These notes have been compiled for a very short and very interdisciplinary and really informal brown bag meeting at the University of Koblenz-Landau (Koblenz campus).

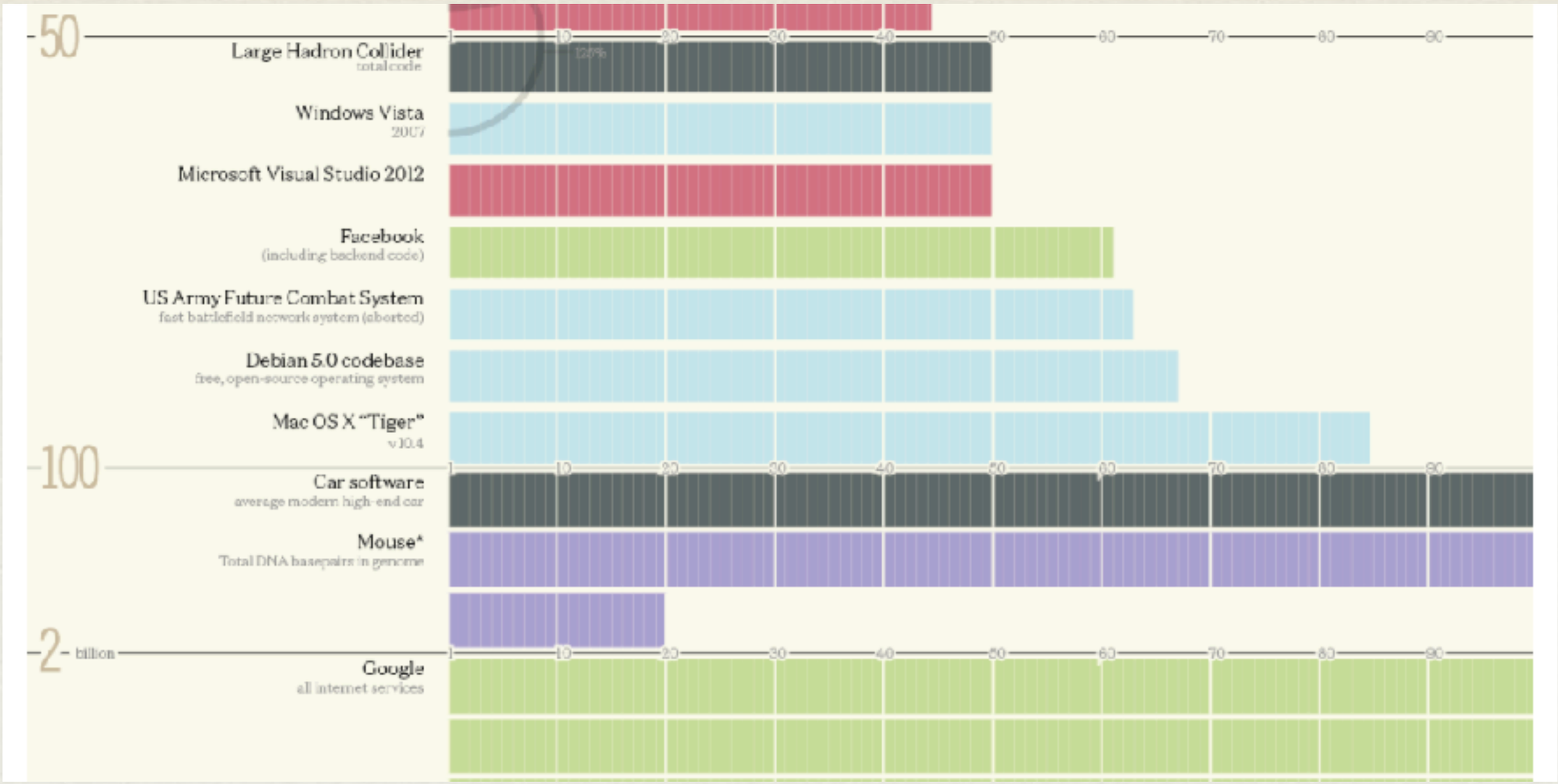
BIG CODE SCIENCE

Ralf Lämmel
Software (Language) Engineer
University of Koblenz-Landau & Facebook

Some wild assumptions

- Let’s say
  - a developer writes 10-30 lines of code per day;
  - each line of code costs 10 US$.

Increasingly bigger code



...

Source: http://www.visualcapitalist.com/millions-lines-of-code/

What's '*Big Code Science*'?

- Code Science is Data Science for code.
- Big Code is like Big Data except that data is code.
- (In fact, we also care about code-related artifacts.)

> Big Code Science is the systematic (scientific) approach to accessing, analyzing, and understanding big data where the data here is code or data related to software development.

214 matches of searching “GitHub” on DBLP (18/03/19)

2018

Raman Goyal, Gabriel Ferreira, Christian Kästner, James D. Herbsleb:
Identifying unusual commits on GitHub. Journal of Software: Evolution and Process 30(1) (2018)

Sebastian Baltes, Stephan Diehl:
Usage and Attribution of Stack Overflow Code Snippets in GitHub Projects. CoRR abs/1802.02938 (2018)

Gede Artha Azriadi Prana, Christoph Treude, Ferdian Thung, Thushari Atapattu, David Lo:
Categorizing the Content of GitHub README Files. CoRR abs/1802.06997 (2018)

Sebastian Baltes, Jascha Knack, Daniel Anastasiou, Ralf Tymann, Stephan Diehl:
(No) Influence of Continuous Integration on the Commit Activity in GitHub Projects. CoRR abs/1802.08441 (2018)

2017

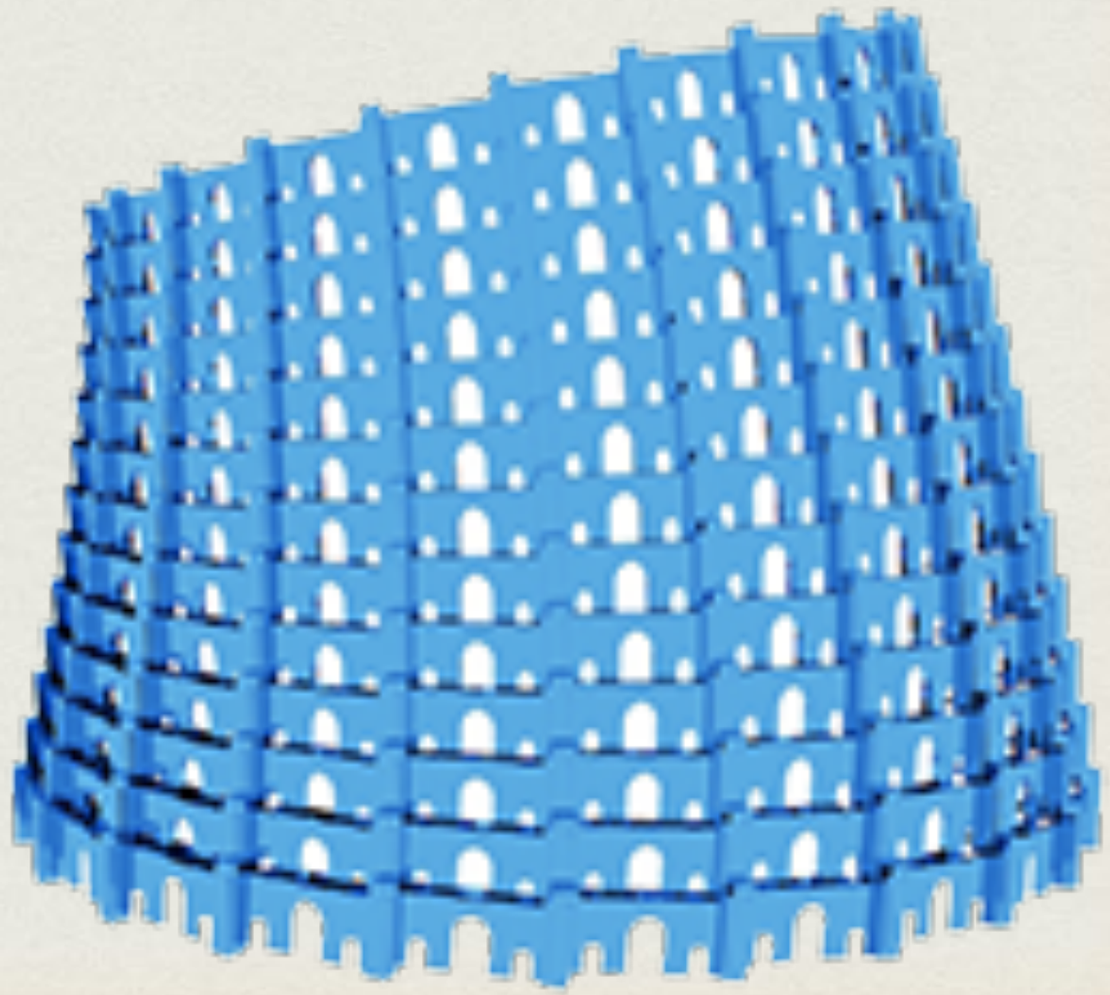
Valerio Cosentino, Javier Luis Cánovas Izquierdo, Jordi Cabot:
A Systematic Mapping Study of Software Development With GitHub. IEEE Access 5: 7173-7192 (2017)

Baishakhi Ray, Daryl Posnett, Premkumar T. Devanbu, Vladimir Filkov:
A large-scale study of programming languages and code quality in GitHub. Commun. ACM 60(10): 91-100 (2017)

Yang Zhang, Huaimin Wang, Gang Yin, Tao Wang, Yue Yu:
Social media in GitHub: the role of @-mention in assisting software development. SCIENCE CHINA Information Sciences 60(3): 32102 (2017)

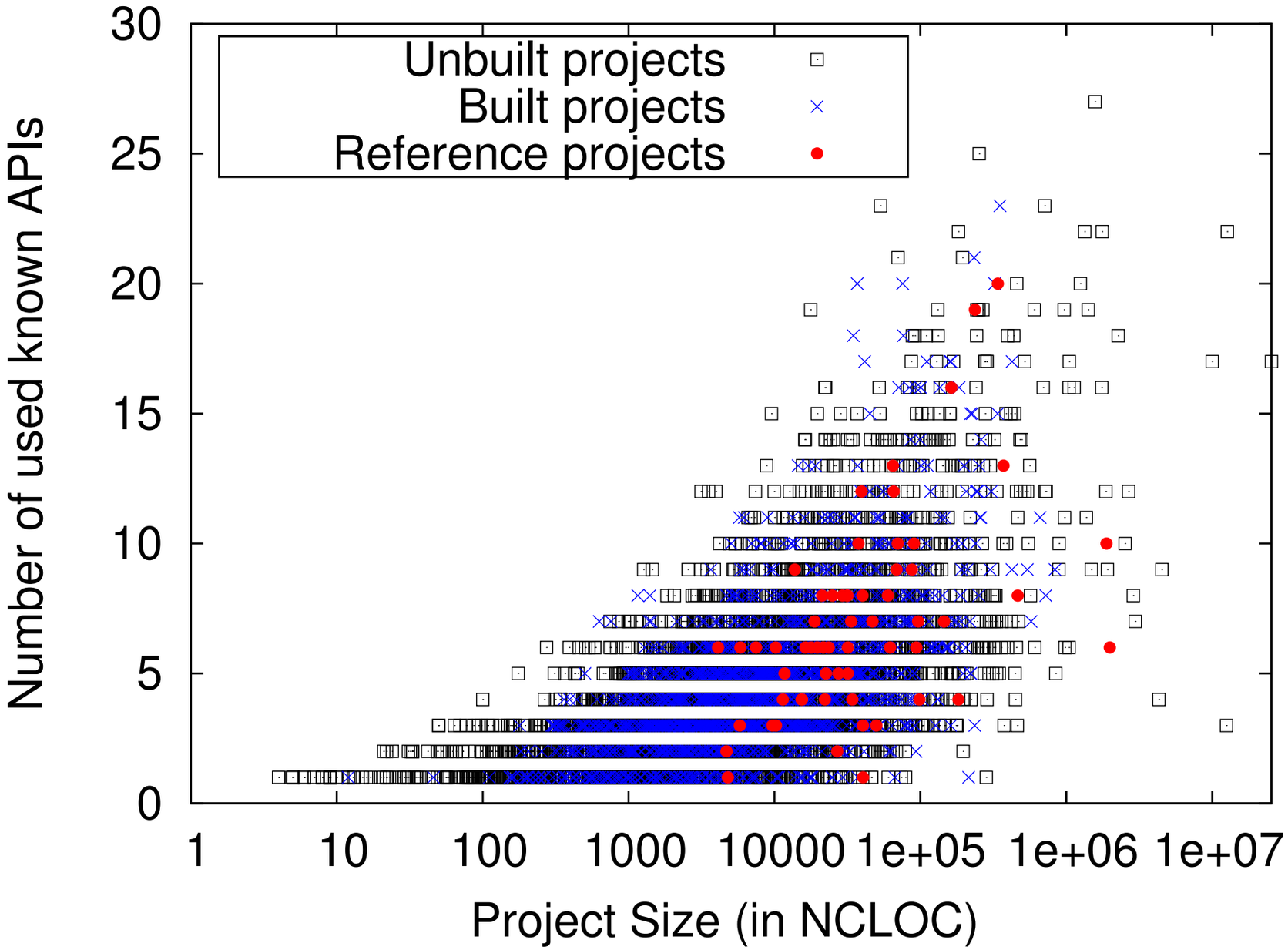
Big Code Science at *SoftLang*

http://www.softlang.org/



An early research question:

How many APIs are and how much of them is used in OSS?



Source: Ralf Lämmel, Ekaterina Pek, Jürgen Starek:
Large-scale, AST-based API-usage analysis of open-source Java projects.
SAC 2011

Unbuilt projects
Built projects
Reference projects

Number of used APIs increases slowly.

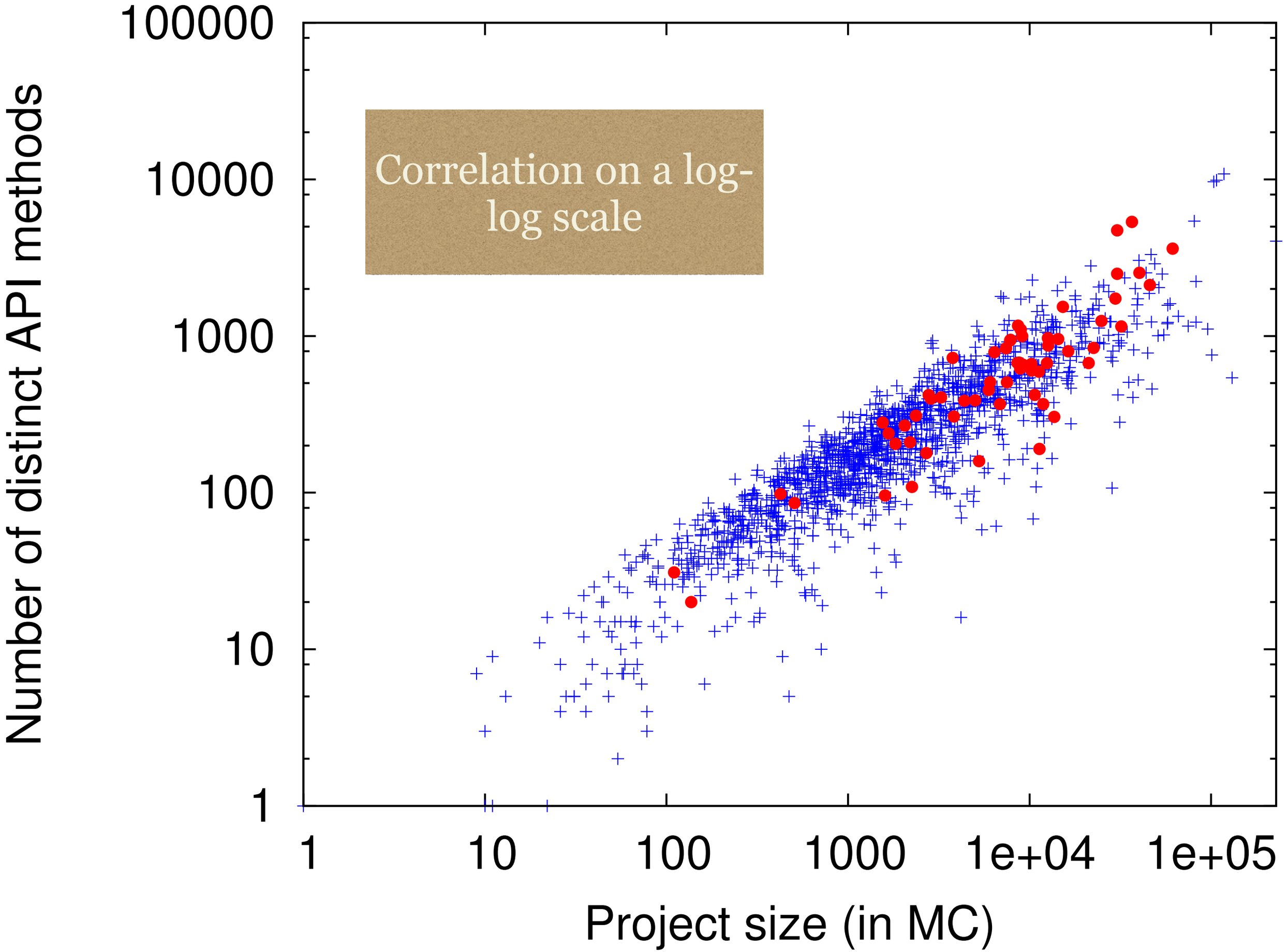
Number of used known APIs

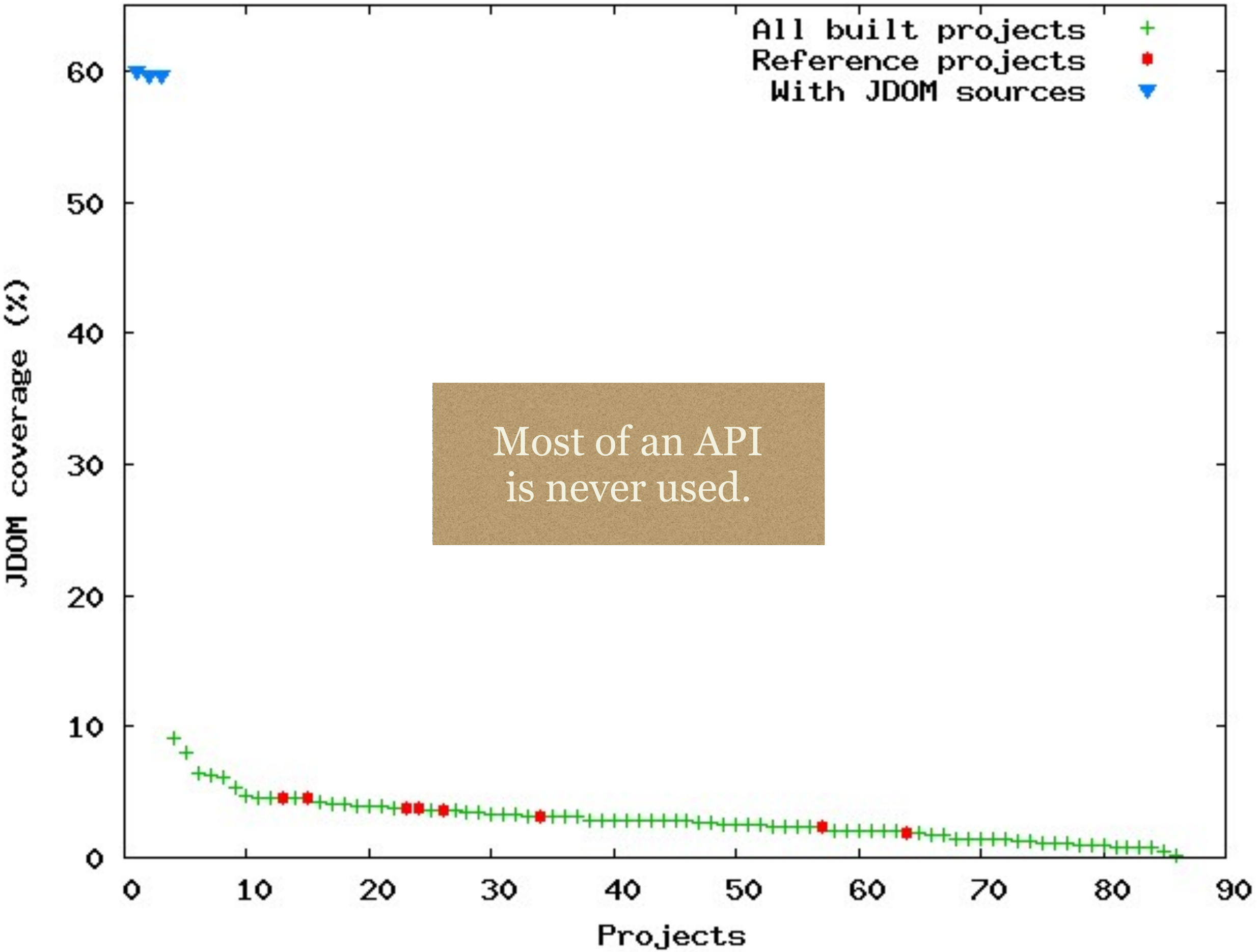
0
5
10
15
20
25
30

1
10
100
1000
10000
1e+05
1e+06
1e+07

Project Size (in NCLOC)

Correlation on a log-log scale





Most of an API is never used.

The deeper research question:

How much OO is there in OSS?

API	# Projects			# Methods		# Distinct methods		# Derived types		# API types	
	impl.	**ext.**	**any**	**impl.**	**over.**	**impl.**	**over.**	**interf.**	**classes**	**interf.**	**classes**
Swing	173	381	391	2512	11150	305	645	443	1859	39	92
AWT	194	75	225	4201	756	593	176	651	120	31	24
Java Collections	120	0	120	986	0	16	0	208	0	3	0
SAX	28	21	42	428	90	85	21	37	29	12	3
JUnit	3	38	40	4	344	4	19	3	46	2	2
Core XML	11	5	14	89	13	17	4	14	5	9	3
SWT	5	8	10	37	86	4	13	25	11	3	3
log4j	1	8	8	25	87	7	9	2	9	2	3
Reflection	7	0	7	10	0	1	0	7	0	1	0
JMF	4	2	6	8	6	6	3	4	3	3	3

API	# Projects			# Methods	
	impl.	**ext.**	**any**	**impl.**	**over.**
Swing	173	381	391	2512	11150
AWT	194	75	225	4201	756
Java Collections	120	0	120	986	0
SAX	28	21	42	428	90
JUnit	3	38	40	4	344
Core XML	11	5	14	89	13
SWT	5	8	10	37	86
log4j	1	8	8	25	87
Reflection	7	0	7	10	0
JMF	4	2	6	8	6

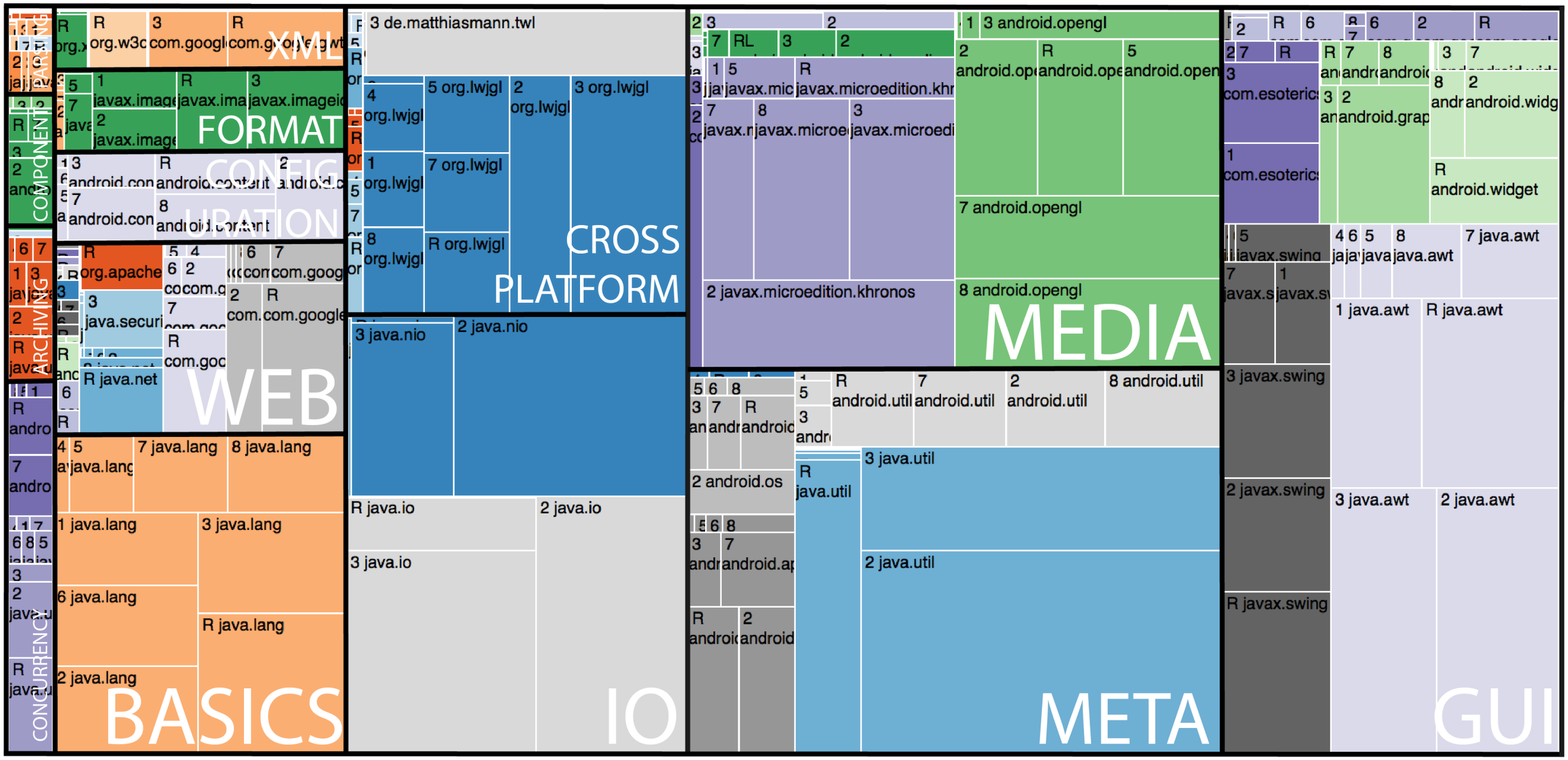
Few APIs (frameworks) are exercised in an OO manner.

# Derived types		# API types	
interf.	**classes**	**interf.**	**classes**
443	1859	39	92
651	120	31	24
208	0	3	0
37	29	12	3
3	46	2	2
14	5	9	3
25	11	3	3
2	9	2	3
7	0	1	0
4	3	3	3

Very few API (framework) types are exercised in an OO manner.

A more recent research question:

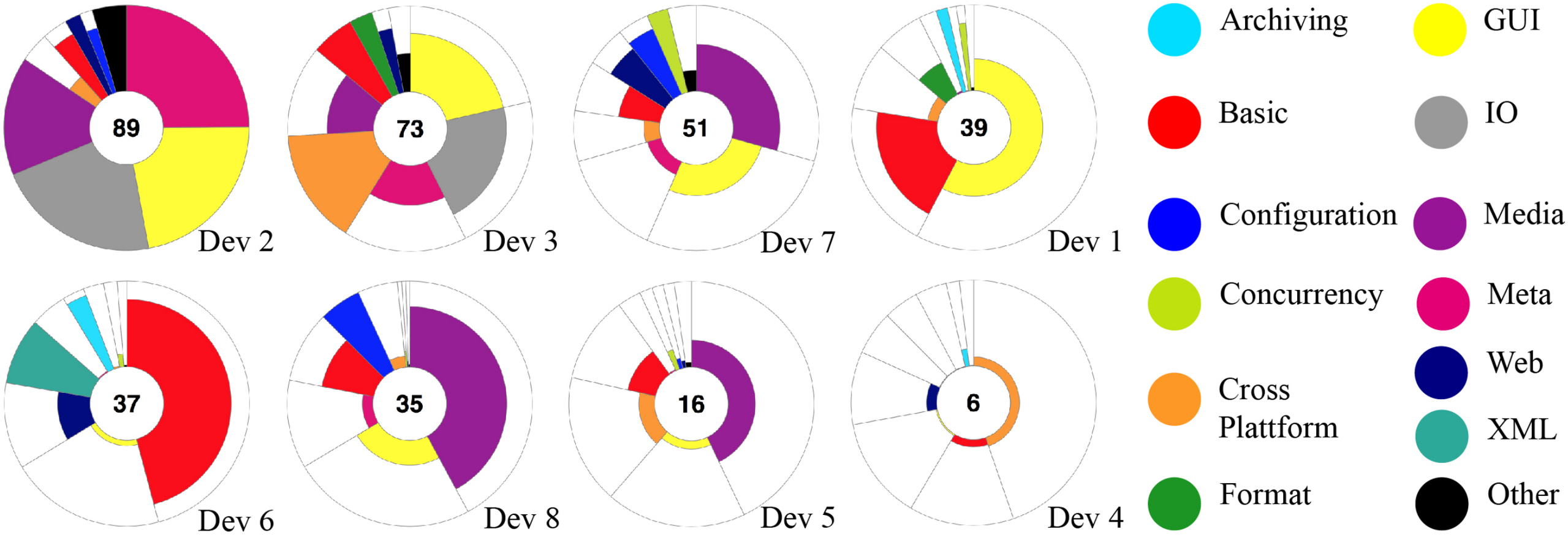
How to profile <u>developers</u> in terms of API experience?



Source: Hakan Aksu, Ralf Lämmel, Wojciech Kwasnik:
Visualization of API Experience.
Softwaretechnik-Trends 36(2) (2016)

The different colors correspond to different APIs in the “GUI” domain.

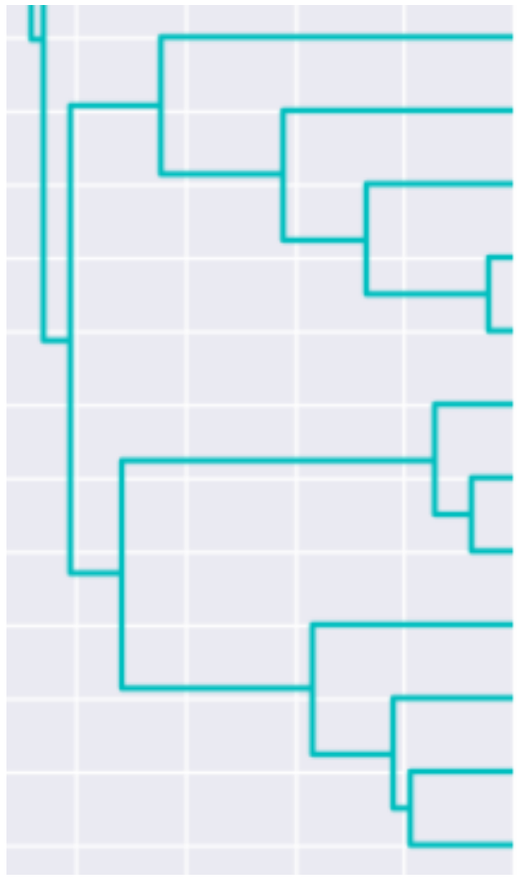
The different squares with the same color correspond to different developers.



The different developers are profiled in terms of different API-related metrics.

A related and recent research question:

How to categorize APIs?



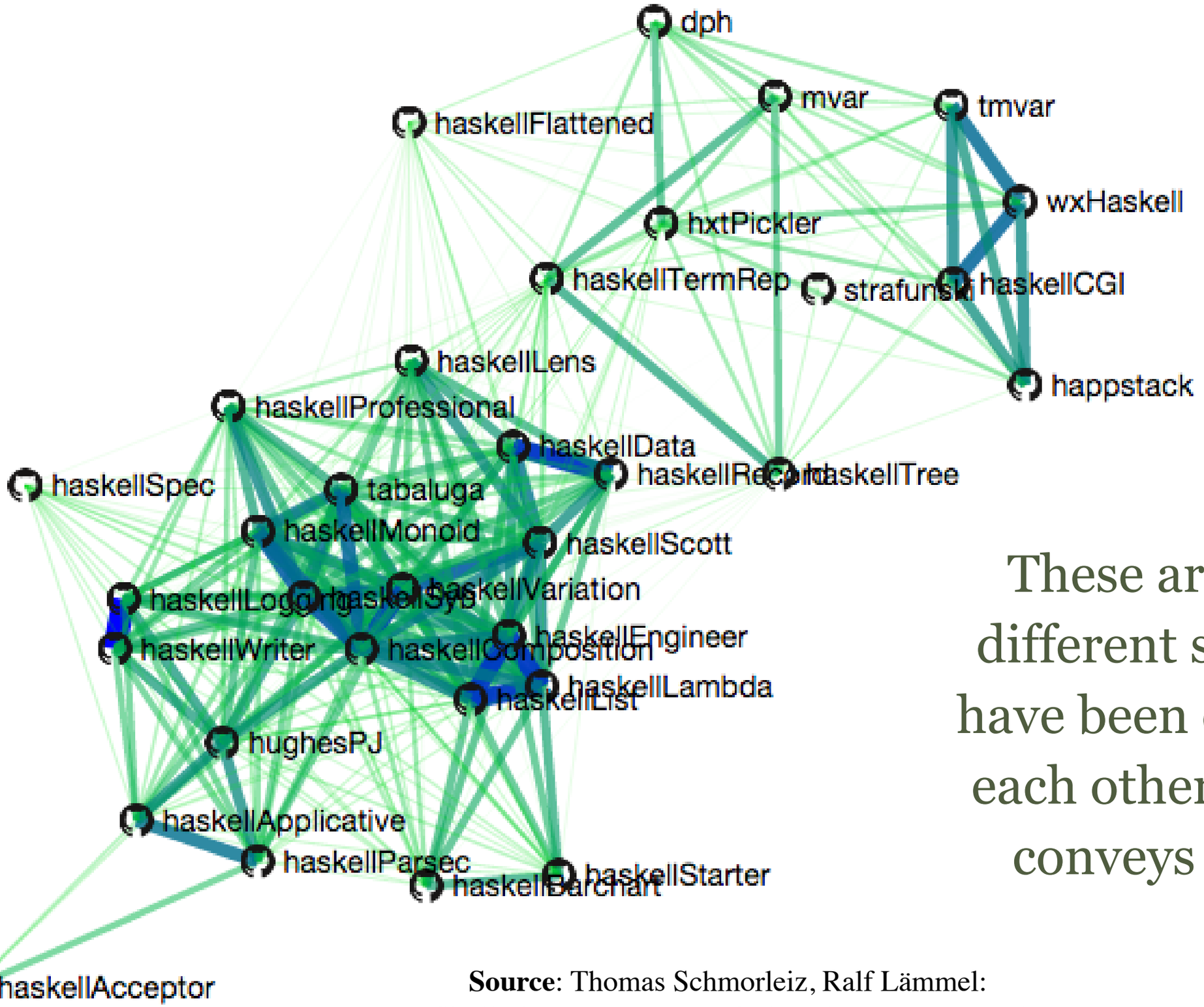
[PostgreSQL Drivers] postgresql:postgresql:7.2
[PostgreSQL Drivers] postgresql:postgresql:7.4.1-jdbc3
[PostgreSQL Drivers] org.postgresql:postgresql:9.4.1212
[PostgreSQL Drivers] org.postgresql:postgresql:9.4-1201-jdbc41
[PostgreSQL Drivers] postgresql:postgresql:9.1-901-1.jdbc4
[JDBC Extensions] org.springframework:spring-jdbc:4.2.5.RELEASE
[JDBC Extensions] org.springframework:spring-jdbc:1.2.9
[JDBC Extensions] org.springframework:spring-jdbc:2.0.6
[MySQL Drivers] mysql:mysql-connector-java:3.0.10
[MySQL Drivers] mysql:mysql-connector-java:5.0.5
[MySQL Drivers] mysql:mysql-connector-java:5.1.38
[MySQL Drivers] mysql:mysql-connector-java:5.1.6

We use the data mining technique of ‘hierarchical clustering’ to compare APIs based on extracted class and method names.

Source: Johannes Härtel, Hakan Aksu, and Ralf Lämmel:
Classification of APIs by Hierarchical Clustering.
ICPC 2018

Yet another research question:

How to detect and consolidate ‘clone and own’?

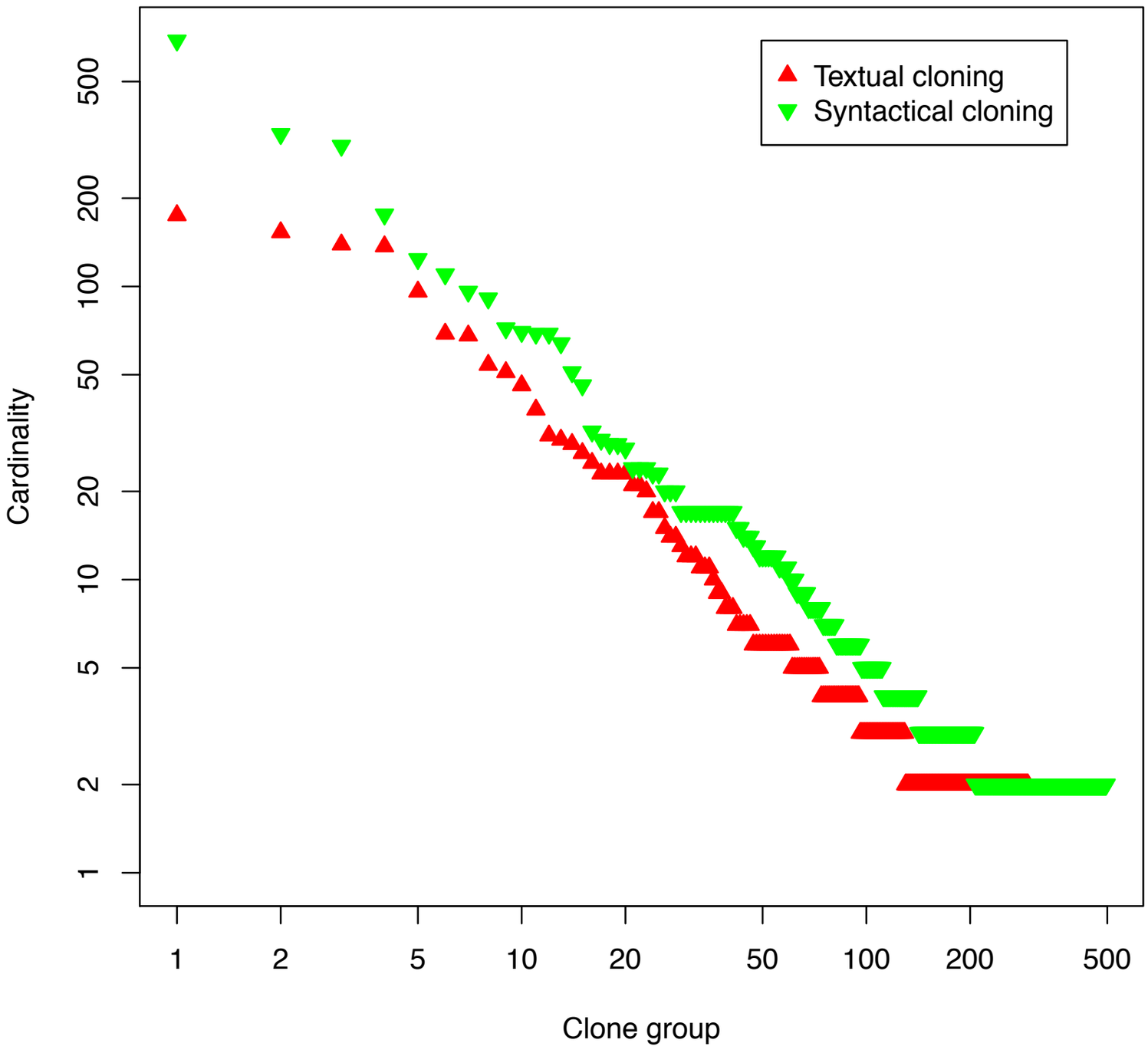


These are about 40 different systems that have been derived from each other. The layout conveys similarity.

Source: Thomas Schmorleiz, Ralf Lämmel:
Similarity management of 'cloned and owned' variants.
SAC 2016

Yet another research question:

How is a language used in the wild?

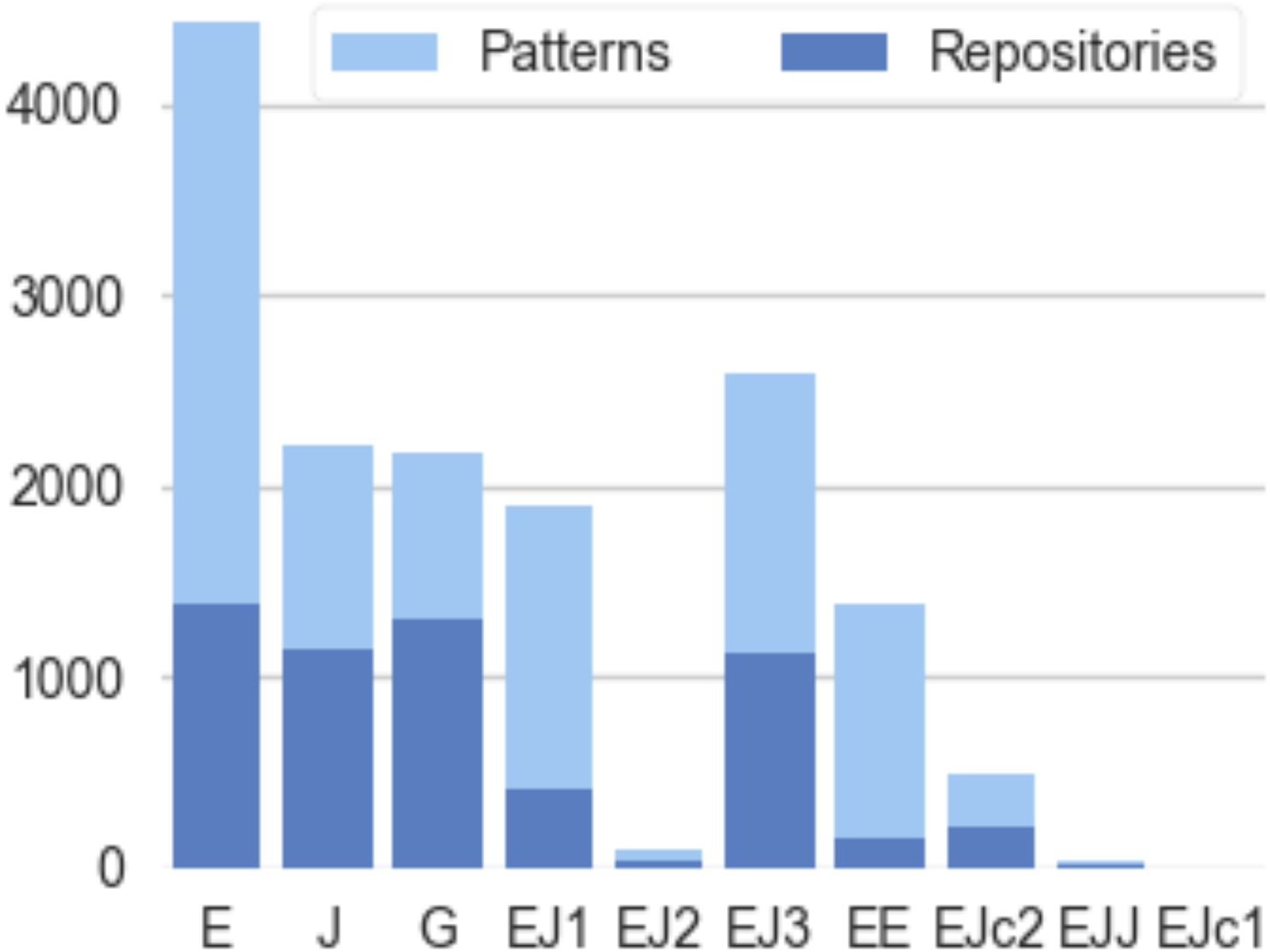


We were able to determine that most privacy policies in the world are ‘clones’ with just a few clone groups defining common use cases.

Source: Ralf Lämmel, Ekaterina Pek:
Understanding privacy policies - A study in empirical analysis of language usage.
Empirical Software Engineering 2013

A current research question:

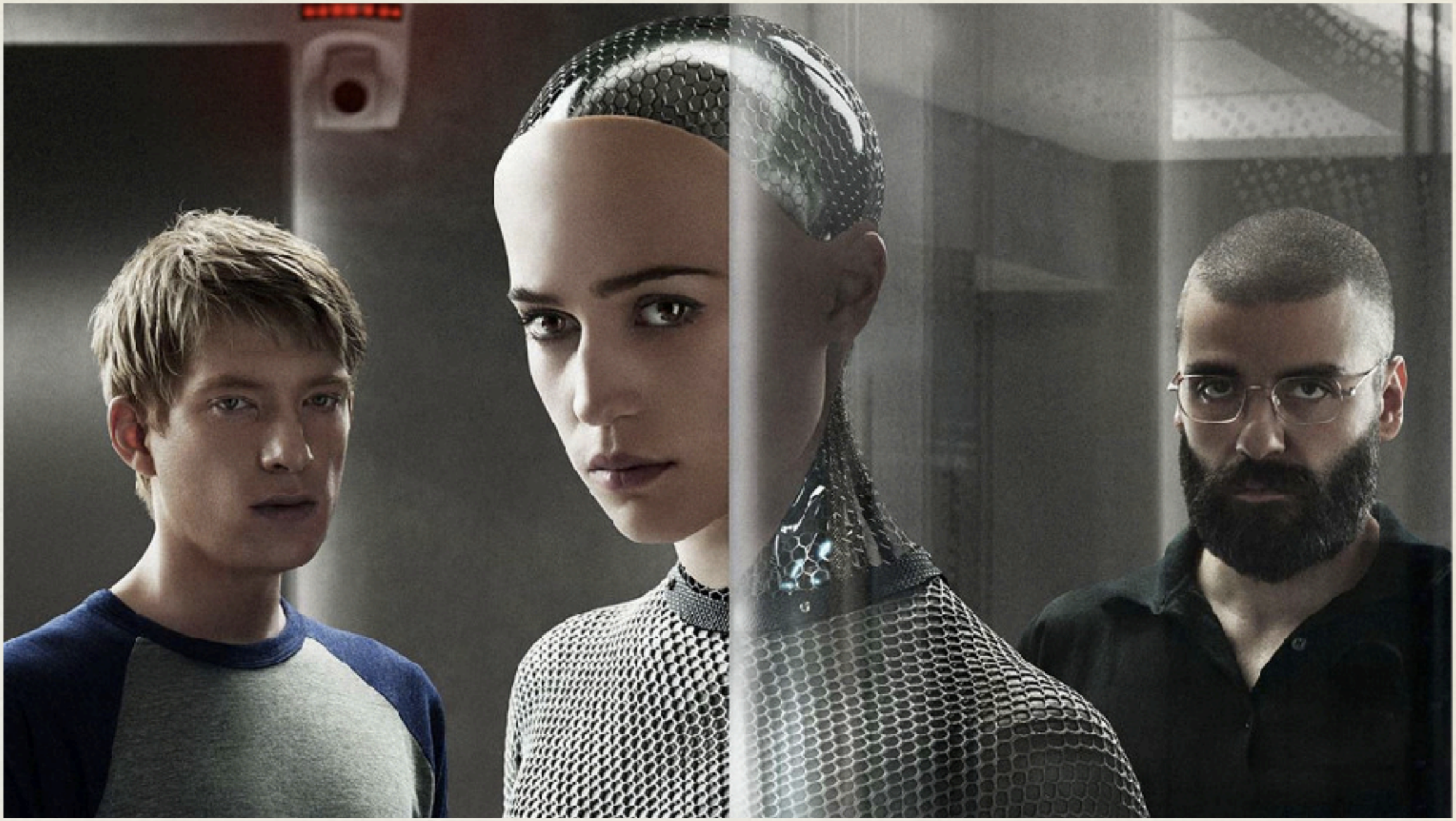
What are the patterns of technology usage across projects?



These are patterns related to the EMF technology. For instance, EJc2 models that metamodel and derived Java code are not in sync in a certain way.

Source: Johannes Härtel, Marcel Heinz, and Ralf Lämmel:
EMF Patterns of Usage on GitHub.
Submitted for publication. 2018

Is it all about *machine learning?*



Big Code Science is about much more than *machine learning!*

Technical aspects

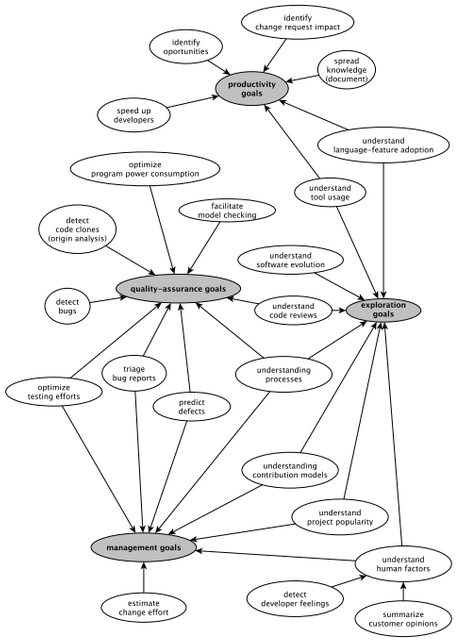
- Syntax and semantics of languages
- Technological spaces (e.g., Java vs. Ruby)
- Software development platforms (e.g., GitHub)

Big Code Science is about much more than *machine learning!*

Scientific and methodological aspects

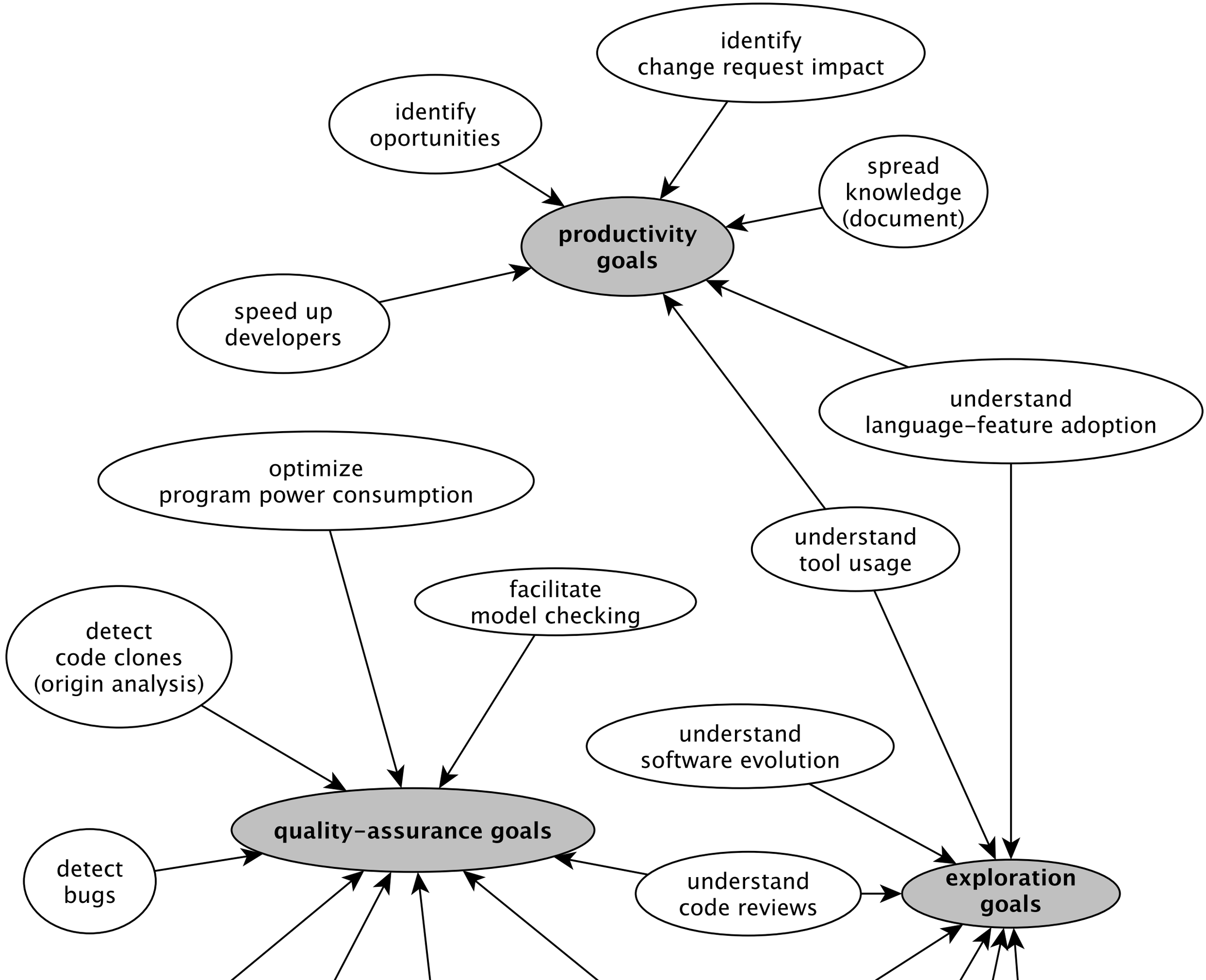
- Development of hypotheses
- Reproducibility of studies
- Realization of benefits
- Parameters

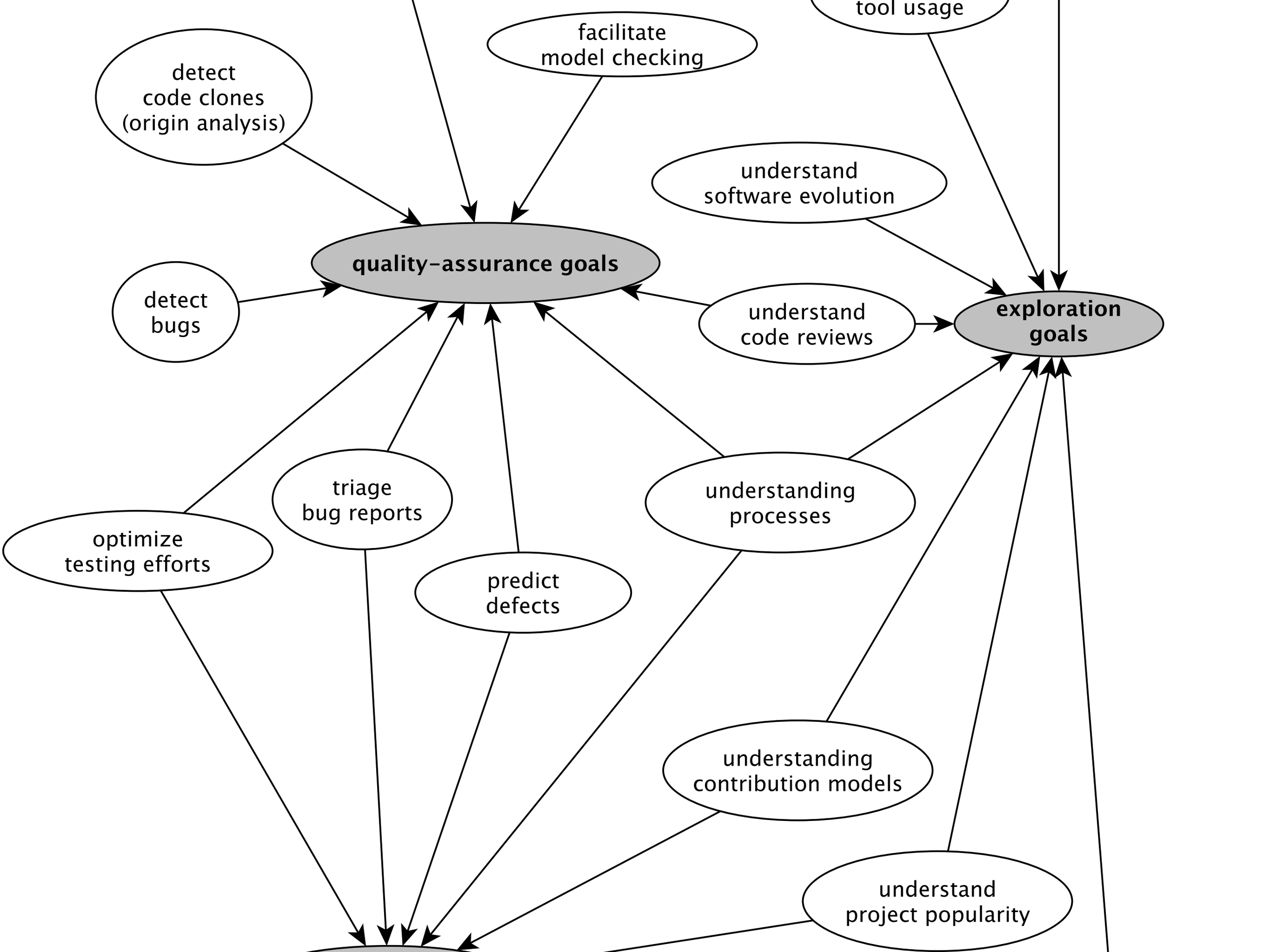
Goals of mining

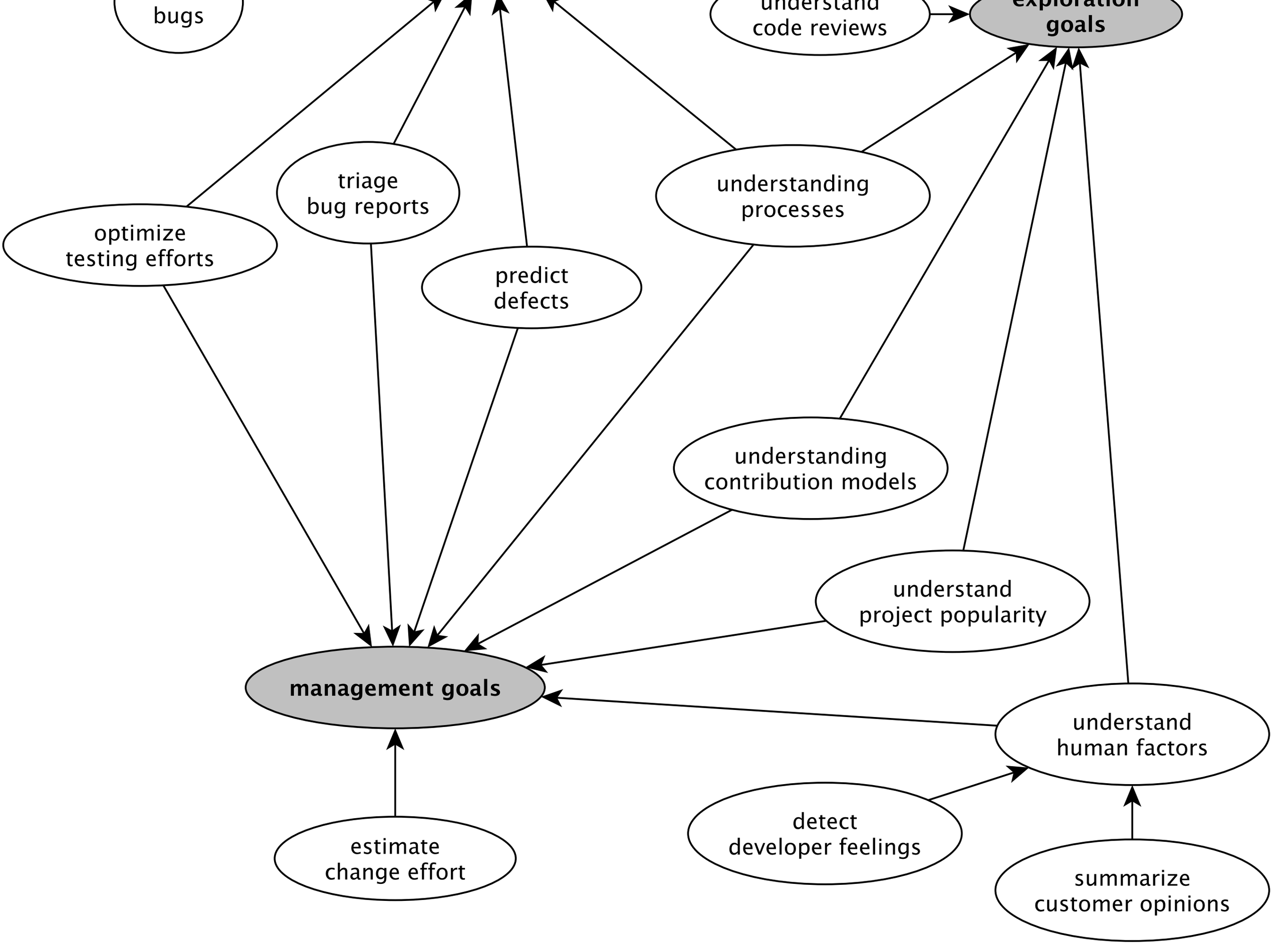


* *Productivity goals*
* *Quality-assurance goals*
* *Management goals*
* *Exploration goals*

Source: Sven Amann, Stefanie Beyer, Katja Kevic, Harald C. Gall: Software Mining Studies: Goals, Approaches, Artifacts, and Replicability. LASER Summer School 2014: 121-158


identify
change request impact
identify
oportunities
spread
knowledge
(document)
productivity
goals
speed up
developers
understand
language-feature adoption
optimize
program power consumption
understand
tool usage
facilitate
model checking
detect
code clones
(origin analysis)
understand
software evolution
quality-assurance goals
detect
bugs
understand
code reviews
exploration
goals


tool usage
facilitate
model checking
detect
code clones
(origin analysis)
understand
software evolution
quality-assurance goals
detect
bugs
understand
code reviews
exploration
goals
triage
bug reports
understanding
processes
optimize
testing efforts
predict
defects
understanding
contribution models
understand
project popularity

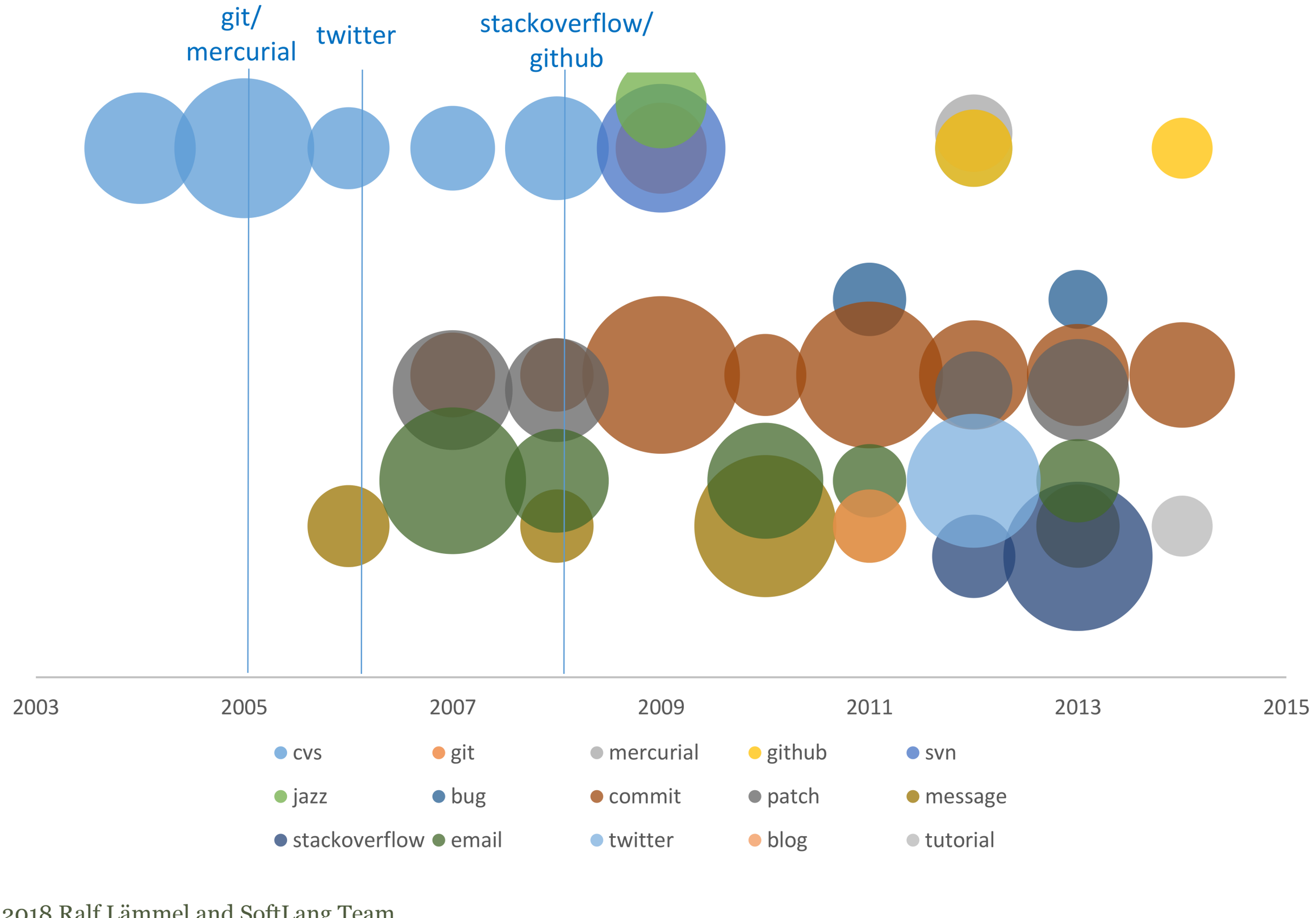

bugs
understand
code reviews
exploration
goals
triage
bug reports
understanding
processes
optimize
testing efforts
predict
defects
understanding
contribution models
understand
project popularity
management goals
understand
human factors
estimate
change effort
detect
developer feelings
summarize
customer opinions

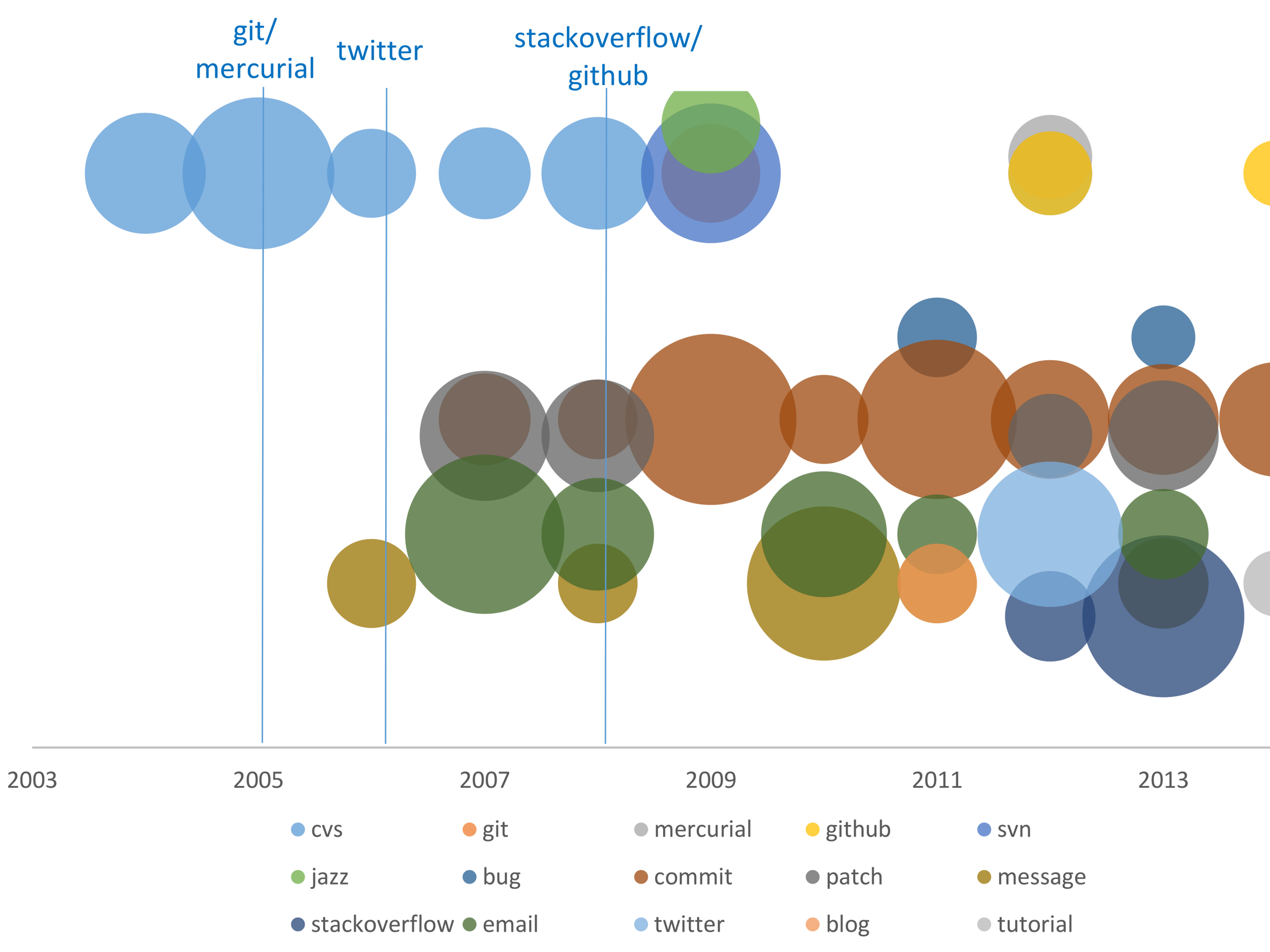
- cvs
- git
- mercurial
- github
- svn
- jazz
- bug
- commit
- patch
- message
- stackoverflow
- email
- twitter
- blog
- tutorial

Artifacts

subjected to mining

Source: Sven Amann, Stefanie Beyer, Katja Kevic, Harald C. Gall: Software Mining Studies: Goals, Approaches, Artifacts, and Replicability. LASER Summer School 2014: 121-158




git/
mercurial
twitter
stackoverflow/
github
2003
2005
2007
2009
2011
2013
cvs
git
mercurial
github
svn
jazz
bug
commit
patch
message
stackoverflow
email
twitter
blog
tutorial

git/
mercurial

twitter

stackoverflow/
github

2005

2007

2009

2011

2013

2015

cvs

git

mercurial

github

svn

jazz

bug

commit

patch

message

stackoverflow

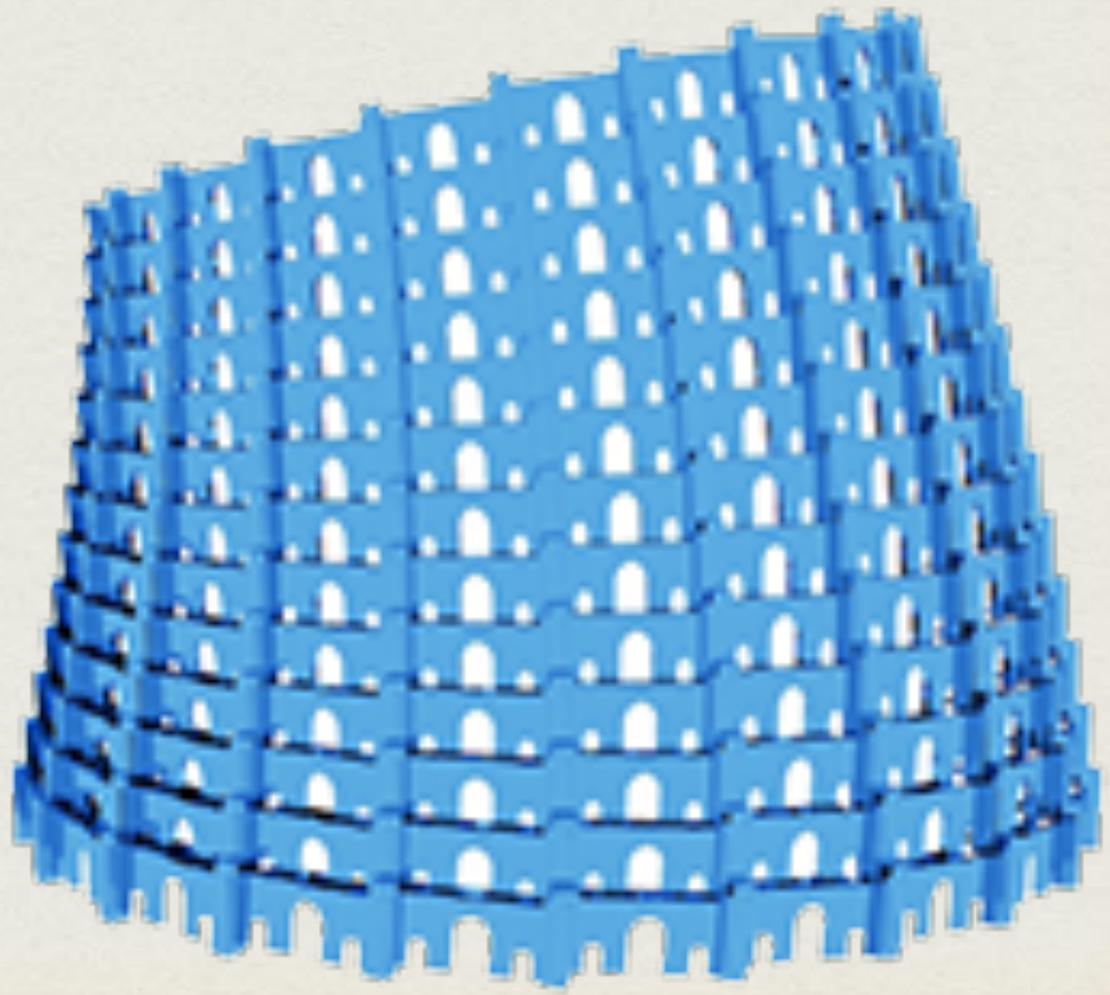
email

twitter

blog

tutorial

Thanks!



http://www.softlang.org/