Mining Sociotechnical Information From Software Repositories

Marco Aurélio Gerosa

University of São Paulo, Brazil

NAPSoL

October/2014

Repositories of repositories

github SOCIAL CODING

11.3 millions repositories
5.4 millions users
In 2013:

- 3 millions new users
- 152 millions pushes
- 25 millions comments
- 14 millions issue
- 7 millions pull requests

https://github.com/about/press
http://octoverse.github.com/

Google code

250K projects

Bitbucket

93K projects
1 million users

Ohloh | BLACK DUCK Open HUB

661K projects
29 billions of lines of codes
3 millions users

http://www.ohloh.net/

launchpad

30K projects

https://launchpad.net

CodePlex

36K projects

http://en.wikipedia.org/wiki/CodePlex

GITORIOUS

33K projects

sourceforge

324K projects
3.4 millions developers

http://sourceforge.net/apps/trac/sourceforge/wiki/What%20is%20SourceForge.net

The Apache Software Foundation http://www.apache.org/

200 projects

http://projects.apache.org/indexes/alpha.html

http://en.wikipedia.org/wiki/Comparison_of_open_source_software_hosting_facilities

Mining

Data mining = "computational process of discovering patterns in large data sets"

"The **Mining Software Repositories** (MSR) field analyzes the rich data available in software repositories to uncover interesting and actionable information about software systems and projects

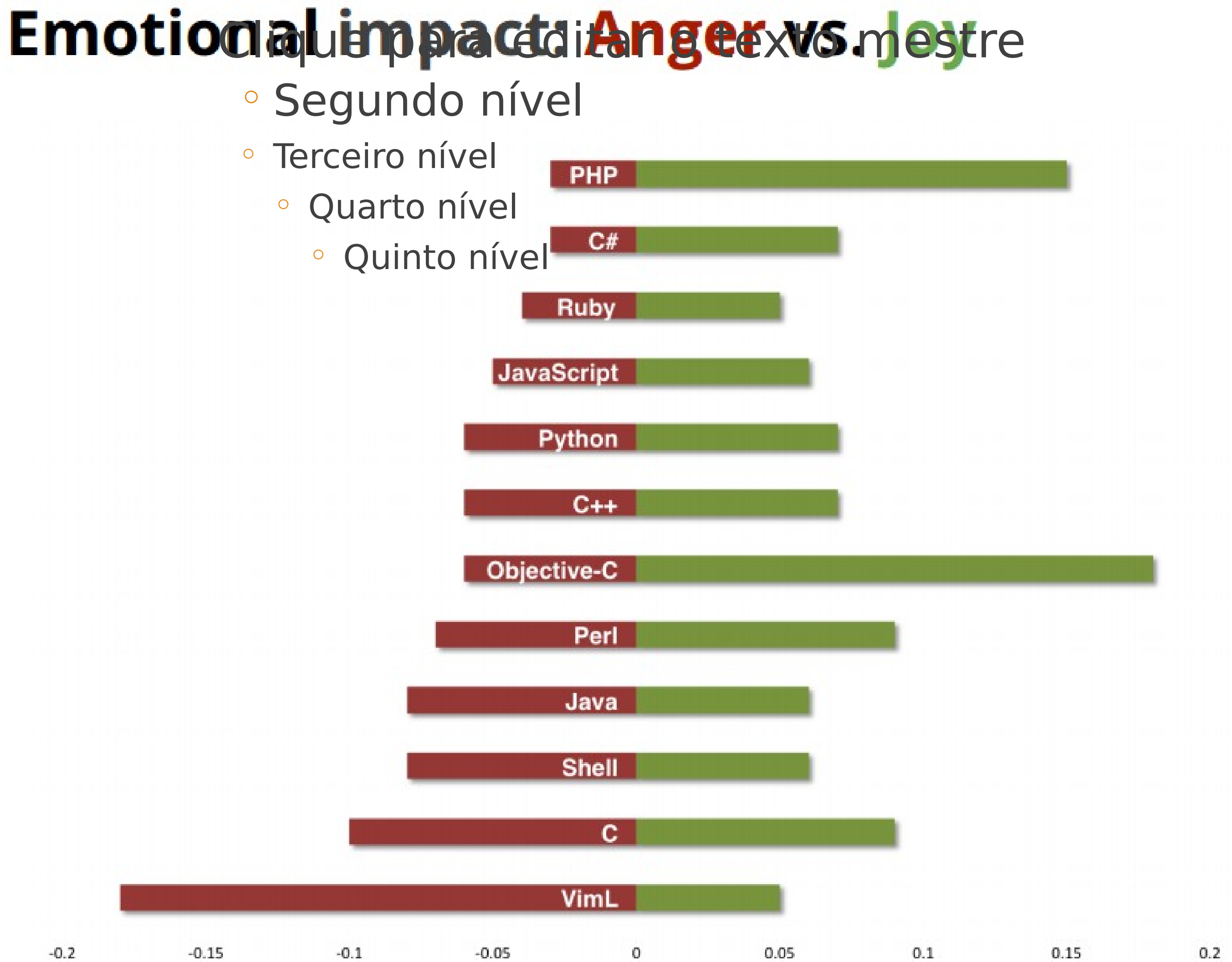
http://2014.msrconf.org/

Programmers of a given language are happier than the others?

Sentiment analysis on commits

Clique para editar o texto mestre

- Segundo nível
- Terceiro nível
  - Quarto nível
    - Quinto nível



GitHub Data Challenge 2nd place

http://www.igvita.com/slides/2012/bigquery-github-strata.pdf

http://geeksta.net/geeklog/exploring-expressions-emotions-github-com messages/

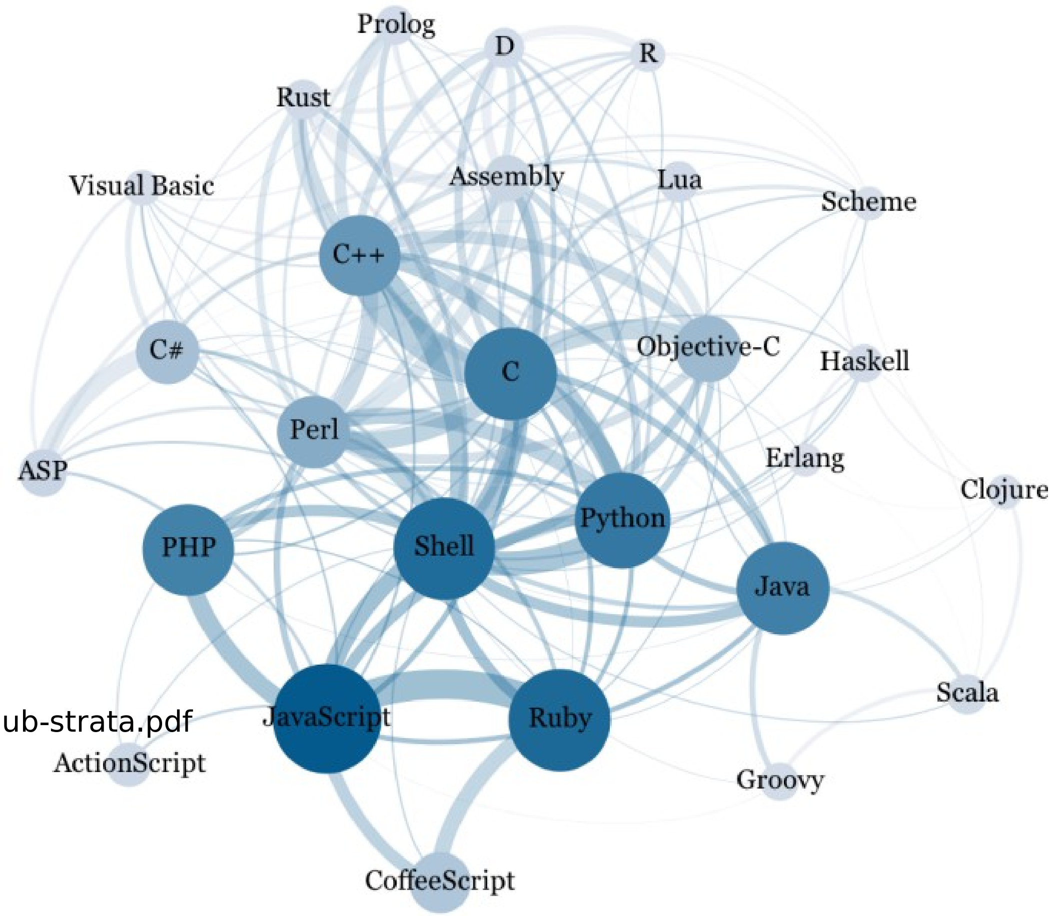
If a programmer knows a language, which others does she know?

Programming language relations

A **Ruby** programmer is ***very likely to know*** **JavaScript**, while a **Perl** programmer is not.

Java is a popular language, but stands primarily alone.

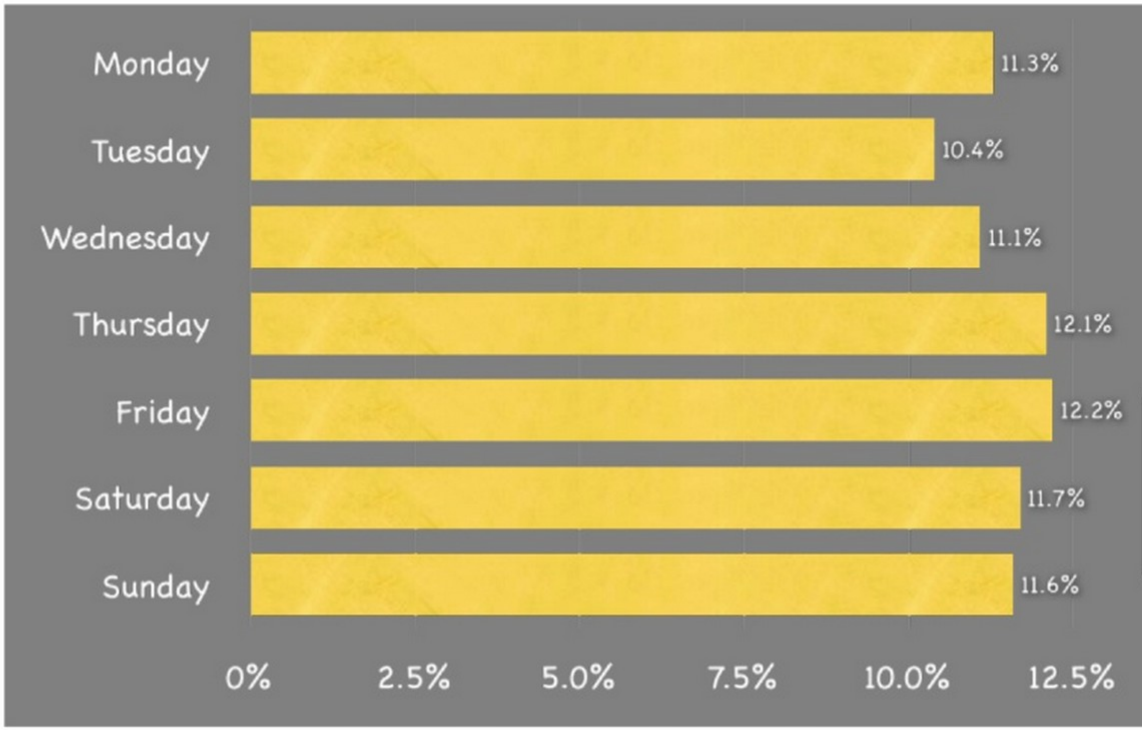
http://www.igvita.com/slides/2012/bigquery-github-strata.pdf



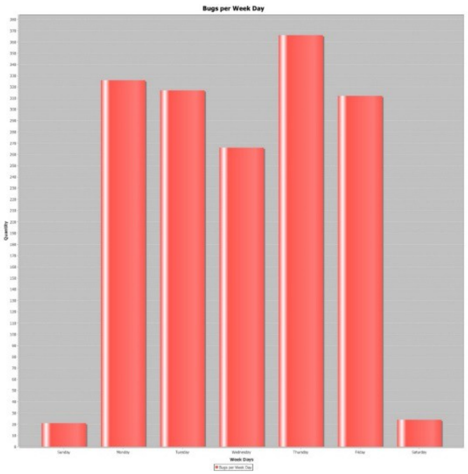
https://github.com/mjwillson/ProgLangVisualise

When are the bugs inserted?

Don’t program on Fridays



Percentage of bug-introducing changes for eclipse

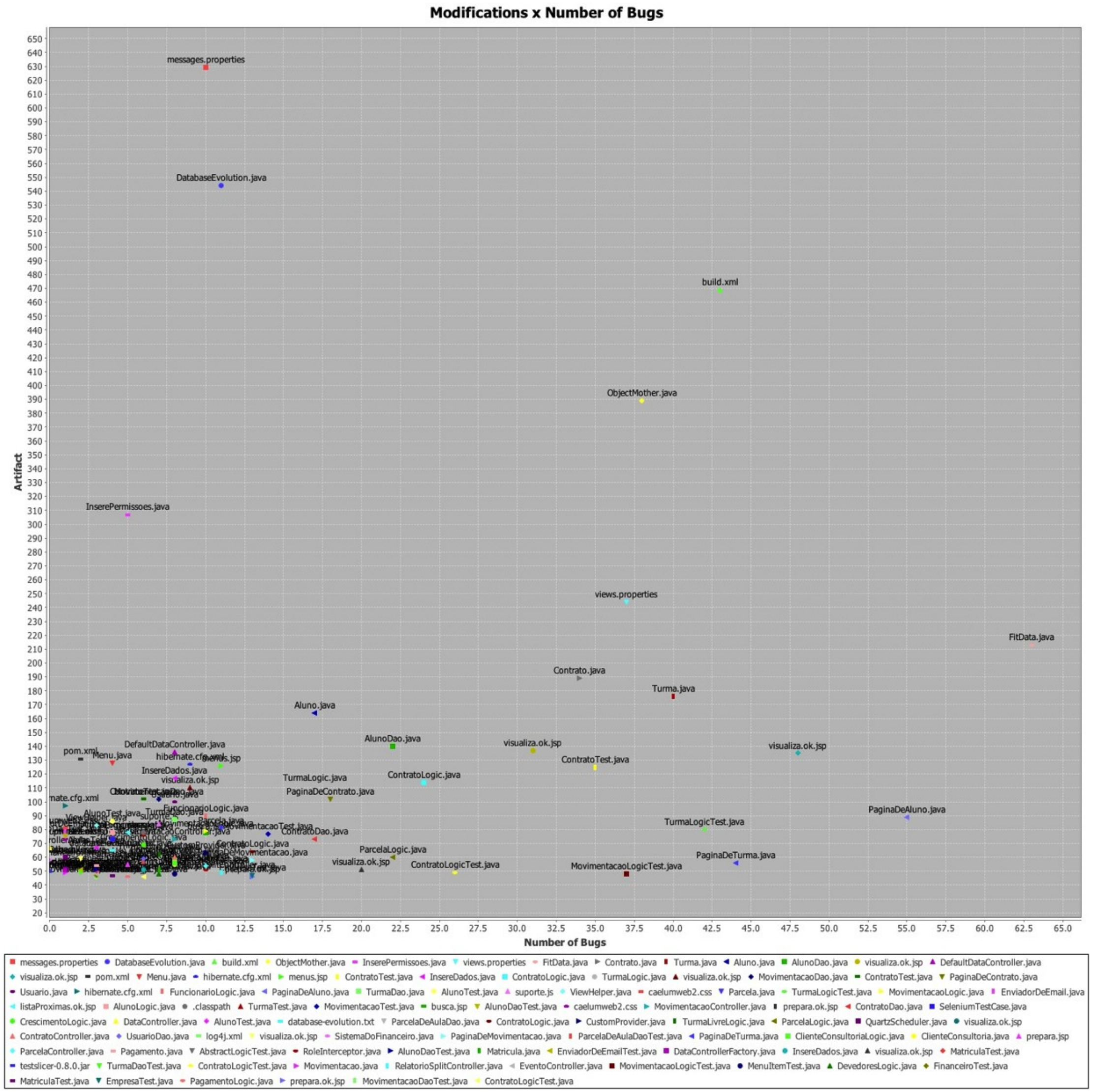


[Zimmermann et al. 05]

http://www.slideshare.net/taoxiease/software-analytics-towards-software-mining that-matters

Which files are more buggy?

Which files are more buggy?



And what about social data?

Is it important for software engineering?

David Parnas

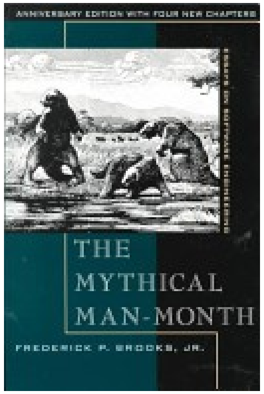
David Parnas

"Software Engineering = **Multi-person** development of multi-version programs"

Frederick Brooks

Frederick Brooks

"**To avoid disaster, all the teams working on a project should remain in** **contact** **with each other in as many ways as possible"**



http://en.wikipedia.org/wiki/The_Mythical_Man-Month#Communication

Conway’s law

design organization criteria

HOW DO COMMITTEES INVENT?

by MELVIN E. CONWAY

That kind of intellectual activity which creates a useful whole from its diverse parts may be called the *design* of a *system*. Whether the particular activity is the creation of specifications for a major weapon system, the formation of a recommendation to meet a social challenge, or the programming of a computer, the general activity is largely the same.

Typically, the objective of a design organization is the creation and assembly of a document containing a coherently structured body of information. We may name this information the *system design*. It is typically produced for a sponsor who usually desires to carry out some activity guided by the system design. For example, a public official may wish to propose legislation to avert a recurrence of a recent disaster, so he appoints a team to explain the catastrophe. Or a manufacturer needs a new product and designates a product planning activity to specify what should be introduced.

The design organization may or may not be involved in the construction of the system it designs. Frequently, in public affairs, there are policies which discourage a group's acting upon its own recommendations, whereas, in private industry, quite the opposite situation often prevails.

It seems reasonable to suppose that the knowledge that one will have to carry out one's own recommendations or that this task will fall to others, probably affects some design choices which the individual designer is called upon to make. Most design activity requires continually making choices. Many of these choices may be more than design decisions; they may also be personal decisions the designer makes about his own future. As we shall see later, the incentives which exist in a conventional management environment can motivate choices which subvert the intent of the sponsor.[1]

ing a design team means that certain design decisions have already been made, explicitly or otherwise. Given any design team organization, there is a class of design alternatives which cannot be effectively pursued by such an organization because the necessary communication paths do not exist. Therefore, there is no such thing as a design group which is both organized and unbiased.

Once the organization of the design team is chosen, it is possible to delegate activities to the subgroups of the organization. Every time a delegation is made and somebody's scope of inquiry is narrowed, the class of design alternatives which can be effectively pursued is also narrowed.

Once scopes of activity are defined, a coordination problem is created. Coordination among task groups, although it appears to lower the productivity of the individual in the small group, provides the only possibility that the separate task groups will be able to consolidate their efforts into a unified system design.

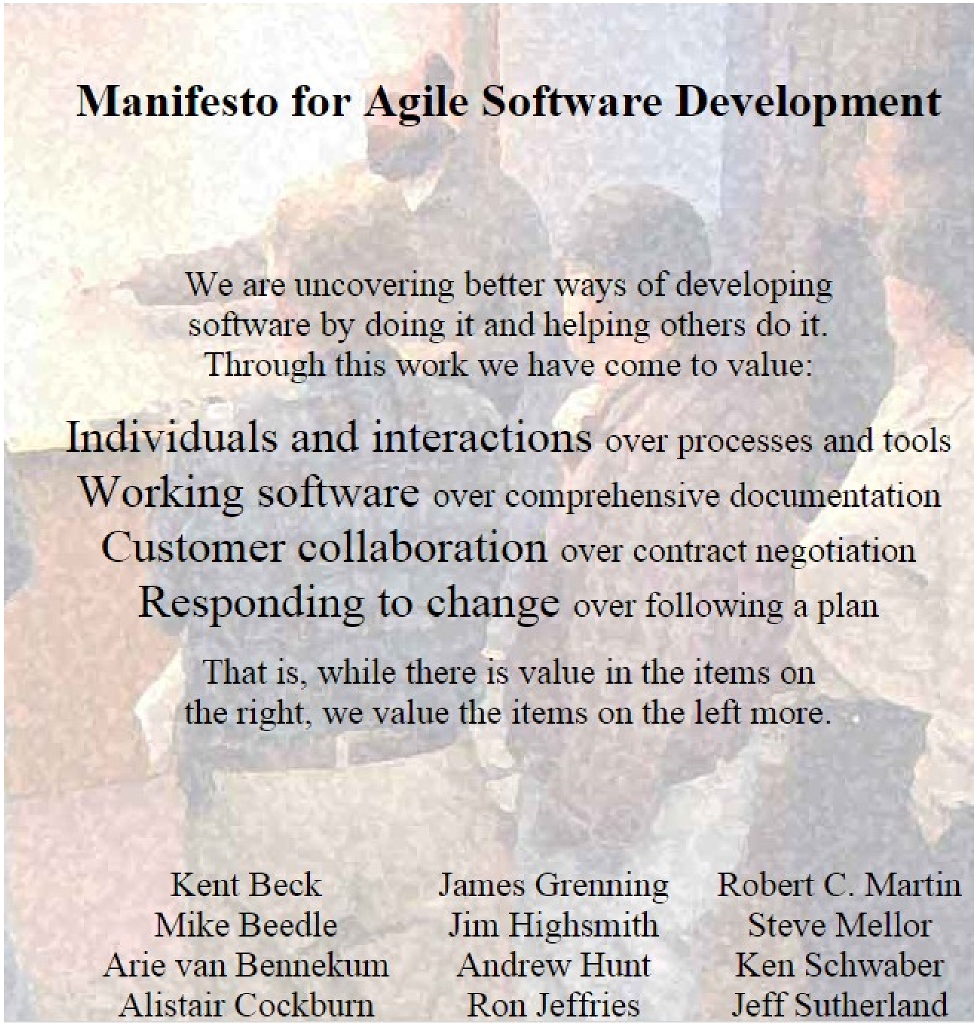
Thus the life cycle of a system design effort proceeds through the following general stages:

1. Drawing of boundaries according to the ground rules.
2. Choice of a preliminary system concept.
3. Organization of the design activity and delegation of tasks according to that concept.
4. Coordination among delegated tasks.
5. Consolidation of subdesigns into a single design.

It is possible that a given design activity will not proceed straight through this list. It might conceivably reorganize upon discovery of a new, and obviously superior, design concept; but such an appearance of uncertainty is unflattering, and the very act of voluntarily abandoning a creation is painful and expensive. Of course, from the

“Organizations which design systems are constrained to produce designs which are copies of the **communication** structures of these organizations”

Agile manifesto



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland

“**Individuals and interactions** over processes and tools”

Social aspects matter!

Mining software repositories

Source code and artifacts

Discussion lists
Comments on issues
Code comments
User reports
Q&A sites
Social media

Issue trackers
Project management systems
Reputation systems

aboration and software production

Mining

Information about a project

Information about an ecosystem

Information about Software Engineering


Practitioner Researcher

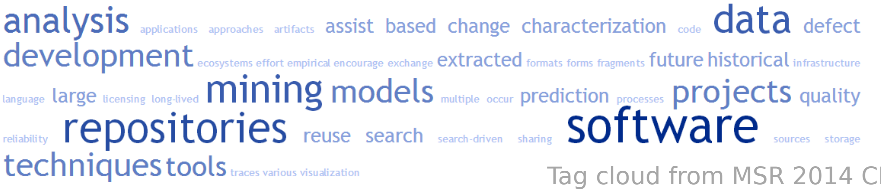
Applications

Decision making

Software understanding

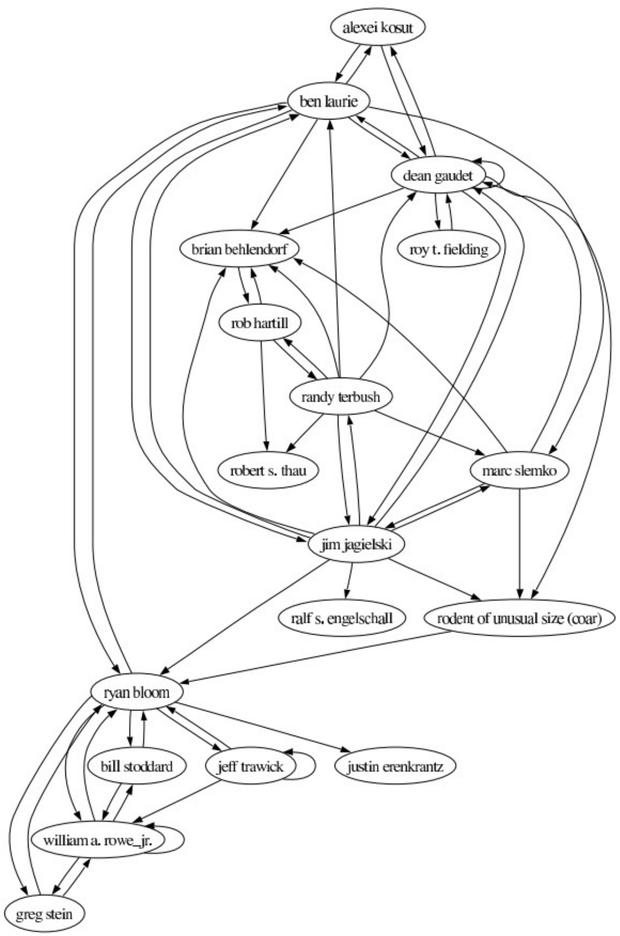
Support maintenance

Empirical validation of ideas & techniques



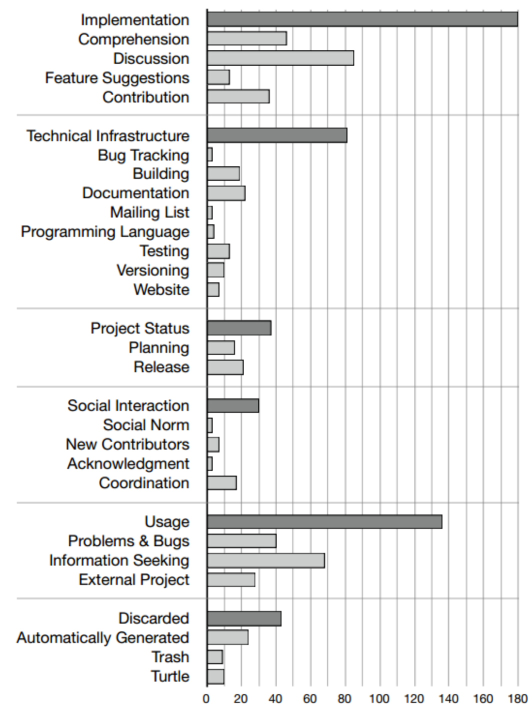
Tag cloud from MSR 2014 CFP

Mining discussion lists



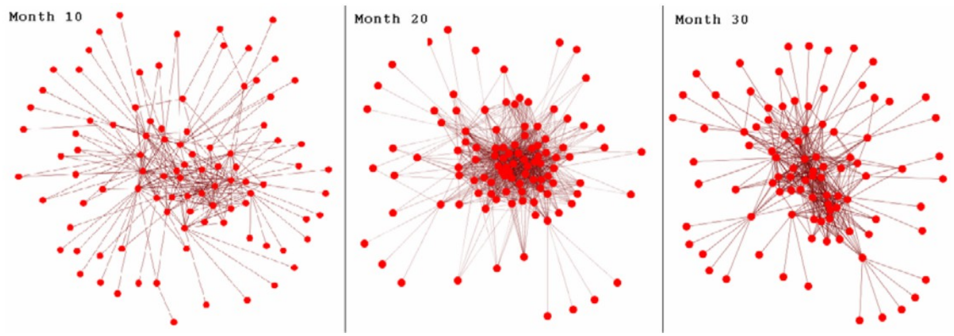
Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. 2006. Mining email social networks. MSR 2006

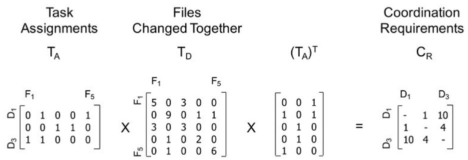
What is the discussion list for?



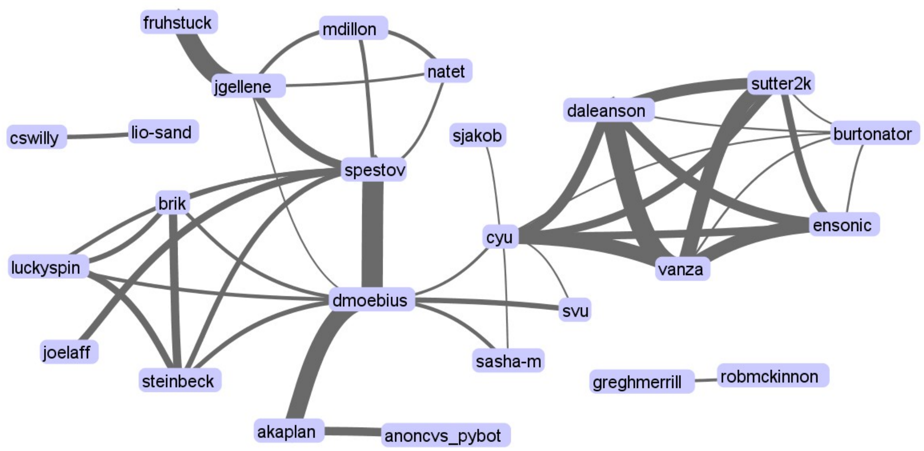
Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, and Arie van Deursen. 2013. Communication in open source software development mailing lists. MSR 2013

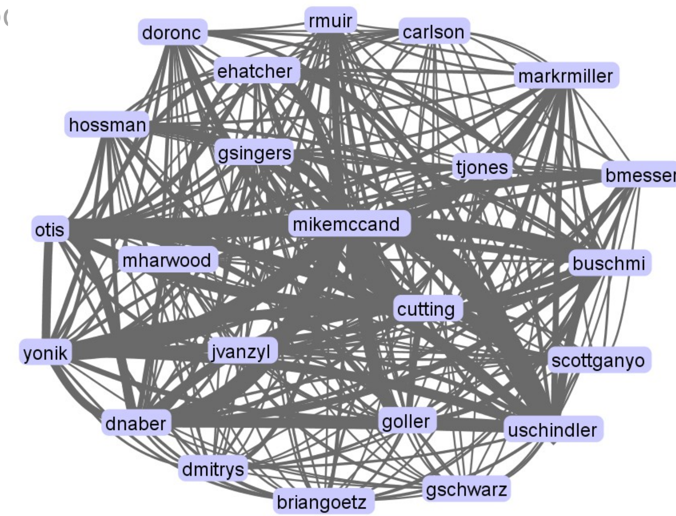
Coordination requirements





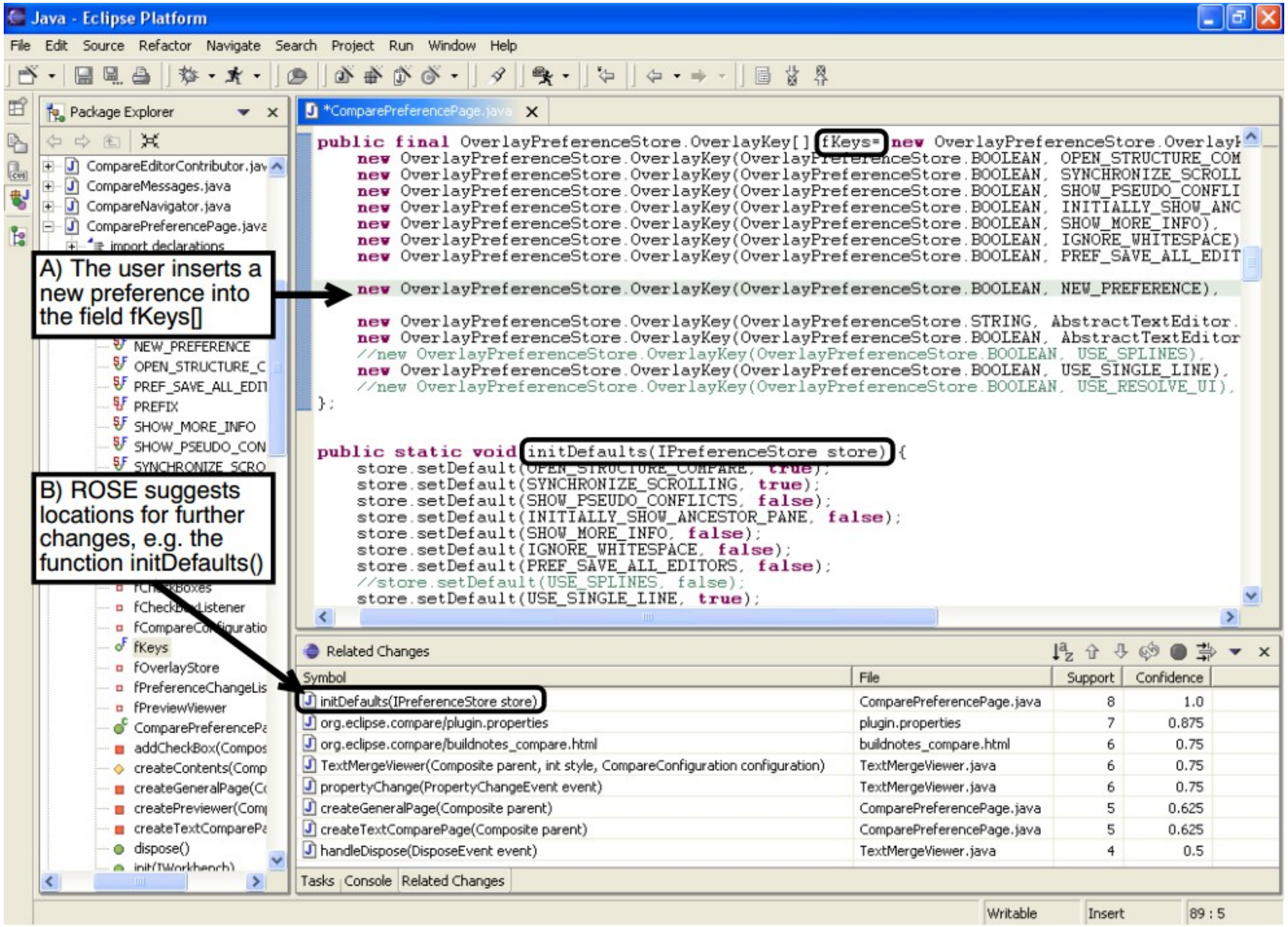
Cataldo, M., Dependencies in geographically distribut
software development: Overcoming the limits of
mod





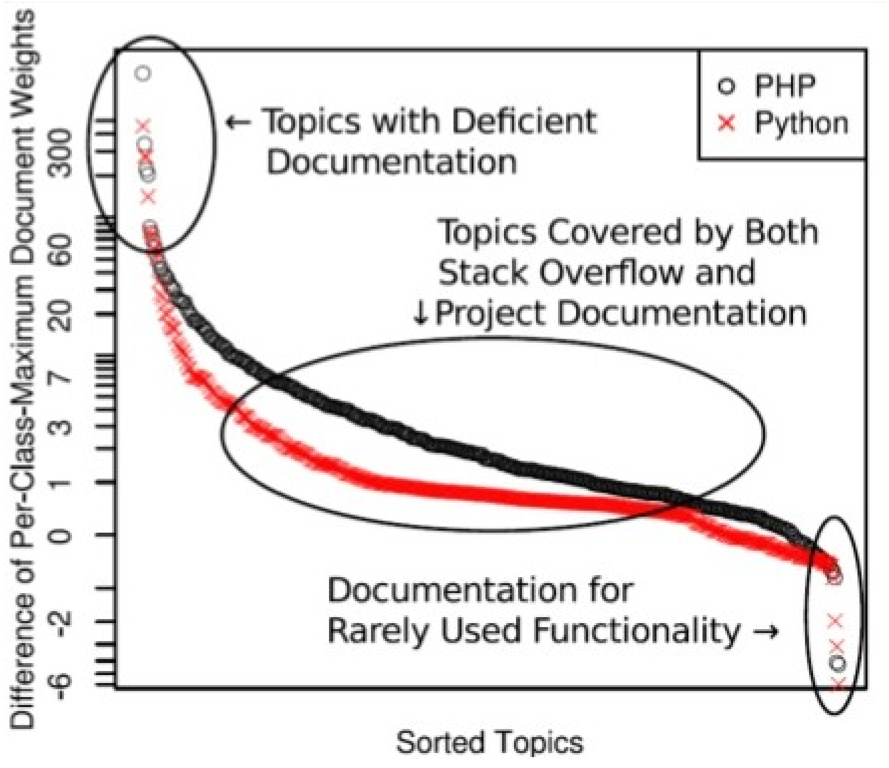
Santana, F. et a. "XFlow: An Extensible Tool for
Empirical Analysis of Software Systems Evolutio
ESELAW 2011

Programmers who changed this function also changed …



Thomas Zimmermann, Peter Weissgerber, Stephan Diehl, and Andreas Zeller. 2005. Mining Version Histories to Guide Software Changes. *IEEE Trans. Software Eng.* 31, 6 (June 2005), 429-445. DOI=10.1109/TSE.2005.72 http://dx.doi.org/10.1109/TSE.2005.72

Lack of documentation



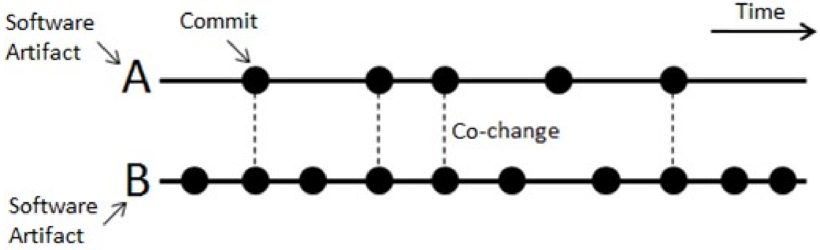
Deficient Documentation Detection: A Methodology to Locate Deficient Project Documentation using Topic Analysis, Joshua Charles Campbell, Chenlei Zhang, Zhen Xu, Abram Hindle, and James Miller, MSR 2013

Some of our work

How to identify change dependencies?

Change dependencies

Files frequently changed together share some sort of dependency [Gall et al. 1998]



Strong change dependency from B to A (the opposite is a much weaker dependency)

A logical dependency denotes an implicit and evolutionary relationship between software artifacts

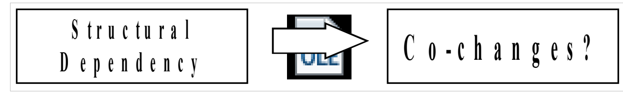
The concept has proven useful in several different studies:

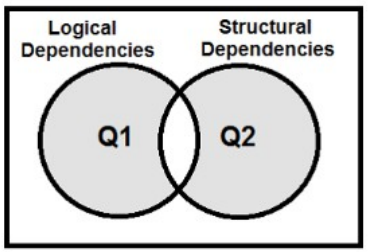
- Software quality analysis [Cataldo & Nambiar, 2010]
- Bugs prediction [D’Ambros et al. , 2009a]
- Change prediction and change impact analysis [Zimmermann et al., 2005]
- Uncover cross-cutting concerns [Adams et al., 2010]
- Uncover design flaws and opportunities for refactoring [D’Ambros et al., 2009b]
- Understand and evaluate software architecture [Zimmermann et al., 2003]
- Requirements traceability [Ali et al., 2013]
- Maintain documentation [Kagdi et al., 2006]

1.1) Structural x Change Dependencies

Gustavo Oliva,
PhD candidate

Overlap between change dependencies
and structural dependencies





Lowly logically coupled

Medium logically coupled

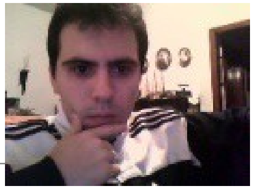
Highly logically coupled

	Number of Logical dependencies	Logical deps. w/o structural counterpart	LCOP
Confidence [0.00, 0.33]	166,378 (61.6%)	151,397	91.0%
Confidence]0.33, 0.66]	103,265 (38.2%)	94,074	91.1%
Confidence]0.66, 1.00]	367 (0.1%)	341	92.9%
Total	270,010 (100%)	245,812	91.0%

lysis of 150K commits of the ASF showed that:
% of the change dependencies did not involve structural dependencies
% of the structural dependencies did not imply in a change dependency

Oliva, G.A., Gerosa, M. A. (2011) "On the Interplay between Structural and Logical Dependencies in Free Software". ***Brazilian Symposium on Software Engineering (SBES 2011)***

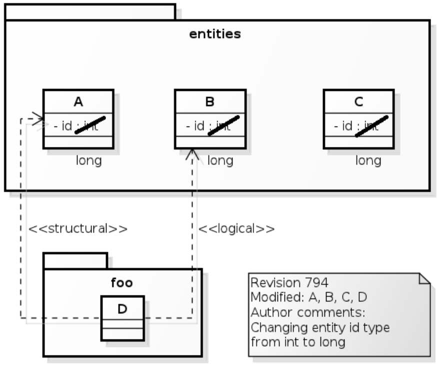
1.2) Change dependencies origins

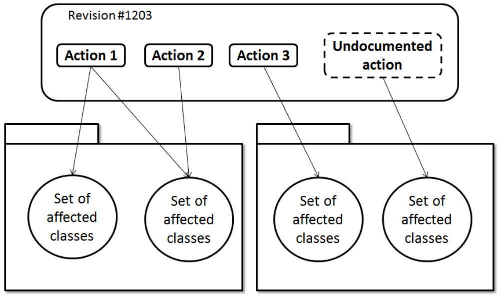


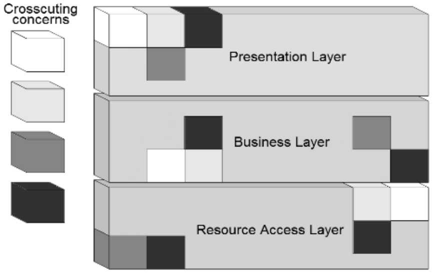
Gustavo Oliva, PhD candidate

Manual classification of commits to understand the origins of change dependencies

Category	Joint-changes	Total %
Refactoring elements that belong to a same semantic class	80	19.6%
Structural dependencies on a changing semantic class	9	2.2%
Cross-cutting concerns	165	40.4%
Overloaded revision	60	14.7%
Repository operations	21	5.1%
Structural dependencies on specific elements	66	16.2%
Other reasons	7	1.7%
Total	408	

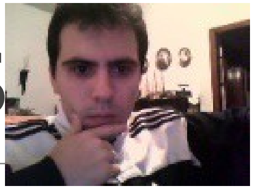






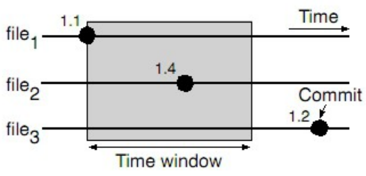
*Oliva, G.A., Santana, F., Gerosa, M. A., Souza, C. (2011) "Towards a Classification of Logical Dependencies Origins: A Case Study". **Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution (IWPSE-EVOL '11)***

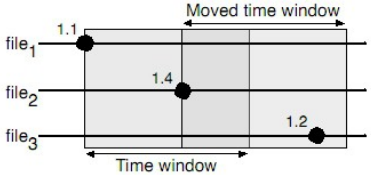
1.3) Preprocessing commits

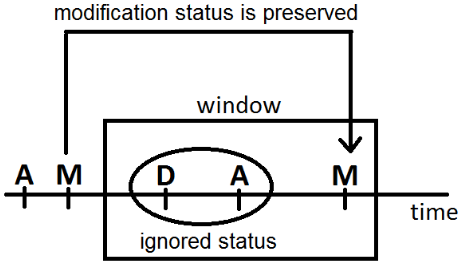


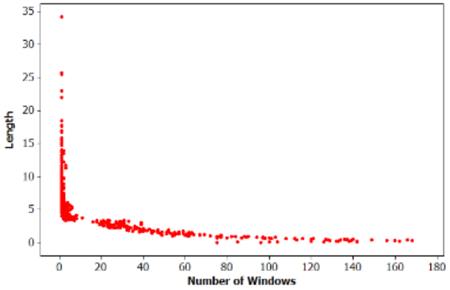
Gustavo Oliva, PhD candidate

Commit = change?
Using the sliding time window approach [Zimmermann & Weißgerber, 2004] to group SVN commits





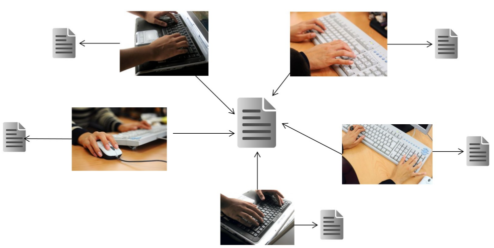




	N	Sum	Mean	StDev	Skewness	Kurtosis
Before	479,794	3,206,900	6.68	37.84	33.80	1,844.00
After	453,865	3,174,051	6.99	40.79	39.56	2,829.94

Evaluation in the Apache code repository showed that the produced grouping corresponded to **4.6%** of the number of commits

What about commit habits/practices/policies?
Social aspects matter!



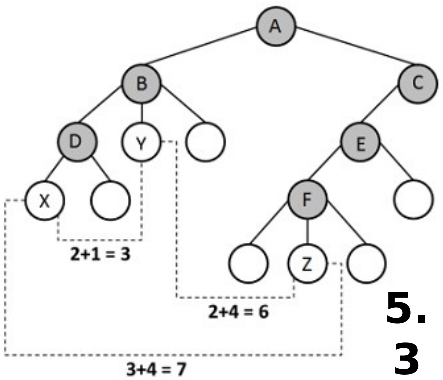
*Oliva, G. A., Santana, F., Gerosa, M. A., Souza, C. (2012), "Preprocessing Change-Sets to Improve Logical Dependencies Identification", **6th Int. Workshop on Software Quality and Maintainability (SQM 2012)***

How to identify design degradation?

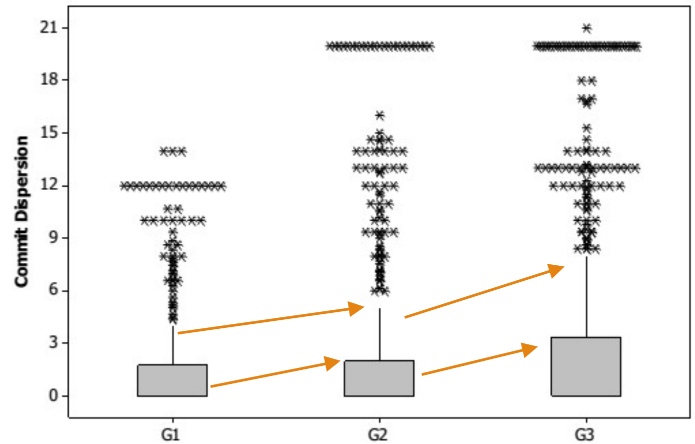
2) Design degradation identification

Gustavo Oliva, PhD candidate

Rigidity and fragility [Martin & Martin, 2006] identification based on commit metadata



5. 3



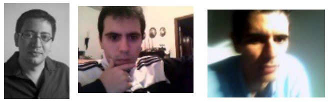
Rigidity => designs difficult to change due to ripple effects => commit density (number of changed files per commit)
Fragility => designs break in different areas when a change is performed => commit dispersion (distance in the directory tree among file paths included in a commit)

	Trend of Increase	Increase in Med.		Increase in IQR		Increase in the % of Outliers		Increase in Mean	
	P1	P2	P3	P2	P3	P2	P3	P2	P3
Rigidity						✓	✓	✓	✓
Fragility	✓			✓	✓	✓		✓	

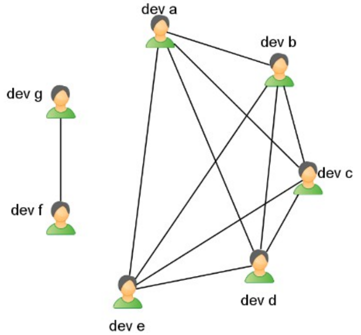
Oliva, G., Steinmacher, I., Wiese, I.S., Gerosa, M.A. "What Can Commit Metadata Tell Us About Design Degradation?", In: ***13th International Workshop on Principles on Software Evolution (IWPSE 2013)****, Saint Petersburg.*

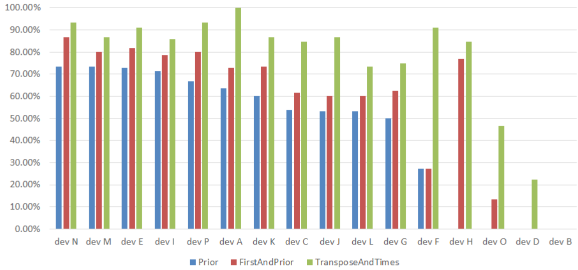
Who are the key developers?

3) Key developers characterization

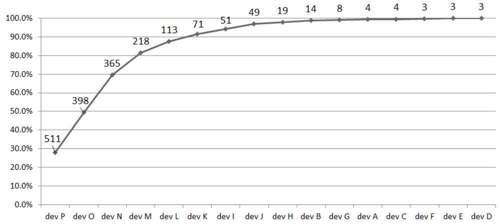


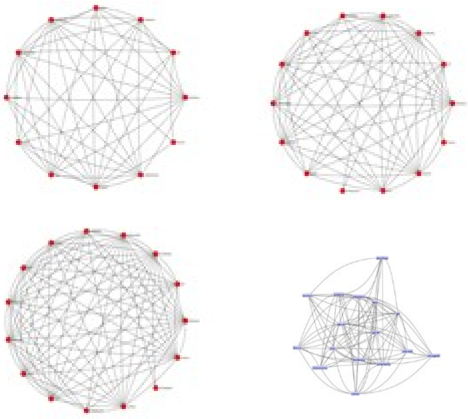
Key developers participation





Developer	Key Developer	Important in Communication Network			Core of Coordination Requirements Network	High Congruence			Top Contributors
		P	FP	TT		C1	C2	C3	
dev P	✔	✔	✔	✔	✔		✔	✔	✔
dev N	✔	✔	✔	✔	✔	✔	✔	✔	✔
dev O	✔				✔				✔
dev M	✔	✔	✔	✔	✔	✔	✔		✔
dev L					✔				
dev J				✔	✔				
dev K				✔	✔				
dev E		✔		✔		✔	✔	✔	
dev H									
dev G									
dev C									
dev F								✔	
dev I		✔				✔			
dev A				✔				✔	
dev D									
dev B									





*Oliva, G., Santana, F.W., da Silva, J. T., Oliveira, K.C.M., Werner, C.M.L., Souza, C.R.B. & Gerosa, M.A., "Evolving the System's Core: A Case Study on the Identification and Characterization of Key Developers in Apache Ant", **Computing and Informatics [to appear]**.*

Do tests characteristics indicate the quality of the code under test?

4) Unit tests' feedback for code quality

Mauricio Aniche
PhD candidate

Number of asserts indicate
- Cyclomatic complexity?
- LOC ?
- Method calls ?

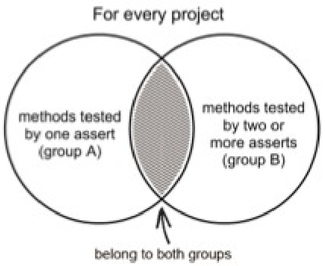


Table V
RESULTING P-VALUES FROM APPLYING WILCOXON TEST TO THE GROUP OF ONE ASSERTED OBJECT AND THE GROUP OF MORE THAN ON

Project	Cyclomatic Complexity	Method Invocations	Lines of Code
commons-codec	0.0376*	0.1376	0.0429*
commons-compress	0.0385*	0.0445*	0.4429
commons-lang	0.9918	0.7207	0.9611
commons-math	0.0262*	0.1873	0.0325*
commons-validator	0.0329*	0.6256	0.0223*
cxf-dosgi	0.5351	0.9986	0.3919
directory-shared	0.9998	0.1408	1.0000
harmony	3.5300E-05*	0.0001*	0.0043*
log4j	0.2749	0.6789	0.3574
log4j-extras	0.0658	0.6339	0.0594
maven-2	0.2912	0.6228	0.1611
maven-doxia-sitetools	0.8012	1.0000	0.8986
maven-enforcer	0.2670	0.8219	0.1348
maven-plugins	0.1461	0.1972	0.0953
maven-sandbox	0.9821	0.1400	0.9815
maven-scm	0.9726	0.9193	0.9999
Industry CW	0.9999	0.0089*	0.9986
Industry CP	0.9103	0.7909	0.3055
Industry WC	0.1274	0.4718	0.0606
rat	0.2213	0.1533	0.1373
shindig	0.0006*	0.0238*	0.0002*
struts-sandbox	0.9994	0.6910	0.9995

Table III
RESULTING P-VALUES FROM APPLYING WILCOXON TEST TO THE GROUP OF ONE ASSERT AND THE GROUP OF MORE THAN ONE ASSERT

Project	Cyclomatic Complexity	Method Invocations	Lines of Code
commons-codec	0.1457	0.0257*	0.2991
commons-compress	0.7899	0.5374	0.9122
commons-lang	0.6766	0.3470	0.8875
commons-math	0.9230	0.9386	0.9369
commons-validator	0.9477	-	0.9635
cxf-dosgi	0.8445	0.9567	0.9463
directory-shared	0.9518	0.1298	0.9972
harmony	0.0174*	0.2822	0.5676
log4j	0.9489	0.6789	0.9885
log4j-extras	0.4811	0.6339	0.5703
maven-2	0.2532	0.9490	0.4427
maven-doxia-sitetools	0.9561	0.9213	0.9595
maven-enforcer	0.9371	0.4064	0.9727
maven-plugins	0.9607	0.6946	0.9847
maven-sandbox	0.4277	0.6499	0.9133
maven-scm	0.9324	0.9846	0.9948
Industry CW	0.9999	0.2567	0.9999
Industry CP	0.2850	0.0003*	0.4425
Industry WC	0.5380	0.5381	0.0561
rat	0.3162	0.4153	0.3263
shindig	0.9968	0.0252*	0.9998
struts-sandbox	0.9758	0.2942	0.9649

- "Asserted Objects" metric presents better results than "number of asserts"
- Statistically difference in 20% of the projects

22 ASF projects
3 industry project

Table VII
SUMMARY OF THE RESULTS OF THE STATISTICAL TESTS: QUANTITY OF PROJECTS THAT REFUTED EACH HYPOTHESES

Hypotheses	Qty of Projects (Asserts)	Qty of Projects (Asserted Objects)
H1 (Cyclom. Complexity)	01 (05%)	06 (27%)
H2 (Lines of Code)	00 (00%)	05 (22%)
H3 (Method Invocations)	03 (13%)	04 (18%)

Aniche, M., Oliva, G.A., Gerosa, M.A., "What Do the Asserts in a Unit Test Tell Us about Code Quality? A Study on Open Source and Industrial Projects", **17th European Conference on Software Maintenance and Reengineering (CSMR 2013)**.

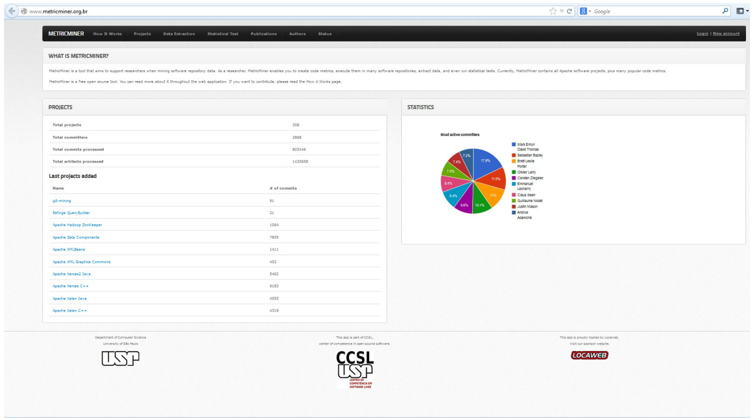
Does refactoring reduce cyclomatic complexity?

5) Refactoring

Francisco Sokol, BSc

Most part of the documented refactoring does not reduce cyclomatic complexity. However, 23% of the documented refactoring reduce cyclomatic complexity while 12% of the other commits have the same effect.

	Decrease CC	Equalized CC	Increased CC
Documented Refactoring	1504	1603	3230
No Documented Refactoring	30145	99580	121239



www.metricminer.org.br

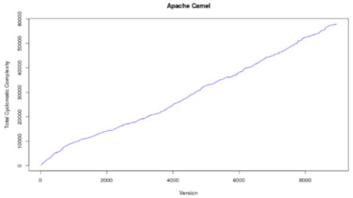


Figure 3. Complexity evolution of Camel project

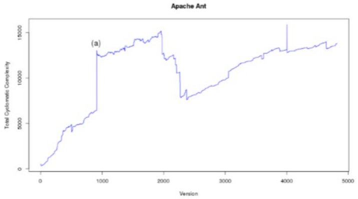


Figure 4. Complexity evolution of Ant project

Category	**Quantity (Percentage)**
Code Removed	17 (68%)
Extract Class	5 (20%)
Extract Method	1 (4%)
Class Removed	6 (24%)
Extract Interface	1 (4%)

Table 3. Distribution of the 25 Commits that Decreased CC

Category	**Quantity (Percentage)**
Functionality added	15 (60%)
Move Method	1 (4%)
Rename Method	2 (8%)
Extract Super Class	4 (16%)
Extract Class	7 (28%)
Pull Up Method	3 (12%)

Table 4. Distribution of the 25 Commits that Increased CC

Sokol, F., Aniche, M.F., Gerosa, M.A., "Does the Act of Refactoring Really Make Code Simpler? A Preliminary Study",. In: ***IV Brazilian Workshop of Agile Methods (WBMA 2013)****.*

Why do newcomers dropout from OSS projects?