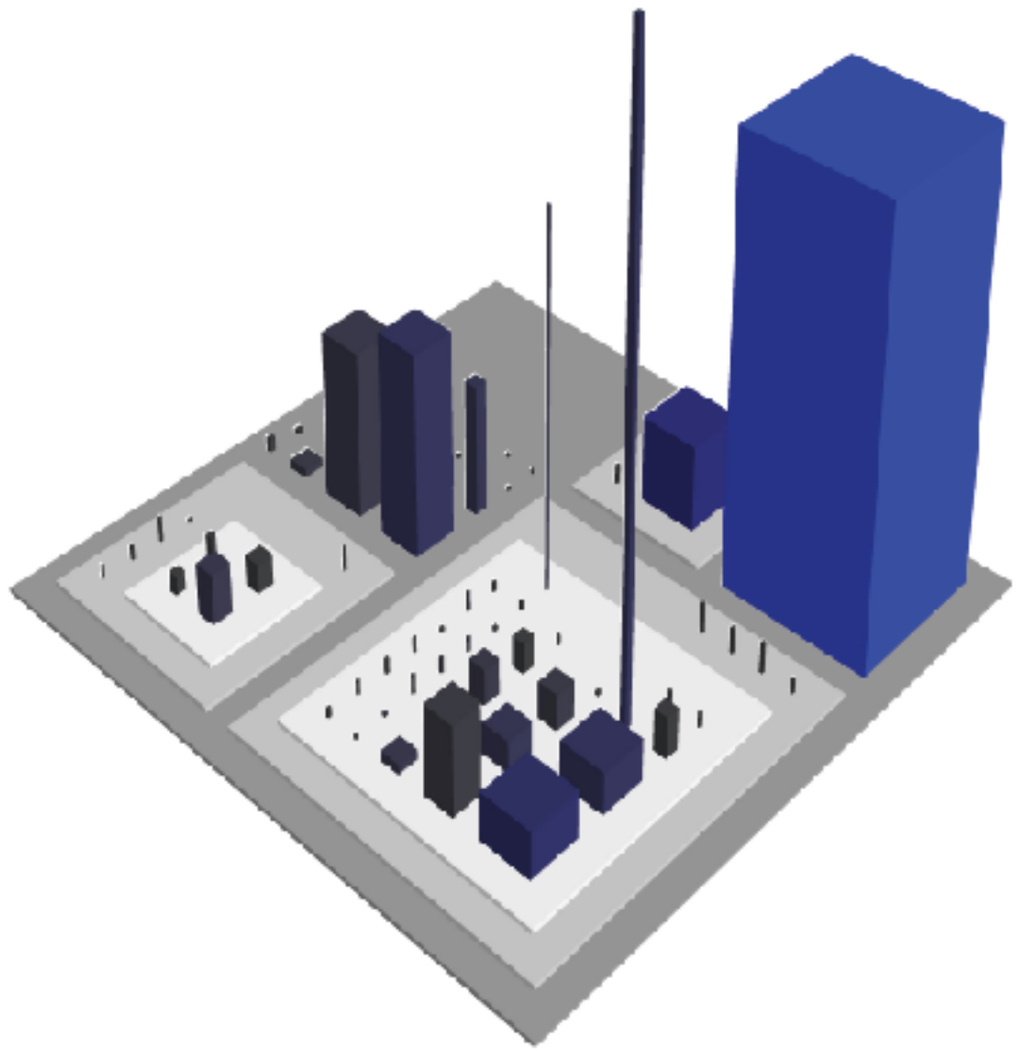
nesting level ~ color

class ~ building

methods (NOM) ~ height

attributes (NOA) ~ width, length

lines (LOC) ~ color



https://en.wikipedia.org/wiki/File:CityMetaphor.png

BUT: WHAT'S HOT?

- Which part is stable, which is not?
- Which part is well tested, which is not?
- What about other files, other languages?
- Static Code Analysis does not tell the whole story!

INFORMATION FROM VCS

- Who changed which file when?
- How often?
- Focussing on a certain period
- Use Color to highlight change frequency
- New Dimension: Time!

CREATING AN OFFENDER PROFILE

- Author created a tool to analyze VCS data

- Name of the tool: Code Maat

HOW DOES IT WORK?

- Collect Data from VCS, e.g. 'git log'

- Inspect Data with Code Maat

COLLECT DATA

- git log --pretty=format:'[%h] %an %ad %s' --date=short --numstat
- %h: abbreviated commit hash
- %an: author name
- %ad: author date
- %s: subject
- -date=short - date without time
- --numstat - number of added and deleted lines

WHAT'S IN THE LOGFILE?

```
[1bafb6d] Adam Petersen 2013-09-24 Release 0.3.1

1    1    project.clj


[0a05fba] Adam Petersen 2013-09-17 Added an identity analysis as debug aid

5    1    README.md

5    5    src/code_maat/analysis/authors.clj

3    4    src/code_maat/app/app.clj

1    1    src/code_maat/core.clj

18   1    test/code_maat/end_to_end/scenario_tests.clj
```

INSPECT DATA WITH CODE MAAT

- Measurement: Change frequency
- Code that changes in the past is likely to change again

FIRST DIMENSION: CHANGE FREQUENCY

```
>maat -l logfile.log -c git -a revisions >freqs.csv
```

```
entity,n-revs
src/code_maat/analysis/logical_coupling.clj,26
src/code_maat/app/app.clj,25
src/code_maat/core.clj,21
test/code_maat/end_to_end/scenario_tests.clj,20
project.clj,19
src/code_maat/parsers/svn.clj,19
src/code_maat/parsers/git.clj,14
src/code_maat/analysis/authors.clj,14
```

LEARN FROM DEVELOPER ACTIVITY

- Relative number of commits is a good predictor of defects and design issues.

- Attractive Starting Point

SECOND DIMENSION: COMPLEXITY

- Metric: lines of code
- Fast and simple analysis
- Language neutral
- Tool: cloc (available on Sourceforge)

CLOC ANALYSIS

>cloc ./ --by-file --csv --quiet >lines.csv

```
language,filename,blank,comment,code

Clojure,.\src\code_maat\analysis\logical_coupling.clj,23,14,145

Clojure,.\test\code_maat\end_to_end\scenario_tests.clj,18,19,91

Clojure,.\test\code_maat\analysis\logical_coupling_test.clj,15,5,89

Clojure,.\src\code_maat\app\app.clj,13,6,85

Clojure,.\test\code_maat\parsers\svn_test.clj,7,5,79
```

CLOC ANALYSIS

lang	filename	blank	comment	code
Clojure	\src\code_maat\analysis\logical_coupling.clj	23	14	145
Clojure	\test\code_maat\end_to_end\scenario_tests.clj	18	19	91
Clojure	\test\code_maat\analysis\logical_coupling_test.clj	15	5	89
Clojure	\src\code_maat\app\app.clj	13	6	85
Clojure	\test\code_maat\parsers\svn_test.clj	7	5	79

MERGE COMPLEXITY AND EFFORT

- Freqs.csv shows change frequency (effort)
- Lines.csv shows complexity
- Merging leads to hotspots

MERGING

- Use filename for combining data
- Author provides python script
- >python merge_comp_freqs.py freqs.csv lines.csv

PRIORITIZED LIST OF SUSPECTS

First Hotspot!

Module	Revisions	Code
src\code_maat\analysis\logical_coupling.clj	26	145
src\code_maat\app\app.clj	25	85
src\code_maat\core.clj	21	35
test\code_maat\end_to_end\scenario_tests.clj	20	91
src\code_maat\parsers\svn.clj	19	53
project.clj	19	17
src\code_maat\analysis\authors.clj	14	47

Changes often!

Complex!

USE HOTSPOTS AS A GUIDE

- Frequent changes to complex code indicates declining quality
- Prioritize design issues
- Identify parts of the system where code reviews are good investment
- Feature areas that could benefit from exploratory testing

LIMITATIONS

- Data is not normalized (is 26 revisions good or bad?)
- Depends on analysis period, so it can be tricky to get right
- Depends on commit styles

QUESTIONS?

ARCHITECTURAL DECAY

https://es.wikipedia.org/wiki/Pruitt-Igoe#/media/File:Pruitt-igoe_collapse-series.jpg

WHAT ARE WE LOOKING FOR?

- Uncover hidden dependencies!

HOW TO DETECT IT?

- Analyze Coupling
- Modules that keep changing together
- Looking for hidden, implicit dependency
- Not looking for explicit dependency

EXAMPLE: EXPLICIT DEPENDENCY

```
[a7abe6c] johndoe 2015-06-29 Improving the test

1   0   CHANGES.md

16  9   src/test/java/WorkflowTest.ja

3   1   src/main/java/W

[6a64190] john oe 2015-06-26 Added test for null

5   0   src/test/java/WorkflowTest.java

3   1   src/main/java/Workflow.java
```

Intentional Coupling!

EXAMPLE: NO EXPLICIT DEPENDENCY

```
[d7abe6c] johndoe 2015-05-29 Correct Calculation
3   1   src/main/java/account/Account.java
3   1   src/main/java/account/Asset.java
3   1   src/main/java/customer/Customer.java
16  9   src/main/test/account/AccountTest.java

[1a64190] johndoe 2015-05-26 Added something
3   1   src/main/java/account/Account.java
2   7   src/main/java/account/Asset.java
2   2   src/main/java/customer/Customer.java
8   0   src/main/java/printing/Printer.java
5   0   src/main/test/account/AccountTest.java
```

EXAMPLE: NO EXPLICIT DEPENDENCY

```
[d7abe6c] johndoe 2015-05-29 Correct Calculation

3    1    src/main/java/account/Account.java

3    1    src/main/java/account/Asset.java

3    1    src/main/java/customer/Customer.ja

16   9    src/main/test/account/Ac


[1a64190] joh                                  omething

3    1    src/mai                 unt/Account.java

2    7    src/main/java/account/Asset.java

2    2    src/main/java/customer/Customer.java

8    0    src/main/java/printing/Printer.java

5    0    src/main/test/account/AccountTest.java
```

Incidential Coupling?

ANALYZE COUPLING

- Do we have a Design Problem?
- Look into Source Code to investigate
- Where to look at?
- We need hints!

PRACTICE / EXAMPLE

- Craft.Net
- A collection of several Minecraft-related .NET libraries.

FIRST: COLLECT DATA

- Analyzing with 'git log'

- Collect files that change together

COLLECT DATA

- git log --pretty=format:'[%h] %an %ad %s' --date=short --numstat --before=2014-08-08 >craft_evo.log

COLLECTED DATA

```
[db70d9a] Drew DeVault 2012-12-31 Added note on submodules

10   0    README.md

[3ee9ba5] Drew DeVault 2012-12-31 Fixed respawn; fixes #142

14   1    Craft.Net.Server/Handlers/LoginHandlers.cs

1    1    externals/fNbt

[1f85519] Drew DeVault 2012-12-30 Added missing file

17   0    Craft.Net.Data.Test/LightingTest.cs

[e623eca] Drew DeVault 2012-12-30 Progress on lighting

1    0    Craft.Net.Data.Test/Craft.Net.Data.Test.csproj

1    1    Craft.Net.Data/Region.cs

10   4    Craft.Net.Data/World.Lighting.cs

1    3    Craft.Net.Data/World.cs
```

SECOND: ANALYZE DATA

- Analysis: Sum of coupling

- Hint which module to inspect

- maat -l craft_evo.log -c git -a soc

SUM OF COUPLING

- Number of shared transactions for a module
- Priority list of the modules that are most frequently changed together with others

SOC EXAMPLE

File	Commit #1	Commit #2	Commit #3	SOC
ClassA	■	■		3
ClassB	■	■		3
ClassC	■		■	2

SOC EXAMPLE

File	Commit #1	Commit #2	Commit #3	SOC
ClassA	■	■		**3**
ClassB	■	■		3
ClassC	■		■	2

SOC EXAMPLE

File	Commit #1	Commit #2	Commit #3	SOC
ClassA	■	■		3
ClassB	■	■		**3**
ClassC	■		■	2

SOC EXAMPLE

File	Commit #1	Commit #2	Commit #3	SOC
ClassA	■	■		3
ClassB	■	■		3
ClassC	■		■	**2**

ANALYZE SOC

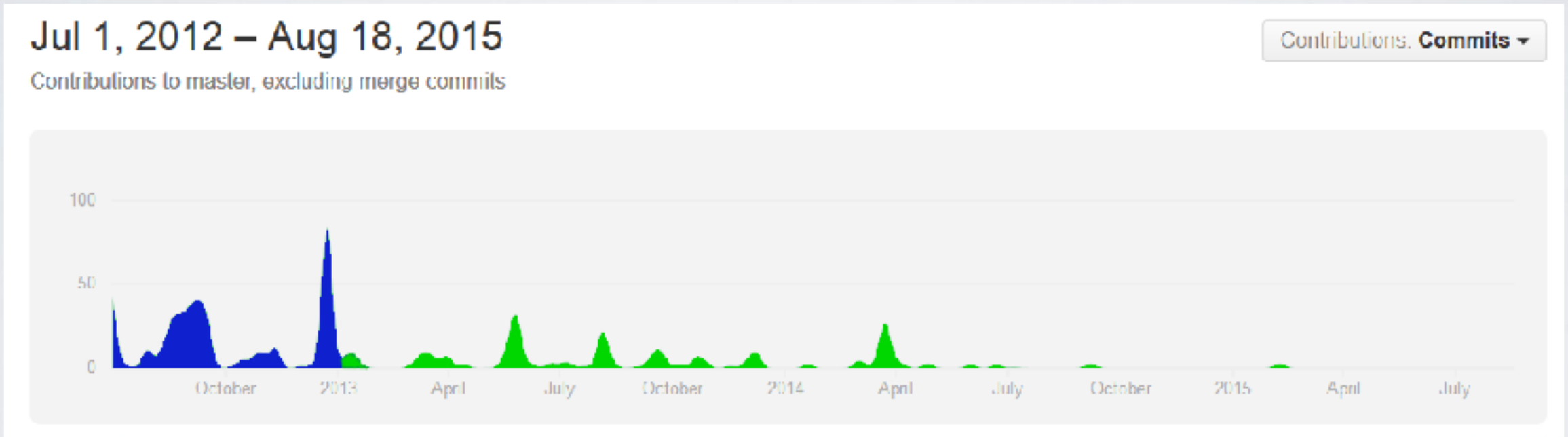
- maat -l craft_evo.log -c git -a soc
- entity,soc
  Craft.Net.Server/Craft.Net.Server.csproj,685
  Craft.Net.Server/**MinecraftServer.cs**,**635**
  Craft.Net.Data/Craft.Net.Data.csproj,521
  Craft.Net.Server/MinecraftClient.cs,464
- MincecraftServer.cs is our first hint!

ANALYSIS PERIODS

- Periods chosen from github activity (Graphs)

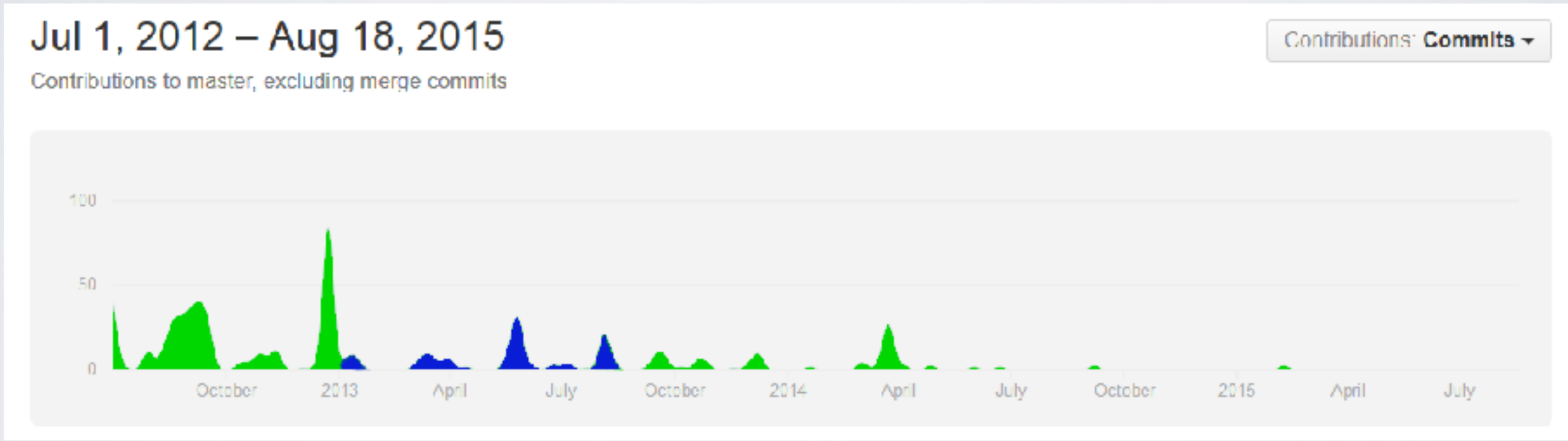
ANALYSIS PERIODS

- Periods chosen from github activity (Graphs)



ANALYSIS PERIODS

- Periods chosen from github activity (Graphs)



ANALYZE FIRST PERIOD

- Analyze Coupling with Code Maat
- git log --pretty=format:'[%h] %an %ad %s' --date=short --numstat --before=2013-01-01 >craft_evo_130101.log
- maat -l craft_evo_130101.log -c git -a coupling >craft_coupling_130101.csv

ANALYZE DATA

Entity	coupled	Degree
Analyze DataCraft.Net.Data/Blocks/CarrotBlock.cs	Craft.Net.Data/Blocks/PotatoBlock.cs	100
Craft.Net.Data/Items/CarrotItem.cs	Craft.Net.Data/Items/PotatoItem.cs	100
Craft.Net.Server/Worlds/Generation/ FlatlandGenerator.cs	Craft.Net.Server/Worlds/Generation/ IWorldGenerator.cs	80
Craft.Net.Data/WindowArea.cs	Craft.Net.Data/Windows/InventoryWindow.cs	76
Craft.Net.Server/Packets/ PlayerPositionAndLookPacket.cs	Craft.Net.Server/Packets/PlayerPositionPacket.cs	72
Craft.Net.Server/Worlds/Chunk.cs	Craft.Net.Server/Worlds/Generation/ FlatlandGenerator.cs	66

Degree = % of shared commits

COUPLING BEFORE 2013

Entity	coupled	Degree
Craft.Net.Server/Craft.Net.Server.csproj	Craft.Net.Server/MinecraftServer.cs	41
Craft.Net.Server/MinecraftClient.cs	Craft.Net.Server/MinecraftServer.cs	38

There are just two lines!
Classic Producer – Consumer Pattern

ANALYZE SECOND PERIOD

- git log --pretty=format:'[%h] %an %ad %s' --date=short --numstat --after=2013-01-01 --before=2014-08-08 >craft_evo_140808.log

- maat -l craft_evo_140808.log -c git -a coupling >craft_coupling_140808.csv

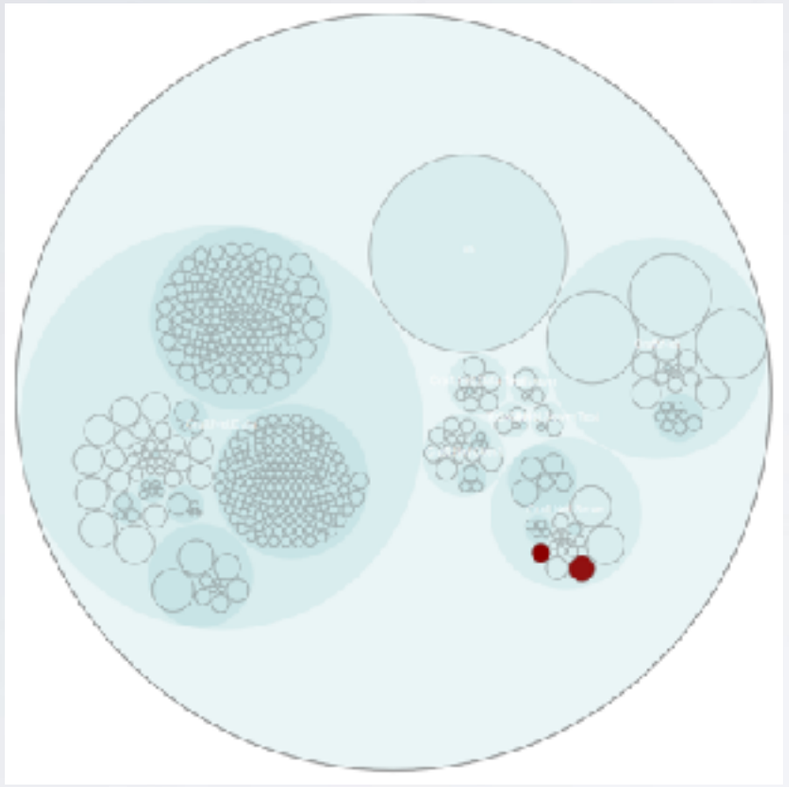
COUPLING AFTER 2013

Entity	coupled	Degree
source/Craft.Net.Server/Handlers/PacketHandlers.cs	MinecraftServer.cs	48
source/Craft.Net.Server/Craft.Net.Server.csproj	MinecraftServer.cs	47
source/Craft.Net.Server/Handlers/LoginHandlers.cs	MinecraftServer.cs	42
source/Craft.Net.Server/EntityManager.cs	MinecraftServer.cs	41
source/Craft.Net.Anvil/Level.cs	MinecraftServer.cs	36

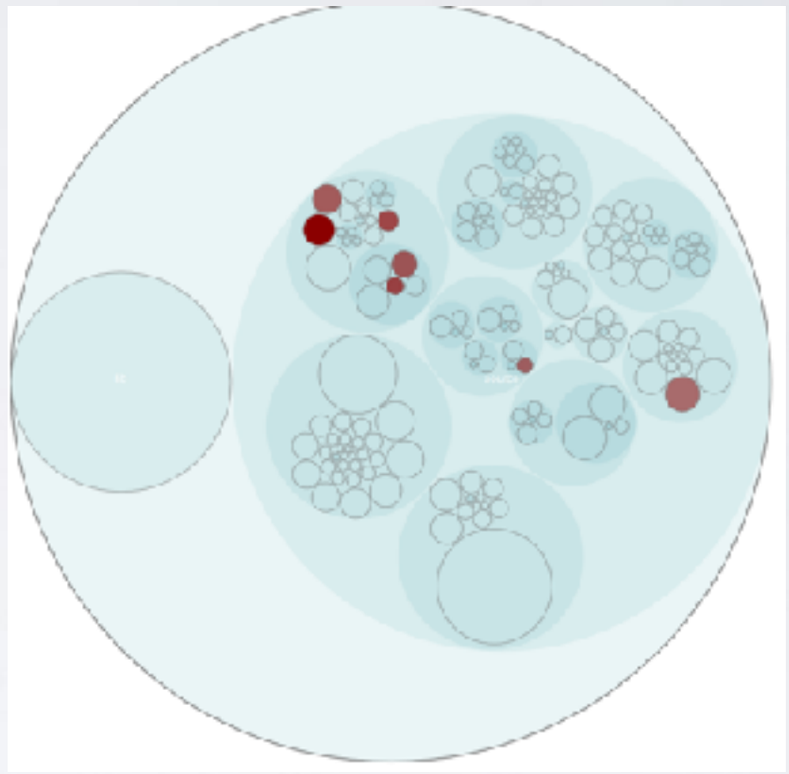
CREATE VISUALIZATION

- D3.js Circle Packing
- python csv_as_enclosure_json.py
  --structure craft_lines.csv
  --weights craft_coupling_130101.csv
  --weightcolumn 3

DEMO!



Before 2013



After 2013

THE FUTURE?

- Will we receive a hint from version control?

- People who changed file A, often changed file B?

QUESTIONS?

COMMIT MESSAGES

- Features we are working on
- Where do we spend our time?

  Features?

  Bugfixes?

  Maintenance?

RETRIEVE COMMIT MESSAGES

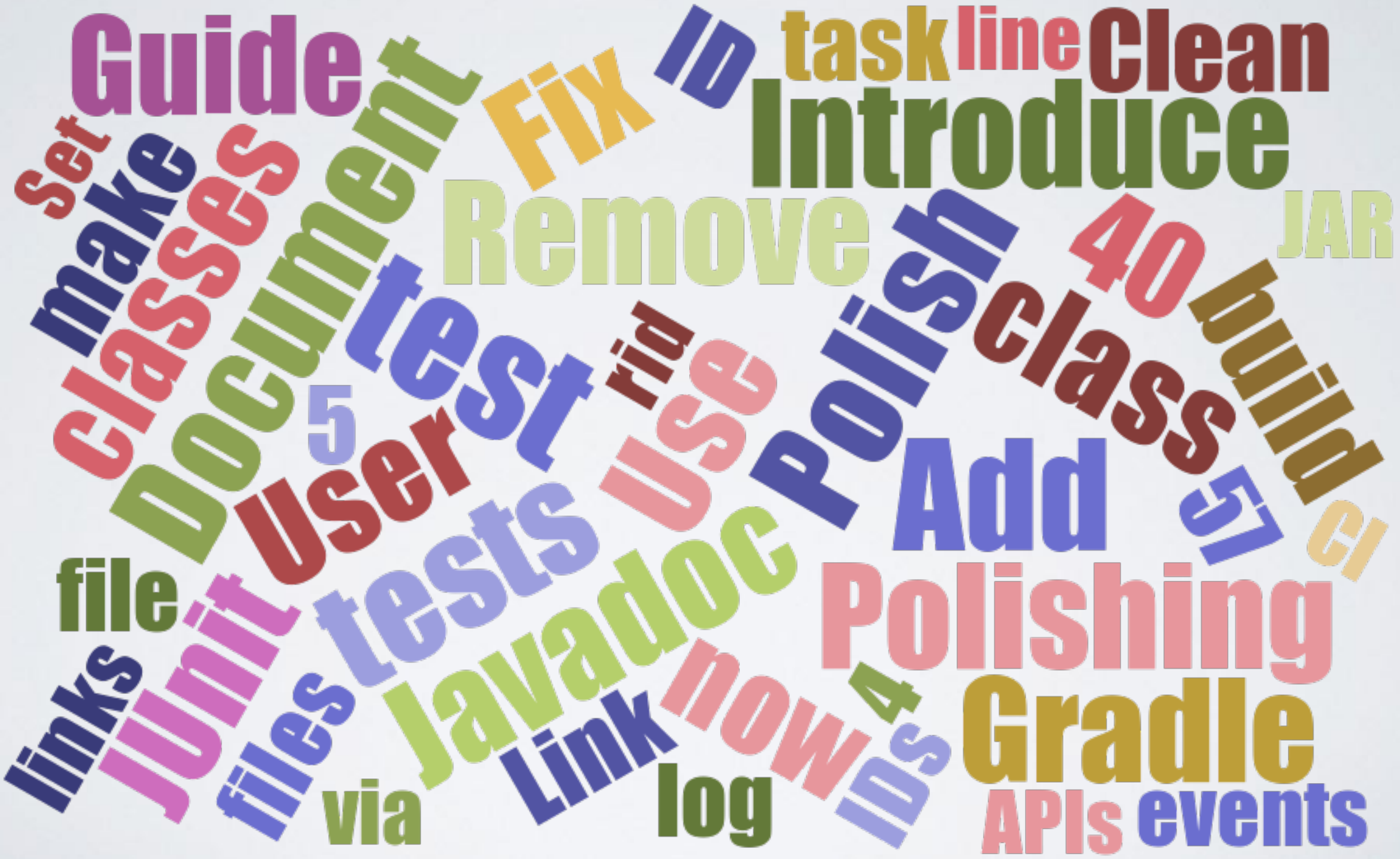
- git log --pretty=format:'%s'

- hg log --template "{desc}\n"

ANALYZE COMMIT MESSAGES

- Just look at them?
- Grep for something?
- Visualize?

WORD CLOUD!


Guide
Set
make
classes
Document
test
ment
Fix
ID
task
line
Clean
Introduce
Remove
JAR
40
class
build
User
5
rid
Use
Polish
57
cl
Add
file
JUnit
tests
Javadoc
Polishing
links
files
Link
now
IDs
Gradle
via
log
APIs
events

COMMIT MESSAGES TELL A STORY

- Good basis for discussion around process and daily work

- We want to see words from our domain in commit clouds

- We do not want to see words indicating quality problems

QUESTIONS?

KNOWLEDGE MAP

- Imagine you're working in a LARGE Codebase
- That system is developed by several teams
- Or: working in an Open Source Project
- People may not know each other

DOING A CHANGE

- Assume you need some input
- To whom should you talk?
- Who has the knowledge?

BLAME / ANNOTATION

- Who changed which line

- Low level information

- But: we likely need high level information

SCALA KNOWLEDGE MAP

- git log --pretty=format:'[%h] %an %ad %s' --date=short --numstat --before=2013-12-31 --after=2011-12-31 >scala_evo.log

- maat -c git -l scala_evo.log -a main-dev >scala_main_dev.csv

MAIN-DEV ANALYSIS

Entity	Main-Dev	added	Total-added	Ownership
GenICode.scala	Paul Phillips	584	1579	0,37
IcodeCheckers.scala	Jason Zaugg	19	44	0,43
Icodes.scala	Grzegorz Kossalowski	16	32	0,5

STRUCTURE

- Collect complexity information

- cloc ./ --by-file --csv --quiet --report-file=scala_lines.csv

PROJECT MAIN-DEVS ON A MAP

- Combine structure (scala_lines.csv) and knowledge owners (scala_main_dev.csv)
- Provide unique color for each developer

SCALA_AUTHOR_COLORS.CSV

```
author,color

Martin Odersky,darkred

Adriaan Moors,orange

Paul Phillips,green

...
```

GENERATE MAP

```
python
csv_main_dev_as_knowledge_json.py
--structure scala_lines.csv
--owners scala_main_dev.csv
--authors scala_author_colors.csv
>scala_knowledge_131231.json
```

DEMO!

KNOWLEDGE LOSS

- Imagine a developer leaves the project
- What parts of the code are left in the wild?
- What parts should the next developer look at?
- Could a knowledge map help?

IDENTIFY ABANDONED CODE

- Assign a color to the developer who left

- ```
  author,color
  Paul Phillips,green
  ```

- Save it as scala_ex_programmers_colors.csv
  ```

GENERATE MAP

```
python
csv_main_dev_as_knowledge_json.py
--structure scala_lines.csv
--owners scala_main_dev.csv
--authors scala_ex_programmers_colors.csv
>scala_knowledge_loss.json
```

DEMO!

USES

- Who can help with code reviews, debugging tasks or design decisions
- Communication aid
- Find a developer who's most likely has knowledge about a feature
- How well fit's the system structure the team structure?
- During refactoring out design issues, main developers should work closely together

MISUSES

- Not a map of productivity

- Can not be used to evaluate people

QUESTIONS?

CODE MAAT ANALYSES

abs-churn
age
author-churn
authors
communication
coupling
entity-churn
entity-effort
entity-ownership
fragmentation
identity
main-dev
main-dev-by-revs
messages
refactoring-main-dev
revisions
soc
summary

CODESCENE

- Commercial Tool (free for Open Source)
- Company Empear founded by Adam Tornhill
- https://codescene.io
- Company Blog: http://www.empear.com/blog/
- Showcases!

TEAMSCALE

- Commercial Tool (Free for Open Source)
- https://www.cqse.eu/de/produkte/teamscale/ueberblick/
- Clone Detection
- Test Gap Analysis

DELTA FLORA

- Analyze Ruby Code History
- Author: Michael Feathers
- https://github.com/michaelfeathers/delta-flora

UPSOURCE

- Commercial Tool from Jetbrains
- Code Review and Repository Browsing
- https://www.jetbrains.com/upsource
- Can suggest Reviewers

GOURCE

- Visualization of commits as animated tree
- Generate videos!
- https://gource.io
- Lots of videos on Youtube (even Linux kernel!)

LINKS

- The Book: Your Code as a crime scene by Adam Tornhill
  https://pragprog.com/book/atcrime/your-code-as-a-crime-scene

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

- http://adamtornhill.com/

- New Book announced: Software (r)Evolution
  http://adamtornhill.com/swevolution/reviewersprogress.html

THANK YOU!